BRIN
BADAN RISET
DAN INOVASI NASIONAL

LANGKAH AWAL MENGUASAI BAHASA PEMROGRAMAN PYTHON

Budhi Gustiandi

LANGKAH AWAL MENGUASAI BAHASA **PEMROGRAMAN PYTHON**

Diterbitkan pertama pada 2023 oleh Penerbit BRIN

Tersedia untuk diunduh secara gratis: penerbit.brin.go.id

LANGKAH AWAL MENGUASAI BAHASA **PEMROGRAMAN PYTHON**

Budhi Gustiandi

Penerbit BRIN

Katalog dalam Terbitan (KDT)

Langkah Awal Menguasai Bahasa Pemrograman Python/Budhi Gustiandi–Jakarta: Penerbit BRIN, 2023.

xiv + 203 hlm.; 14,8 x 21 cm

ISBN 978-623-8372-21-8 *(e-book)*

1. Bahasa Pemrograman 2. Python
3. Komputer

005.262

Editor Akuisisi : Wijananto
Copy editor : Sonny Heru Kusuma
Proofreader : Annisa' Eskahita Azizah & Noviastuti Putri Indrasari
Penata isi : Hilda Yunita & S. Imam Setyawan
Desainer Sampul : Budhi Gustiandi & Hilda Yunita
Ilustrasi Sampul : Hasil olahan AI

Cetakan Pertama : November 2023

Diterbitkan oleh:
Penerbit BRIN, Anggota Ikapi
Direktorat Repositori, Multimedia, dan Penerbitan Ilmiah
Gedung B.J. Habibie Lt. 8, Jl. M.H. Thamrin No. 8,
Kb. Sirih, Kec. Menteng, Kota Jakarta Pusat,
Daerah Khusus Ibukota Jakarta 10340
Whatsapp: +62 811-1064-6770
E-mail: penerbit@brin.go.id
Website: penerbit.brin.go.id
Penerbit BRIN
@Penerbit_BRIN
@penerbit.brin

Daftar Isi

Daftar Gambar

Daftar Tabel

Pengantar Penerbit

Sebagai penerbit ilmiah, Penerbit BRIN mempunyai tanggung jawab untuk terus berupaya menyediakan terbitan ilmiah yang berkualitas. Upaya tersebut merupakan salah satu perwujudan tugas Penerbit BRIN untuk turut serta membangun sumber daya manusia unggul dan mencerdaskan kehidupan bangsa sebagaimana yang diamanatkan dalam pembukaan UUD 1945.

Buku ini mengulas tentang Python yang menjadi salah satu bahasa pemrograman yang paling populer digunakan di dunia. Buku ini selain mengajarkan cara memahami bahasa pemrograman Python, juga membimbing untuk menjalankan program tersebut secara cepat dan fokus. Bahasa pemrograman Python dapat digunakan di dalam berbagai bidang, mulai dari aplikasi-aplikasi umum yang biasa kita temukan di berbagai komputer, aplikasi-aplikasi pada telepon pintar, hingga aplikasi-aplikasi khusus yang berjalan pada perangkat tertentu.

Buku ini diterbitkan melalui Program Akuisisi Pengetahuan Lokal dengan skema akses terbuka yang diharapkan dapat menjadi salah satu referensi bagi pembaca untuk mengetahui salah satu bahasa pemrograman paling populer di dunia, yaitu Python. Akhir kata, kami mengucapkan terima kasih kepada semua pihak yang telah membantu proses penerbitan buku ini.

Penerbit BRIN

Prakata

Python merupakan salah satu bahasa pemrograman yang paling populer di dunia saat ini dan yang paling sesuai bagi para pemula apabila mereka ingin belajar cara membuat sebuah program komputer. Sayangnya, sebagian besar buku atau referensi yang tersedia saat ini masih memiliki konten yang terlalu detail untuk sebuah topik, menggunakan banyak jargon, urutan pembahasan yang meloncat-loncat, dan sepertinya belum benar-benar ditujukan bagi yang ingin belajar pada level pemula, yang belum pernah membuat sebuah program komputer sama sekali. Penulis sudah pernah mengalaminya dan tidak ingin Anda mengalami hal yang sama.

Di dalam buku ini, Anda akan mempelajari bahasa pemrograman Python benar-benar dari awal, di lingkungan yang paling sesuai untuk para pemula, dan tentunya dengan cara yang paling mudah dan menyenangkan. Anda akan belajar bahasa pemrograman Python langkah demi langkah dengan pertama-tama mempelajari konsep dari suatu topik, mewujudkan konsep tersebut dalam baris kode, dan

menyelesaikan tantangan-tantangan untuk memastikan bahwa Anda benar-benar memahami apa yang telah Anda pelajari. Uniknya, Anda tidak akan dipusingkan oleh spesifikasi minimum komputer yang harus Anda gunakan karena pembelajaran di dalam buku ini tidak memerlukan instalasi perangkat lunak sama sekali. Selama komputer Anda memiliki peramban web (*web browser*) yang terhubung dengan internet, Anda dapat melakukan penulisan baris-baris kode program langsung di peramban web Anda. Anda juga akan diberikan tiga contoh proyek lengkap beserta contoh solusi yang dapat memandu Anda dalam menyelesaikan proyek tersebut secara detail, yaitu "Bilangan dan Kutipan Acak", "Penganalisis Teks", dan "Permainan Tebak Angka". Jadi, apabila Anda telah siap untuk mempelajari salah satu bahasa pemrograman yang paling populer di dunia dalam cara yang mudah, menyenangkan, dan ramah bagi pemula, mari kita mulai!

Penulis

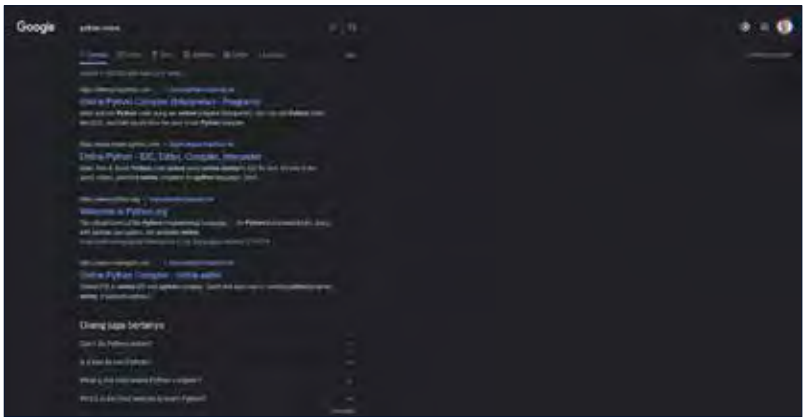
Bab 1 Pendahuluan

Selamat datang di awal pembelajaran Anda. Apabila Anda benar-benar memulai dari awal untuk belajar bahasa pemrograman Python, Anda berada di tempat yang tepat. Hal yang paling menyenangkan dalam belajar Python adalah bahasa pemrograman tersebut sangat populer sehingga dapat dijalankan hampir di mana pun, baik di dalam komputer berbasis sistem operasi Windows, Mac, Linux, maupun hanya dengan menggunakan peramban web (*web browser*) saja. Bahkan, bahasa pemrograman Python dapat diterapkan di berbagai sistem tertanam (*embedded system*), seperti pada mikrokontroler. Bahasa pemrograman tersebut dapat digunakan di dalam berbagai bidang, mulai dari aplikasi-aplikasi umum yang biasa Anda temukan di berbagai komputer, aplikasi-aplikasi pada telepon pintar Anda, hingga aplikasi-aplikasi khusus yang berjalan pada perangkat tertentu.

Di dalam buku ini, untuk membantu Anda mempelajari Python secara cepat dan fokus, Anda akan dibimbing dan diarahkan untuk menulis dan menjalankan kode dalam bahasa pemrograman Python

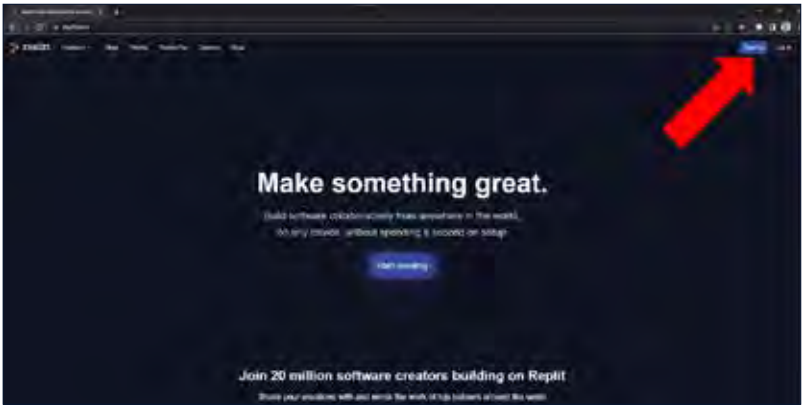
di dalam peramban web (*web browser*) saja sehingga untuk langkah awal ini, Anda tidak akan direpotkan dan dipusingkan dengan berbagai aspek teknis dalam hal instalasi Python serta aplikasi-aplikasi pendukung lainnya. Untuk itu, tentunya, Anda akan dapat melakukan pemrograman di mana pun menggunakan komputer apa pun selama terhubung dengan internet. Anda tidak perlu melakukan instalasi ulang ketika Anda berpindah komputer.

Banyak sekali alternatif situs web yang dapat digunakan untuk menulis, menjalankan, dan menguji kode-kode yang dibuat dengan bahasa pemrograman Python, seperti yang diperlihatkan pada Gambar 1.1. Namun, untuk kemudahan penggunaan, situs web yang akan digunakan di dalam buku ini adalah situs web Replit dengan alamat https://replit.com, seperti yang diperlihatkan pada Gambar 1.2.



Sumber: Google.com (2023)

Gambar 1.1 Tampilan Hasil Pencarian dengan Menggunakan Kata Kunci "Python online"



Sumber: Replit.com (8 Maret 2023)

Gambar 1.2 Tampilan Halaman Depan Situs Web Replit (Tampilan bisa berubah sewaktu-waktu)

Hal pertama yang harus Anda lakukan agar dapat menggunakan situs web Replit adalah membuat sebuah akun dengan menekan tombol **"Sign Up"** yang berada di bagian kanan atas halaman depan situs web, sebagaimana diperlihatkan oleh tanda panah berwarna merah pada Gambar 1.2. Kemudian, Anda akan diarahkan ke sebuah halaman yang menampilkan kotak-kotak isian sebagaimana diperlihatkan pada Gambar 1.3. Anda perlu mengisikan nama pengguna yang ingin Anda pakai di dalam kolom isian **"Username"**, alamat *email* aktif Anda pada kolom isian **"Email"**, dan kata kunci untuk akun Replit Anda pada kolom isian **"Password"**, kemudian menekan tombol **"+ Create account"**.

Anda juga dapat membuat akun Replit baru dengan menggunakan akun Google, Github, atau Facebook Anda dengan menekan tombol **"Continue with Google"**, **"Continue with Github"**, atau **"Continue with Facebook"** pada bagian bawah halaman tersebut. Selebihnya, Anda dapat mengikuti petunjuk-petunjuk yang diberikan untuk menyelesaikan pendaftaran Anda di dalam situs web tersebut, seperti

memilih nama yang akan ditampilkan di dalam profil Anda, foto profil Anda, dan siapa saja yang dapat mengakses profil Anda.

Setelah Anda berhasil memiliki akun di dalam situs web tersebut dan masuk (*log in*) ke dalam situs web tersebut, Anda perlu membuat sebuah **Repl**. **Repl** merupakan istilah yang digunakan di dalam situs web tersebut untuk lingkungan kerja yang dapat Anda gunakan untuk menulis, menyunting, dan menguji kode-kode program yang Anda tulis. Jadi, apabila Anda perlu membuat sebuah proyek baru, Anda harus membuat sebuah **Repl**. Satu **Repl** dapat berisi lebih dari satu berkas sehingga dapat disesuaikan dengan kompleksitas program yang Anda tulis. Penulis menyarankan agar Anda membuat sebuah **Repl** baru untuk setiap tahapan pembelajaran Anda.

Gambar 1.3 Tampilan Kolom-Kolom Isian bagi Para Pengguna yang Baru Melakukan Pendaftaran di dalam Situs Web Replit

A. Sekilas tentang Bahasa Pemrograman Python

Tak kenal maka tak sayang. Karena itu, pada bagian ini, penulis ingin memperkenalkan secara sekilas tentang perkembangan bahasa pemrograman Python. Python adalah sebuah bahasa pemrograman yang pertama kali dikembangkan pada akhir tahun 1980-an oleh seorang pemrogram periset berkebangsaan Belanda bernama Guido van Rossum. Awalnya, bahasa pemrograman ini beliau namakan ABC. Namun, kemudian, beliau memberi nama bahasa pemrograman yang beliau ciptakan tersebut menjadi Python, terinspirasi dari sebuah acara televisi favorit beliau bernama *Monty Python's Flying Circus*. Beliau tetap menjadi pengembang utama bahasa pemrograman Python sampai saat ini dan diakui oleh komunitas pengembang bahasa pemrograman Python sebagai *Benevolent Dictator for Life* (BDFL), yang berarti beliaulah pengambil keputusan terakhir terkait dengan pengembangan bahasa pemrograman Python apabila diperlukan.

Rilis pertama bahasa pemrograman Python adalah pada Januari tahun 1994 dengan nama Python 1.0. Versi 2.0 dari Python dirilis pada bulan Desember tahun 2000. Bahasa pemrograman Python terus-menerus dikembangkan sesuai dengan kebutuhan para penggunanya. Saat ini terdapat dua varian utama dari bahasa pemrograman Python yang banyak digunakan, yaitu Python 2 dan Python 3. Python 3 sendiri dirilis pertama kali pada bulan Desember tahun 2008. Buku ini akan mengacu ke bahasa pemrograman Python 3, Anda akan disajikan jenis data yang lebih sederhana dibandingkan yang ada pada bahasa pemrograman Python 2. Di samping itu, bahasa pemrograman Python 3 sudah berkembang sejak lama. Walaupun masih ada program-program tertentu yang menggunakan bahasa pemrograman Python 2, untuk program-program yang baru, hampir seluruhnya menggunakan bahasa pemrograman Python 3. Bahasa pemrograman Python 2 sudah mulai ditinggalkan oleh pemrogram-pemrogram terkini. Penulis

sangat menganjurkan Anda untuk menggunakan bahasa pemrograman Python 3 alih-alih bahasa pemrograman Python 2, apalagi dukungan resmi terhadap versi 2 telah diakhiri sejak tahun 2020 yang lalu.

Saat ini, bahasa pemrograman Python menjadi salah satu bahasa pemrograman yang paling populer digunakan di dunia. Bahkan, bahasa pemrograman ini pun sangat populer di tingkat nasional, baik di dunia akademisi maupun di dunia kerja (Supardi & Dede, 2020). Hal ini tidak terlepas dari penggunaan kode-kodenya yang mendekati bahasa manusia.

Kepopuleran bahasa pemrograman ini juga dikarenakan bahasa tersebut sangat mendukung penerapan di bidang-bidang ilmu data (*data science*) yang sedang "naik daun" saat ini, seperti pembelajaran mesin (*machine learning*), kecerdasan buatan (*artificial intelligence*), pengenalan pola (*pattern recognition*), bioinformatika, robotika, antariksa, dan bidang-bidang lainnya (Rangkuti dkk., 2021). Bahkan, perusahaan-perusahaan teknologi besar dunia yang dikenal sebagai GAMAM (Google, Amazon, Meta, Apple, dan Microsoft) juga telah menggunakan Python dalam mengembangkan banyak aplikasi milik mereka. Di samping itu, yang terpenting adalah bahasa pemrograman Python memiliki lisensi *open source* dan *general public license* (GPL) yang berarti bahasa pemrograman tersebut dapat digunakan secara gratis dan didistribusikan dengan bebas bahkan untuk kepentingan komersial (Enterprise, 2016, 2017, 2019; Huda & Ardi, 2020; JUD, 2016).

Python adalah bahasa yang sangat lengkap (Mastrodomenico, 2022). Python dapat mendukung berbagai teknik pemrograman, baik pemrograman berorientasi object (*object oriented programming*/OOP) maupun pemrograman berbasis fungsi (*functional programming*). Banyak sekali pustaka-pustaka standar (*standard libraries*) yang

disediakan secara bawaan oleh bahasa pemrograman Python sehingga para pengguna tidak perlu membuat baris-baris kode dari awal untuk fungsi-fungsi yang sudah biasa digunakan secara umum di dalam pembuatan program. Tidaklah mungkin bagi penulis untuk menjabarkan semuanya di dalam buku ini, apalagi di dalam buku ini materi yang disajikan adalah materi-materi yang paling sesuai untuk pemelajar pemula Python. Dokumentasi bahasa pemrograman Python secara daring tersedia di situs web dengan alamat https://www.python.org/doc/, seperti yang diperlihatkan pada Gambar 1.4. Penulis menyarankan setelah Anda menyelesaikan buku ini, Anda dapat membaca dokumentasi tersebut dan terbiasa dengannya untuk mencari hal-hal yang lebih mendalam lagi mengenai topik-topik yang dibahas di dalam buku ini maupun untuk topik-topik yang belum dibahas di dalam buku ini.

Sumber: Python (2023)

Gambar 1.4 Tampilan Halaman Situs Web yang Memberikan Dokumentasi Bahasa Pemrograman Python secara Lengkap

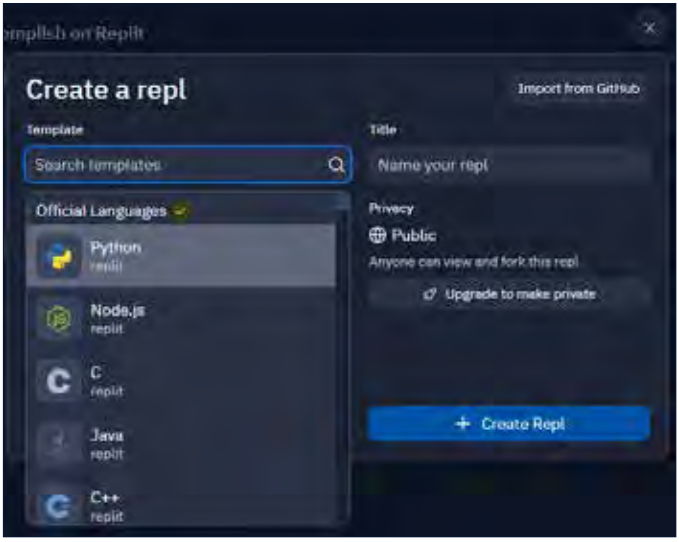
B. Baris Kode Pertama Anda

Tampilan peramban web Anda mungkin berbeda sebagaimana dengan yang diperlihatkan pada Gambar 1.5 ketika Anda telah masuk ke dalam situs web Replit. Namun, secara garis besar, fungsi-fungsi yang ditampilkan adalah sama. Perbedaan tampilan tersebut hanyalah dikarenakan pilihan-pilihan yang Anda buat ketika melakukan pengaturan di awal Anda mengaktifkan akun Anda. Untuk membuat sebuah **Repl** baru, Anda dapat melakukan langkah-langkah sebagai berikut.

1) Klik tombol **"Plus"** yang terdapat pada bagian kanan atas atau yang berada tepat di bawah bagian **"Create"** sebagaimana ditunjukkan oleh tanda panah merah pada Gambar 1.5.

Gambar 1.5 Tampilan Halaman Situs Web Replit setelah Anda *Log in* ke Situs Web Tersebut

2) Setelah Anda mengeklik tombol **"Plus"** tersebut, sebuah jendela **"Create a repl"** akan ditampilkan sebagaimana diperlihatkan pada Gambar 1.6.



Gambar 1.6 Tampilan Jendela "**Create a repl**" yang Dimunculkan setelah Anda Menekan Tombol "**Plus**" untuk Membuat Sebuah **Repl** Baru

3) Pilihlah "**Python**" pada bagian kiri pilihan bahasa program yang ingin Anda gunakan.
4) Berilah judul program yang akan dibuat pada kolom isian "**Title**". Pada pembelajaran kali ini, penulis memberi nama **Baris Kode Pertamaku** pada kolom isian tersebut. Anda dapat memberikan nama apa pun yang relevan sesuai dengan nama program yang ingin Anda buat.
5) Kemudian, klik tombol "**+ Create repl**" pada bagian kanan bawah jendela tersebut. Tunggu beberapa saat, Replit akan menyiapkan lingkungan pemrograman untuk Anda sebagaimana diperlihatkan pada Gambar 1.7.

Gambar 1.7 Tampilan Lingkungan Pemrograman dalam Situs Web Replit

Lingkungan pemrograman tersebut terbagi atas tiga kolom utama. Kolom sebelah kiri (**"Files"**) memberikan informasi mengenai berkas-berkas apa saja yang terdapat di dalam **Repl** yang Anda buat. Pada Gambar 1.7 terlihat hanya ada satu berkas saja, yaitu **"main.py"**. Berkas-berkas yang digunakan di dalam bahasa pemrograman Python akan memiliki nama dengan ekstensi **".py"** seperti pada berkas tersebut. Untuk program yang lebih kompleks, akan terdapat banyak berkas yang ditampilkan di kolom **"Files"** tersebut. Namun, untuk saat ini, Anda cukup fokus pada satu berkas **"main.py"** saja.

Kolom bagian tengah merupakan tempat Anda melakukan penyuntingan terhadap berkas yang ditampilkan di dalam kolom sebelah kiri. Pada Gambar 1.7 terlihat bahwa kolom tengah memiliki judul **"main.py"** yang berarti Anda melakukan penyuntingan untuk berkas **"main.py"** yang ditampilkan di sebelah kiri kolom.

Mari mulai menuliskan baris kode pertama Anda dengan mengeklik kolom tengah tersebut. Namun, sebelumnya, ada yang ingin penulis tanyakan kepada Anda. Apa yang Anda lakukan jika bertemu seseorang yang baru Anda kenal? Biasanya Anda atau seseorang tersebut akan memberikan kalimat salam dan memperkenalkan diri.

Kalimat apa yang biasanya Anda ucapkan? Anda dapat mengucapkan "selamat pagi", "selamat siang", "selamat malam", atau "salam kenal". Nah, sekarang kalimat salam apa yang biasanya disampaikan oleh sebuah program komputer kepada pengguna baru seperti Anda? Di dalam dunia pemrograman, terdapat satu ungkapan yang digunakan secara universal oleh sebuah program ketika menyapa pengguna untuk pertama kalinya. Ungkapan tersebut adalah "Hello World!". Ungkapan tersebut digunakan untuk memastikan bahwa program sudah dapat berjalan dengan baik pada komputer yang digunakan atau dapat berkomunikasi dengan baik dengan penggunanya.

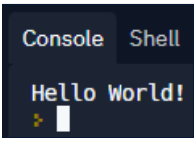
Mari kita kembali ke kolom tengah pada Gambar 1.7. Ketikkan **`print('Hello World!')`** sebagaimana diperlihatkan pada Gambar 1.8.

Gambar 1.8 Tampilan Kolom Penyuntingan (Kolom Tengah pada Gambar 1.7) setelah Baris **Kode print("Hello World!")** Dituliskan pada Kolom Tersebut

Perhatikan pada saat Anda mengetikkan tanda kurung buka pada baris kode tersebut, maka Replit juga akan menyediakan tanda kurung tutup secara otomatis. Begitu juga ketika Anda mengetikkan tanda petik tunggal pada baris kode tersebut, Replit akan menyediakan tanda petik tunggal kedua secara otomatis. Hal tersebut untuk mempermudah seseorang dalam menuliskan baris kode dan untuk menghindari kesalahan penulisan baris kode yang disebabkan karena kurangnya tanda kurung tutup atau tanda petik tunggal di akhir baris kode. Kekurangan hanya satu tanda baca saja dapat menyebabkan baris kode tidak dikenali oleh komputer sehingga program akan gagal dijalankan. Anda juga dapat menggunakan tanda petik ganda alih-alih tanda petik tunggal yang penulis gunakan pada contoh ini.

Namun, Anda harus memastikan bahwa jenis tanda petik yang Anda gunakan untuk membuka dan menutup kalimat yang ingin Anda tuliskan harus sama. Perbedaan penggunaan jenis tanda petik juga dapat menyebabkan kesalahan dalam program dan mengakibatkan program tidak dapat dijalankan.

Langkah selanjutnya adalah mengeklik tombol **"► Run"** sebagaimana ditunjukkan oleh tanda panah merah pada Gambar 1.7. Kolom sebelah kanan, yang dinamakan **"Console"**, akan menampilkan keluaran dari program sebagaimana diperlihatkan pada Gambar 1.9. **Console** inilah yang akan digunakan oleh program untuk berkomunikasi dengan pengguna. Selain menampilkan keluaran dari program, pengguna juga akan dapat berinteraksi dengan program seperti memberikan masukan ke dalam program untuk dijalankan oleh program tersebut. Hal ini akan dibahas lebih detail pada bagian-bagian selanjutnya.



Gambar 1.9 Tampilan Keluaran Baris Perintah **print('Hello World!')** pada Kolom **Console** ketika Dijalankan

Untuk mempermudah pembacaan baris-baris kode program di bagian-bagian selanjutnya pada buku ini, tampilan kolom penyuntingan seperti yang diperlihatkan pada Gambar 1.8 akan penulis sajikan seperti yang diperlihatkan pada `Kode Program 1`. Penulis pun akan menandai dengan warna berbeda baris-baris kode yang menjadi perhatian utama di dalam sebuah kode program yang harus Anda ubah atau tambahkan.

Kode Program 1

```
1 print('Hello World!')
```

Angka pada kolom sebelah kiri menunjukkan nomor baris kode Anda dan kolom sebelah kanan memperlihatkan baris kode yang harus Anda tuliskan di dalam kolom penyuntingan tersebut. Sementara itu, untuk tampilan kolom **Console** seperti yang diperlihatkan pada Gambar 1.9 akan penulis sajikan seperti yang diperlihatkan pada `Console 1`.

```
Console 1
Hello World!
> 
```

Hal ini dikarenakan di bagian-bagian selanjutnya pada buku ini jumlah baris kode yang Anda tuliskan lambat laun akan bertambah seiring dengan tingkat kerumitan program yang akan Anda buat. Tanda "▓" pada kolom **Console** tersebut menunjukkan posisi *pointer* seperti yang ditampilkan pada Gambar 1.9.

Selamat! Anda telah berhasil membuat program pertama Anda dengan menggunakan bahasa pemrograman Python. Di sini, Anda telah belajar konsep paling awal dari bahasa pemrograman Python, yaitu apabila Anda ingin menampilkan sebuah teks di kolom **Console**, Anda dapat melakukannya dengan mengetikkan **`print()`** kemudian mengetikkan teks yang ingin ditampilkan di antara tanda kurung **`print()`** dengan memberi tanda kutip tunggal (') atau ganda (") yang mengapit teks tersebut. Untuk menjalankan program yang telah Anda tulis dan melihat hasilnya di kolom **Console**, klik tombol **"► Run"** sebagaimana dijelaskan sebelumnya.

Hal seperti inilah yang juga membuat bahasa pemrograman Python menjadi populer. Untuk menghasilkan keluaran yang sama dengan menggunakan bahasa pemrograman yang lain, Anda tidak akan cukup hanya dengan menuliskan satu baris kode seperti yang telah Anda lakukan sebelumnya dengan menggunakan bahasa pemrograman Python. Sebagai gambaran, untuk menampilkan teks

“Hello World!” pada kolom **Console** dengan menggunakan bahasa C, C++, dan Java adalah seperti yang diperlihatkan sebagai berikut.

Bahasa pemrograman C

```
1  #include <stdio.h>
2  int main(void)
3  {
4   print(“Hello World!”);
5  }
```

Bahasa pemrograman C++

```
1  #include <iostream.h>
2  int main()
3  {
4   std::cout << “Hello World!”;
5   return 0;
6  }
```

Bahasa pemrograman Java

```
1  class HelloWorld()
2  {
3   public static void main(String[] args)
4   {
5    System.out.println(“Hello World!);
6   }
7  }
```

Apa yang terjadi apabila Anda tidak memasukkan teks apa pun ke dalam tanda kurung **`print`**() tersebut? Kolom **Console** akan menampilkan satu baris kosong untuk setiap **`print`**() yang Anda ketikkan tanpa memberikan teks di dalamnya. Misalkan Anda ingin menampilkan sebuah kalimat baru di bawah kalimat **“Hello World!”** yang telah Anda buat sebelumnya dengan memberi jeda sebuah baris kosong di antara kedua kalimat tersebut maka Anda dapat mengetikkan **`print`**() tanpa teks di baris kedua kemudian mengetikkan **`print`**() dengan teks yang Anda inginkan di baris ketiga

seperti yang diperlihatkan pada Kode Program 2. Ketika kode program tersebut dijalankan, di kolom **Console** akan ditampilkan dua kalimat yang Anda masukkan di dalam tanda kurung **print**() dengan jeda satu baris kosong di antara kedua kalimat tersebut seperti yang diperlihatkan pada Console 2.

Kode Program 2

```
1  print('Hello World!')
2  print()
3  print('Semangat pagi dan salam sehat!')
```

Console 2

```
Hello World!

Semangat pagi dan salam sehat!
>
```

Cara lain yang lebih mudah untuk menampilkan lebih dari satu baris di kolom **Console** adalah dengan menggunakan tanda kutip tunggal tiga kali di dalam tanda kurung **print**() seperti yang diperlihatkan pada Kode Program 3. Di dalam tanda kutip tunggal tiga kali tersebut, Anda dapat memasukkan beberapa baris teks untuk dapat ditampilkan di dalam kolom **Console** secara bersamaan. Ketika kode program tersebut dijalankan, tampilan pada kolom **Console** akan serupa seperti sebelumnya, yaitu yang ditampilkan pada Console 3. Terdapat sedikit perbedaan di antara kedua hasil tersebut. Ketika menggunakan tanda petik tunggal tiga kali, pada baris pertama tampilan di kolom **Console** akan ditampilkan sebuah baris kosong. Hal ini dikarenakan baris pertama kode di atas akan diterjemahkan sebagai baris kosong oleh program.

Kode Program 3

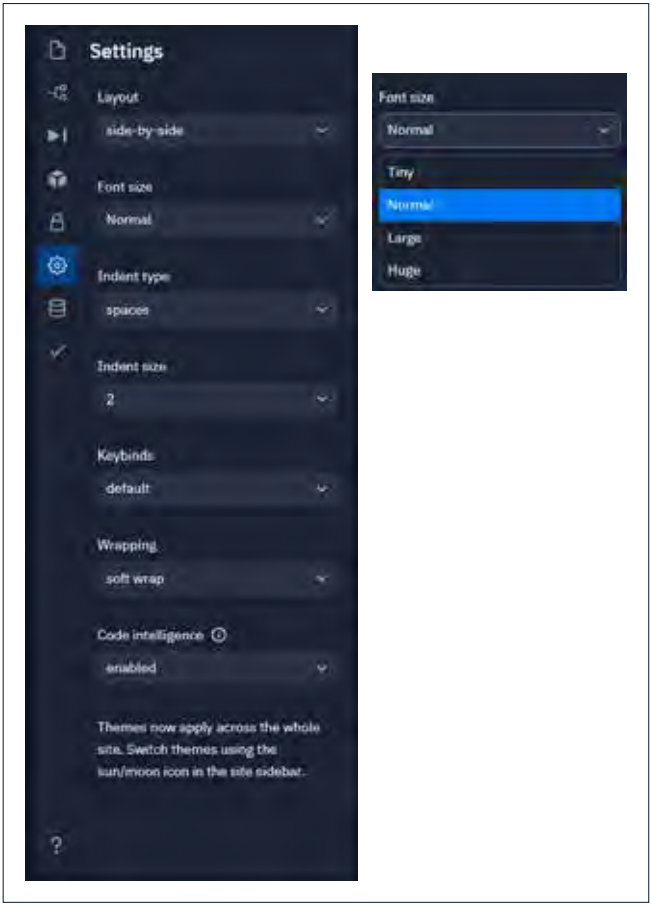
```
1  print('''
2  Hello World!
3
4  Semangat pagi dan salam sehat! ''')
```

Console 3

```
Hello World!

Semangat pagi dan salam sehat!
>
```

Sebelum mengakhiri bagian ini, penulis ingin memberikan sedikit tips terkait penggunaan aplikasi Replit. Anda dapat melakukan pengaturan bagaimana tampilan pada Gambar 1.7 sesuai dengan kebutuhan Anda dengan mengeklik tombol **"Settings"** yang merupakan tombol keenam dari atas kolom tombol yang terdapat pada bagian kiri, sebagaimana ditunjukkan oleh tanda panah kuning pada Gambar 1.7. Jendela **"Settings"** akan ditampilkan pada kolom sebelah kiri seperti diperlihatkan pada Gambar 1.10. Pada jendela tersebut, Anda dapat melakukan pengaturan seperti ukuran huruf atau karakter (*font*) yang ingin ditampilkan pada lingkungan pemrograman Anda dengan mengeklik menu *drop-down* **"Font size"** dan memilih ukuran font seperti kecil **"Tiny"**, normal **"Normal"**, besar **"Large"**, dan sangat besar **"Huge"**. Anda juga dapat mengatur indentasi pada baris-baris kode program Anda melalui menu *drop-down* **"Indent size"**. Indentasi menjadi hal yang sangat penting di dalam bahasa pemrograman Python. Hal ini dibahas pada bab-bab selanjutnya.



Gambar 1.10 Tampilan Jendela **"Settings"** untuk Melakukan Pengaturan Tampilan Lingkungan Pemrograman Sesuai Kebutuhan Anda

Dasar-Dasar Bahasa Pemrograman Python

Bab 2

Sekarang Anda telah berhasil membuat baris kode pertama Anda dengan menggunakan bahasa pemrograman Python dan mungkin juga sudah mulai terbiasa dengan situs web Replit yang Anda gunakan untuk menuliskan dan menjalankan baris-baris kode program Anda. Pada bab ini, Anda mempelajari fondasi-fondasi yang paling penting dalam semua bahasa pemrograman, termasuk Python. Anda dapat belajar mengenai variabel, data yang dapat dimasukkan ke dalam variabel (seperti numerik, *string*, dan *boolean*), dan alur pengambilan keputusan di dalam sebuah program komputer.

A. Variabel

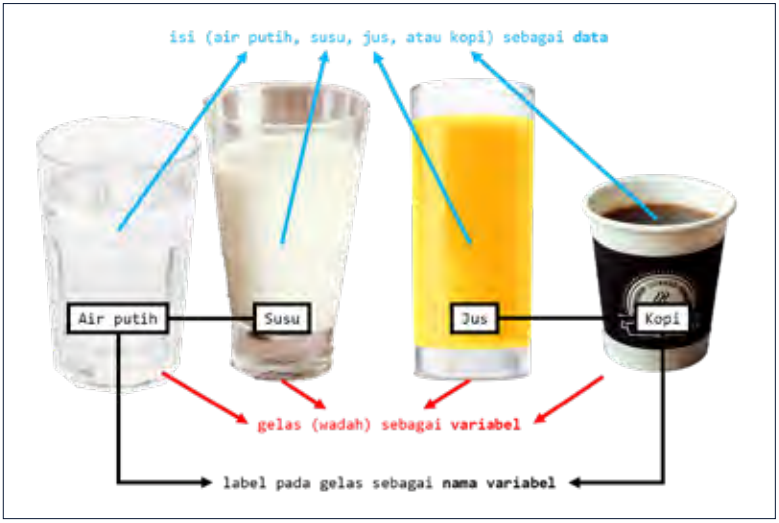
Salah satu fondasi yang paling penting dari bahasa pemrograman adalah variabel. Apa itu variabel? Sederhananya, variabel adalah tempat yang dapat menyimpan data di dalam sebuah program. Variabel dapat menyimpan angka atau numerik, karakter atau teks, *boolean*,

dan jenis-jenis data lainnya. Kita akan membahas masing-masing jenis data tersebut pada bagian-bagian selanjutnya. Pada bagian ini, kita akan fokus membahas tempat untuk menyimpan data tersebut, yaitu variabel.

Bayangkan terdapat empat buah gelas dengan isi minuman yang berbeda-beda, seperti yang diperlihatkan pada Gambar 2.1. Sebagaimana disebutkan sebelumnya, variabel adalah sebuah tempat (atau wadah) untuk menyimpan data. Dalam hal ini, gelas yang merupakan wadah dapat diumpamakan sebagai variabel, sedangkan isi dari gelas tersebut dapat diumpamakan sebagai data yang disimpan di dalam variabel. Isi dari gelas dapat bermacam-macam. Pada gambar tersebut dapat terlihat bahwa gelas dapat berisi air putih, susu, jus, atau kopi. Hal ini mengumpamakan bahwa variabel di dalam sebuah program dapat berisi data yang jenisnya dapat disesuaikan dengan kebutuhan penggunanya.

Gelas yang digunakan di dalam contoh ini adalah gelas yang transparan sehingga Anda dapat dengan mudah mengetahui isi dari masing-masing gelas. Namun, bagaimana apabila gelas yang digunakan tidak tembus pandang? Bagaimana cara Anda mengetahui apa isi yang ada di dalam sebuah gelas? Anda dapat memberikan label pada masing-masing gelas sesuai dengan isinya sehingga Anda dapat mengetahui isi dari gelas tersebut hanya dengan melihat labelnya saja. Hal ini mengumpamakan Anda memberikan nama pada sebuah variabel yang menunjukkan data yang ada di dalam variabel tersebut. Oleh karena itu, dalam pemberian nama sebuah variabel, Anda disarankan untuk memberikan nama yang sesuai dengan isi data yang terdapat di dalam variabel tersebut agar tidak tertukar penggunaannya dengan variabel yang lainnya.

Misalkan sekarang Anda ingin membuat sebuah variabel yang merepresentasikan tentang jumlah uang yang ada di dalam dompet



Sumber: fawaz-qureshi (2023), Penelope883 (2022), Ragabz (2021), mojzagrebinfo (2014)

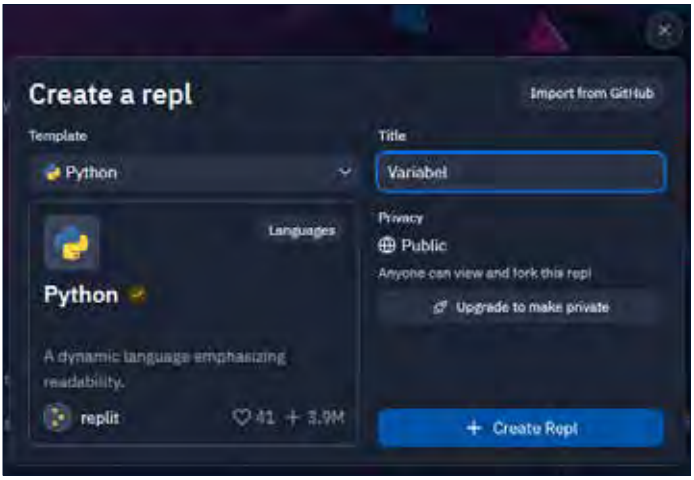
Gambar 2.1 Perumpamaan Variabel, Data, dan Nama Variabel pada Berbagai Macam Gelas yang Berisi Minuman Berbeda

Anda. Tentunya jumlah uang tersebut tidak akan tetap sepanjang waktu, tetapi bisa saja bertambah atau berkurang sesuai dengan kondisi keuangan Anda. Situasi seperti inilah yang sesuai untuk menerapkan konsep variabel karena isi dari variabel dapat Anda ubah sesuai dengan kebutuhan program yang Anda buat.

Karena Anda akan belajar sebuah konsep yang baru, Anda sebaiknya membuat sebuah **Repl** baru. Anda dapat kembali ke halaman *dashboard* akun Anda dengan mengeklik logo Replit yang terdapat pada bagian kiri atas halaman tempat Anda memasukkan dan menjalankan baris kode program sebagaimana diperlihatkan oleh tanda panah merah pada Gambar 2.2. Anda akan kembali diarahkan ke halaman seperti yang diperlihatkan pada Gambar 1.5 di bab sebelumnya.

Sekarang, buatlah sebuah **Repl** baru. Jangan lupa pastikan bahwa bahasa pemrograman Python dipilih di dalam **Repl** baru Anda. Karena Anda akan belajar mengenai variabel, Anda dapat menamai **Repl** tersebut dengan nama **variabel** seperti yang diperlihatkan pada Gambar 2.3. Setelah Anda mengeklik tombol **"+ Create Repl"** pada jendela **"Create a repl"**, Anda akan kembali diarahkan ke halaman penyuntingan sebagaimana diperlihatkan pada Gambar 2.2.

Gambar 2.2 Tampilan Halaman Tempat Menuliskan dan Menjalankan Baris Kode Program pada Situs Web Replit



Gambar 2.3 Tampilan Jendela "Create a repl" untuk Belajar mengenai Variabel

Bayangkan Anda memiliki uang sejumlah Rp100.000,00 di dalam dompet Anda. Anda ingin merepresentasikan situasi tersebut di dalam baris kode program Python. Hal pertama yang harus Anda lakukan adalah memberi nama variabel yang ingin Anda gunakan untuk menyimpan informasi jumlah uang di dalam dompet Anda tersebut.

Misalkan Anda memberi nama variabel Anda sebagai **`dompet`**. Seperti yang telah dijelaskan sebelumnya, variabel berisi data. Nah, bagaimana cara kita memasukkan data tersebut (dalam hal ini jumlah uang yang ada di dalam dompet Anda) ke dalam variabel yang telah Anda buat (dalam hal ini variabel **`dompet`**)? Caranya adalah Anda ketikkan nama variabel Anda, berikan spasi (untuk mempermudah pembacaan baris kode), ketikkan tanda sama dengan (=), berikan spasi lagi, dan ketikkan nilai dari variabel tersebut sebagaimana diperlihatkan pada `Kode Program 4`. Perhatikan bahwa nilai yang dimasukkan tidak memakai tanda titik untuk memisahkan bilangan ribuan.

Kode Program 4

```
1  dompet = 100000
```

Kemudian Anda ingin melihat hasil dari baris kode yang Anda ketikkan tersebut di kolom **Console** (kolom sebelah kanan pada Gambar 2.2) untuk memastikan bahwa baris kode yang Anda ketikkan sudah benar. Anda pun mengeklik tombol **"► Run"** sebagaimana ditunjukkan oleh tanda panah merah pada Gambar 1.7. Namun, Anda ternyata tidak melihat apa pun yang muncul di kolom **Console** tersebut. Anda pun berpikir bahwa mungkin Anda salah mengetikkan baris kode. Alasan sebenarnya mengapa tidak ada informasi apa pun yang ditampilkan di kolom **Console** tersebut adalah di dalam pemrograman Python, apabila Anda ingin menampilkan data yang Anda miliki di dalam baris-baris kode Anda, Anda harus mengetikkan

`print()`sebagaimana dijelaskan pada bab sebelumnya. Itulah sebabnya **`print()`** tersebut dijelaskan paling pertama kali ketika Anda memulai untuk belajar pemrograman dengan menggunakan bahasa pemrograman Python ini.

Mari kita tambahkan baris kode baru ke dalam baris kode yang sudah Anda buat. Berikan jeda satu baris kosong di antara kedua baris untuk mempermudah pembacaan program. Biasanya seorang pemrogram akan memberikan satu baris kosong yang memisahkan antara baris-baris yang berfungsi untuk membuat variabel dan baris-baris yang berfungsi untuk mengeksekusi perintah-perintah tertentu di dalam sebuah program. Ketikkan **`print(dompet)`** pada baris ketiga kode Anda sebagaimana diperlihatkan pada `Kode Program 5`. Kemudian klik tombol **"► Run"**. Anda akan melihat angka **`100000`** ditampilkan di kolom **Console** sebagaimana diperlihatkan pada `Console 4`.

Kode Program 5

```
1  dompet = 100000
2
3  print(dompet)
```

Console 4

```
100000
>
```

Dari contoh tersebut dapat terlihat bahwa baris kode **`print()`** tidak hanya dapat menampilkan teks di kolom **Console**, tetapi juga dapat digunakan untuk menampilkan isi dari sebuah variabel. Perhatikan bahwa ketika Anda ingin menampilkan isi dari sebuah variabel di kolom **Console** dengan menggunakan baris kode **`print()`**, Anda tidak perlu menggunakan tanda kutip tunggal atau ganda di dalam tanda kurungnya. Apabila Anda menggunakan tanda petik tunggal atau ganda di dalam tanda kurungnya seperti yang diperlihatkan

pada Kode Program 6, yang muncul di kolom **Console** adalah teks **dompet** alih-alih isi dari variabel **dompet** tersebut sebagaimana diperlihatkan pada Console 5.

Kode Program 6

```
1  dompet = 100000
2
3  print('dompet')
```

Console 5

```
dompet
> 
```

Terdapat beberapa aturan yang harus Anda ikuti ketika memberikan nama ke sebuah variabel, sebagai berikut.

1) Karakter pertama dari nama variabel harus berupa huruf atau garis bawah.
2) Karakter pertama dari nama variabel tidak boleh berupa angka.
3) Karakter-karakter yang dapat digunakan di dalam nama sebuah variabel hanyalah karakter-karakter alfanumerik (A–Z, a–z, 0–9) dan garis bawah (_).
4) Nama variabel tidak boleh menggunakan kata-kata kunci di dalam bahasa pemrograman Python dikarenakan kata-kata tersebut digunakan sebagai perintah-perintah untuk menjalankan fungsi tertentu di dalam program. Kata-kata kunci tersebut adalah **and**, **assert**, **break**, **class**, **continue**, **def**, **del**, **elif**, **else**, **except**, **exec**, **finally**, **for**, **from**, **global**, **if**, **import**, **in**, **is**, **lambda**, **not**, **or**, **pass**, **print**, **raise**, **return**, **try**, dan **while**.

Jadi, untuk contoh di atas, nama-nama variabel yang dapat Anda gunakan misalnya **dompet**, **_dompet**, **DOMPET**, **D0MP37**, **D_0_M_P_3_7**, **_D0MP37**, dan sebagainya. Namun, Anda tidak dapat menggunakan nama-nama variabel seperti **3dompet**, **7_dompet**, **d@mp#t**, dan sebagainya. Perlu diketahui bahwa nama-nama variabel sensitif terhadap besar kecilnya huruf yang digunakan. Jadi, variabel **dompet**, **Dompet**, dan **DOMPET** adalah tiga variabel yang berbeda dan masing-masing dapat memiliki isi yang berbeda pula.

Nilai di dalam variabel dapat berubah-ubah, misalnya Anda membeli sesuatu dengan menggunakan sejumlah uang di dalam dompet Anda sejumlah Rp20.000,00. Sekarang Anda hanya memiliki uang sejumlah Rp80.000,00 tersisa di dalam dompet Anda. Anda ingin merepresentasikan kembali jumlah uang Anda tersebut di dalam baris kode. Anda cukup menambahkan baris kode baru dan mengetikkan nama variabel yang ingin Anda ubah nilainya (dalam hal ini **dompet**), mengetikkan spasi, mengetikkan tanda sama dengan (=), mengetikkan spasi kembali, dan mengetikkan nilai baru yang ingin Anda masukkan ke dalam variabel **dompet** tersebut (dalam hal ini **80000**) sebagaimana diperlihatkan pada Kode Program 7.

Kode Program 7

```
1  dompet = 100000
2
3  print(dompet)
4
5  dompet = 80000
6
7  print(dompet)
```

Jalankan baris kode yang telah Anda ubah tersebut dengan mengeklik tombol **"► Run"** pada bagian atas halaman situs web seperti yang telah Anda lakukan pada langkah-langkah sebelumnya. Di kolom **Console** akan ditampilkan dua nilai, yaitu **`100000`** dan **`80000`** seperti yang diperlihatkan pada `Console 6`.

Console 6

```
100000
80000
>
```

Nilai yang pertama merepresentasikan jumlah awal uang yang ada di dalam variabel **`dompet`** yang telah Anda buat, yaitu Rp100.000,00, sedangkan nilai yang kedua merepresentasikan jumlah uang yang ada di dalam variabel **`dompet`** setelah Anda membelanjakan sebagian jumlah uang yang tersedia di dalam variabel tersebut, yaitu Rp80.000,00. Mengapa ada dua nilai yang ditampilkan? Hal tersebut dikarenakan pada baris-baris kode yang Anda ketikkan terdapat dua baris kode **`print(dompet)`** di mana masing-masing baris tersebut akan menampilkan isi dari variabel **`dompet`** sebelum dan sesudah diubah nilainya oleh Anda.

Untuk memastikan bahwa Anda memahami konsep variabel, penulis ingin memberikan sebuah tantangan kepada Anda. Penulis akan memberikan contoh solusi terhadap tantangan yang telah diberikan. Anda bisa saja memiliki solusi yang berbeda dengan solusi yang diberikan oleh penulis. Anda bisa membandingkan solusi Anda sendiri dan solusi yang penulis sediakan untuk lebih memahami konsep dari baris-baris kode program yang Anda buat untuk menjawab tantangan tersebut.

Tantangan 1:
Anda diminta untuk membuat sebuah variabel yang diberi nama **tanggal** dan memberi variabel tersebut nilai dengan tanggal pada saat Anda belajar bagian ini. Anda juga diminta untuk menampilkan nilai variabel tersebut di kolom **Console.**

Contoh solusi Tantangan 1:

Kode Program 8

```
1 tanggal = 12
2
3 print(tanggal)
```

Console 7

```
12
>
```

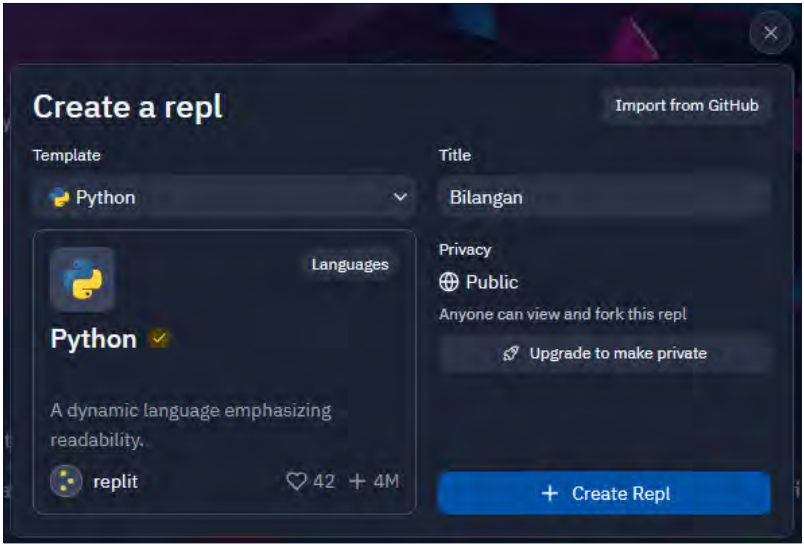
B. Numerik—*Integer* dan *Float*

Terdapat banyak jenis data utama dalam bahasa pemrograman Python. Pada bagian ini, Anda akan mulai mempelajari jenis-jenis data tersebut yang dapat disimpan di dalam variabel dan dimanfaatkan di dalam baris-baris kode program Anda. Jenis data pertama yang akan Anda pelajari adalah numerik atau angka. Secara umum, di dalam bahasa pemrograman Python, sebuah variabel dapat menyimpan dua jenis data numerik, yaitu ***integer*** dan ***float***.

Integer adalah bilangan bulat, sebuah bilangan yang tidak memiliki nilai desimal di dalamnya. Sebagai contoh, pada solusi Tantangan 1 di bagian sebelumnya, penulis menuliskan baris kode **`tanggal = 12`**. Ini adalah contoh dari jenis data ***integer*** karena bilangan tersebut tidak memiliki nilai desimal. ***Integer*** dapat berupa nilai positif atau negatif. Misalnya, ketika Anda melihat tulisan di salah satu kemasan makanan beku, "Simpan di bawah suhu -18°C." Apabila bilangan yang digunakan memiliki nilai negatif, tetapi tidak memiliki nilai desimal di dalamnya, bilangan tersebut masuk ke dalam kategori bilangan ***integer***.

Float adalah bilangan-bilangan yang memiliki nilai desimal di dalamnya. Sebagai contoh, ketika seseorang menimbang berat badan dengan menggunakan timbangan digital, pada timbangan tersebut tertulis 79,20 kg. Ini adalah contoh dari bilangan ***float***. Bilangan tersebut memiliki nilai desimal. Bilangan ***float*** tidak dibatasi hanya dua angka desimal seperti pada contoh tersebut, tetapi dapat disesuaikan dengan kebutuhan bilangan yang ingin direpresentasikan di dalam sebuah variabel. Seperti halnya bilangan ***integer***, bilangan ***float*** juga dapat memiliki nilai positif atau negatif.

Apa yang dapat Anda lakukan dengan kedua jenis data numerik tersebut? Umumnya, di dalam bahasa pemrograman, jenis data numerik digunakan untuk menyelesaikan operasi-operasi matematika, seperti penjumlahan, pengurangan, perkalian, dan pembagian secara langsung (seperti pada penggunaan kalkulator) maupun melalui penggunaan variabel-variabel yang berisi nilai dengan jenis data numerik tersebut.



Gambar 2.4 Tampilan Jendela **"Create a repl"** untuk Membuat Sebuah **Repl** Baru dengan Nama **Bilangan**

Mari kita praktikkan konsep jenis data numerik tersebut di dalam baris kode bahasa pemrograman Python. Anda dapat membuat **Repl** baru seperti pada bagian sebelumnya dan memberi nama **"bilangan"** seperti yang diperlihatkan pada Gambar 2.4 atau Anda juga dapat melanjutkan dari **Repl** yang telah Anda miliki sebelumnya.

Sebagai contoh, Anda ingin mengetahui berapa hasil dari tujuh ditambah delapan. Anda dapat dengan mudah menuliskan baris perintah **`print(7 + 8)`** seperti yang diperlihatkan pada `Kode Program 9`. Ketika Anda mengeklik tombol **"► Run"**, Anda akan melihat **15** sebagai hasil dari penjumlahan kedua bilangan tersebut di kolom **Console** seperti yang diperlihatkan pada `Console 8`.

`Kode Program 9`

```
1  print(7 + 8)
```

`Console 8`

```
15
>
```

Bagaimana jika Anda ingin mengetahui tujuh hari dari sekarang tanggal berapa? Misalkan hari ini adalah tanggal 12 seperti yang diperlihatkan pada `Kode Program 8` untuk solusi Tantangan 1 di bagian sebelumnya. Karena Anda sudah memasukkan bilangan 12 ke dalam sebuah variabel **`tanggal`**, Anda cukup menambahkan variabel tersebut dengan bilangan yang ingin Anda gunakan. Anda dapat langsung mengubah baris kode **`print(tanggal)`** menjadi **`print(tanggal + 7)`** pada baris ketiga seperti yang diperlihatkan pada `Kode Program 10`. Kolom **Console** akan menampilkan bilangan **19** yang berarti tujuh hari dari tanggal 12 adalah tanggal 19 seperti yang diperlihatkan pada `Console 9`.

Kode Program 10

```
1  tanggal = 12
2
3  print(tanggal + 7)
```

Console 9

```
Console
19
>
```

Apa yang terjadi apabila Anda melakukan penjumlahan jenis data numerik yang berbeda? Misalkan Anda menjumlahkan dua variabel yang berisi bilangan ***integer*** dan ***float***? Anda bisa mengetahuinya dengan membuat baris-baris kode seperti yang diperlihatkan pada `Kode Program 11`. Ketika Anda menjalankan kode program tersebut, di kolom **Console** akan ditampilkan hasil seperti yang diperlihatkan pada `Console 10`. Jadi, penjumlahan bilangan ***integer*** dan ***float*** akan menghasilkan bilangan ***float*** di dalam bahasa pemrograman Python.

Kode Program 11

```
1  x = 7
2  y = 8.0
3  z = x + y
4  print(z)
```

Console 10

```
15.0
>
```

Hal lain yang dapat Anda lakukan adalah melakukan operasi perkalian. Misalnya, Anda ingin mengetahui berapa berat badan seseorang apabila dikalikan dengan bilangan dua. Dari contoh informasi sebelumnya, Anda dapat mengetahui bahwa berat badan seseorang adalah 79,20 kg. Anda dapat membuat sebuah baris kode yang memasukkan nilai 79,20 ke sebuah variabel (misalnya **`berat_badan`**) kemudian baris kode lain yang melakukan operasi

perkalian sekaligus untuk menampilkannya di kolom **Console**. Kedua baris kode tersebut diperlihatkan pada `Kode Program 12`. Operasi perkalian dalam bahasa pemrograman Python menggunakan tanda bintang atau asteriks (*). Jangan lupa untuk menggunakan tanda titik alih-alih tanda koma untuk menyatakan bilangan dalam bentuk desimal. Setelah Anda mengeklik tombol **"► Run"**, Anda dapat langsung melihat hasil perkaliannya pada kolom **Console**, yaitu 158.4 (yang berarti 158,4 kg), seperti yang diperlihatkan pada `Console 11.`

Kode Program 12

```
1  berat_badan = 79.20
2
3  print(berat_badan * 2)
```

Console 11

```
158.4
>
```

Jenis operasi lain yang dapat Anda lakukan adalah pembagian. Untuk pembagian, Anda dapat menggunakan tanda garis miring (/) di antara bilangan yang ingin Anda bagi (pembilangnya) dengan bilangan pembaginya (penyebutnya). Misalkan untuk contoh di atas, Anda ingin mengetahui setengah dari berat badan seseorang, maka Anda dapat mengetikkan **`print(berat_badan / 2)`** pada baris kode Anda seperti yang diperlihatkan pada `Kode Program 13.` Pada kolom **Console** dapat terlihat bahwa jawabannya adalah 39.6 (yang artinya 39,6 kg) seperti yang diperlihatkan pada `Console 12`.

Kode Program 13

```
1  berat_badan = 79.20
2
3  print(berat_badan / 2)
```

Console 12

```
39.6
>
```

Sama seperti operasi-operasi yang telah dijelaskan sebelumnya, Anda pun dapat melakukan operasi pengurangan dengan menggunakan tanda minus (-), baik secara langsung maupun di antara variabel-variabel yang telah ditentukan sebelumnya. Itulah jenis-jenis operasi matematika dasar yang dapat dilakukan di dalam bahasa pemrograman Python.

Anda dapat menggunakan tanda baca garis miring ganda (//) untuk mengetahui seberapa banyak sebuah bilangan dapat dibagi dengan bilangan lainnya dan menyatakan hasilnya dalam bentuk bilangan bulat seperti yang diperlihatkan pada `Kode Program 14`. Bilangan 7 dapat dibagi dengan bilangan 2 sebanyak 3 kali sehingga hasil dari **7 // 2** adalah **3** seperti yang diperlihatkan pada `Console 13.`

Kode Program 14

```
1  x = 7 // 2
2
3  print(x)
```

Console 13

```
3
>
```

Terakhir, Anda juga dapat mengetahui sisa dari hasil pembagian dua bilangan atau yang sering disebut dengan *modulo* menggunakan tanda persen (%) pada bahasa pemrograman Python seperti yang diperlihatkan pada `Kode Program 15`. Bilangan 7 apabila dibagi dengan bilangan 2 akan menghasilkan sisa sebesar 1 sehingga hasil dari **`7 % 2`** adalah **`1`** seperti yang diperlihatkan pada `Console 14` sebagai berikut.

Kode Program 15

```
1  x = 7 % 2
2
3  print(x)
```

Console 14

```
1
> 
```

Dari penjelasan tersebut, dapat diketahui bahwa bahasa pemrograman Python sangat mendukung operasi-operasi matematika yang dapat diterapkan untuk jenis data numerik (bilangan), baik itu *integer* (bilangan bulat) maupun *float* (bilangan dengan desimal). Adapun operasi-operasi matematika yang dapat diterapkan adalah penjumlahan, pengurangan, perkalian, pembagian, pembagian bulat, dan *modulo*.

Tiba waktunya untuk memastikan bahwa Anda mengerti mengenai apa yang telah Anda pelajari di bagian ini, yaitu konsep jenis data numerik. Mari kerjakan sebuah tantangan.

Tantangan 2:
Penulis ingin Anda menemukan sesuatu di sekitar Anda yang dapat direpresentasikan dalam sebuah bilangan ***integer*** dan ***float***. Kemudian masukkan kedua bilangan tersebut ke dalam variabel-variabel.

Contoh solusi Tantangan 2:

Kode Program 16

```
1  umur = 39
2
3  tinggi_badan = 173.7
```

C. *String*—Karakter dan Teks

Sekarang Anda sudah mempelajari tentang dua jenis data numerik berbeda, yaitu ***integer*** dan ***float***. Anda akan mempelajari sebuah jenis data baru yang dinamakan ***string***. ***String*** merupakan sebuah cara untuk merepresentasikan teks (satu karakter atau lebih) di dalam bahasa pemrograman Python.

Mari membuat sebuah variabel yang menyimpan jenis data ***string*** yang merepresentasikan warna favorit Anda. Misalnya, warna favorit Anda adalah merah. Anda dapat memberi nama variabel tersebut dengan nama **`warna_favorit`** dan memberi nilai dari variabel tersebut dengan **`merah`**. Anda juga ingin menampilkan isi dari variabel tersebut di kolom **Console** dengan menggunakan perintah **`print(warna_favorit)`** seperti yang diperlihatkan pada `Kode Program 17`.

`Kode Program 17`

```
1  warna_favorit = merah
2
3  print(warna_favorit)
```

Namun, ketika Anda mengeklik tombol **"► Run"**, kolom **Console** memperlihatkan bahwa terdapat kesalahan di dalam baris-baris kode yang Anda tulis seperti yang diperlihatkan pada `Console 15`. Biasanya ketika terdapat kesalahan di dalam baris-baris kode, kolom **Console** akan menampilkan teks berwarna merah. Kesalahannya menyebutkan bahwa nilai **`merah`** tidak didefinisikan. Permasalahannya adalah di dalam bahasa pemrograman Python, kapan pun Anda ingin merepresentasikan ***string*** atau teks, Anda harus menyertakan tanda petik sebelum dan sesudah ***string*** atau teks tersebut.

Console 15

```
Traceback (most recent call last):
 File "main.py", line 1, in <module>
 warna_favorit = merah
NameError: name 'merah' is not defined
>
```

Untuk itu, mari berikan tanda petik tunggal sebelum dan sesudah kata **merah** yang telah Anda ketikkan sebelumnya sehingga baris pertama kode Anda menjadi **warna_favorit = 'merah'** seperti yang diperlihatkan pada Kode Program 18. Ketika Anda mengeklik tombol "▶ **Run**" kembali, kolom **Console** tidak akan memperlihatkan pesan kesalahan, tetapi isi dari variabel **warna_favorit** yang telah Anda tentukan, yaitu **merah** seperti yang diperlihatkan pada Console 16. Anda akan menyadari bahwa nilai **merah** yang ditampilkan pada kolom **Console** tidak memiliki tanda petik tunggal sebagaimana ketika Anda menuliskannya di dalam baris kode Anda. Tanda petik tunggal yang Anda tuliskan berfungsi untuk memberi tahu bahasa pemrograman Python bahwa nilai yang Anda masukkan adalah memiliki jenis ***string*** atau teks.

Kode Program 18

```
1  warna_favorit = 'merah'
2
3  print(warna_favorit)
```

Console 16

```
merah
>
```

Hal lain yang perlu Anda ketahui mengenai bahasa pemrograman Python adalah bahwa untuk merepresentasikan nilai ***string*** atau teks, Anda juga dapat menggunakan tanda petik ganda. Jadi, apabila tanda petik tunggal yang Anda ketikkan sebelumnya diganti dengan tanda petik ganda, kolom **Console** akan tetap menampilkan hasil yang sama seperti yang diperlihatkan pada Console 16.

Serupa seperti penjelasan pada bagian sebelumnya, hal yang tidak diperkenankan di dalam baris kode Anda ketika ingin merepresentasikan nilai ***string*** atau teks di dalam sebuah variabel adalah menggunakan tanda petik campuran. Artinya, Anda menggunakan tanda petik ganda sebelum nilai ***string*** atau teks dan menggunakan tanda petik tunggal setelah nilai tersebut, atau sebaliknya. Contoh baris kode yang salah adalah seperti **`warna_favorit = “merah’`** atau **`warna_favorit = ‘merah”.`** Anda harus menggunakan tanda petik secara konsisten sebelum dan sesudah nilai ***string*** atau teks yang ingin Anda representasikan.

Selain itu, serupa dengan perintah **`print()`** yang telah dibahas pada bagian sebelumnya, ketika Anda ingin memasukkan jenis data ***string*** atau teks ke dalam sebuah variabel, Anda juga dimungkinkan untuk menggunakan tanda petik tunggal atau ganda tiga kali di depan dan di belakang teks yang ingin Anda masukkan tersebut. Fungsinya adalah apabila Anda ingin menyajikan sebuah ***string*** atau teks yang dituliskan lebih dari satu baris seperti yang diperlihatkan pada `Kode Program 19`. Pada saat variabel tersebut ditampilkan di kolom **Console**, sama seperti teks yang dimasukkan di dalam variabelnya, akan ditampilkan dalam lebih dari satu baris juga seperti yang diperlihatkan pada `Console 17`.

Kode Program 19

```
1  warna_favorit = '''
2  merah
3  kuning
4  hijau '''
5
6  print(warna_favorit)
```

Console 17

```
merah
kuning
hijau
>
```

Bagaimana apabila di dalam ***string*** atau teks yang ingin Anda masukkan di dalam sebuah variabel memiliki tanda petik tunggal atau tanda petik ganda? Anda harus menggunakan tanda petik yang berbeda dengan tanda petik yang Anda gunakan di dalam nilai ***string*** atau teks Anda. Apabila di dalam nilai ***string*** atau teks Anda terdapat tanda petik tunggal, gunakanlah tanda petik ganda sebelum dan sesudah nilai tersebut, dan sebaliknya.

Tanda petik tunggal biasanya digunakan untuk menyatakan kepemilikan di dalam bahasa inggris. Misalnya teks **Ratih's Bookstore** yang artinya toko buku milik Ratih. Apabila Anda ingin merepresentasikan teks tersebut di dalam variabel, Anda perlu untuk mengapit teks tersebut dengan tanda petik ganda karena di dalam teks tersebut terdapat tanda petik tunggal. Sebagai contoh, Anda ingin memasukkan teks tersebut ke dalam sebuah variabel dengan nama **`nama_toko`** maka baris kode yang harus Anda tulis adalah **`nama_toko = "Ratih's Bookstore"`**, seperti yang diperlihatkan pada `Kode Program 20`.

Kode Program 20

```
1  nama_toko = "Ratih's Bookstore"
2
3  print(nama_toko)
```

Console 18

```
Ratih's Bookstore
>
```

Apabila Anda melakukan kesalahan dengan tetap memasukkan tanda petik tunggal untuk mengapit ***string*** atau teks yang mengandung tanda petik tunggal seperti yang diperlihatkan pada `Kode Program 21`, setelah Anda mengeklik tombol **"▶ Run"**, kolom **Console** akan menampilkan pesan kesalahan bahwa terdapat sintaks yang tidak tepat

di dalam baris-baris kode yang Anda tulis sebagaimana diperlihatkan pada `Console 19`.

Kode Program 21

```
1 nama_toko = 'Ratih's Bookstore'
2
3 print(nama_toko)
```

Console 19

```
File "main.py", line 1
 Nama_toko = 'Ratih's Bookstore'
   ^
SyntaxError: invalid syntax
>
```

Sebenarnya Anda tidak perlu selalu mengeklik tombol "**► Run**" untuk mengetahui bahwa terdapat kesalahan di dalam baris-baris kode yang Anda tulis. Ketika terjadi sebuah kesalahan, biasanya Replit akan menampilkan sebuah garis bergelombang berwarna merah pada bagian-bagian di dalam baris-baris kode Anda yang belum tepat sesuai dengan kaidah penulisan dalam bahasa pemrograman Python.

Bagaimana apabila ***string*** atau teks yang ingin Anda masukkan ke dalam variabel memiliki baik tanda petik tunggal maupun tanda petik ganda di dalamnya? Misalnya frasa yang ingin Anda masukkan adalah **Berbelanja di Ratih's Bookstore akan meningkatkan "popularitas" Anda**. Apabila kita menggunakan tanda petik tunggal atau ganda untuk mengapit frasa tersebut, baris kode yang Anda ketikkan akan menjadi tidak valid. Bagaimana cara Anda menangani situasi seperti ini? Anda dapat menggunakan tanda *backslash* (\) sebelum tanda petik tunggal dan tanda petik ganda yang Anda gunakan di dalam teks yang ingin Anda masukkan ke dalam variabel. Misalkan Anda ingin memasukkan kalimat di atas ke dalam sebuah variabel dengan nama **`slogan`**, baris kode yang dapat Anda ketikkan adalah **`slogan = 'Berbelanja di Ratih\'s Bookstore akan meningkatkan`**

“popularitas” Anda.’ seperti yang diperlihatkan pada Kode Program 22.

Kode Program 22

```
1  slogan = 'Berbelanja di Ratih\'s Bookstore akan
   meningkatkan "popularitas" Anda. '
2
3  print(slogan)
```

Console 20

```
Berbelanja di Ratih’s Bookstore akan meningkatkan
"popularitas" Anda.
>
```

Pada baris kode pertama di Kode Program 22 Anda menggunakan *backslash* di depan tanda petik tunggal yang ada di dalam teks karena Anda menggunakan tanda petik tunggal untuk mengapit teks tersebut. Apabila Anda menggunakan tanda petik ganda untuk mengapit teks tersebut, yang perlu diberi tanda *backslash* adalah tanda petik ganda yang ada di dalam teks seperti yang diperlihatkan pada Kode Program 23.

Kode Program 23

```
1  slogan = "Berbelanja di Ratih’s Bookstore akan
   meningkatkan \”popularitas\” Anda."
2
3  print(slogan)
```

Console 21

```
Berbelanja di Ratih’s Bookstore akan meningkatkan
"popularitas" Anda.
>
```

Pada bagian sebelumnya Anda telah mempelajari operasi matematika pada jenis data numerik. Apa yang terjadi apabila operasi tersebut diterapkan pada data berjenis ***string*** atau teks? Misalnya Anda

mencoba menjumlahkan dua variabel yang memiliki nilai ***string*** atau teks di dalamnya. Variabel pertama berisi **"Hello"** dan variabel kedua berisi **"World!"** seperti yang diperlihatkan pada Kode Program 24. Karena itu, ketika Anda menjalankan baris-baris kode tersebut, kolom **Console** akan menampilkan keluaran seperti yang diperlihatkan pada Console 22. Dari contoh tersebut dapat disimpulkan bahwa operasi penjumlahan yang dilakukan pada dua variabel bernilai ***string*** atau teks akan menjadikan sebuah ***string*** atau teks yang merupakan penggabungan dari dua ***string*** atau teks awal.

Kode Program 24

```
1  x = "Hello"
2  y = "World!"
3  z = x + y
4
5  print(z)
```

Console 22

```
HelloWorld!
>
```

Bagaimana dengan operasi pengurangan? Anda dapat mencobanya dengan memodifikasi baris-baris kode pada Kode Program 24 di atas menjadi seperti yang diperlihatkan pada Kode Program 25. Ketika Anda menjalankan baris-baris kode tersebut, sebuah pesan kesalahan akan ditampilkan di kolom **Console** seperti yang diperlihatkan pada Console 23. Hal tersebut memperlihatkan bahwa operasi pengurangan tidak berlaku pada variabel-variabel yang memiliki nilai ***string*** atau teks. Hasil serupa akan ditampilkan apabila Anda melakukan operasi perkalian, pembagian, dan *modulo* pada variabel-variabel yang memiliki nilai ***string*** atau teks.

Kode Program 25

main.py

```
1  x = "Hello"
2  y = "World!"
3  z = x - y
4
5  print(z)
```

Console 23

```
Traceback (most recent call last):
 File "main.py", line 3, in <module>
 Z = x - y
TypeError: unsupported operand type(s) for -: 'str' and
'str'
>
```

Sekarang Anda telah mempelajari tiga jenis nilai untuk dimasukkan ke dalam variabel, yaitu ***integer***, ***float***, dan ***string***. Untuk memastikan Anda memahami ***string*** dengan baik, kerjakan tantangan di bawah ini.

Tantangan 3:
Penulis ingin Anda menuliskan judul buku favorit Anda ke dalam sebuah variabel yang dinamakan **buku**. Kemudian, tampilkan isi dari variabel **buku** tersebut di kolom **Console**. Sekali lagi, jangan lupa menggunakan tanda petik yang sesuai dengan judul buku yang ingin Anda masukkan ke dalam variabel tersebut.

Contoh solusi Tantangan 3:

Kode Program 26

```
1  buku = "Langkah Awal Menguasai Bahasa Pemrograman
   Python"
2
3  print(buku)
```

Console 24

```
Langkah Awal Menguasai Bahasa Pemrograman Python
>
```

D. *String-f*

Pada bagian ini, Anda akan mempelajari bagaimana Anda dapat menempatkan nilai yang merupakan gabungan dari variabel dan *string* di dalam baris kode **`print()`**. Misalkan Anda ingin membuat baris-baris kode yang menghasilkan informasi tanggal dan prakiraan cuaca pada tanggal tersebut, salah satu contoh yang dapat Anda tuliskan adalah **`print('Kondisi cuaca pada tanggal 24 April 2022 adalah cerah')`** seperti yang diperlihatkan pada `Kode Program 27`. Ketika kode tersebut dijalankan, yang ditampilkan pada kolom **Console** adalah teks sesuai dengan yang diapit kedua tanda petik tunggal di dalam **`print()`** tersebut seperti yang diperlihatkan pada `Console 25`.

Kode Program 27

```
1  print('Kondisi cuaca pada tanggal 24 April 2022 adalah
   cerah.')
```

Console 25

```
Kondisi cuaca pada tanggal 21 April 2022 adalah cerah.
>
```

Bagaimana jika Anda ingin memperbarui informasi yang ditampilkan di kolom **Console** tersebut? Tentunya Anda harus mengubah baris kode Anda yang terdapat di antara kedua tanda petik di dalam **`print()`** tersebut secara menyeluruh. Sementara itu, apabila Anda perhatikan, informasi yang ditampilkan tersebut dapat dipecah-pecah lagi menjadi bagian-bagian yang lebih kecil, terutama bagian-bagian yang dapat berubah-ubah. Informasi tersebut memuat tanggal, bulan, tahun, dan kondisi cuaca pada tanggal tersebut. Bagian-bagian yang lebih kecil inilah yang dapat Anda wakilkan dengan menggunakan variabel untuk memudahkan perubahan baris kode ketika Anda ingin mengubah informasi yang ingin ditampilkan di

kolom **Console** sehingga baris kode pada Kode Program 27 dapat Anda ubah menjadi seperti yang diperlihatkan pada Kode Program 28. Apabila kode program tersebut dijalankan, kolom **Console** akan menampilkan informasi yang sama dengan yang ditampilkan pada Console 25.

Kode Program 28

```
1  tanggal = 24
2  bulan = 'April'
3  tahun = 2022
4  kondisi_cuaca = 'cerah'
5
6  print('Kondisi cuaca pada tanggal 24 April 2022 adalah
   cerah.')
```

Ketika Anda mengubah nilai-nilai dari variabel-variabel **tanggal**, **bulan**, **tahun**, atau **kondisi_cuaca** seperti yang diperlihatkan pada Kode Program 29, tampilan di kolom **Console** masih memperlihatkan informasi yang tetap, tidak berubah mengikuti perubahan yang Anda buat di dalam variabel-variabel tersebut. Bagaimana cara untuk menghubungkan antara nilai-nilai di dalam variabel-variabel tersebut dengan **print()** yang saat ini hanya bisa menampilkan nilai yang konstan?

Kode Program 29

```
1  tanggal = 12
2  bulan = 'Maret'
3  tahun = 2022
4  kondisi_cuaca = 'hujan'
5
6  print('Kondisi cuaca pada tanggal 24 April 2022 adalah
   cerah.')
```

Jawabannya adalah dengan menggunakan apa yang disebut ***string-f*** di dalam bahasa pemrograman Python. Jadi, Anda tinggal mengetikkan huruf ***f*** tepat sebelum tanda petik (baik tunggal maupun ganda) yang Anda gunakan untuk mengawali ***string*** atau teks Anda seperti yang diperlihatkan pada `Kode Program 30`.

Kode Program 30

```
1  tanggal = 12
2  bulan = 'Maret'
3  tahun = 2022
4  kondisi_cuaca = 'hujan'
5
6  print(f'Kondisi cuaca pada tanggal 24 April 2022
   adalah cerah.')
```

Huruf ***f*** ini menandakan bahwa Anda ingin menempatkan variabel di dalam ***string*** atau teks yang ingin ditampilkan di kolom **Console** sehingga satu bagian atau lebih dari ***string*** atau teks tersebut dapat digantikan oleh nilai dari variabel yang ditempatkan tersebut. Bagaimana cara menempatkan variabel di dalam ***string*** atau teks tersebut? Anda dapat melakukannya dengan menggunakan tanda kurung kurawal ({ dan }) untuk mengapit variabel yang ingin Anda tempatkan di dalam ***string*** atau teks. Misalknya angka **`24`** di dalam teks yang ditampilkan pada baris-baris kode di atas ingin Anda ubah dengan variabel **`tanggal`**, Anda cukup menggantikan angka **`24`** tersebut dengan **`{tanggal}`** seperti yang diperlihatkan pada `Kode Program 31`. Ketika baris-baris kode tersebut dijalankan, informasi yang ditampilkan menyesuaikan dengan nilai dari variabel **`tanggal`** yang telah dimasukkan ke dalam ***string*** atau teks yang ditampilkan tersebut. Selanjutnya, Anda dapat mengubah baris-baris kode Anda seperti yang diperlihatkan pada `Kode Program 32`.

Kode Program 31

```
1 tanggal = 12
2 bulan = 'Maret'
3 tahun = 2022
4 kondisi_cuaca = 'hujan'
5
6 print(f'Kondisi cuaca pada tanggal {tanggal} April 2022
  adalah cerah.')
```

Kode Program 32

```
1 tanggal = 12
2 bulan = 'Maret'
3 tahun = 2022
4 kondisi_cuaca = 'hujan'
5
6 print(f'Kondisi cuaca pada tanggal {tanggal} {bulan}
  {tahun} adalah {kondisi_cuaca}.')
```

Perubahan tersebut bertujuan untuk menampilkan nilai-nilai dari seluruh variabel yang telah Anda tentukan. Meskipun pada contoh di atas penulis hanya memperlihatkan nilai-nilai variabel yang berjenis ***integer*** dan ***string*** atau teks, nilai dari variabel apa pun dapat dimasukkan ke dalam teks yang ingin ditampilkan dengan menggunakan **`print()`** tersebut. Kini, ketika kode program tersebut dijalankan, tampilan di kolom **Console** akan menyesuaikan dengan nilai-nilai dari variabel-variabel yang Anda tuliskan pada baris ke-1 sampai dengan baris ke-4 `Kode Program 32` seperti yang diperlihatkan pada `Console 26`.

Console 26

```
Kondisi cuaca pada tanggal 12 Maret 2022 adalah hujan.
>
```

Tantangan 4:
Untuk memastikan Anda memahami bagian ini, penulis ingin Anda memodifikasi baris-baris kode yang ditampilkan pada Kode Program 32 dengan menyertakan sebuah variabel baru yang

dinamakan hari. Anda dapat mengisi variabel tersebut dengan nilai sesuai dengan tanggal yang ingin Anda tampilkan di dalam teks. Kemudian, Anda masukkan variabel tersebut ke dalam teks sehingga apabila Anda ingin mengubah teks yang ditampilkan, Anda cukup mengubah nilai di dalam variabel yang relevan saja.

Contoh solusi Tantangan 4:

Kode Program 33

```
1  hari = 'Sabtu'
2  tanggal = 12
3  bulan = 'Maret'
4  tahun = 2022
5  kondisi_cuaca = 'hujan'
6
7  print(f'Kondisi cuaca pada hari {hari}, tanggal
   {tanggal} {bulan} {tahun} adalah {kondisi_cuaca}.')
```

Console 27

```
Kondisi cuaca pada hari Sabtu, tanggal 12 Maret 2022
adalah hujan.
>
```

Sebelum mengakhiri bagian ini, penulis ingin memberikan sedikit tips bagi Anda dalam membuat nama variabel yang terdiri lebih dari satu kata, misalnya **`kondisi_cuaca`** pada contoh di atas. Dalam bahasa pemrograman Python, sebagian besar *programmer* profesional membuat variabel yang memiliki lebih dari satu kata dengan menggunakan tanda garis bawah sebagai pemisah antarkatanya sehingga, untuk contoh di atas, alih-alih menggunakan nama variabel **`kondisicuaca`** atau **`KondisiCuaca`** atau **`kondisiCuaca`**, penulis menggunakan nama **`kondisi_cuaca`**. Hal ini bertujuan untuk memudahkan dalam proses pembacaan baris-baris kode program, baik oleh orang lain yang menjadi mitra *programmer* maupun oleh *programmer* itu sendiri ketika sudah sekian lama tidak membaca baris-baris kode yang telah ia tulis.

E. *Boolean* dan *If Statement*

Anda telah mengetahui tiga jenis data sekarang, yaitu ***integer***, ***float***, dan ***string***. Anda sekarang akan mempelajari jenis yang keempat, yang dinamakan ***boolean***. ***Boolean*** adalah cara untuk merepresentasikan situasi "ya atau tidak", "benar atau salah". Apabila Anda pernah mendengar istilah "biner", yang merepresentasikan segala sesuatu dalam bilangan satu dan nol, itu termasuk ke dalam jenis data ***boolean***.

Anda mungkin berpikir jenis data tersebut berbeda dengan jenis data lainnya dan ingin mengetahui jenis data tersebut sesuai untuk situasi apa saja. Hal yang paling umum adalah jenis data tersebut digunakan untuk menyatakan apakah sesuatu itu aktif atau tidak. Sebagai contoh, apabila Anda ingin membuat sebuah variabel yang merepresentasikan apakah kondisi listrik di ruangan Anda sedang menyala atau tidak, Anda dapat membuat variabel bernama **`listrik`** dan memberi nilai **`menyala`** atau **`padam`**. Namun, nilai ***boolean*** yang valid di dalam bahasa pemrograman Python hanyalah ***True*** atau ***False*** sehingga apabila Anda ingin menyatakan bahwa listrik di ruangan Anda sedang dalam keadaan menyala, Anda dapat menuliskannya dalam baris kode **`listrik = True`**. Sementara itu, apabila Anda ingin menyatakan bahwa listrik di ruangan Anda sedang dalam keadaan padam (mungkin dikarenakan sedang adanya pemadaman bergilir atau Anda lupa mengisi ulang token listrik), Anda dapat menuliskannya dalam baris kode **`listrik = False`**. Kedua contoh tersebut dapat dinyatakan dalam baris-baris kode seperti yang diperlihatkan pada `Kode Program 34`.

`Kode Program 34`

```
1  listrik = True
2  listrik = False
```

Perhatikan bahwa Anda tidak menuliskan nilai-nilai ***boolean*** tersebut dengan diapit oleh tanda kutip (baik tunggal maupun ganda). Hal itu karena apabila Anda mengapit nilai-nilai tersebut dengan tanda kutip maka nilai-nilai tersebut akan menjadi memiliki jenis ***string***. Jadi, pastikan ketika Anda memberi nilai ***boolean*** ke dalam sebuah variabel, cara penulisannya adalah seperti yang dijelaskan sebelumnya.

Integer memungkinkan Anda untuk memasukkan bilangan bulat apa pun ke dalam sebuah variabel. Bisa sangat kecil (negatif). Bisa juga sangat besar (positif). ***Float*** memungkinkan Anda untuk memasukkan bilangan dengan desimal sesuai dengan ketelitian yang Anda butuhkan ke dalam sebuah variabel. ***String*** memungkinkan Anda untuk memasukkan karakter atau huruf apa pun ke dalam sebuah variabel. Namun, hanya terdapat dua pilihan untuk jenis ***boolean***, yaitu ***True*** dan ***False***.

Pada awalnya mungkin Anda akan mengira bahwa jenis nilai ini kurang berguna dibandingkan jenis lainnya. Namun, setelah Anda mempelajari tentang ***if statement***, Anda akan melihat bahwa jenis nilai ini sangat berguna dalam membuat program Anda menjadi lebih "hidup" karena penggunaan ***boolean*** dalam ***if statement*** akan menjadikan program Anda mampu untuk membuat keputusan berdasarkan data yang tersedia.

Sebagai contoh, Anda ingin membuat baris-baris kode yang akan menghasilkan keluaran atau tampilan di dalam kolom **Console** berdasarkan status dari listrik di ruangan Anda. Ketika listrik di ruangan Anda menyala, Anda ingin menampilkan informasi di dalam kolom **Console** semisal "Listrik di ruangan Anda menyala dengan normal." Anda dapat menuliskannya dalam baris-baris kode seperti yang diperlihatkan pada `Kode Program 35`. Ketika Anda menjalankannya, kolom **Console** akan menampilkan seperti yang diperlihatkan pada `Console 28`.

Kode Program 35

```
1  listrik = True
2
3  if listrik:
4    print('Listrik di ruangan Anda menyala dengan
   normal.')
```

Console 28

```
Listrik di ruangan Anda menyala dengan normal.
>
```

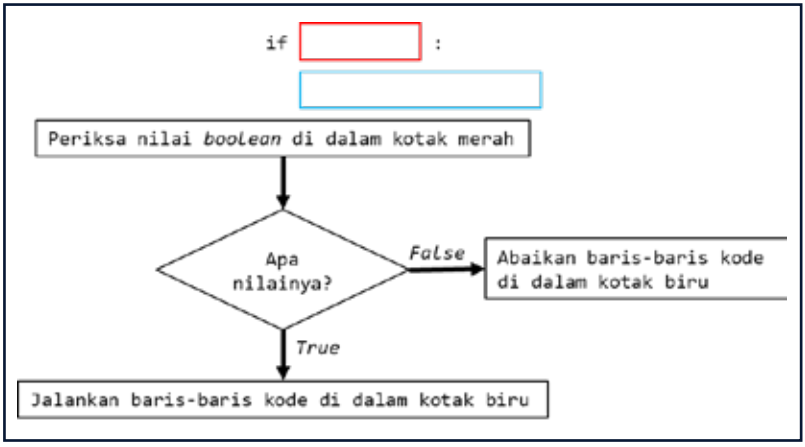
Contoh baris-baris kode di atas memiliki arti bahwa kondisi listrik di ruangan Anda sedang menyala. Hal ini diperlihatkan oleh baris kode **`listrik = True`**. Kemudian, pada baris kode ketiga diperlihatkan sebuah ***if statement*** yang memiliki arti bahwa, "Jika nilai variabel **`listrik`** adalah ***True*** maka jalankan baris kode yang ada setelah tanda titik dua di bawahnya (baris kode keempat pada `Kode Program 35`), yaitu **`print('Listrik di ruangan Anda menyala dengan normal.')`**." Oleh karena itu, pada contoh tersebut teks "Listrik di ruangan Anda menyala dengan normal" ditampilkan di kolom **Console** pada saat kode tersebut dijalankan.

Bagaimana jika listrik di ruangan Anda padam? Anda tinggal mengubah nilai variabel **`listrik`** menjadi **`False`** seperti yang diperlihatkan pada `Kode Program 36`. ***If statement*** pada baris ketiga dan keempat kode tersebut tidak mengalami perubahan. Pada saat Anda menjalankan kode tersebut, pada kolom **Console** tidak ditampilkan informasi apa pun. Hal ini disebabkan nilai variabel **`listrik`** adalah **`False`** sehingga kode pada baris ketiga tidak dijalankan lebih lanjut dan kode pada baris keempat pun akan diabaikan oleh program. Jadi, baris-baris kode setelah tanda titik dua dari sebuah ***if statement*** hanya dijalankan apabila setelah ***if statement*** tersebut dievaluasi memiliki nilai ***boolean True***.

Kode Program 36

```
1 listrik = False
2
3 if listrik:
4   print('Listrik di ruangan Anda menyala dengan
  normal.')
```

Secara umum, proses eksekusi program di dalam ***if statement*** adalah seperti yang diperlihatkan pada Gambar 2.5. Pada gambar tersebut dapat terlihat bahwa ***if statement*** sangat berkaitan erat dengan jenis data ***boolean*** dalam menentukan langkah selanjutnya yang harus dilakukan oleh program. Dengan menggabungkan kedua konsep tersebut, dapat terlihat bahwa kini program Anda telah mampu untuk membuat sebuah keputusan. Program Anda seakan-akan bertanya "Apakah listrik di ruangan Anda menyala atau tidak?" Apabila menyala, kode akan melanjutkan menjalankan baris kode keempat. Apabila padam, kode tidak akan menjalankan baris kode keempat tersebut.



Gambar 2.5 Diagram Alur *If Statement*

Hal lain yang harus Anda perhatikan adalah penempatan dari baris kode setelah tanda titik dua. Terdapat beberapa spasi sebelum baris kode keempat ditulis. Pada contoh di atas terdapat empat karakter spasi sehingga baris kode tersebut terlihat seolah-olah menjorok ke arah kanan. Anda juga dapat menggunakan tombol **Tab** pada papan ketik (*keyboard*) Anda untuk membuat spasi tersebut. Spasi ini sangat penting di dalam bahasa pemrograman Python. Spasi ini menandakan bahwa baris kode yang menjorok tersebut merupakan bagian dari ***if statement*** atau fungsi-fungsi lain di atasnya. Apabila baris kode keempat tidak diberi spasi atau tab, baris tersebut tidak akan dianggap sebagai bagian dari ***if statement*** atau fungsi-fungsi lain di atasnya.

Sebagai contoh, Anda tambahkan baris kode baru di bawah baris keempat kode tersebut seperti yang diperlihatkan pada `Kode Program 37`. Karena nilai dari variabel **`listrik`** adalah **`True`**, baris-baris keempat dan kelima kode akan dijalankan. Hal ini terlihat kedua ***string*** yang Anda masukkan di baris keempat dan kelima akan ditampilkan di kolom **Console** ketika kode program dijalankan seperti diperlihatkan pada `Console 29.`

Kode Program 37

```
1  listrik = True
2
3  if listrik:
4    print('Listrik di ruangan Anda menyala dengan normal.')
5    print(‘Anda dapat menggunakan komputer Anda.’)
```

Console 29

```
Listrik di ruangan Anda menyala dengan normal.
Anda dapat menggunakan komputer Anda.
>
```

Sekarang Anda coba hilangkan spasi di depan baris kelima kode seperti yang diperlihatkan pada `Kode Program 38`. Ketika Anda menjalankan kode program tersebut, tidak ada perubahan signifikan

yang terlihat dan kolom **Console** akan memperlihatkan tampilan sama seperti sebelumnya. Namun, ketika Anda mengubah nilai dari variabel **`listrik`** menjadi **`False`** yang menandakan bahwa listrik di ruangan Anda padam sebagaimana diperlihatkan pada `Kode Program 39`, baris kode kelima akan tetap ditampilkan di kolom **Console** ketika kode program dijalankan seperti yang diperlihatkan pada `Console 30`.

Kode Program 38

```
1  listrik = True
2
3  if listrik:
4    print('Listrik di ruangan Anda menyala dengan normal. ')
5    print('Anda dapat menggunakan komputer Anda.')
```

Kode Program 39

```
1  listrik = False
2
3  if listrik:
4   print('Listrik di ruangan Anda menyala dengan normal.')
5  print('Anda dapat menggunakan komputer Anda.')
```

Console 30

```
Anda dapat menggunakan komputer Anda.
>
```

Sementara itu, baris kode keempat tidak dijalankan karena nilai dari ***if statement*** pada baris ketiga adalah ***False***. Hal ini memperlihatkan bahwa baris kode kelima tersebut sudah bukan lagi bagian dari ***if statement*** di atasnya sehingga terlepas dari variabel **`listrik`** bernilai **`True`** atau **`False`**, baris kode tersebut tetap dijalankan. Oleh karena itu, Anda harus berhati-hati pada saat menuliskan baris-baris kode yang berhubungan dengan ***if statement*** di dalam bahasa pemrograman Python karena penempatan kode menjadi hal yang menentukan.

Bagaimana jika baris-baris kode yang berada di bawah ***if statement*** tidak memiliki jarak yang sama ketika ditulis? Anda dapat mencobanya dengan menuliskan baris-baris kode seperti yang diperlihatkan pada `Kode Program 40`. Pada baris-baris kode tersebut dapat terlihat bahwa baris keempat dan kelima sama-sama diberi spasi, tetapi dengan jumlah spasi yang tidak sama, padahal kedua baris tersebut ditujukan sebagai bagian dari ***if statement*** di atasnya. Ketika Anda menjalankan kode program tersebut, di kolom **Console** akan menampilkan sebuah pesan kesalahan bahwa baris kelima kode program Anda tersebut memiliki kesalahan dalam hal pemberian indentasi seperti yang diperlihatkan pada `Console 31`. Untuk itu, pastikan bahwa indentasi yang Anda gunakan telah sama dan sesuai dengan ***if statement*** atau fungsi-fungsi lain di atasnya.

Kode Program 40

```
1  listrik = False
2
3  if listrik:
4   print('Listrik di ruangan Anda menyala dengan normal.')
5    print('Anda dapat menggunakan komputer Anda.')
```

Console 31

```
Console
File "main.py", line 5
 print('Anda dapat menggunakan komputer Anda.')
 ^
IndentationError: unexpected indent
>
```

Tantangan 5:
Untuk memastikan Anda memahami tentang ***boolean***, penulis ingin Anda mencari sesuatu di sekitar Anda yang dapat direpresentasikan dengan nilai ***boolean*** dan membuat variabel untuk menyimpan nilai tersebut.

Contoh solusi Tantangan 5:

Kode Program 41

```
1  kipas_angin = True
2
3  if kipas_angin:
4   print('Anda memiliki sirkulasi udara yang baik.')
```

F. *Else Statement* dan Perbandingan

Mari kita lanjutkan pembelajaran mengenai semua hal yang dapat dilakukan oleh ***if statement***. Pada bagian sebelumnya Anda telah memiliki contoh mengenai jenis data ***boolean***. Apabila listrik di ruangan menyala, Anda memberikan nilai ***True*** dan apabila listrik di ruangan padam, Anda memberikan nilai ***False***. Anda menggunakan ***if statement*** untuk menentukan apa yang harus dilakukan oleh kode program Anda jika nilai dari variabel **`listrik`** tersebut **`True`** atau **`False`**. Apabila nilai dari variabel **`listrik`** adalah **`True`** yang berarti listrik di ruangan Anda menyala, kode akan menampilkan pesan di kolom **Console** bahwa "Listrik di ruangan Anda menyala dengan normal." dan kolom **Console** tidak akan menampilkan pesan apa pun ketika nilai variable **`listrik`** adalah **`False`**.

Namun, Anda ingin membuat kode program Anda menjadi lebih baik lagi. Misalkan, apabila variabel **`listrik`** bernilai **`False`**, Anda juga ingin menampilkan sesuatu di kolom **Console** untuk menginformasikan kepada pengguna program apa yang sedang terjadi karena pada umumnya pengguna program tidak akan membaca baris-baris kode yang membentuk sebuah program. Mereka hanya akan berinteraksi dengan program yang sudah dijalankan.

Anda dapat melakukannya dalam bahasa pemrograman Python. Perhatikan baris-baris kode yang diperlihatkan pada `Kode Program 42`. Baris-baris kode tersebut merupakan modifikasi dari baris-baris

kode yang diperlihatkan pada Kode Program 36 sebelumnya dengan penambahan dua baris kode baru di bawah baris-baris kode yang sudah ada. Jangan lupa untuk memperhatikan bagian-bagian mana saja yang harus diberi spasi tambahan dan bagian mana saja yang tidak. Pada saat variabel **listrik** bernilai ***True***, baris keempat dari kode tersebut akan dijalankan dan teks "Listrik di ruangan Anda menyala dengan normal." Kode tersebut akan ditampilkan pada kolom **Console** sebagaimana penjelasan pada bagian sebelumnya.

Kode Program 42

```
1  listrik = True
2
3  if listrik:
4   print('Listrik di ruangan Anda menyala dengan normal.')
5  else:
6   print('Anda mengalami pemadaman listrik.')
```

Sekarang ubahlah nilai dari variabel **`listrik`** menjadi ***False*** seperti yang diperlihatkan pada Kode Program 43. Pada saat Anda menjalankan kode tersebut, baris keempat tidak akan dijalankan oleh program. Sebagai gantinya, program akan menjalankan baris kode keenam dan teks "Anda mengalami pemadaman listrik." akan ditampilkan pada kolom **Console** seperti yang diperlihatkan pada Console 32.

Kode Program 43

```
1  listrik = False
2
3  if listrik:
4   print('Listrik di ruangan Anda menyala dengan normal.')
5  else:
6   print('Anda mengalami pemadaman listrik.')
```

Console 32

```
Anda mengalami pemadaman listrik.
>
```

Jadi, ***else statement*** berfungsi sebagai alternatif bagian yang harus dijalankan oleh program pada saat kondisi pada ***if statement*** di atasnya bernilai ***False***. Dengan memadukan ***if statement*** dan ***else statement***, kode program Anda kini dapat menjalankan bagian-bagian dari baris-baris kode sesuai dengan nilai yang Anda berikan pada variabel yang dievaluasi di bagian ***if statement***.

Mari kita kembangkan konsep ini lebih jauh lagi ke perbandingan di dalam kode program. Bayangkan situasi sebagai berikut. Di Indonesia, menurut peraturan perundang-undangan yang berlaku saat ini, seseorang dinyatakan telah dewasa apabila telah berumur lebih atau sama dengan 18 tahun atau telah menikah. Anda ingin membuat sebuah kode program yang mengklasifikasikan seseorang telah dewasa atau tidak berdasarkan umurnya (untuk saat ini kita abaikan status telah menikah atau belum).

Hal pertama yang Anda lakukan tentunya adalah membuat sebuah variabel untuk dibandingkan dengan suatu nilai tertentu untuk menghasilkan nilai ***True*** atau ***False*** di dalam ***if statement*** dan ***else statement*** sehingga kode Anda dapat menjalankan bagian-bagian baris kode sesuai dengan nilai tersebut. Misalkan variabel yang Anda buat adalah **`umur`** dan Anda memberi nilai variabel tersebut sebesar 30 yang merepresentasikan umur seseorang 30 tahun. Baris pertama kode Anda akan seperti yang diperlihatkan pada `Kode Program 44`.

`Kode Program 44`

```
1  umur = 30
```

Langkah selanjutnya adalah Anda membuat sebuah ***if statement*** yang disesuaikan dengan contoh kasus tersebut, yaitu bernilai ***True*** apabila variabel **umur** memiliki nilai 18 atau lebih agar seseorang dapat dinyatakan telah dewasa. Anda dapat menggunakan tanda perbandingan "lebih besar atau sama dengan" (>=) sebagaimana biasa

digunakan pada pernyataan persamaan matematika. Baris-baris kode Anda akan seperti yang diperlihatkan pada Kode Program 45.

Kode Program 45

```
1  umur = 30
2
3  if umur >= 18:
```

Kemudian, Anda membuat sebuah baris kode yang dijalankan apabila nilai dari variabel **umur** memenuhi kriteria yang ditentukan di dalam ***if statement***, yaitu "lebih besar atau sama dengan 18 tahun." Pada bagian ini, Anda dapat menampilkan informasi di dalam kolom **Console** semisal "Anda masuk ke dalam kategori dewasa." seperti yang diperlihatkan pada Kode Program 46. Untuk memastikan bahwa baris-baris kode tersebut sudah benar, Anda dapat menjalankannya sehingga terlihat informasi yang ditampilkan di kolom **Console** seperti diperlihatkan pada Console 33.

Kode Program 46

```
1  umur = 30
2
3  if umur >= 18:
4    print('Anda masuk ke dalam kategori dewasa.')
```

Console 33

```
Anda masuk ke dalam kategori dewasa.
>
```

Anda juga menyiapkan baris-baris kode ***else statement*** dalam rangka menampilkan informasi apabila kriteria di dalam ***if statement*** tidak terpenuhi. Misalkan Anda ingin menampilkan pesan di dalam kolom **Console** "Anda masih belum dewasa" untuk nilai variabel **umur** yang kurang dari 18 tahun, baris-baris kode Anda akan menjadi seperti yang diperlihatkan pada Kode Program 47.

Kode Program 47

```
1  umur = 30
2
3  if umur >= 18:
4   print('Anda masuk ke dalam kategori dewasa.')
5  else:
6   print('Anda masih belum dewasa.')
```

Kode Anda telah selesai. Anda dapat mengujinya dengan memasukkan nilai di bawah 18 tahun pada variabel **umur** untuk memastikan bahwa baris keenam pada kode Anda tersebut akan dijalankan oleh program. Misalkan Anda memasukkan nilai 15 ke dalam variabel **umur** seperti yang diperlihatkan pada Kode Program 48, ketika Anda menjalankan kode tersebut, informasi "Anda masih belum dewasa." akan ditampilkan pada kolom **Console** seperti diperlihatkan pada Console 34. Hal ini memperlihatkan bahwa kode program Anda telah berjalan dengan baik.

Kode Program 48

```
1  umur = 15
2
3  if umur >= 18:
4   print('Anda masuk ke dalam kategori dewasa.')
5  else:
6   print('Anda masih belum dewasa.')
```

Console 34

```
Anda masih belum dewasa.
>
```

Dari kode program di atas yang telah Anda buat, pada intinya Anda membuat sebuah perbandingan. Ketika perbandingan tersebut bernilai benar, program akan memberi nilai ***True*** pada baris kode ***if statement*** dan baris kode di bawah ***if statement*** tersebut akan dijalankan oleh program. Sebaliknya, ketika perbandingan tersebut bernilai salah, program akan memberi nilai ***False*** pada baris kode ***if***

statement dan akan mencari apakah ada baris kode ***else statement*** yang tersedia. Apabila ada, baris kode di bawah ***else statement*** tersebut akan dijalankan oleh program. Selain tanda "lebih besar atau sama dengan" yang diperlihatkan pada contoh, beberapa tanda perbandingan juga dapat digunakan di dalam pemrograman Python seperti yang diperlihatkan pada Tabel 2.1.

Tabel 2.1 Beberapa Tanda Perbandingan yang Dapat Digunakan di dalam Bahasa Pemrograman Python

Tanda Perbandingan	Arti
>	Lebih besar
>=	Lebih besar atau sama dengan
<	Lebih kecil
<=	Lebih kecil atau sama dengan
==	Sama dengan
!=	Tidak sama dengan

Tantangan 6:
Untuk memastikan bahwa Anda telah memahami mengenai ***else statement*** dan perbandingan di dalam bahasa pemrograman Python, penulis ingin Anda menuliskan kode sebagai berikut. Bayangkan Anda sedang berada di sebuah taman bermain. Di dalam taman bermain tersebut terdapat sebuah wahana yang dapat dimainkan apabila pengunjung memiliki tinggi badan lebih dari 100 cm. Buatlah kode yang menyatakan situasi tersebut!

Contoh solusi Tantangan 6:

Kode Program 49

```
1  tinggi = 170
2
3  if tinggi >= 100:
4   print('Anda berhak menaiki wahana ini.')
5  else:
6   print('Anda tidak boleh menaiki wahana ini.')
```

Console 35

```
Anda berhak menaiki wahana ini.
> 
```

Kode Program 50

```
1  tinggi = 90
2
3  if tinggi >= 100:
4   print('Anda berhak menaiki wahana ini.')
5  else:
6   print('Anda tidak boleh menaiki wahana ini.')
```

Console 36

```
Anda tidak boleh menaiki wahana ini.
> 
```

G. Operator Logika

Selain tanda-tanda perbandingan yang dapat digunakan di dalam ***if statement*** sebagaimana yang diberikan pada Tabel 2.1, ***if statement*** juga dapat menggunakan tanda-tanda yang disebut sebagai operator kondisional untuk mengevaluasi ***if statement*** yang menggunakan lebih dari satu perbandingan. Ada tiga operator kondisional utama yang sering digunakan di dalam bahasa pemrograman Python, yaitu operator logika, operator identitas, dan operator keanggotaan. Anda akan mempelajari masing-masing operator tersebut satu per satu agar lebih mudah memahami. Pada bagian ini, Anda akan mempelajari tentang operator logika. Operator identitas dan operator keanggotaan akan dibahas pada bagian-bagian selanjutnya.

Ada tiga jenis operator logika, yaitu ***and***, ***or***, dan ***not***. Operator ***and*** akan menghasilkan nilai ***True*** apabila seluruh kondisi yang dibandingkan di dalam ***if statement*** bernilai ***True***. Apabila terdapat salah satu kondisi yang bernilai ***False,*** kondisi ***if statement***-nya juga akan bernilai ***False***. Misalkan pada contoh Tantangan 6, selain harus

memiliki tinggi badan lebih dari 100 cm, Anda juga ingin menambahkan sebuah kondisi di mana seorang pengunjung yang diperbolehkan menaiki sebuah wahana permainan adalah yang memiliki berat badan kurang dari 100 kg. Anda pun membuat sebuah variabel baru selain **`tinggi`**, misalnya **`berat`**. Untuk itu, ***if statement*** yang Anda buat harus mencakup kedua kondisi yang dapat menghasilkan nilai ***True***, yaitu **`tinggi >= 100`** dan **`berat < 100`** sehingga baris ketiga pada kode program Anda sebelumnya dapat Anda ubah menjadi **`if tinggi >= 100 and berat < 100:`** seperti yang diperlihatkan baris keempat pada `Kode Program 51`. Pada baris-baris kode tersebut diperlihatkan seseorang yang memiliki tinggi badan 170 cm dan berat 80 kg. Dikarenakan ketika dievaluasi pada baris keempat ***if statement*** sebelah kiri operator ***and*** menghasilkan nilai ***True*** dan sebelah kanan operator ***and*** juga menghasilkan nilai ***True*** juga, baris tersebut akan bernilai ***True*** sehingga baris kelima **`print('Anda berhak menaiki wahana ini.')`** akan dijalankan oleh program. Hal ini terlihat di kolom **Console** yang menampilkan pesan bahwa "Anda berhak menaiki wahana ini." seperti diperlihatkan pada `Console 37`.

Kode Program 51

```
1  tinggi = 170
2  berat = 80
3
4  if tinggi >= 100 and berat < 100:
5   print('Anda berhak menaiki wahana ini.')
6  else:
7   print('Anda tidak boleh menaiki wahana ini.')
```

Console 37

```
Anda berhak menaiki wahana ini.
>
```

Seandainya salah satu dari kondisi tersebut memiliki nilai ***False***, baris keempat pun akan memiliki nilai ***False*** sehingga baris ketujuh

yang akan dijalankan oleh program, yaitu baris yang berada di bawah ***else statement***. Contohnya diperlihatkan pada `Kode Program 52` dan `Kode Program 53`. Tentunya apabila kedua kondisi tidak memenuhi persyaratan yang ditentukan di dalam ***if statement***, baris keempat pun akan bernilai ***False*** dan baris ketujuh yang akan dijalankan oleh program.

Kode Program 52

```
1  tinggi = 90
2  berat = 60
3
4  if tinggi >= 100 and berat < 100:
5   print('Anda berhak menaiki wahana ini.')
6  else:
7   print('Anda tidak boleh menaiki wahana ini.')
```

Console 38

```
Anda tidak boleh menaiki wahana ini.
>
```

Kode Program 53

```
1  tinggi = 170
2  berat = 110
3
4  if tinggi >= 100 and berat < 100:
5   print('Anda berhak menaiki wahana ini.')
6  else:
7   print('Anda tidak boleh menaiki wahana ini.')
```

Console 39

```
Anda tidak boleh menaiki wahana ini.
>
```

Sementara itu, operator ***or*** akan menghasilkan nilai ***True*** apabila kondisi-kondisi yang dihubungkan oleh operator tersebut salah satunya memiliki nilai ***True***. Mari kita kembali ke contoh kategori orang dewasa di Indonesia. Seperti disebutkan sebelumnya, seseorang dikatakan sudah dewasa apabila telah berumur lebih dari 18 tahun atau

telah menikah. Pada contoh sebelumnya, kita hanya menggunakan variabel umur saja untuk menyatakan seseorang sudah dewasa atau belum. Apabila Anda ingin menyertakan status dari seseorang sebagai pertimbangan apakah seseorang sudah dewasa atau belum, Anda dapat menambahkan sebuah variabel ke dalam baris-baris kode tersebut (misalnya **`menikah`**) dan mengubah ***if statement*** di baris ketiga pada baris-baris di `Kode Program 48` sebelumnya menjadi seperti yang diperlihatkan pada `Kode Program 54`. Pada baris-baris kode tersebut jelas sekali baris kelima dari kode yang akan dijalankan program karena kedua kondisi yang dipisahkan oleh operator logika ***or*** bernilai ***True***, baik **`umur >= 18`** maupun **`menikah`** sehingga ***if statement***-nya pun akan bernilai ***True***.

Kode Program 54

```
1  umur = 15
2  menikah = True
3
4  if umur >= 18 or menikah:
5   print('Anda masuk ke dalam kategori dewasa.')
6  else:
7   print('Anda masih belum dewasa.')
```

Console 40

```
Anda masuk ke dalam kategori dewasa.
>
```

Misalnya Anda mengubah salah satu dari variabel **umur** atau **menikah** sehingga kondisi yang dipisahkan oleh operator logika ***or*** tersebut menjadi memiliki nilai ***False*** seperti yang diperlihatkan pada `Kode Program 55` dan `Kode Program 56`. Program akan tetap menjalankan baris kode yang kelima dikarenakan operator logika ***or*** hanya membutuhkan salah satu kondisi saja yang bernilai ***True*** agar ***if statement***-nya bernilai ***True***. Apabila seluruh kondisi yang dihubungkan oleh operator logika ***or*** bernilai ***False***, barulah ***if***

statement-nya akan bernilai ***False*** seperti yang diperlihatkan pada Kode Program 57. Pada contoh tersebut dapat terlihat bahwa baris kode yang dijalankan adalah baris ketujuh yang berada di bawah ***else statement***.

Kode Program 55

```
1  umur = 17
2  menikah = True
3
4  if umur >= 18 or menikah:
5   print('Anda masuk ke dalam kategori dewasa.')
6  else:
7   print('Anda masih belum dewasa.')
```

Kode Program 56

```
1  umur = 25
2  menikah = False
3
4  if umur >= 18 or menikah:
5   print('Anda masuk ke dalam kategori dewasa.')
6  else:
7   print('Anda masih belum dewasa.')
```

Kode Program 57

main.py

```
1  umur = 16
2  menikah = False
3
4  if umur >= 18 or menikah:
5   print('Anda masuk ke dalam kategori dewasa.')
6  else:
7   print(‘Anda masih belum dewasa.’)
```

Console 41

```
Anda masih belum dewasa.
>
```

Operator logika ***not*** berfungsi untuk membalikkan nilai dari ***True*** menjadi ***False*** dan sebaliknya. Jadi, apabila suatu kondisi yang memiliki nilai ***True*** diberi operator logika ***not*** di depannya, kondisi tersebut berubah menjadi memiliki nilai ***False***. Begitu pun sebaliknya. Misalkan pada baris keempat `Kode Program 57` Anda sisipkan operator logika ***not*** sebelum kondisi **menikah** pada ***if statement*** seperti yang diperlihatkan pada `Kode Program 58`, maka pada saat Anda menjalankan kode tersebut, program akan menjalankan baris kelima seperti yang ditampilkan keluarannya di kolom **Console**, yang diperlihatkan pada Console 42. Hal ini dikarenakan variabel **menikah** yang sebelumnya bernilai ***False***, di baris ***if statement***-nya (baris keempat) berubah menjadi bernilai ***True*** karena diawali dengan operator logika ***not*** sehingga kedua kondisi yang dihubungkan oleh operator logika ***or*** menjadi bernilai ***True*** dan akhirnya ***if statement*** tersebut pun bernilai ***True***.

Kode Program 58

```
1  umur = 16
2  menikah = False
3
4  if umur >= 18 or not menikah:
5   print('Anda masuk ke dalam kategori dewasa.')
6  else:
7   print('Anda masih belum dewasa.')
```

Console 42

```
Anda masuk ke dalam kategori dewasa.
>
```

Secara ringkas, operator-operator logika berperilaku sebagaimana yang diperlihatkan pada Tabel 2.2.

Tabel 2.2 Operator-Operator Logika di dalam Bahasa Pemrograman Python

True	*and*	*True*	=	*True*
True	*and*	*False*	=	*False*
False	*and*	*True*	=	*False*
False	*and*	*False*	=	*False*
True	*or*	*True*	=	*True*
True	*or*	*False*	=	*True*
False	*or*	*True*	=	*True*
False	*or*	*False*	=	*False*
	not	*True*	=	*False*
	not	*False*	=	*True*

Proyek 1: Bilangan dan Kutipan Acak

Bab 3

Setelah Anda mempelajari beberapa dasar dari bahasa pemrograman Python, tibalah saatnya Anda menerapkannya dalam sebuah proyek pemrograman agar Anda dapat lebih memahami tentang penggunaan dari masing-masing konsep dan teknik yang telah Anda pelajari pada bagian-bagian sebelumnya. Pada bab ini, Anda akan membangun sebuah program yang menampilkan teks kutipan (*quote*) secara acak. Namun, seperti yang disebutkan bahwa Anda akan membuat sebuah program yang memiliki perilaku acak di dalamnya, pada bagian ini Anda akan mempelajari terlebih dahulu cara membuat perilaku acak tersebut.

A. Bilangan Acak

Dalam sebuah program, perilaku acak biasanya berhubungan erat dengan bilangan acak. Sederhananya, program akan menghasilkan bilangan secara acak terlebih dahulu dan akan berperilaku sesuai

dengan bilangan yang dihasilkan tersebut. Bahasa pemrograman Python telah menyediakan baris-baris kode untuk menghasilkan sebuah bilangan acak tersebut secara *default*. Hal yang harus Anda lakukan adalah memberi tahu program bahwa Anda akan menggunakan baris-baris kode tersebut. Baris-baris kode biasanya dikemas dalam sebuah blok kode yang dinamakan fungsi. Anda akan mempelajarinya lebih detail pada bagian-bagian selanjutnya. Lebih jauh lagi, berbagai macam fungsi yang serupa biasanya dikelompokkan oleh program ke dalam sebuah modul.

Pada saat bahasa pemrograman Python diinstal di sebuah komputer, terdapat modul-modul bawaan yang juga ikut disertakan di dalam proses instalasinya. Modul-modul bawaan ini berisi fungsi-fungsi yang sering kali digunakan oleh para pemrogram sehingga pemrogram lain cukup menggunakan fungsi-fungsi tersebut tanpa harus menuliskan baris-baris kodenya dari awal. Ketika Anda menggunakan sistem pemrograman berbasis *cloud* seperti Replit, modul-modul bawaan tersebut juga sudah terinstal di dalam server yang Anda akses melalui peramban web. Berarti langkah pertama Anda di dalam kegiatan ini adalah memanggil modul tersebut ke dalam kode program Anda agar Anda dapat menggunakan fungsi-fungsi yang tersedia di dalam modul tersebut di dalam kode program Anda.

Bagaimana cara memanggil sebuah modul ke dalam kode program Anda? Anda cukup mengetikkan **`import nama_modul_yang_ingin_diimpor`** di bagian awal baris kode Anda. Fungsi acak terdapat di dalam modul yang benama **`random`** sehingga untuk memanggil modul random tersebut Anda dapat mengetikkan **`import random`** ke dalam baris pertama kode Anda, seperti yang diperlihatkan pada Kode Program 59.

Kode Program 59

```
1  import random
```

Modul random telah diimpor ke dalam kode program Anda. Kemudian, bagaimana cara menggunakan fungsi-fungsi yang ada di dalam modul tersebut? Misalkan Anda ingin kode program Anda dapat menghasilkan bilangan ***integer*** secara acak. Di dalam modul random tersebut terdapat fungsi **`randint()`** yang dapat menghasilkan bilangan tersebut. Cara menggunakannya adalah dengan mengetikkan nama modul terlebih dahulu, yaitu **`random`**, kemudian diikuti oleh tanda titik dan nama fungsi yang ingin Anda gunakan. Untuk menghasilkan bilangan ***integer*** secara acak, Anda dapat mengetikkan **`random.randint()`** ke dalam baris kode Anda, seperti yang diperlihatkan pada `Kode Program 60`.

`Kode Program 60`

```
1  import random
2
3  random.randint()
```

Fungsi **`randint()`** memerlukan dua bilangan ***integer*** di dalam tanda kurungnya agar dapat berfungsi sebagaimana mestinya. Dua bilangan tersebut menunjukkan bilangan minimum dan bilangan maksimum dari nilai yang ingin dihasilkan oleh fungsi tersebut. Misalkan Anda ingin menghasilkan bilangan ***integer*** secara acak dengan nilai di antara 1 dan 7, Anda harus mengetikkan bilangan 1 dan 7 di dalam tanda kurung tersebut dengan kedua bilangan dipisahkan oleh tanda baca koma sehingga baris-baris kode di atas akan menjadi seperti yang diperlihatkan pada `Kode Program 61`.

`Kode Program 61`

```
1  import random
2
3  random.randint(1, 7)
```

Untuk menampilkan keluaran dari kode program tersebut di kolom **Console**, seperti biasa Anda perlu menggunakan fungsi **`print()`** yang telah Anda gunakan di kode-kode program sebelumnya sehingga baris-baris kode Anda akan menjadi seperti yang diperlihatkan pada `Kode Program 62`.

`Kode Program 62`

```
1  import random
2
3  print(random.randint(1, 7))
```

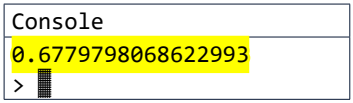
Pada saat Anda menjalankan kode program tersebut, di kolom **Console** akan ditampilkan bilangan ***integer*** acak dengan nilai minimum 1 dan nilai maksimum 7. Cobalah menjalankannya berkali-kali untuk memastikan bahwa kemunculan bilangan-bilangan tersebut bersifat acak.

Selain bilangan ***integer***, Anda juga bisa membuat sebuah bilangan ***float*** secara acak. Anda dapat menggunakan fungsi **`random()`** untuk membuat bilangan tersebut. Tidak seperti pada fungsi **`randint()`**, Anda tidak perlu memasukkan bilangan apa pun di dalam fungsi **`random()`** sehingga Anda cukup mengetikkan **`random.random()`** seperti yang diperlihatkan pada `Kode Program 63`. Ingat bahwa fungsi **`print()`** hanyalah untuk menampilkan hasil dari bilangan yang dibuat pada kolom **Console**. Ketika Anda menjalankan kode program tersebut, pada kolom **Console** akan menampilkan bilangan ***float*** secara acak dengan rentang 0 sampai 1 seperti yang diperlihatkan pada `Console 43`. Pada contoh ini dapat terlihat terdapat 16 angka di belakang koma.

`Kode Program 63`

```
1  import random
2
3  print(random.random())
```

Console 43



Anda mungkin bertanya adakah fungsi lain di dalam modul **`random`** tersebut selain fungsi **`randint()`** dan **`random()`** yang telah kita bahas? Apabila ada, seperti apa kegunaan dan bagaimana cara menggunakan fungsi-fungsi tersebut? Anda bisa melakukan penelusuran di internet dengan menggunakan kata kunci *"Python random module"* dan Anda akan mendapatkan banyak sekali informasi mengenai modul random tersebut. Namun, dari pengalaman penulis, sumber yang paling mudah dalam menjelaskan modul random tersebut beserta fungsi-fungsi di dalamnya yang disertai dengan contoh-contoh baris kode adalah salah satu halaman dari situs web W3Schools yang beralamat di https://www.w3schools.com/python/module_random.asp. Python merupakan salah satu bahasa pemrograman yang paling populer di dunia sehingga banyak sekali yang membahas mengenai bahasa pemrograman ini. Selain itu, ketika Anda ingin menggali lebih dalam informasi yang tersedia mengenai sebuah modul atau sebuah fungsi, Anda dapat melakukan penelusuran di internet dengan mudah.

Tantangan 7:
Untuk memastikan bahwa Anda telah memahami konsep bilangan acak dalam bahasa pemrograman Python, penulis ingin memberikan sebuah tantangan. Buatlah sebuah program yang menghasilkan salah satu dari tujuh warna pelangi (merah, jingga, kuning, hijau, biru, nila, atau ungu) secara acak setiap kali program tersebut dijalankan! Catatan: pada proyek mini ini Anda akan mengombinasikan berbagai macam hal yang telah Anda pelajari pada bab-bab sebelumnya, seperti ***`if statement`***.

Contoh solusi Tantangan 7:
Terdapat tujuh warna di dalam pelangi, yaitu merah, jingga, kuning, hijau, biru, nila, dan ungu. Oleh karena itu, program yang dibuat pertama-tama harus dapat menghasilkan tujuh

bilangan secara acak. Anda dapat menggunakan fungsi **randint(1, 7)** untuk menghasilkan bilangan tersebut seperti yang dijelaskan sebelumnya. Bilangan yang dihasilkan kemudian dimasukkan ke dalam sebuah variabel. Kemudian, dengan menggunakan ***if statement***, Anda akan membuat program berperilaku sesuai dengan bilangan yang dimasukkan ke dalam variabel tersebut. Dikarenakan terdapat tujuh kemungkinan bilangan yang dihasilkan, akan ada tujuh ***if statement*** juga yang akan diperlukan di dalam program Anda. Jangan lupa ketika melakukan perbandingan di dalam masing-masing ***if statement***, Anda menggunakan tanda "sama dengan" sebanyak dua kali (==), bukan hanya satu kali. Salah satu contoh solusi dari tantangan ketujuh tersebut secara lengkap diperlihatkan oleh baris-baris pada Kode Program 64. Jalankan program yang Anda buat tersebut berkali-kali untuk memastikan bahwa program tersebut sudah berjalan dengan baik sesuai dengan spesifikasi tantangan yang telah diberikan.

Catatan: Cara yang lebih tepat dalam merepresentasikan ***if statement*** pada contoh solusi yang diperlihatkan pada Kode Program 64 adalah dengan menggunakan ***elif*** alih-alih dari ***if*** untuk ***if statement*** kedua pada baris ke-7 sampai dengan ***if statement*** ketujuh pada baris ke-17 seperti yang diperlihatkan pada Kode Program 65.

Kode Program 64

```
1  import random
2
3  bilangan_acak = random.randint()
4
5  if bilangan_acak == 1:
6   print('Merah')
7  if bilangan_acak == 2:
8   print('Jingga')
9  if bilangan_acak == 3:
10  print('Kuning')
11 if bilangan_acak == 4:
12  print('Hijau')
13 if bilangan_acak == 5:
14  print('Biru')
15 if bilangan_acak == 6:
16  print('Nila')
17 if bilangan_acak == 7:
18  print('Ungu')
```

Kode Program 65

main.py

```
1  import random
2
3  bilangan_acak = random.randint()
4
5  if bilangan_acak == 1:
6   print('Merah')
7  elif bilangan_acak == 2:
8   print('Jingga')
9  elif bilangan_acak == 3:
10  print('Kuning')
11 elif bilangan_acak == 4:
12  print('Hijau')
13 elif bilangan_acak == 5:
14  print('Biru')
15 elif bilangan_acak == 6:
16  print('Nila')
17 elif bilangan_acak == 7:
18  print('Ungu')
```

Secara umum, ***if*** dan ***elif*** memiliki fungsi yang sama. Namun, apabila Anda menggunakan ***elif,*** Anda mengelompokkan sebuah ***if statement*** dan ***elif statement*** di bawahnya menjadi satu kesatuan. Artinya, apabila salah satu kondisi dari kelompok tersebut terpenuhi, baris-baris kode lainnya tidak akan dijalankan. Misalnya pada baris-baris di `Kode Program 65` ketika dijalankan menghasilkan bilangan acak bernilai **`1`,** maka setelah menjalankan baris ke-5 dan ke-6 (karena nilai variabel **`bilangan_acak == 1`** bernilai **`True`**), program tidak akan menjalankan baris-baris kode ke-7 sampai ke-18. Berbeda apabila Anda menggunakan ***if statement*** saja, seluruh baris kode akan dievaluasi sehingga ***elif statement*** akan menjadikan kode program Anda lebih efisien apabila digunakan pada kasus memilih dari sekian banyak pilihan tersebut.

B. Kutipan Acak

Mari kita mulai membuat proyek program yang lebih kompleks lagi dibandingkan tantangan-tantangan sebelumnya. Dalam memulai sebuah proyek, sangat disarankan untuk menggunakan pendekatan "dimulai dari akhir". Artinya, Anda harus mengetahui terlebih dahulu apa yang ingin Anda tampilkan di akhir program Anda. Misalnya, Anda ingin menampilkan kutipan secara acak di kolom **Console** yang disertai dengan nama penulisnya. Anda juga ingin menampilkan nomor secara acak (sebut saja nomor keberuntungan Anda) di bawah kutipan tersebut seperti yang diperlihatkan pada `Console 44`, Anda dapat mulai dengan membuat baris program dengan fungsi **`print()`** seperti yang diperlihatkan pada `Kode Program 66`.

`Console 44`

```
Kutipan hari ini:
"Tulislah sendiri buku yang tak kau temukan ketika kau
cari"
~ Budhi Gustiandi, 2022 ~

Nomor keberuntungan Anda: 123-456-789
>
```

`Kode Program 66`

```
1  print('''
2  Kutipan hari ini:
3  "Tulislah sendiri buku yang tak kau temukan ketika kau
   cari"
4  ~ Budhi Gustiandi, 2022 ~
5
6  Nomor keberuntungan Anda: 123-456-789
7  ''')
```

Langkah selanjutnya adalah menentukan bagian-bagian dari keluaran tersebut yang berubah-ubah setiap kali program dijalankan. Dalam kasus ini, kutipan yang ditampilkan (dan nama penulisnya)

serta nomor keberuntungan merupakan bagian yang berubah-ubah dari tampilan program, sehingga bagian-bagian ini dapat digantikan dengan menggunakan variabel yang kemudian variabel itu akan digunakan di dalam ***string*** yang ditampilkan di dalam fungsi **`print()`** tersebut.

Dari kedua bagian tersebut, bagian nomor keberuntungan adalah yang paling mudah untuk dikerjakan selanjutnya. Anda akan mengganti ketiga bagian di dalam nomor keberuntungan tersebut dengan tiga buah variabel, sebut saja **`nomor_1`**, **`nomor_2`**, dan **`nomor_3`**. Untuk itu, baris-baris kode tersebut dapat diubah menjadi seperti yang diperlihatkan pada `Kode Program 67`.

`Kode Program 67`

```
1  nomor_1 = 123
2  nomor_2 = 456
3  nomor_3 = 789
4
5  print(f'''
6  Kutipan hari ini:
7  "Tulislah sendiri buku yang tak kau temukan ketika kau
   cari"
8  ~ Budhi Gustiandi, 2022 ~
9
10 Nomor keberuntungan Anda: {nomor_1}-{nomor_2}-{nomor_3}
11 ''')
```

Jangan lupa untuk menyertakan tanda kurung kurawal untuk mengapit masing-masing variabel yang disertakan di dalam fungsi **`print()`** tersebut dan menambahkan huruf **`f`** tepat sebelum tanda kutip pertama yang mengapit teks yang ingin ditampilkan dengan menggunakan fungsi **`print()`** tersebut. Apabila Anda menjalankan kode tersebut, kolom **Console** akan tetap menampilkan seperti yang diperlihatkan pada `Console 44`.

Berikutnya adalah Anda dapat mengganti masing-masing nilai untuk variabel **nomor_1**, **nomor_2**, dan **nomor_3** dengan nomor acak dari 0 sampai 999 dengan menggunakan fungsi **randint(0, 999)**. Jangan lupa untuk memasukkan modul **random** ke dalam program yang Anda buat dengan menggunakan kode **import random**. Baris-baris kode yang Anda buat akan menjadi seperti yang diperlihatkan pada Kode Program 68.

Kode Program 68

```
1  import random
2
3  nomor_1 = random.randint(0, 999)
4  nomor_2 = random.randint(0, 999)
5  nomor_3 = random.randint(0, 999)
6
7  print(f'''
8  Kutipan hari ini:
9  "Tulislah sendiri buku yang tak kau temukan ketika kau
   cari"
10 ~ Budhi Gustiandi, 2022 ~
11
12 Nomor keberuntungan Anda: {nomor_1}-{nomor_2}-{nomor_3}
13 ''')
```

Anda akan mendapatkan tampilan serupa dengan yang diperlihatkan pada Console 44 sebelumnya. Namun, nomor keberuntungan akan berbeda-beda setiap kali Anda menjalankan programnya.

Setelah Anda berhasil mengganti nomor keberuntungan Anda dari bilangan-bilangan yang tetap menjadi bilangan-bilangan acak, selanjutnya adalah mengubah isi dari kutipan yang ingin ditampilkan. Serupa dengan bagian nomor keberuntungan sebelumnya, Anda dapat menggantikan bagian kutipan dengan sebuah variabel. Misalkan Anda menggunakan variabel **kutipan** untuk menyimpan teks yang ingin ditampilkan, kode program Anda dapat menjadi seperti yang diperlihatkan pada Kode Program 69.

Kode Program 69

```
import random

nomor_1 = random.randint(0, 999)
nomor_2 = random.randint(0, 999)
nomor_3 = random.randint(0, 999)
kutipan = '''
"Tulislah sendiri buku yang tak kau temukan ketika kau cari"
~ Budhi Gustiandi, 2022 ~
'''

print(f'''
Kutipan hari ini:
{kutipan}

Nomor keberuntungan Anda: {nomor_1}-{nomor_2}-{nomor_3}
''')
```

Tentunya Anda tidak ingin teks yang ditampilkan di dalam kolom **Console** selalu sama setiap kali Anda menjalankan program Anda. Misalnya, terdapat tiga buah kutipan yang ingin Anda tampilkan secara acak setiap kali program dijalankan. Untuk itu, Anda dapat menggunakan ***if statement*** dan ***elif statement*** seperti pada solusi tantangan ketujuh pada bagian sebelumnya. Dikarenakan jumlah kutipan yang ingin ditampilkan dibatasi tiga saja pada contoh proyek ini, bilangan ***integer*** acak yang harus dihasilkan juga dibatasi hanya satu sampai tiga saja. Anda dapat menggunakan fungsi **`randint(1, 3)`** untuk menghasilkan bilangan tersebut, serupa dengan ketika Anda membuat bilangan-bilangan acak untuk bagian nomor keberuntungan. Dikarenakan modul random sudah disertakan di dalam program pada langkah sebelumnya, Anda cukup langsung memanggil fungsi tersebut. Baris-baris kode program yang Anda tulis kini menjadi seperti yang diperlihatkan pada `Kode Program 70`.

Kode Program 70

```
import random

nomor_1 = random.randint(0, 999)
nomor_2 = random.randint(0, 999)
nomor_3 = random.randint(0, 999)
pilihan_kutipan = random.randint(1, 3)

if pilihan_kutipan == 1:
  kutipan = '''
  "Tulislah sendiri buku yang tak kau temukan ketika kau cari."
  ~ Budhi Gustiandi, 2022 ~
  '''
elif pilihan_kutipan == 2:
  kutipan = '''
  "Jika kau merasa sakit karena suatu perbuatan, belajarlah untuk tidak melakukan perbuatan tersebut ke orang lain."
  ~ Anonim ~
  '''
elif pilihan_kutipan == 3:
  kutipan = '''
  "Suatu pekerjaan yang paling tak kunjung bisa diselesaikan adalah pekerjaan yang tak kunjung pernah dimulai. "
  ~ J.R.R. Tolkien ~
  '''

print(f'''
Kutipan hari ini:
{kutipan}

Nomor keberuntungan Anda: {nomor_1}-{nomor_2}-{nomor_3}
''')
```

Ingat bahwa kutipan-kutipan yang disertakan di dalam kode tersebut hanyalah contoh dari penulis. Anda dapat menggunakan kutipan-kutipan yang Anda peroleh sendiri. Jangan lupa menguji kode program yang telah Anda buat dengan menjalankannya berkali-kali

untuk melihat bahwa kutipan-kutipan yang Anda sertakan akan ditampilkan secara acak. Begitu juga dengan nomor keberuntungan yang ditampilkan secara acak juga.

Selamat! Anda telah berhasil menyelesaikan proyek pertama Anda. Kita akan membahas dua proyek lainnya pada bab-bab selanjutnya dari buku ini. Namun, sebelum melakukannya, ada beberapa konsep bahasa pemrograman Python yang masih harus Anda pelajari. Mari kita lanjutkan pembelajaran pada bab-bab berikutnya.

Bab 4

List, Tuple, Loop, dan *Dictionary*

Setelah Anda menyelesaikan proyek pertama Anda dengan menggunakan bahasa pemrograman Python, Anda pasti sudah makin memahami alur pembelajaran pada buku ini. Anda belajar beberapa konsep bahasa pemrograman Python dan menyatukannya semuanya ketika Anda membuat sebuah proyek. Sekarang waktunya untuk belajar beberapa konsep bahasa pemrograman Python baru sebagai persiapan untuk proyek Anda selanjutnya.

Pada bab ini Anda akan belajar mengenai data yang tersusun dalam bentuk daftar yang dapat digunakan di dalam bahasa pemrograman Python. Anda akan belajar bagaimana cara memperbarui daftar tersebut, seperti menambahkan data, menyisipkan data, dan menghapus data yang ada di dalam daftar. Dua jenis operator juga akan diperkenalkan di dalam bab ini, yaitu operator identitas dan operator keanggotaan. Konsep-konsep daftar dan operator tersebut kemudian akan diterapkan pada konsep pengulangan di dalam pemrograman melalui pernyataan *for loop* yang juga akan Anda pelajari pada bab

ini. Terakhir, Anda akan mempelajari mengenai konsep penyusunan data dalam bentuk kamus yang berfungsi untuk menghubungkan kata kunci dengan isi dari data di dalam sebuah daftar.

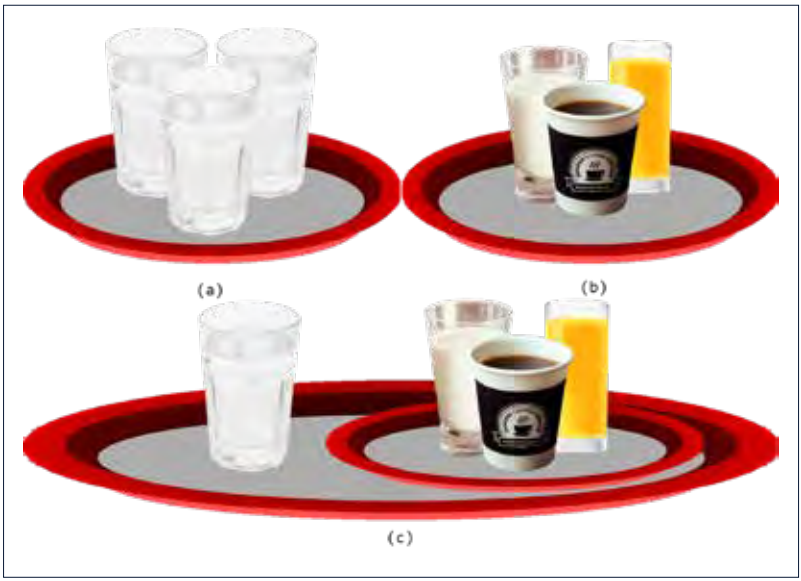
A. *List*

Pada bagian ini, Anda akan belajar mengenai ***list***. Ini adalah sebuah jenis data baru bagi Anda dalam bahasa pemrograman Python dan ini sebenarnya hanyalah sebuah daftar yang berisi data secara berurutan. Contohnya adalah daftar tentang buku bacaan, film favorit, atau makanan favorit Anda. Perlu dicatat bahwa ***list*** di dalam bahasa pemrograman Python memiliki urutan tertentu di dalamnya.

Ketika Anda mempelajari konsep variabel pada bab sebelumnya, Anda hanya diberikan contoh menyimpan satu data saja di dalam satu variabel. Anda mungkin bertanya-tanya, apakah sebuah variabel dapat menyimpan lebih dari satu data? Jawabannya adalah iya, Anda dapat menyimpan lebih dari satu data di dalam sebuah variabel. Inilah yang disebut dengan ***list***. Bagaimana dengan isi dari variabel tersebut? Apakah harus memiliki jenis data yang sama atau jenis-jenis data yang berbeda dapat dimasukkan di dalam sebuah variabel? Anda dapat memasukkan ***integer*** ke dalam ***list*** yang telah Anda buat. Anda juga dapat memasukkan ***float*** ke dalam ***list*** tersebut. ***String*** dan ***boolean*** juga dapat Anda masukkan ke dalam sebuah ***list***. Bahkan, Anda juga dapat memasukkan ***list*** ke dalam sebuah ***list*** untuk membuat sebuah ***list*** bertingkat. Intinya, sebuah ***list*** dapat berisi campuran dari jenis-jenis data yang dapat digunakan di dalam bahasa pemrograman Python.

Bayangkan terdapat beberapa kumpulan minuman yang akan disajikan seperti yang diperlihatkan pada Gambar 4.1. Pada gambar tersebut, ***list*** dapat diumpamakan sebagai baki yang dapat digunakan untuk menyimpan gelas-gelas minuman tersebut. Di dalam sebuah

baki, Anda dapat menyimpan gelas-gelas minuman dengan jenis yang sama seperti yang diperlihatkan pada Gambar 4.1(a). Ini diumpamakan sebagai sebuah ***list*** yang terdiri dari data yang jenisnya sama. Anda juga dapat menyimpan gelas-gelas minuman dengan jenis berbeda seperti yang diperlihatkan pada Gambar 4.1(b). Ini diibaratkan sebagai sebuah ***list*** yang terdiri dari data dengan jenis yang berbeda. Bahkan Anda juga dapat menyimpan sebuah baki di dalam baki lain yang lebih besar seperti yang diperlihatkan pada Gambar 4.1(c). Hal ini mengumpamakan bahwa di dalam sebuah ***list*** dapat mengandung ***list*** lain atau yang lebih dikenal sebagai ***list*** bertingkat.



Keterangan: (a) *List* dengan jenis data yang sama, (b) *List* dengan jenis data yang berbeda, dan (c) *List* dengan jenis data *list* di dalamnya atau *list* bertingkat

Sumber: fawaz-qureshi (2023), Penelope883 (2022), Ragabz (2021), mojzagrebinfo (2014), OpenClipart-Vectors (2013)

Gambar 4.1 Perumpamaan Konsep *List* dengan Menggunakan Berbagai Minuman yang Disajikan dengan Baki

Mari kita buat contoh pertama dari sebuah ***list***. Buatlah sebuah daftar berisi makanan favorit Anda. Jadi, Anda bebas untuk mengisi daftar tersebut, tidak harus sama dengan contoh yang penulis berikan di sini.

Cara membuat sebuah ***list*** dalam bahasa pemrograman Python sangat sederhana. Anda tinggal membuat sebuah variabel, memberikan tanda sama dengan, dan memberikan kurung siku buka dan tutup seperti yang diperlihatkan pada `Kode Program 71`.

Kode Program 71

```
1  makanan_favorit = []
```

Ini adalah sebuah contoh dari ***list*** yang masih kosong. Untuk memasukkan data ke dalam ***list*** tersebut, Anda dapat mengetikkan apa pun yang Anda inginkan dengan memisahkan masing-masing data tersebut dengan tanda baca koma.

Untuk contoh ***list*** dengan nama **`makanan_favorit`** yang telah Anda buat sebelumnya, sepertinya jenis data yang paling sesuai untuk dimasukkan ke dalam ***list*** tersebut adalah ***string*** atau teks. Mari kita masukkan nama makanan favorit Anda ke dalam ***list*** tersebut. Misalkan Anda ingin memasukkan **`rendang`**, **`sate`**, dan **`nasi goreng`** ke dalam ***list*** **`makanan_favorit`** Anda, Anda hanya perlu menuliskan nama masing-masing makanan favorit Anda tersebut di antara tanda kurung siku dan dipisahkan dengan tanda koma. Jangan lupa untuk memberi tanda petik tunggal atau ganda untuk menyatakan bahwa jenis data yang Anda masukkan ke dalam ***list*** adalah ***string*** atau teks seperti yang diperlihatkan pada `Kode Program 72`.

Kode Program 72

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
```

Seperti biasa, apabila Anda ingin melihat hasilnya di kolom **Console**, Anda dapat menggunakan fungsi **print()** seperti pada bagian-bagian sebelumnya. Anda cukup memasukkan nama variabel yang ingin Anda tampilkan di kolom **Console** seperti yang diperlihatkan baris ke-3 pada `Kode Program 73`. Pada baris-baris kode tersebut dapat terlihat bahwa isi dari ***list*** berhasil ditampilkan di kolom ***Console*** yang sama dengan yang Anda isikan pada saat Anda membuat ***list*** tersebut.

Kode Program 73

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2
3  print(makanan_favorit)
```

Console 45

```
['rendang', 'sate', 'nasi goreng']
>
```

Anda mungkin penasaran dengan kegunaan dari ***list*** seperti ini. Mengapa Anda harus membuat jenis data seperti ini? Apa yang bisa Anda lakukan dengannya? Salah satu kegunaan dari ***list*** adalah untuk mengelompokkan data yang sejenis walaupun bisa saja di dalam sebuah ***list*** berisi data dengan jenis yang beragam (***integer***, ***float***, ***string***, atau ***boolean***).

Anda dapat mengambil satu atau lebih data yang tertera di dalam sebuah ***list*** sesuai dengan kebutuhan Anda (atau bahkan semuanya seperti yang diperlihatkan di atas). Sebagai contoh, Anda ingin mengambil isi pertama dari ***list*** `makanan_favorit` Anda seperti yang telah dibuat pada contoh sebelumnya. Bagaimana cara Anda melakukannya? Pertama, Anda menuliskan nama ***list*** yang ingin Anda tampilkan isinya, kemudian menuliskan tanda kurung siku setelah nama ***list*** yang Anda inginkan. Kemudian Anda tentukan isi yang mana yang Anda inginkan. Misalkan Anda ingin menggunakan data

pertama dalam ***list*** **makanan_favorit** Anda, yaitu **rendang** dalam contoh di atas, Anda tuliskan **print(makanan_favorit[1])** untuk menampilkannya di kolom **Console** seperti yang diperlihatkan pada Kode Program 74.

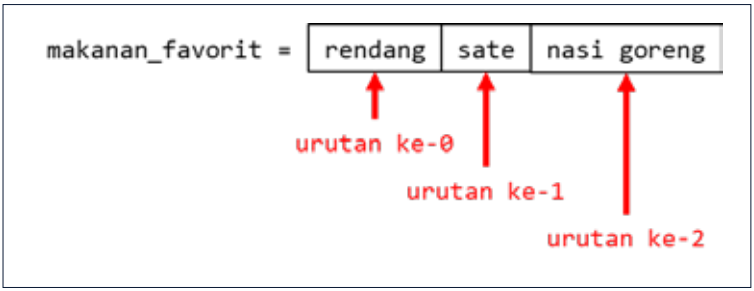
Kode Program 74

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2
3  print(makanan_favorit[1])
```

Console 46

```
Console
sate
>
```

Namun, setelah Anda menjalankan program, kolom ***Console*** justru menampilkan **sate** yang merupakan data kedua dari ***list*** **makanan_favorit** Anda seperti diperlihatkan pada Console 46. Mengapa demikian? Hal ini dikarenakan ***list*** menggunakan perhitungan berbasis bilangan nol. Artinya, di dalam sebuah ***list***, data pertama adalah urutan yang ke-nol. Pada contoh di atas, rendang berada pada urutan nol, sate pada urutan pertama, dan nasi goreng pada urutan kedua, seperti yang diperlihatkan pada Gambar 4.2 sehingga apabila Anda ingin menampilkan rendang di dalam kolom **Console**, Anda harus menuliskan **print(makanan_favorit[0])** pada baris kode Anda seperti yang diperlihatkan pada Kode Program 75.



Gambar 4.2 Urutan di dalam Sebuah *List* yang Dimulai dari Urutan Ke-0 (bukan ke-1).

Kode Program 75

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2
3  print(makanan_favorit[0])
```

Console 47

```
rendang
>
```

Begitulah cara Anda mengakses isi dari sebuah ***list***.

Tantangan 8:
Buatlah sebuah ***list*** yang berisi tentang judul film favorit Anda. ***List*** cukup berisi tiga sampai lima judul di dalamnya. Kemudian, tampilkan isi terakhir dari ***list*** yang Anda buat pada kolom **Console**.

Contoh solusi Tantangan 8:

Kode Program 76

```
1  film_favorit = ['Iqro: Petualangan Meraih Bintang',
   'Iqro: My Universe', 'Habibie & Ainun', 'Keluarga
   Cemara', 'Rumah Tanpa Jendela']
2
3  print(film_favorit[4])
```

Console 48

```
Rumah Tanpa Jendela
>
```

B. Menambahkan, Menyisipkan, dan Menghapus Isi *List*

Sekarang Anda telah memahami dasar-dasar tentang ***list***, mari kita lanjutkan pembelajaran Anda mengenai beberapa detail yang dapat membuat jenis data tersebut sangat berguna di dalam bahasa pemrograman Python. Kita mulai dengan cara mengetahui panjang dari sebuah ***list*** yang Anda buat. Jadi, kita akan gunakan ***list*** tentang makanan favorit yang telah Anda buat di bagian sebelumnya. Sekarang, bagaimana apabila Anda ingin mengetahui panjang dari ***list*** tersebut atau dengan kata lain berapa jumlah data yang ada di dalam ***list*** tersebut dengan menggunakan bahasa pemrograman Python? Anda cukup menuliskan **`len`**, yang merupakan kependekan dari *length* ("panjang" dalam bahasa Inggris) kemudian memberi tanda kurung buka dan tutup setelahnya. Di dalam tanda kurung tersebut, tuliskan variabel yang ingin Anda ketahui panjangnya, yaitu **`makanan_favorit`**. Seperti biasa, agar Anda dapat melihat hasilnya di kolom **Console**, Anda harus menggunakan fungsi **`print()`**. Baris-baris kode yang Anda buat akan menjadi seperti yang diperlihatkan pada `Kode Program 77`. Di kolom **Console** yang diperlihatkan pada `Console 49` terlihat bahwa panjang ***list*** **`makanan_favorit`** adalah **`3`**.

Kode Program 77

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2
3  print(len(makanan_favorit))
```

Console 49

```
3
>
```

Sekarang, misalkan Anda ingin menambahkan makanan favorit baru ke dalam ***list*** Anda, yang sebelumnya berjumlah tiga menjadi empat. Bagaimana caranya? Anda dapat menuliskan nama ***list*** yang ingin Anda tambahkan isinya, yaitu **`makanan_favorit`** pada kasus ini, kemudian menuliskan **`.append()`**setelah nama ***list*** tersebut. Lalu, masukkan ke dalam tanda kurung tersebut nama makanan favorit yang ingin Anda tambahkan. Misalkan Anda ingin memasukkan 'bakso' ke dalam ***list*** **makanan_favorit** Anda maka baris-baris kodenya akan menjadi seperti yang diperlihatkan pada `Kode Program 78`. Dari tampilan yang diperlihatkan pada `Console 50`, ketika program dijalankan dapat terlihat bahwa isi dari ***list*** yang sebelumya berjumlah tiga kini menjadi empat dengan adanya isi baru, yaitu bakso.

Kode Program 78

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2
3  print(len(makanan_favorit))
4
5  makanan_favorit.append('bakso')
6
7  print(len(makanan_favorit))
```

Console 50

```
3
4
>
```

Di manakah posisi dari 'bakso' tersebut di dalam ***list*** Anda? Untuk mengetahuinya, Anda dapat menampilkan ***list*** yang sudah ditambahkan 'bakso' tersebut di kolom **Console** sebagaimana diperlihatkan pada `Console 51`. Dapat terlihat bahwa 'bakso' ditambahkan pada bagian terakhir dari ***list*** **`makanan_favorit`** Anda, atau pada urutan yang keempat.

Kode Program 79

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2  makanan_favorit.append('bakso')
3
4  print(makanan_favorit)
```

Console 51

```
['rendang', 'sate', 'nasi goreng', 'bakso']
>
```

Anda kemudian ingin memasukkan nama makanan baru ke dalam *list* **makanan_favorit** Anda, Anda tidak ingin memasukkannya ke urutan paling belakang. Anda ingin menyisipkan 'soto' di antara 'sate' dan 'nasi goreng'. Anda ingin menyisipkan masakan favorit Anda yang baru tersebut ke urutan ketiga dari *list* **makanan_favorit** Anda. Bagaimana caranya? Anda dapat melakukannya dengan menggunakan fungsi **insert()**. Sama seperti ketika Anda menggunakan fungsi **append()**, pertama-tama Anda menuliskan nama *list* yang ingin disisipkan (dalam hal ini **makanan_favorit**), kemudian Anda menuliskan **.insert()** setelah nama *list* tersebut. Namun, tidak seperti pada fungsi **.append()** yang hanya memerlukan nilai yang ingin Anda masukkan saja, pada fungsi **.insert()** Anda harus memasukkan dua nilai ke dalam tanda kurungnya dengan masing-masing nilai dipisahkan oleh tanda koma. Nilai yang pertama adalah urutan ke berapa Anda ingin menyisipkan isi yang baru dan nilai yang kedua adalah isi yang ingin Anda masukkan tersebut, dalam hal ini 'soto'. Ingat bahwa *list* menggunakan urutan dari nol. Untuk itu, apabila Anda ingin memasukkan 'soto' ke dalam urutan ketiga dari *list* Anda, Anda harus memasukkan angka 2 pada nilai pertama di dalam tanda kurung fungsi **.insert()** sehingga baris-baris kode Anda akan menjadi seperti yang diperlihatkan pada Kode Program 80. Pada Console 52, yang merupakan tampilan ketika program tersebut dijalankan, dapat terlihat bahwa panjang *list* menjadi lima

dan nilai 'soto' disisipkan di antara 'sate' dan 'nasi goreng', bukan pada akhir ***list*** seperti pada kasus 'bakso' sebelumnya.

Kode Program 80

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2  makanan_favorit.append('bakso')
3  makanan_favorit.insert(2, 'soto')
4
5  print(len(makanan_favorit))
6  print(makanan_favorit)
```

Console 52

```
['rendang', 'sate', 'soto', 'nasi goreng', 'bakso']
>
```

Dikarenakan Anda terlalu banyak makan 'sate', Anda menjadi bosan dan memutuskan 'sate' tidak menjadi makanan favorit Anda lagi. Bagaimana caranya Anda mengeluarkan dari ***list*** **makanan_favorit** Anda? Anda dapat menggunakan fungsi **`del()`**. Namun, pertama-tama, Anda harus mengetahui terlebih dahulu urutan dari isi ***list*** yang ingin Anda hapus. Dalam kasus ini 'sate' berada dalam urutan kedua di dalam ***list*** tersebut. Dikarenakan urutan ***list*** dimulai dari angka nol, 'sate' adalah urutan kesatu di ***list*** tersebut. Karena itu, untuk merujuk ke 'sate', Anda harus menuliskannya dengan **`makanan_favorit[1]`**. Masukkan ke dalam tanda kurung dari fungsi **`del()`**seperti yang diperlihatkan pada `Kode Program 81`. Ketika Anda menampilkan kembali isi dari ***list*** **`makanan_favorit`** Anda di kolom **Console**, akan terlihat bahwa 'sate' sudah dikeluarkan dari ***list*** tersebut seperti yang ditampilkan pada `Console 53`.

Kode Program 81

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2  makanan_favorit.append('bakso')
3  makanan_favorit.insert(2, 'soto')
4
5  print(makanan_favorit)
6
7  del(makanan_favorit[1])
9  print(makanan_favorit)
```

Console 53

```
['rendang', 'sate', 'soto', 'nasi goreng', 'bakso']
['rendang', 'soto', 'nasi goreng', 'bakso']
>
```

Tantangan 9:
Modifikasi baris-baris kode yang diperlihatkan pada Kode Program 81 sehingga Anda hanya menyisakan 'nasi goreng' saja di dalam ***list*** tersebut! Anda tidak diperbolehkan untuk mengubah baris-baris kode pertama sampai kesembilan yang sudah diberikan di dalam kode program tersebut.

Contoh solusi Tantangan 9:

Kode Program 82

```
1  makanan_favorit = ['rendang', 'sate', 'nasi goreng']
2  makanan_favorit.append('bakso')
3  makanan_favorit.insert(2, 'soto')
4
5  print(makanan_favorit)
6
7  del(makanan_favorit[1])
9  print(makanan_favorit)
10
11 del(makanan_favorit[0])
12 del(makanan_favorit[0])
13 del(makanan_favorit[1])
14 print(makanan_favorit)
```

Console 54

```
['rendang', 'sate', 'soto', 'nasi goreng', 'bakso']
['rendang', 'soto', 'nasi goreng', 'bakso']
['nasi goreng']
>
```

C. *Tuple*

Ada sebuah jenis data yang sangat mirip dengan ***list***, yaitu ***tuple***. Cara membuat ***tuple*** serupa dengan ketika Anda membuat ***list***. Perbedaannya adalah tanda kurung yang Anda gunakan bukan tanda kurung siku, melainkan tanda kurung biasa seperti yang diperlihatkan pada `Kode Program 83`. Mirip dengan contoh pada bagian sebelumnya, Anda membuat sebuah ***tuple*** bernama **`makanan_favorit`** yang berisi tiga data di dalamnya, yaitu rendang, sate, dan nasi goreng.

Kode Program 83

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
```

Terdapat perbedaan mendasar di antara jenis data ***list*** dan ***tuple***. Seperti yang telah Anda pelajari sebelumnya, untuk jenis data ***list***, Anda dapat mengubah-ubah isi dari ***list*** tersebut, seperti menambahkan isi yang baru, menyisipkan isi baru di antara isi-isi yang sudah ada, dan menghapus isi yang sudah tidak Anda kehendaki berada di dalam ***list*** tersebut. Hal-hal tersebut tidak akan dapat Anda lakukan di dalam jenis data ***tuple***. Apabila Anda berusaha untuk menambahkan isi baru ke dalam sebuah ***tuple***, Anda akan memperoleh pesan kesalahan di kolom ***Console*** bahwa ***tuple*** **makanan_favorit** Anda tidak memiliki atribut **.append()** yang artinya Anda tidak dapat menambahkan isi baru ke dalam ***tuple*** tersebut seperti yang diperlihatkan pada `Console 55` ketika `Kode Program 84` dijalankan.

Kode Program 84

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2
3  makanan_favorit.append('bakso')
```

Console 55

```
Traceback (most recent call last):
 File "main.py", line 3, in <module>
 makanan_favorit.append('bakso')
AttributeError: 'tuple' object has no attribute 'append'
>
```

Selain itu, pesan kesalahan serupa pun ditampilkan pada saat Anda berusaha menyisipkan sebuah data baru ke dalam ***tuple*** yang sudah dibuat menggunakan fungsi **insert()** seperti yang diperlihatkan pada `Console 56` ketika `Kode Program 85` dijalankan. Bahkan, Anda pun tidak dapat menghapus data yang sudah ada di dalam sebuah ***tuple*** menggunakan fungsi **del()** seperti yang diperlihatkan pada `Console 57` ketika `Kode Program 86` dijalankan.

Kode Program 85

```
1 makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2
3 makanan_favorit.insert(1, 'soto')
```

Console 56

```
Traceback (most recent call last):
 File "main.py", line 3, in <module>
 makanan_favorit.insert(1, 'soto')
AttributeError: 'tuple' object has no attribute 'insert'
>
```

Kode Program 86

```
1 makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2
3 del(makanan_favorit[1])
```

Console 57

```
Traceback (most recent call last):
 File "main.py", line 3, in <module>
 del(makanan_favorit[1])
AttributeError: 'tuple' object doesn't support item deletion
>
```

Dari ketiga contoh kode program tersebut, jelas sekali bahwa ketika sebuah ***tuple*** sudah dibuat, Anda tidak akan dapat mengubah isi dari ***tuple*** tersebut. Inilah yang menjadi perbedaan mendasar antara jenis data ***list*** dan ***tuple***. Lalu, untuk apa jenis data tersebut digunakan? Jenis data ***tuple*** biasanya digunakan untuk merepresentasikan data yang tidak akan berubah di dalam baris-baris kode di dalam sebuah program. Kalaupun Anda ingin mengubah isi dari ***tuple*** tersebut, Anda harus mengubahnya langsung pada baris kode ***tuple*** tersebut ditentukan. Hal ini berguna untuk mencegah perubahan data yang tidak diinginkan di baris-baris kode lain pada saat program dijalankan.

D. Operator Identitas

Pada bab sebelumnya, Anda telah mempelajari tentang operator logika. Pada bagian ini, Anda akan mempelajari jenis operator yang kedua, yaitu **operator identitas**. Sama seperti operator logika, operator ini dapat juga digunakan di dalam baris-baris kode yang membutuhkan nilai *boolean True* atau *False* seperti yang digunakan pada ***if statement*** atau ***elif statement***.

Biasanya **operator identitas** digunakan untuk membandingkan dua data di dalam sebuah program untuk mengetahui apakah kedua data tersebut sama atau tidak sama. Terdapat dua jenis **operator identitas**, yaitu **`is`** dan **`is not`**. Operator **`is`** digunakan untuk mengetahui apakah dua data yang dibandingkan adalah sama. Sementara itu, operator **`is not`** digunakan untuk mengetahui apakah dua data yang dibandingkan adalah berbeda.

Misalkan Anda memiliki tiga buah variabel bernama **`makanan_favorit_1`**, **`makanan_favorit_2`**, dan **`makanan_favorit_3`** yang masing-masing memiliki nilai berjenis ***string*** **`'rendang'`**, **`"rendang"`**, dan **`'sate'`** seperti yang diperlihatkan pada `Kode Program 87`. Kemudian Anda ingin membandingkan apakah nilai dari variabel **`makanan_favorit_1`** dan **`makanan_favorit_2`**

adalah sama atau tidak, Anda dapat menggunakan operator **is** seperti yang diperlihatkan pada baris kelima di Kode Program 88.

Kode Program 87

```
1  makanan_favorit_1 = 'rendang''
2  makanan_favorit_2 = "rendang"
3  makanan_favorit_3 = 'sate'
```

Kode Program 88

```
1  makanan_favorit_1 = 'rendang'
2  makanan_favorit_2 = "rendang"
3  makanan_favorit_3 = 'sate'
4
5  if makanan_favorit_1 is makanan_favorit_2:
6   print('Kedua data adalah sama. ')
7  else:
8   print('Kedua data adalah berbeda. 'a)
```

Ketika Anda menjalankan baris-baris kode tersebut, kolom **Console** akan menampilkan bahwa kedua variabel tersebut merujuk ke nilai yang sama seperti yang diperlihatkan pada Console 58. Hal ini memperlihatkan bahwa walaupun Anda menggunakan dua cara dalam merepresentasikan sebuah nilai ***string***, yaitu tanda petik tunggal dan tanda petik ganda, namun program akan menganggap bahwa Anda merujuk ke sebuah nilai yang sama, yaitu **rendang**. Selain itu, walaupun Anda menggunakan dua variabel yang berbeda untuk menyimpan data tersebut, secara penyimpanan di dalam memori komputer, kedua variabel tersebut merujuk ke data yang sama.

Console 58

```
Kedua data adalah sama.
>
```

Apabila Anda mengubah operator **is** di dalam Kode Program 88 tersebut menjadi operator **is not**, nilai pada baris kode kelima akan menjadi ***False*** dan program tidak akan menjalankan baris

kode keenam di bawahnya, tetapi akan menjalankan baris kode kedelapan yang berada di bawah baris kode ***else statement*** seperti yang diperlihatkan pada `Kode Program 89`.

Kode Program 89

```
1  makanan_favorit_1 = 'rendang'
2  makanan_favorit_2 = "rendang"
3  makanan_favorit_3 = 'sate'
4
5  if makanan_favorit_1 is not makanan_favorit_2:
6   print('Kedua data adalah sama.')
7  else:
8   print('Kedua data adalah berbeda.')
```

Console 59

```
Kedua data adalah berbeda.
>
```

Hasil tersebut tentunya tidak tepat karena kedua variabel yang Anda bandingkan memiliki nilai yang sama. Hal ini dikarenakan operator **`is not`** hanya akan menghasilkan nilai ***True*** apabila kedua nilai yang dibandingkan adalah berbeda. Karena itu, untuk penggunaan operator **is not**, sebaiknya Anda mengubah baris-baris kode pada `Kode Program 89` dengan menukar baris keenam dengan baris kedelapan, seperti yang diperlihatkan pada `Kode Program 90`, sehingga ketika Anda menjalankan kembali kode program tersebut, kolom **Console** akan menampilkan hasil yang benar seperti yang diperlihatkan pada `Console 60`.

Kode Program 90

```
1  makanan_favorit_1 = 'rendang'
2  makanan_favorit_2 = "rendang"
3  makanan_favorit_3 = 'sate'
4
5  if makanan_favorit_1 is not makanan_favorit_2:
6   print('Kedua data adalah berbeda.')
7  else:
8   print('Kedua data adalah sama.')
```

Console 60

```
Kedua data adalah sama.
>
```

Untuk memastikan bahwa baris-baris kode tersebut sudah benar, Anda juga dapat membandingkan nilai dari variabel **makanan_favorit_1** dan **makanan_favorit_3** dengan menggunakan operator **is not** dengan mengubah baris kelima kode tersebut menjadi seperti yang diperlihatkan pada Kode Program 91. Dapat terlihat bahwa kode program telah berjalan dengan benar karena memang nilai dari variabel **makanan_favorit_1** dan **makanan_favorit_3** adalah berbeda.

Kode Program 91

```
1 makanan_favorit_1 = 'rendang'
2 makanan_favorit_2 = "rendang"
3 makanan_favorit_3 = 'sate'
4
5 if makanan_favorit_1 is not makanan_favorit_3:
6  print('Kedua data adalah berbeda.')
7 else:
8  print('Kedua data adalah sama.')
```

Console 61

```
Kedua data adalah berbeda.
>
```

Sebenarnya fungsi dari operator identitas serupa dengan fungsi dari operator perbandingan "==" dan "!=" yang diberikan pada Tabel 2.1 pada bab sebelumnya. Jadi, Anda bisa memilih akan menggunakan gaya penulisan kode dengan menggunakan operator identitas atau dengan menggunakan operator perbandingan.

E. Operator Keanggotaan

Operator terakhir yang akan Anda pelajari di dalam buku ini adalah **operator keanggotaan**. Operator ini berfungsi untuk mengetahui apakah sebuah nilai yang ingin Anda bandingkan merupakan anggota dari sekumpulan nilai yang biasanya direpresentasikan di dalam sebuah ***list*** atau ***tuple***. Serupa dengan operator identitas yang Anda pelajari pada bagian sebelumnya, **operator keanggotaan** terdiri dari dua macam, yaitu **`in`** dan **`not in`**.

Misalnya, kita kembali menggunakan ***list*** **`makanan_favorit`** yang telah Anda buat pada bagian sebelumnya seperti yang diperlihatkan kembali pada `Kode Program 92`. Di dalam program, Anda ingin mengetahui apakah sebuah nilai merupakan anggota dari ***list*** tersebut, Anda dapat menggunakan operator **`in`** seperti yang diperlihatkan pada `Kode Program 93`.

Kode Program 92

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
```

Kode Program 93

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2
3  if 'nasi goreng' in makanan_favorit:
4   print('Makanan tersebut adalah makanan favorit Anda.')
5  else:
6   print('Makanan tersebut bukan makanan favorit Anda.')
```

Dikarenakan nasi goreng adalah salah satu nilai yang ada di dalam ***list*** **makanan_favorit**, ketika program dijalankan akan menampilkan bahwa nasi goreng adalah makanan favorit Anda. Tampilan program diperlihatkan seperti pada `Console 62`.

Console 62

```
Makanan tersebut adalah makanan favorit Anda.
>
```

Anda juga dapat menggunakan variabel untuk mewakili nilai yang ingin Anda bandingkan seperti yang diperlihatkan pada baris-baris kode contoh penggunaan operator **`is`** dan **`is not`** pada bagian sebelumnya, seperti yang diperlihatkan pada `Kode Program 94`. Ketika baris-baris kode tersebut dijalankan, kolom **Console** akan menampilkan hasil yang sama seperti yang diperlihatkan pada `Console 62`.

Kode Program 94

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2  makanan = 'nasi goreng'
3
4  if makanan in makanan_favorit:
5   print('Makanan tersebut adalah makanan favorit Anda.')
6  else:
7   print('Makanan tersebut bukan makanan favorit Anda.')
```

Misalkan sekarang Anda memasukkan sebuah nilai ke dalam variabel **makanan** pada baris kedua kode di atas yang tidak ada di dalam ***list* makanan_favorit** tersebut sebagaimana diperlihatkan pada `Kode Program 95`, ketika Anda menjalankan kode program tersebut, baris ketujuh kode di bawah ***else statement*** akan dijalankan karena baris keempat kode akan menjadi bernilai ***False***. Kolom **Console** akan menampilkan hasil seperti yang diperlihatkan pada Console 63.

Kode Program 95

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2  makanan = 'bakso'
3
4  if makanan in makanan_favorit:
5   print('Makanan tersebut adalah makanan favorit Anda.')
6  else:
7   print('Makanan tersebut bukan makanan favorit Anda.')
```

Console 63

```
Makanan tersebut bukan makanan favorit Anda.
>
```

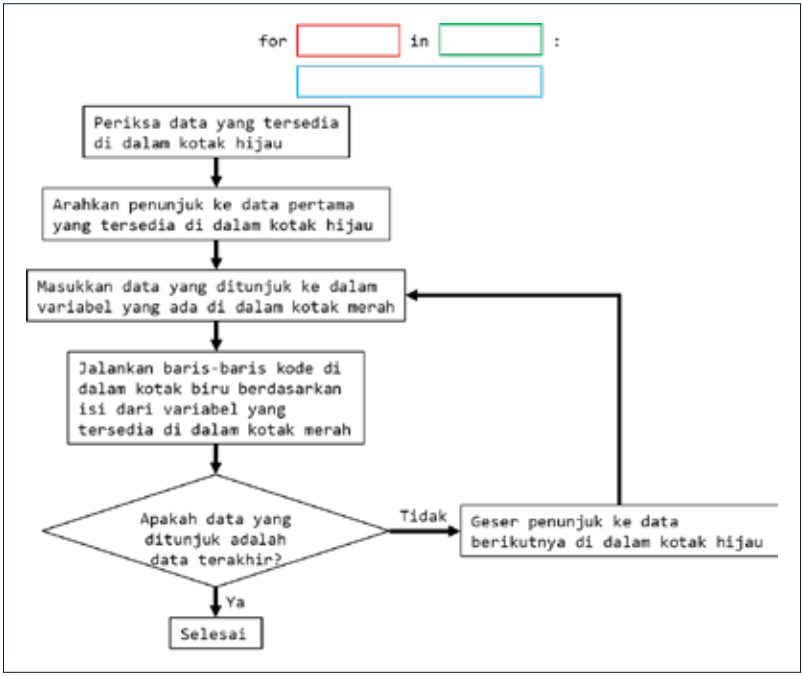
Apabila Anda ingin menggunakan operator **not in** sebagai alternatif dari baris-baris kode di atas, jangan lupa untuk menukar baris kelima kode dengan baris ketujuh kode seperti yang diperlihatkan pada Kode Program 96. Perbedaannya adalah sekarang baris keempat kode akan memiliki nilai ***True*** karena 'bakso' yang merupakan nilai dari variabel makanan tidak ada di dalam ***list* makanan_favorit** tersebut. Dengan demikian, ketika baris-baris kode tersebut dijalankan, kolom **Console** akan tetap menampilkan bahwa **"Makanan tersebut bukan makanan favorit Anda."** seperti yang diperlihatkan sebelumnya.

Kode Program 96

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2  makanan = 'bakso'
3
4  if makanan not in makanan_favorit:
5   print('Makanan tersebut bukan makanan favorit Anda.')
6  else:
7   print('Makanan tersebut adalah makanan favorit Anda.')
```

F. *For Loop*

Pada bagian ini, Anda akan mempelajari tentang ***for loop*** dan bagaimana fungsinya sebagai salah satu kekuatan utama dalam pemrograman. Sebuah ***for loop*** dalam bahasa pemrograman Python dapat membuat Anda untuk menjalankan suatu bagian kode beberapa kali. Secara umum, diagram alur ***for loop*** adalah seperti yang diperlihatkan pada Gambar 4.3. Pada diagram tersebut dapat terlihat adanya perulangan pada saat menjalankan baris-baris kode yang terdapat di dalam sebuah program berdasarkan ketersediaan data yang menjadi acuan di dalam perulangan tersebut. Untuk lebih jelasnya, Anda akan mempraktikkan cara membuat ***for loop*** di dalam bahasa pemrograman Python.



Gambar 4.3 *Diagram Alur For Statement*

Jadi, mari buat ***for loop*** pertama kalinya. Caranya adalah dengan mengetikkan **`for`**, diikuti spasi, diikuti oleh nama variabel (misalkan **`nomor`**), diikuti spasi, diikuti **`in`**, diikuti spasi, dan diikuti **`range()`** dan masukkan sejumlah angka di dalam tanda kurung tersebut. Misalnya Anda memasukkan angka 5 di dalam tanda kurung tersebut. Kemudian berikan tanda titik dua di bagian akhir barisnya. Baris kodenya seperti yang diperlihatkan pada `Kode Program 97`.

`Kode Program 97`

```
1  for nomor in range(5):
```

Kemudian pada baris berikutnya (jangan lupa untuk memberikan spasi atau *tab* sehingga baris kode menjadi agak menjorok ke dalam dan agar baris kode tersebut menjadi bagian dari ***for loop***), Anda ketikkan **`print('Selamat datang!')`** seperti yang diperlihatkan pada `Kode Program 98`. Pada saat Anda menjalankan kode program tersebut, pada kolom **Console** akan menampilkan teks **`Selamat datang!`** sebanyak lima kali seperti yang diperlihatkan pada `Console 64`.

`Kode Program 98`

```
1  for nomor in range(5):
2    print('Selamat datang!')
```

`Console 64`

```
Selamat datang!
Selamat datang!
Selamat datang!
Selamat datang!
Selamat datang!
>
```

Itulah gambaran bagaimana ***for loop*** bekerja. Fungsi ***for loop*** akan mengulangi baris kode yang berada di dalamnya sesuai dengan jumlah yang ditentukan di baris ***for loop*** tersebut. Karena pada baris ***for loop*** Anda mengetikkan **`range(5)`**, artinya Anda memerintahkan fungsi ***for loop*** tersebut untuk mengulangi sebanyak lima kali baris kode **`print('Selamat datang!')`** yang ada di dalamnya.

Untuk lebih memperjelas fungsi ***for loop***, Anda ketikkan **`print('Selamat belajar Python.')`** di baris ketiga seperti yang diperlihatkan pada `Kode Program 99`. Pada saat Anda menjalankan kode program tersebut, kolom **Console** akan menampilkan dua teks yang berada di dalam tanda kurung fungsi **`print()`** masing-masing sebanyak lima kali seperti yang diperlihatkan pada `Console 65`.

Kode Program 99

```
1  for nomor in range(5):
2   print('Selamat datang!')
3   print('Selamat belajar Python.')
```

Console 65

```
Selamat datang!
Selamat belajar Python.
Selamat datang!
Selamat belajar Python.
Selamat datang!
Selamat belajar Python.
Selamat datang!
Selamat belajar Python.
Selamat datang!
Selamat belajar Python.
>
```

Anda pasti akan bertanya-tanya, "Lalu untuk apa variabel **nomor** pada baris ***for loop*** tersebut?". Jadi, setiap kali fungsi ***for loop*** berjalan, variabel **nomor** tersebut akan menyimpan nilai yang berbeda tergantung pada nilai apa yang diasosiasikannya. Pada contoh di atas, variabel **nomor** akan menyimpan angka dari fungsi **range()**. Untuk lebih jelasnya, mari kita tampilkan nilai dari variabel **nomor** setiap kali fungsi ***for loop*** berjalan. Ketikkan **print(nomor)** pada baris kedua sehingga baris-baris kode program Anda akan menjadi seperti yang diperlihatkan pada Kode Program 100.

Kode Program 100

```
1  for nomor in range(5):
2   print(nomor)
3   print('Selamat datang!')
4   print('Selamat belajar Python.')
```

Ketika Anda menjalankan kode program tersebut, dapat terlihat pada kolom **Console** bahwa setiap kali ***for loop*** berjalan, urutan nomor dari fungsi **range()** akan disimpan di dalam variabel **nomor**

seperti yang diperlihatkan pada `Console 66`. Ingat bahwa bahasa pemrograman Python memulai angka dari nol sehingga nomor yang diperlihatkan adalah dari angka nol sampai dengan angka empat.

Console 66

```
0
Selamat datang!
Selamat belajar Python.
1
Selamat datang!
Selamat belajar Python.
2
Selamat datang!
Selamat belajar Python.
3
Selamat datang!
Selamat belajar Python.
4
Selamat datang!
Selamat belajar Python.
>
```

Nama variabel yang digunakan tidak harus **`nomor`** seperti pada contoh di atas. Anda dapat menggantinya dengan nama variabel apa pun. Namun, sebaiknya nama variabel disesuaikan dengan maksud atau arti dari variabel tersebut. Dikarenakan pada contoh di atas Anda akan menggunakan variabel untuk merepresentasikan urutan dari ***for loop***, penggunaan nama variabel **`nomor`** adalah yang paling sesuai untuk kebutuhan tersebut.

Anda telah membuat sebuah ***list*** mengenai makanan favorit Anda di bagian sebelumnya. Sekarang, Anda akan mempelajari bagaimana ***list*** dan ***for loop*** dapat dipadukan agar kode program Anda menjadi makin sinergis.

Sebagai contoh, Anda ingin menampilkan isi dari ***list*** Anda pada kolom **Console**, tetapi Anda ingin menampilkannya satu per satu, tidak sekaligus seperti pada bagian sebelumnya. Pertama-tama,

Anda harus memasukkan masing-masing isi dari ***list*** Anda tersebut ke dalam sebuah variabel, kita sebut saja namanya variabel **`makanan`**. Anda bisa saja memberi nama apa pun untuk variabel tersebut, tetapi agar konteksnya sesuai dengan isi dari variabelnya, yaitu makanan yang diambil dari ***list*** **`makanan_favorit`** Anda, penulis sarankan memberi nama yang sesuai. Kemudian, Anda gunakan ***for loop*** untuk menampilkan satu per satu isi dari ***list*** **makanan_favorit** Anda. Baris-baris kodenya akan menjadi seperti yang diperlihatkan pada `Kode Program 101`.

Kode Program 101

```
1  makanan_favorit = ('rendang', 'sate', 'nasi goreng')
2
3  for makanan in makanan_favorit:
4   print(makanan)
```

Ketika Anda menjalankan kode program tersebut, Anda dapat melihat pada kolom **Console** bahwa isi dari ***list*** **makanan_favorit** Anda akan ditampilkan satu per satu untuk masing-masing baris seperti yang diperlihatkan pada `Console 67`. Hal ini memungkinkan Anda untuk memproses masing-masing isi dari ***list*** sesuai dengan kebutuhan Anda di dalam program.

Console 67

```
rendang
sate
nasi goreng
>
```

Sekarang, penulis ingin memperlihatkan salah satu kekuatan lain dari ***for loop*** yang bisa Anda gunakan di dalam kode program yang Anda tulis. Misalnya, Anda ingin menghasilkan ratusan bilangan tanpa harus mengetikkannya satu per satu. Anda bisa menggunakan ***for loop***. Bayangkan apabila Anda melakukannya secara manual. Berapa

banyak waktu yang Anda butuhkan untuk melakukannya? Sementara itu, dengan menggunakan ***for loop***, Anda dapat melakukannya hanya dengan mengetikkan dua baris kode saja seperti yang diperlihatkan pada `Kode Program 102`. Ketika kode program tersebut dijalankan, kolom **Console** akan memperlihatkan tampilan seperti yang diperlihatkan pada `Console 68`. Pada gambar tersebut bilangan 3 sampai dengan 481 tidak ditampilkan (diwakilkan oleh tiga tanda titik) dikarenakan keterbatasan tampilan pada kolom **Console**. Namun, penulis yakin Anda sudah dapat memahami seperti apa ***for loop*** berjalan di dalam kode Anda.

Kode Program 102

```
1 for nomor in range(500):
2  print(nomor)
```

Console 68

```
0
1
2
...
497
498
499
>
```

Apabila Anda masih kurang yakin dengan kekuatan dari ***for loop*** di dalam kode program Anda, Anda bisa juga mencoba untuk memasukkan angka yang sangat besar di dalam baris-baris kode pada contoh di atas, misalkan 50.000. Memang mungkin program akan membutuhkan waktu yang lebih lama daripada ketika Anda menjalankannya untuk menghasilkan hanya 500 bilangan saja. Namun, apabila Anda melakukannya secara manual, penulis tidak yakin Anda dapat melakukannya dalam waktu yang sangat singkat. Belum lagi bisa saja ada bilangan yang Anda lewatkan selama melakukan proses

tersebut. Di komputer penulis, penulis hanya membutuhkan waktu sekitar lima detik untuk menjalankan program yang menghasilkan puluhan ribu bilangan tersebut.

Tantangan 10:
Penulis ingin Anda melakukan *Loop* sebanyak 20 kali. Setiap *Loop* yang Anda lakukan, Anda menampilkan bilangan kelipatan tiga di dalam kolom **Console** sehingga Anda akan menampilkan bilangan 3, 6, 9, dan seterusnya hingga 20 kali di dalam kolom **Console** Anda.

Contoh solusi Tantangan 10:

Kode Program 103

```
1 for nomor in range(20):
2   print((nomor + 1) * 3)
```

Console 69

```
3
6
9
...
54
57
60
>
```

Catatan: Ingat bahwa bahasa pemrograman Python menggunakan angka nol sebagai angka pertama ketika menggunakan fungsi **`range()`**. Itulah mengapa pada fungsi **`print()`**, penulis menambahkan dengan bilangan satu terlebih dahulu agar baris kode memulai dari angka satu. Selain itu, perlu diperhatikan bahwa di dalam matematika perkalian memiliki prioritas lebih tinggi dibandingkan penjumlahan sehingga di dalam baris kode perlu menyertakan tanda kurung agar proses penjumlahan dilakukan terlebih dahulu dibandingkan proses perkalian.

G. *Dictionary*

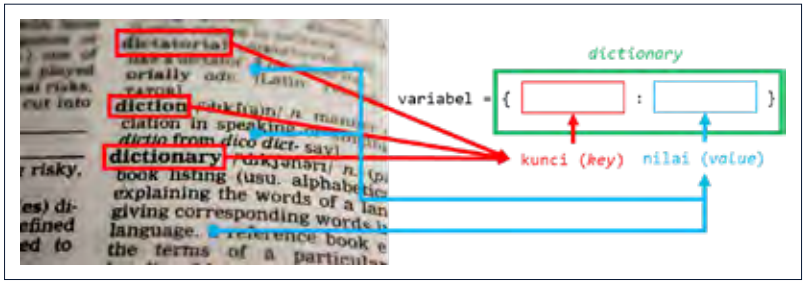
Sekarang Anda akan mempelajari tentang ***dictionary*** yang sebenarnya mirip dengan ***list*** dalam beberapa hal, tetapi juga memiliki beberapa perbedaan yang mendasar. ***Dictionary*** dan ***list*** diibaratkan bagaikan sepupu karena mereka saling berhubungan, tetapi tidak sama persis. ***Dictionary*** dalam bahasa Indonesia berarti kamus, yaitu sebuah buku, baik secara fisik maupun elektronik menyajikan berbagai macam kata beserta definisi dari kata-kata tersebut. Mengapa demikian? Jawabannya adalah karena ***dictionary*** memiliki penampilan mirip seperti kamus. Namun, di dalam ***dictionary*** digunakan istilah pasangan ***key*** dan ***value*** (atau dalam bahasa Indonesia berarti **kunci** dan **nilai**) seperti yang diperlihatkan pada Gambar 4.4. Jadi, ***key*** di dalam ***dictionary*** itu dapat dianalogikan dengan **kata** yang ingin Anda cari artinya di dalam kamus, sedangkan ***value*** di dalam ***dictionary*** itu dapat Anda analogikan dengan **definisi** atas kata yang Anda cari di dalam kamus.

Mari kita pelajari lebih mendalam melalui sebuah contoh agar lebih jelas lagi. Misalnya di dalam sebuah keluarga terdapat anggota-anggota keluarga dengan bulan kelahiran berbeda-beda. Buatlah sebuah variabel dengan nama **keluarga** dan untuk membuat sebuah ***dictionary*** di dalam bahasa pemrograman Python, Anda harus menggunakan kurung kurawal. Jadi, ketika Anda akan membuat sebuah ***list***, Anda menggunakan kurung siku. Sementara itu, ketika Anda akan membuat sebuah ***dictionary***, Anda menggunakan kurung kurawal dan di dalam kurung kurawal ini, Anda akan pertama-tama memberikan sebuah ***key*** (atau **kata** di dalam kamus). Kemudian barulah Anda memberikan ***value*** (atau **definisi** di dalam kamus). Langkah pertama dalam pembuatan ***dictionary*** tersebut diperlihatkan pada `Kode Program 104`.

`Kode Program 104`

```
1  keluarga = {}
```

Untuk variabel **keluarga** yang Anda buat, nama dari anggota keluarga akan menjadi ***key***-nya, sedangkan ***value***-nya adalah bulan kelahiran dari anggota keluarga tersebut. Misalnya untuk ***key*** pertama Anda gunakan nama Budhi sebagai anggota keluarga pertama (Anda bebas untuk menggunakan nama keluarga Anda sendiri) dan karena Budhi lahir di bulan Agustus, Anda menggunakan Agustus sebagai ***value*** pertama Anda. Pisahkan ***key*** dan ***value*** dengan menggunakan tanda titik dua. Sekarang, Anda memiliki satu isi di dalam ***dictionary*** Anda seperti yang diperlihatkan pada `Kode Program 105`. Jangan lupa untuk menggunakan tanda petik tunggal atau ganda untuk merepresentasikan teks yang digunakan sebagai ***key*** dan ***value*** tersebut.



Sumber: PDPics (2014)

Gambar 4.4 Perumpamaan Kunci (*Key*) dan Nilai (*Value*) pada Jenis Data *Dictionary* dengan Kata dan Definisi dalam Sebuah Kamus

Kode Program 105

```
1  keluarga = {'Budhi' : 'Agustus'}
```

Mari kita tambahkan isi yang kedua. Misalnya nama anggota keluarga kedua yang ingin Anda tambahkan adalah Ratih. Untuk memisahkan antara isi yang satu dengan yang lainnya di dalam ***dictionary***, Anda dapat menggunakan tanda baca koma. Penggunaan tanda baca koma ini sebenarnya serupa ketika Anda memisahkan isi-isi di dalam ***list***. Jadi, Anda memiliki Ratih sebagai ***key*** kedua kita,

kemudian Anda ketikkan tanda baca titik dua dan mengetikkan bulan kelahiran dari Ratih, misalkan bulan September. Baris kode yang Anda buat akan menjadi seperti yang diperlihatkan pada `Kode Program 106`.

Kode Program 106

```
1 keluarga = {'Budhi' : 'Agustus', 'Ratih' : 'September'}
```

Sekarang, Anda tambahkan isi yang ketiga. Anda berikan nama anggota keluarga yang lain, misalkan Pradya yang lahir di bulan November. Baris kodenya akan menjadi seperti yang diperlihatkan pada `Kode Program 107`.

Kode Program 107

```
1 keluarga = {'Budhi' : 'Agustus', 'Ratih' : 'September',
  'Pradya' : 'November'}
```

Dapat terlihat bahwa masing-masing isi dipisahkan oleh tanda baca koma dan pasangan ***key-value*** dipisahkan oleh tanda baca titik dua. Seperti apa isi dari ***dictionary*** tersebut ketika ditampilkan di kolom **Console**? Anda dapat menampilkannya dengan menggunakan fungsi **`print()`** sama ketika Anda menampilkan ***list*** seperti yang diperlihatkan pada `Kode Program 108`. Dari contoh tersebut dapat terlihat bahwa tampilan di dalam kolom **Console** yang diperlihatkan pada `Console 70` akan sama dengan tampilan ketika kita membuat ***dictionary*** tersebut.

Kode Program 108

```
1 keluarga = {'Budhi' : 'Agustus', 'Ratih' : 'September',
  'Pradya' : 'November'}
2
3 print(keluarga)
```

Console 70

```
{'Budhi' : 'Agustus', 'Ratih' : 'September', 'Pradya' :
'November'}
>
```

Dalam membuat sebuah ***dictionary*** yang memiliki isi yang banyak, Anda juga dapat menuliskan masing-masing anggota ***dictionary*** tersebut pada baris-baris terpisah untuk memudahkan pengisiannya seperti yang diperlihatkan pada `Kode Program 109` untuk memudahkan pembacaan kode programnya. Apabila kode program tersebut dijalankan, kolom **Console** akan menampilkan hasil yang sama seperti yang ditampilkan pada `Console 70`.

Kode Program 109

```
1  keluarga = {'Budhi' : 'Agustus',
2              'Ratih' : 'September',
3              'Pradya' : 'November'}
4
5  print(keluarga)
```

Anda ingin menampilkan isi tertentu dari sebuah ***dictionary***. Bagaimana caranya? Mirip seperti pada ***list***, Anda dapat menggunakan nama ***dictionary*** tersebut dan menggunakan tanda kurung siku setelahnya. Perbedaannya adalah ketika pada ***list***, Anda menggunakan urutan dari isi di dalam ***list*** tersebut, sedangkan pada ***dictionary***, Anda menggunakan nama dari ***key*** yang terdapat di dalam ***dictionary*** tersebut. Sebagai contoh, apabila Anda ingin mengetahui pada bulan apa Budhi dilahirkan, Anda mengetikkan **`keluarga['Budhi']`** dan untuk menampilkannya pada kolom **Console**, Anda menggunakan fungsi **`print()`** seperti yang diperlihatkan pada `Kode Program 110`. Dapat terlihat bahwa pada kolom ***Console*** ditampilkan teks "Agustus" yang merupakan bulan kelahiran dari Budhi seperti yang diberikan pada ***dictionary*** yang telah Anda buat.

Kode Program 110

```
1  keluarga = {'Budhi' : 'Agustus',
2              'Ratih' : 'September',
3              'Pradya' : 'November'}
4
5  print(keluarga['Budhi'])
```

Console 71

```
Agustus
>
```

Perlu diingat bahwa penulisan di dalam ***dictionary*** sensitif terhadap besar kecilnya huruf atau ejaannya. Jadi, apabila Anda salah mengetikkan nama 'budhi' alih-alih dari 'Budhi', Anda akan memperoleh pesan kesalahan pada kolom **Console** seperti yang diperlihatkan pada `Kode Program 111` dan `Console 72`.

Kode Program 111

main.py

```
1  keluarga = {'Budhi' : 'Agustus',
2              'Ratih' : 'September',
3              'Pradya' : 'November'}
4
5  print(keluarga['budhi'])
```

Console 72

Console

```
Traceback (most recent call last):
 File "main.py", line 5, in <module>
 print(keluarga['budhi'])
KeyError: 'budhi'
>
```

Pesan kesalahan tersebut memiliki arti bahwa program tidak menemukan kata 'budhi' sebagai salah satu ***key*** yang digunakan di dalam ***dictionary*** yang telah Anda buat sebelumnya. Jadi, perhatikan ejaan pada saat Anda membuat ***dictionary*** maupun pada saat Anda memanggil isi dari ***dictionary*** tersebut.

Pemanggilan isi dari ***dictionary*** juga bersifat satu arah. Artinya Anda dapat memanggil ***value*** dari sebuah isi ***dictionary*** yang telah Anda buat melalui ***key***, Anda tidak dapat memanggil ***key*** dari sebuah isi ***dictionary*** dengan menggunakan ***value***-nya. Dengan kata lain, pada contoh di atas, Anda dapat mencari bulan kelahiran dari salah satu anggota keluarga yang terdapat di dalam ***dictionary* keluarga** yang telah Anda buat sebelumnya, tetapi Anda tidak dapat mencari nama dari anggota keluarga yang lahir pada bulan November meskipun salah satu anggota keluarga (Pradya) lahir di bulan November. Hal ini dikarenakan nilai ***key*** yang berbeda dapat memiliki nilai ***value*** yang sama. Anda hanya dapat melakukan pencarian isi di dalam sebuah ***dictionary*** dengan menggunakan ***key***-nya.

Ada beberapa hal lagi yang dapat Anda lakukan dengan ***dictionary***. Misalnya Anda ingin menambahkan isi yang baru ke dalam ***dictionary*** yang telah Anda buat. Anda dapat melakukannya dengan mengetikkan nama ***dictionary*** yang ingin Anda tambahkan, memberikan tanda kurung siku, mengisi ***key*** baru ke dalam kurung siku tersebut, dan memberikan ***value*** ke dalamnya sebagaimana Anda memberikan nilai ke sebuah variabel. Contoh baris-baris kodenya diperlihatkan baris kedelapan dan kesembilan pada `Kode Program 112`.

Kode Program 112

```
1  keluarga = {'Budhi' : 'Agustus',
2              'Ratih' : 'September',
3              'Pradya' : 'November'}
4
5  print(keluarga)
6  print()
7
8  keluarga['Vrasya'] = 'Mei'
9  keluarga['Farsya'] = 'Oktober'
10
11 print(keluarga)
```

Pada Kode Program 112 terlihat ada dua anggota keluarga baru yang ditambahkan ke dalam ***dictionary*** **keluarga**, yaitu 'Vrasya' yang lahir pada bulan Mei dan 'Farsya' yang lahir pada bulan 'Oktober'. Ketika Anda menjalankan kode program tersebut, di kolom **Console** dapat terlihat bahwa dua anggota keluarga baru tersebut berhasil ditambahkan ke dalam ***dictionary*** **keluarga** seperti yang diperlihatkan pada Console 73.

Console 73

```
{'Budhi' : 'Agustus', 'Ratih' : 'September', 'Pradya' :
'November'}

{'Budhi' : 'Agustus', 'Ratih' : 'September', 'Pradya' :
'November', 'Vrasya' : 'Mei', 'Farsya' : 'Oktober'}
>
```

Apabila Anda ingin menghapus salah satu isi dari ***dictionary***, Anda dapat menggunakan fungsi **del()** sebagaimana yang Anda lakukan sebelumnya pada ***list***. Anda masukkan nama ***dictionary*** ke dalam tanda kurung di dalam fungsi **del()** tersebut, menambahkan tanda kurung siku setelah nama ***dictionary***-nya dan berikan ***key*** yang ingin Anda hapus di dalam tanda kurung siku tersebut. Misalkan Anda ingin mengeluarkan 'Ratih' dari ***dictionary*** **keluarga** yang telah Anda buat sebelumnya, Anda dapat mengetikkan **del(keluarga['Ratih'])** seperti yang diperlihatkan pada Kode Program 113.

Kode Program 113

```
1  keluarga = {'Budhi' : 'Agustus',
2              'Ratih' : 'September',
3              'Pradya' : 'November'}
4
5  print(keluarga)
6  print()
7
8  del(keluarga['Ratih'])
9
10 print(keluarga)
```

Console 74

```
{'Budhi' : 'Agustus', 'Ratih' : 'September', 'Pradya' :
'November'}

{'Budhi' : 'Agustus', 'Pradya' : 'November'}
>
```

Anda sudah mempelajari cara menambahkan dan menghapus isi dari sebuah ***dictionary***. Semuanya serupa dengan yang Anda pelajari di bagian ***list***. Apabila Anda ingin mengetahui jumlah dari sebuah ***dictionary***, Anda juga dapat melakukannya sebagaimana Anda melakukan pada ***list***. Anda dapat mengetikkan **`len(keluarga)`** untuk mengetahui jumlah anggota keluarga yang ada di dalam ***dictionary*** **`keluarga`** yang telah Anda buat seperti yang diperlihatkan pada `Kode Program 114`.

Kode Program 114

main.py

```
1  keluarga = {'Budhi' : 'Agustus',
2              'Ratih' : 'September',
3              'Pradya' : 'November'}
4
5  print(len(keluarga))
```

Console 75

```
3
>
```

Pada gambar tersebut dapat terlihat bahwa di kolom ***Console*** diperlihatkan angka tiga sebagai jumlah anggota keluarga. Angka ini telah Anda definisikan pada ***dictionary*** **`keluarga`** di atasnya.

Tantangan 11:
Buatlah sebuah ***dictionary*** dengan bilangan ***integer*** sebagai ***key***-nya dan ***boolean*** sebagai ***value***-nya yang menunjukkan bahwa bilangan tersebut adalah bilangan prima atau bukan.

Contoh solusi Tantangan 11:

Kode Program 115

```
1  Bilangan prima = {2 : True, 3 : True, 4 : False, 5 : True,
2                    6 : False, 7 : True, 18 : False, 28 :
3                    False, 29 : True}
```

Catatan: Bilangan prima adalah bilangan yang hanya dapat dibagi oleh 1 dan bilangan itu sendiri. Untuk mempermudah pembacaan ***dictionary***, penulis menyarankan Anda untuk memberikan spasi di antara isi yang satu dengan isi yang lainnya.

Proyek 2: Penganalisis Teks

Bab 5

Selamat! Anda telah mempelajari cukup bahasa pemrograman Python untuk membuat proyek lainnya. Pada bab ini, Anda akan membuat sebuah program yang dapat menghitung jumlah kata yang terdapat di dalam sebuah teks sebagai salah satu contoh analisis teks yang sering dilakukan. Idenya adalah Anda akan membaca sebuah teks yang panjang (bisa dari sebuah pidato, sebuah lagu, atau bahkan sebuah buku) dan program yang Anda buat nanti akan memiliki kemampuan untuk menghitung jumlah kata yang terdapat di dalam teks tersebut. Program Anda juga akan dapat menghitung berapa kali masing-masing kata muncul di dalam teks tersebut.

Pada bagian awal bab ini, Anda akan mempelajari satu konsep tentang cara mengubah teks menjadi sebuah ***list*** sehingga Anda akan dapat menjalankan fungsi ***for loop*** dan melakukan analisis terhadap isi ***list*** tersebut yang merupakan kata-kata yang terdapat di dalam teks yang ingin Anda analisis. Untuk contoh teks pada bab ini, penulis akan menggunakan lima paragraf pertama dari karya tulis berjudul

"Metaverse, membangun kehidupan dalam dunia virtual: menjanjikan tapi juga potensial bermasalah" (Rastati, 2022). Tentunya Anda dapat memilih teks Anda sendiri untuk dianalisis.

A. Mengubah Teks Menjadi *List*

Buatlah sebuah variabel bernama **`teks`** dan masukkan teks Anda ke dalam variabel tersebut seperti yang diperlihatkan pada `Kode Program 116`. Jangan lupa untuk menggunakan tanda kutip tiga kali (baik itu kutip tunggal maupun kutip ganda) untuk mengapit teks panjang yang ingin Anda masukkan ke dalam variabel **`teks`** tersebut sehingga teks akan ditampilkan "apa adanya" sebagaimana Anda tuliskan di dalam variabel seperti yang diperlihatkan pada `Console 76`.

Kode Program 116

```
1  teks = '''
2  Pandemi COVID-19 telah memaksa dan menjadi katalisator
   penting dalam mengubah secara radikal aktivitas hidup
   manusia dari offline ke online. Sistem kerja dari rumah,
   webinar, dan rapat online yang dulu sulit terjadi, kini
   menjadi gaya hidup yang lazim.
3
4  Hampir semua aktivitas tatap muka seperti rapat, sekolah,
   dan hiburan berubah menjadi online. Misalnya, film-film
   box office yang dulu tayang eksklusif di bioskop, kini
   beralih perilisannya secara streaming.
5
6  Dengan kondisi pandemi yang belum menunjukkan tanda
   berakhir, metaverse dapat menjadi solusi ampuh untuk
   alternatif kehidupan selain di dunia fisik. Metaverse
   diciptakan dengan memadukan berbagai unsur teknologi
   seperti konferensi video, media sosial, hiburan, game,
   pendidikan, pekerjaan, dan mata uang kripto.
7
8  Keunggulan inilah yang akan membuat metaverse tampil
   sebagai revolusi internet.
```

```
 9
10  Sebuah riset menunjukkan metaverse memberikan manfaat
    signifikan dalam kehidupan manusia seperti menyediakan
    ruang tanpa batas kegiatan kampanye kepresidenan 2020
    Joe Biden di Animal Crossing Nintendo, wisuda mahasiswa
    UC Berkeley di Minecraft, dan perpustakaan virtual yang
    dikembangkan Standford University di Second Life.
11  '''
12
13  print(teks)
14
```

Console 76

```
Pandemi COVID-19 telah memaksa dan menjadi katalisator
penting dalam mengubah secara radikal aktivitas hidup manusia
dari offline ke online. Sistem kerja dari rumah, webinar, dan
rapat online yang dulu sulit terjadi, kini menjadi gaya
hidup yang lazim.

Hampir semua aktivitas tatap muka seperti rapat, sekolah,
dan hiburan berubah menjadi online. Misalnya, film-film box
office yang dulu tayang eksklusif di bioskop, kini beralih
perilisannya secara streaming.

Dengan kondisi pandemi yang belum menunjukkan tanda berakhir,
metaverse dapat menjadi solusi ampuh untuk alternatif
kehidupan selain di dunia fisik. Metaverse diciptakan dengan
memadukan berbagai unsur teknologi seperti konferensi video,
media sosial, hiburan, game, pendidikan, pekerjaan, dan mata
uang kripto.

Keunggulan inilah yang akan membuat metaverse tampil sebagai
revolusi internet.

Sebuah riset menunjukkan metaverse memberikan manfaat
signifikan dalam kehidupan manusia seperti menyediakan ruang
tanpa batas kegiatan kampanye kepresidenan 2020 Joe Biden
di Animal Crossing Nintendo, wisuda mahasiswa UC Berkeley
di Minecraft, dan perpustakaan virtual yang dikembangkan
Standford University di Second Life.
>
```

Apabila Anda ingin melakukan analisis seperti menghitung jumlah kata yang ada di dalam teks tersebut, Anda harus memecah teks tersebut menjadi bagian-bagian yang lebih kecil, yaitu kata-kata itu sendiri. Mungkin Anda berpikir kenapa tidak menggunakan fungsi **`len()`** yang telah dipelajari sebelumnya? Bukankah fungsi tersebut juga melakukan perhitungan dari jumlah isi di dalam sebuah ***list***? Apabila kita menggunakan fungsi **`len()`** tanpa memecah teks tersebut menjadi sebuah ***list***, yang akan ditampilkan bukanlah jumlah kata yang ada di dalam teks, melainkan jumlah karakter yang ada di dalam teks tersebut. Contoh penggunaan fungsi **`len()`** secara langsung pada teks di atas (tanpa mengubahnya menjadi ***list*** terlebih dahulu) diperlihatkan pada `Kode Program 117`. Pada kolom **Console** terlihat bahwa jumlah yang ditampilkan adalah 1.183 seperti yang diperlihatkan pada `Console 77`. Jumlah tersebut adalah jumlah karakter yang berada di dalam teks tersebut, bukan jumlah kata yang terdapat di dalam teks tersebut (Anda dapat menghitungnya apabila Anda kurang yakin dengan hal tersebut).

Kode Program 117

```
1   teks = '''
2   Pandemi COVID-19 telah memaksa dan menjadi katalisator
    penting dalam mengubah secara radikal aktivitas hidup
    manusia dari offline ke online. Sistem kerja dari rumah,
    webinar, dan rapat online yang dulu sulit terjadi, kini
    menjadi gaya hidup yang lazim.
.   .
.   .
.   .
14  print(len(teks))
```

Console 77

```
1183
>
```

Untuk memecah sebuah teks menjadi sebuah ***list*** yang berisi kata-kata dari teks tersebut, Anda dapat menggunakan fungsi **`split()`**.

Cara penggunaannya mudah, Anda ketikkan variabel yang berisi teks yang ingin Anda ubah menjadi ***list***, kemudian berikan tanda titik, lalu ketikkan **`split()`** seperti yang diperlihatkan pada `Kode Program 118`. Teks Anda pun akan diubah menjadi sebuah ***list*** yang berisi kata-kata yang berasal dari teks Anda seperti yang diperlihatkan pada `Console 78`. Perhatikan bahwa pada kolom **Console** menampilkan keluaran yang diawali dengan kurung siku dan diakhiri dengan kurung siku juga yang merupakan salah satu ciri ***list*** di dalam bahasa pemrograman Python.

Kode Program 118

```
1   teks = '''
2   Pandemi COVID-19 telah memaksa dan menjadi katalisator
    penting dalam mengubah secara radikal aktivitas hidup
    manusia dari offline ke online. Sistem kerja dari rumah,
    webinar, dan rapat online yang dulu sulit terjadi, kini
    menjadi gaya hidup yang lazim.
.   .
.   .
.   .
14  teks_dalam_list = teks.split()
15
16  print(teks_dalam_list)
```

Console 78

```
['Pandemi', 'COVID-19', 'telah', 'memaksa', 'dan', 'menjadi',
'katalisator', 'penting', 'dalam', 'mengubah', 'secara',
'radikal', 'aktivitas', 'hidup', 'manusia', 'dari', 'offline',
'ke', 'online.', 'Sistem', 'kerja', 'dari', 'rumah,',
'webinar,', 'dan', 'rapat', 'online', 'yang', 'dulu',
'sulit', 'terjadi,', 'kini', 'menjadi', 'gaya', 'hidup',
'yang', 'lazim.', 'Hampir', 'semua', 'aktivitas', 'tatap',
'muka', 'seperti', 'rapat,', 'sekolah,', 'dan', 'hiburan',
'berubah', 'menjadi', 'online.', 'Misalnya,', 'film-film',
'box', 'office', 'yang', 'dulu', 'tayang', 'eksklusif', 'di',
'bioskop,', 'kini', 'beralih', 'perilisannya', 'secara',
'streaming.', 'Dengan', 'kondisi', 'pandemi', 'yang',
'belum', 'menunjukkan', 'tanda', 'berakhir,', 'metaverse',
'dapat', 'menjadi', 'solusi', 'ampuh', 'untuk', 'alternatif',
```

```
'kehidupan', 'selain', 'di', 'dunia', 'fisik.', 'Metaverse',
'diciptakan', 'dengan', 'memadukan', 'berbagai', 'unsur',
'teknologi', 'seperti', 'konferensi', 'video,', 'media',
'sosial,', 'hiburan,', 'game,', 'pendidikan,', 'pekerjaan,',
'dan', 'mata', 'uang', 'kripto.', 'Keunggulan', 'inilah',
'yang', 'akan', 'membuat', 'metaverse', 'tampil', 'sebagai',
'revolusi', 'internet.', 'Sebuah', 'riset', 'menunjukkan',
'metaverse', 'memberikan', 'manfaat', 'signifikan', 'dalam',
'kehidupan', 'manusia', 'seperti', 'menyediakan', 'ruang',
'tanpa', 'batas', 'kegiatan', 'kampanye', 'kepresidenan',
'2020', 'Joe', 'Biden', 'di', 'Animal', 'Crossing',
'Nintendo,', 'wisuda', 'mahasiswa', 'UC', 'Berkeley', 'di',
'Minecraft,', 'dan', 'perpustakaan', 'virtual', 'yang',
'dikembangkan', 'Standford', 'University', 'di', 'Second',
'Life.']
>
```

Sekarang apabila Anda menggunakan fungsi **`len()`** pada ***list*** tersebut, Anda akan memperoleh jumlah kata yang terdapat di dalam teks tersebut karena fungsi **`len()`** akan menghitung jumlah isi di dalam ***list*** yang tiada lain merupakan kata-kata yang berasal dari teks tersebut. Hal tersebut seperti yang diperlihatkan pada `Kode Program 119` dan `Console 79`. Dapat terlihat di kolom **Console** bahwa terdapat 156 kata di dalam teks tersebut.

Kode Program 119

```
 1  teks = '''
 2  Pandemi COVID-19 telah memaksa dan menjadi katalisator
    penting dalam mengubah secara radikal aktivitas hidup
    manusia dari offline ke online. Sistem kerja dari rumah,
    webinar, dan rapat online yang dulu sulit terjadi, kini
    menjadi gaya hidup yang lazim.
 .  .
 .  .
 .  .
14  teks_dalam_list = teks.split()
15
16  print(len(teks_dalam_list))
```

Console 79

Sampai saat ini, Anda telah belajar bagaimana caranya untuk mengubah teks menjadi ***list***. Anda juga telah mempelajari cara menampilkan teks apa adanya di dalam sebuah variabel. Selain itu, Anda juga telah mempelajari cara menghitung isi dari ***list*** sebagai jembatan untuk menghitung jumlah kata yang ada di dalam sebuah teks. Langkah selanjutnya adalah Anda akan menghitung seberapa sering masing-masing kata tersebut muncul di dalam teks.

B. Menghitung Kemunculan (Frekuensi) Kata di Dalam *List*

Sampai dengan saat ini, kode program yang telah Anda buat telah dapat menghitung seberapa banyak jumlah kata di dalam sebuah teks. Namun, kode program tersebut belum dapat menghitung seberapa banyak sebuah kata muncul di dalam sebuah teks. Untuk dapat melakukannya, mari kita sederhanakan terlebih dahulu teks yang Anda gunakan dalam bagian sebelumnya menjadi beberapa kata saja. Anda akan menggunakan kata-kata **`apel belimbing ceri durian apel belimbing belimbing ceri durian durian durian`** seperti yang diperlihatkan pada `Kode Program 120`. Perhatikan bahwa ada beberapa kata yang diulang di dalam teks tersebut.

Kode Program 120

```
1  teks = '''
2  apel belimbing ceri durian apel belimbing belimbing
   ceri durian durian durian
3  '''
```

Untuk menghitung jumlah karakter di dalam teks yang disederhanakan tersebut, Anda dapat menggunakan fungsi **`len()`** dan untuk menampilkan kata-kata di dalam teks tersebut dalam bentuk ***list***, Anda juga dapat menggunakan fungsi **`.split()`** seperti yang dijelaskan pada bagian sebelumnya. Penulis perlihatkan kembali kode programnya sebagaimana ditampilkan pada `Kode Program 121` dan `Console 80`. Dari contoh tersebut dapat terlihat bahwa setelah baris-baris kode tersebut dijalankan, pada kolom **Console** akan diperlihatkan kata-kata di dalam teks yang sudah diubah menjadi bentuk ***list*** beserta hasil perhitungan jumlah kata di dalam teks tersebut, yaitu sebelas.

Kode Program 121

```
1   teks = '''
2   apel belimbing ceri durian apel belimbing belimbing
    ceri durian durian durian
3   '''
4
5   teks_dalam_list = teks.split()
6
7   print(teks_dalam_list)
8   print()
9
10  jumlah_kata = len(teks_dalam_list)
11
12  print(f'Jumlah kata di dalam teks: {jumlah_kata}')
```

Console 80

```
['apel', 'belimbing', 'ceri', 'durian', 'apel', 'belimbing',
'belimbing', 'ceri', 'durian', 'durian', 'durian']

Jumlah kata di dalam teks: 11
>
```

Lalu, bagaimana cara agar Anda dapat menghitung jumlah masing-masing kata atau frekuensi kemunculan kata-kata tersebut?

Anda dapat menggunakan ***dictionary***. Sebagaimana telah Anda ketahui, ***dictionary*** terdiri atas dua bagian, yaitu ***key*** dan ***value***. Anda dapat menyimpan nama kata yang ingin Anda hitung sebagai ***key*** dan jumlah kemunculan kata tersebut sebagai ***value*** dari ***dictionary*** tersebut.

Pertama-tama, buatlah sebuah ***dictionary*** kosong. Anda dapat memberinya nama **`jumlah_per_kata`** seperti yang diperlihatkan pada `Kode Program 122`. Jangan lupa gunakan kurung kurawal untuk membuat sebuah ***dictionary***.

Kode Program 122

```
1  teks = '''
2  apel belimbing ceri durian apel belimbing belimbing
   ceri durian durian durian
3  '''
4
5  teks_dalam_list = teks.split()
6
7  print(teks_dalam_list)
8  print()
9
10 jumlah_kata = len(teks_dalam_list)
11
12 print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14 jumlah_per_kata = {}
```

Selanjutnya, ingat bahwa isi dari variabel **`teks_dalam_list`** adalah kata-kata yang berada di dalam teks karena telah menggunakan fungsi **`teks.split()`** seperti yang diperlihatkan pada baris kelima pada `Kode Program 122`. Jadi, Anda dapat menggunakan ***for loop*** untuk mengakses masing-masing dari kata-kata tersebut dan menampilkannya seperti yang diperlihatkan pada `Kode Program 123` dan `Console 81`.

Kode Program 123

```
1  teks = '''
2  apel belimbing ceri durian apel belimbing belimbing
   ceri durian durian durian
3  '''
4
5  teks_dalam_list = teks.split()
6
7  print(teks_dalam_list)
8  print()
9
10 jumlah_kata = len(teks_dalam_list)
11
12 print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14 jumlah_per_kata = {}
15
16 for kata in teks_dalam_list:
17  print(kata)
```

Console 81

```
['apel', 'belimbing', 'ceri', 'durian', 'apel', 'belimbing',
'belimbing', 'ceri', 'durian', 'durian', 'durian']

Jumlah kata di dalam teks: 11
apel
belimbing
ceri
durian
apel
belimbing
belimbing
ceri
durian
durian
durian
>
```

Sekarang yang harus Anda lakukan adalah menjadikan kata-kata tersebut sebagai ***key*** dari ***dictionary*** yang telah dibuat sebelumnya, yaitu **`jumlah_per_kata`**. Caranya adalah dengan mengubah fungsi

`print(kata)` yang ada di dalam ***for loop*** dengan **`jumlah_per_kata[kata] = 1`** seperti yang diperlihatkan pada `Kode Program 124` dan `Console 82`. Pada baris-baris kode tersebut juga diperlihatkan fungsi **`print(jumlah_per_kata)`** untuk menampilkan seperti apa ***dictionary*** **`jumlah_per_kata`** yang awalnya kosong menjadi berisi kata-kata yang berasal dari ***list*** **`teks_dalam_list`**. Namun, jumlah kemunculan masing-masing kata di dalam teks masih belum sesuai dikarenakan berapa kali pun sebuah kata muncul, Anda baru memberikan nilai 1 untuk kata-kata tersebut.

Kode Program 124

```
1   teks = '''
2   apel belimbing ceri durian apel belimbing belimbing ceri
    durian durian durian
3   '''
4
5   teks_dalam_list = teks.split()
6
7   print(teks_dalam_list)
8   print()
9
10  jumlah_kata = len(teks_dalam_list)
11
12  print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14  jumlah_per_kata = {}
15
16  for kata in teks_dalam_list:
17   jumlah_per_kata[kata] = 1
18
19  print(jumlah_per_kata)
```

Console 82

```
['apel', 'belimbing', 'ceri', 'durian', 'apel', 'belimbing',
'belimbing', 'ceri', 'durian', 'durian', 'durian']

Jumlah kata di dalam teks: 11
{'apel': 1, 'belimbing': 1, 'ceri': 1, 'durian': 1}
> _
```

Lalu, bagaimana caranya agar jumlah tersebut menjadi sesuai? Di dalam ***for loop*** tersebut, Anda harus membuat baris-baris kode yang dijalankan dapat disesuaikan dengan isi dari ***dictionary*** dan masing-masing kata yang menjadi perhatian. Apabila kata yang menjadi perhatian belum tersedia di dalam ***dictionary***, Anda akan menambahkannya ke dalam ***dictionary*** tersebut dan memberikan nilai 1 sebagai kemunculan pertama dari kata tersebut seperti yang diperlihatkan baris-baris pada `Kode Program 124`. Namun, apabila kata yang menjadi perhatian telah ada di dalam ***dictionary*** tersebut, Anda cukup menambahkan nilai sebanyak 1 ke dalam kata yang telah ada di dalam ***dictionary*** tersebut. Untuk kondisi seperti ini, Anda dapat menggunakan ***`if statement`*** seperti yang diperlihatkan pada `Kode Program 125` dan `Console 83`.

Kode Program 125

```
1   teks = '''
2   apel belimbing ceri durian apel belimbing belimbing
    ceri durian durian durian
3   '''
4
5   teks_dalam_list = teks.split()
6
7   print(teks_dalam_list)
8   print()
9
10  jumlah_kata = len(teks_dalam_list)
11
12  print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14  jumlah_per_kata = {}
15
16  for kata in teks_dalam_list:
17   if kata not in jumlah_per_kata:
18     jumlah_per_kata[kata] = 1
19   else:
20     jumlah_per_kata[kata] = jumlah_per_kata[kata] + 1
21
22  print(jumlah_per_kata)
```

Console 83

```
['apel', 'belimbing', 'ceri', 'durian', 'apel', 'belimbing',
'belimbing', 'ceri', 'durian', 'durian', 'durian']

Jumlah kata di dalam teks: 11
{'apel': 2, 'belimbing': 3, 'ceri': 2, 'durian': 4}
>
```

Pada baris ke-17 Kode Program 125, Anda dapat melihat bahwa ketika kata di dalam teks belum ada di dalam ***dictionary*** **jumlah_per_kata**, kata akan dimasukkan menjadi ***key*** baru dari ***dictionary*** tersebut dan diberi nilai 1 sebagai kemunculan pertamanya di baris ke-18. Sementara itu, baris ke-19 (***else statement***) menunjukkan bahwa apabila kata sudah ada di dalam ***dictionary*** **jumlah_per_kata**, nilai dari ***dictionary*** yang menggunakan kata sebagai ***key***-nya akan dinaikkan menjadi 1 di baris ke-20. Jangan lupa bahwa posisi ***if statement*** dan ***else*** harus diketikkan secara sejajar agar menjadi satu kesatuan.

Di dalam bahasa pemrograman Python, baris kode seperti **jumlah_per_kata[kata] = jumlah_per_kata[kata] + 1** sebagaimana diperlihatkan pada baris ke-18 Kode Program 125 tersebut dapat diperingkas menjadi **jumlah_per_kata[kata] += 1** seperti yang diperlihatkan pada Kode Program 126.

Kode Program 126

```
1   teks = '''
2   apel belimbing ceri durian apel belimbing belimbing
    ceri durian durian durian
3   '''
4
5   teks_dalam_list = teks.split()
6
7   print(teks_dalam_list)
8   print()
9
10  jumlah_kata = len(teks_dalam_list)
```

```
11
12  print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14  jumlah_per_kata = {}
15
16  for kata in teks_dalam_list:
17   if kata not in jumlah_per_kata:
18     jumlah_per_kata[kata] = 1
19   else:
20     jumlah_per_kata[kata] += 1
21
22  print(jumlah_per_kata)
```

Keduanya memiliki arti yang sama, yaitu tambahkan nilai 1 pada nilai yang ada sebelumnya. Hal tersebut dapat terlihat bahwa ketika baris kode diubah dan program dijalankan, hasil yang ditampilkan pada kolom **Console** tetap sama seperti yang diperlihatkan pada `Console 83`.

Kini kode program Anda pun dapat menghitung jumlah kata sesuai dengan kemunculannya di dalam teks. Pada contoh di atas dapat terlihat bahwa kata **`apel`** muncul dua kali di dalam teks, yaitu di urutan pertama dan kelima. Kata **`belimbing`** muncul tiga kali di dalam teks yaitu di urutan kedua, keenam, dan ketujuh. Kata **`ceri`** muncul dua kali di dalam teks dan **`durian`** muncul empat kali di dalam teks. Apabila Anda menghitungnya secara manual, dapat dipastikan jumlah tersebut telah sesuai.

Satu hal penting yang ingin penulis tekankan adalah bahwa kode program akan membaca berdasarkan besar kecilnya huruf yang Anda gunakan di dalam teks. Jadi, **`apel`** akan berbeda dengan **`Apel`** atau **`APEL`**. Dengan demikian, apabila terdapat kata-kata yang berbeda tersebut di dalam teks, masing-masing kata tersebut akan menjadi ***key*** yang berbeda di dalam ***dictionary***-nya. Misal, ketika Anda mengubah kata **`apel`** kedua dari teks yang digunakan sebelumnya menjadi **`Apel`** seperti yang diperlihatkan pada `Kode Program 127`, kode program akan membaca bahwa **`Apel`** merupakan kata yang berbeda dari

apel. Hal ini terlihat pada kolom **Console** di mana terdapat anggota ***dictionary*** baru, yaitu **Apel** dengan nilai kemunculan satu seperti yang diperlihatkan pada Console 84. Lalu, **apel** yang sebelumnya berjumlah dua, kini menjadi bernilai satu dikarenakan nilai yang satu lagi telah digunakan oleh **Apel**.

Kode Program 127

```
1   teks = '''
2   apel belimbing ceri durian Apel belimbing belimbing
    ceri durian durian durian
3   '''
4
5   teks_dalam_list = teks.split()
6
7   print(teks_dalam_list)
8   print()
9
10  jumlah_kata = len(teks_dalam_list)
11
12  print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14  jumlah_per_kata = {}
15
16  for kata in teks_dalam_list:
17   if kata not in jumlah_per_kata:
18     jumlah_per_kata[kata] = 1
19   else:
20     jumlah_per_kata[kata] += 1
21
22  print(jumlah_per_kata)
```

Console 84

```
['apel', 'belimbing', 'ceri', 'durian', 'apel', 'belimbing',
'belimbing', 'ceri', 'durian', 'durian', 'durian']

Jumlah kata di dalam teks: 11
{'apel': 1, 'belimbing': 3, 'ceri': 2, 'durian': 4, 'Apel': 1}
>
```

Dalam kenyataannya, untuk program penghitung jumlah kata seperti yang sedang Anda buat tidak akan memedulikan besar kecilnya huruf yang membentuk sebuah kata yang sama. Yang paling penting adalah seberapa sering kata tersebut muncul di dalam teks sehingga Anda dapat memperbarui baris-baris kode program Anda dengan menjadikan setiap kata yang diperhitungkan di dalam ***for loop*** menjadi huruf kecil semuanya atau huruf besar semuanya. Hal tersebut dapat dilakukan dengan menggunakan fungsi **`.lower()`** (untuk menjadikan seluruh kata di dalam teks menjadi huruf kecil) sebelum teks tersebut diberikan fungsi **`.split()`** seperti yang diperlihatkan baris kelima pada `Kode Program 128`. Dapat terlihat pada kolom **Console** bahwa jumlah kemunculan kata **`apel`** menjadi dua kali karena kata **`Apel`** diubah terlebih dahulu menjadi kata **`apel`** sebelum kemudian dilakukan perhitungan jumlah kemunculan masing-masing kata di dalam ***for loop*** seperti yang diperlihatkan pada `Console 85`.

Kode Program 128

```
1  teks = '''
2  apel belimbing ceri durian Apel belimbing belimbing
   ceri durian durian durian
3  '''
4
5  teks_dalam_list = teks.lower().split()
6
7  print(teks_dalam_list)
8  print()
9
10 jumlah_kata = len(teks_dalam_list)
11
12 print(f'Jumlah kata di dalam teks: {jumlah_kata}')
13
14 jumlah_per_kata = {}
15
16 for kata in teks_dalam_list:
17  if kata not in jumlah_per_kata:
```

```
 if kata not in jumlah_per_kata:
  jumlah_per_kata[kata] = 1
 else:
  jumlah_per_kata[kata] += 1
print(jumlah_per_kata)
```

Console 85

```
['apel', 'belimbing', 'ceri', 'durian', 'apel', 'belimbing',
'belimbing', 'ceri', 'durian', 'durian', 'durian']

Jumlah kata di dalam teks: 11
{'apel': 2, 'belimbing': 3, 'ceri': 2, 'durian': 4}
>
```

Sekarang mari kita masukkan teks dari tulisan tentang metaverse yang digunakan pada bagian sebelumnya ke dalam baris-baris kode program yang telah Anda buat seperti yang diperlihatkan pada Kode Program 129. Setelah Anda menjalankan program tersebut, Anda akan diberikan tampilan kolom **Console** seperti yang diperlihatkan pada Console 86.

Kode Program 129

```
teks = '''
Pandemi COVID-19 telah memaksa dan menjadi katalisator
penting dalam mengubah secara radikal aktivitas hidup
manusia dari offline ke online. Sistem kerja dari rumah,
webinar, dan rapat online yang dulu sulit terjadi, kini
menjadi gaya hidup yang lazim.

Hampir semua aktivitas tatap muka seperti rapat, sekolah,
dan hiburan berubah menjadi online. Misalnya, film-film box
office yang dulu tayang eksklusif di bioskop, kini beralih
perilisannya secara streaming.

Dengan kondisi pandemi yang belum menunjukkan tanda
berakhir, metaverse dapat menjadi solusi ampuh untuk
alternatif kehidupan selain di dunia fisik. Metaverse
diciptakan dengan memadukan berbagai unsur teknologi
seperti konferensi video, media sosial, hiburan, game,
pendidikan, pekerjaan, dan mata uang kripto.
```

```
7
8  Keunggulan inilah yang akan membuat metaverse tampil
   sebagai revolusi internet.
9
10 Sebuah riset menunjukkan metaverse memberikan manfaat
   signifikan dalam kehidupan manusia seperti menyediakan
   ruang tanpa batas kegiatan kampanye kepresidenan 2020
   Joe Biden di Animal Crossing Nintendo, wisuda mahasiswa
   UC Berkeley di Minecraft, dan perpustakaan virtual yang
   dikembangkan Standford University di Second Life.
   '''
11
12 teks_dalam_list = teks.lower().split()
13 jumlah_kata = len(teks_dalam_list)
14
15 print(f'Jumlah kata di dalam teks: {jumlah_kata}')
16 print()
17
18 jumlah_per_kata = {}
19
20 for kata in teks_dalam_list:
21  if kata not in jumlah_per_kata:
22    jumlah_per_kata[kata] = 1
23  else:
24    jumlah_per_kata[kata] = jumlah_per_kata[kata] + 1
25
26 print(jumlah_per_kata)
```

Console 86

```
Jumlah kata di dalam teks: 156

{'pandemi': 2, 'covid-19': 1, 'telah': 1, 'memaksa': 1,
'dan': 5, 'menjadi': 4, 'katalisator': 1, 'penting': 1,
'dalam': 2, 'mengubah': 1, 'secara': 2, 'radikal': 1,
'aktivitas': 2, 'hidup': 2, 'manusia': 2, 'dari': 2,
'offline': 1, 'ke': 1, 'online.': 2, 'sistem': 1, 'kerja':
1, 'rumah,': 1, 'webinar,': 1, 'rapat': 1, 'online': 1,
'yang': 6, 'dulu': 2, 'sulit': 1, 'terjadi,': 1, 'kini': 2,
'gaya': 1, 'lazim.': 1, 'hampir': 1, 'semua': 1, 'tatap':
1, 'muka': 1, 'seperti': 3, 'rapat,': 1, 'sekolah,': 1,
'hiburan': 1, 'berubah': 1, 'misalnya,': 1, 'film-film':
```

```
1, 'box': 1, 'office': 1, 'tayang': 1, 'eksklusif': 1,
'di': 5, 'bioskop,': 1, 'beralih': 1, 'perilisannya': 1,
'streaming.': 1, 'dengan': 2, 'kondisi': 1, 'belum': 1,
'menunjukkan': 2, 'tanda': 1, 'berakhir,': 1, 'metaverse':
4, 'dapat': 1, 'solusi': 1, 'ampuh': 1, 'untuk': 1,
'alternatif': 1, 'kehidupan': 2, 'selain': 1, 'dunia': 1,
'fisik.': 1, 'diciptakan': 1, 'memadukan': 1, 'berbagai':
1, 'unsur': 1, 'teknologi': 1, 'konferensi': 1, 'video,':
1, 'media': 1, 'sosial,': 1, 'hiburan,': 1, 'game,': 1,
'pendidikan,': 1, 'pekerjaan,': 1, 'mata': 1, 'uang': 1,
'kripto.': 1, 'keunggulan': 1, 'inilah': 1, 'akan': 1,
'membuat': 1, 'tampil': 1, 'sebagai': 1, 'revolusi': 1,
'internet.': 1, 'sebuah': 1, 'riset': 1, 'memberikan': 1,
'manfaat': 1, 'signifikan': 1, 'menyediakan': 1, 'ruang':
1, 'tanpa': 1, 'batas': 1, 'kegiatan': 1, 'kampanye':
1, 'kepresidenan': 1, '2020': 1, 'joe': 1, 'biden': 1,
'animal': 1, 'crossing': 1, 'nintendo,': 1, 'wisuda': 1,
'mahasiswa': 1, 'uc': 1, 'berkeley': 1, 'minecraft,':
1, 'perpustakaan': 1, 'virtual': 1, 'dikembangkan': 1,
'standford': 1, 'university': 1, 'second': 1, 'life.': 1}
>
```

Pada contoh tersebut dapat terlihat bahwa terdapat 156 kata di dalam teks yang digunakan di dalam program. Pada bagian bawah kolom **Console** tersebut dapat terlihat bahwa kata-kata yang digunakan di dalam teks akan ditampilkan beserta jumlah kemunculannya di dalam teks. Untuk kata **`metaverse,`** misalnya, digunakan sebanyak empat kali di dalam teks tersebut.

Selamat! Anda telah berhasil menyelesaikan proyek yang kedua.

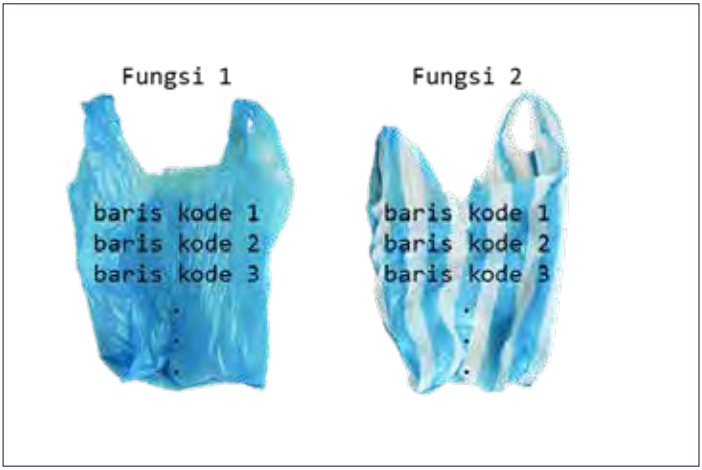
Bab 6 Fungsi

Selamat! Sekarang Anda telah berhasil menyelesaikan proyek pembuatan program dengan menggunakan bahasa pemrograman Python kedua Anda dan Anda akan kembali belajar konsep-konsep yang baru agar program Anda menjadi makin berkembang. Anda akan mengerjakan satu proyek pemrograman lagi di dalam buku ini untuk menyelesaikan keseluruhan pelatihan. Jadi, mari kita pelajari konsep dan keterampilan yang Anda butuhkan untuk mengerjakannya.

Pada bab ini, Anda akan belajar mengenai fungsi. Anda akan belajar bagaimana cara membuat fungsi dan bagaimana cara memanggil fungsi yang telah Anda buat tersebut di dalam program. Anda juga akan belajar cara memberi masukan ke dalam sebuah fungsi yang Anda buat melalui fitur yang disebut dengan parameter. Terakhir, Anda akan belajar mengenai cara mengeluarkan informasi yang telah diproses oleh sebuah fungsi melalui fitur yang disebut dengan *return*. Mari kita bahas satu per satu terkait hal-hal tersebut.

A. Fungsi

Sederhananya, fungsi merupakan sebuah cara untuk memberi nama pada satu atau beberapa baris kode yang terdapat di dalam program Anda. Fungsi dapat diumpamakan sebagai wadah yang menyimpan satu atau beberapa baris kode seperti yang diperlihatkan pada Gambar 6.1 sehingga ketika Anda akan menggunakan baris-baris kode tersebut di tempat lain di program Anda, Anda tidak perlu mengetikkan ulang baris-baris kode tersebut. Anda cukup menggunakan fungsi yang terdiri dari baris-baris kode yang ingin Anda gunakan kembali tersebut.



Sumber: divotomezove (2021)

Gambar 6.1 Perumpamaan Fungsi dengan Wadah untuk Menyimpan Baris-Baris Kode di dalam Satu Tempat

Jadi, bagaimana cara Anda membuat sebuah fungsi di dalam bahasa pemrograman Python? Pertama-tama, Anda ketikkan **`def`**. **`def`** merupakan kependekan dari ***define*** atau dalam bahasa Indonesia-nya adalah "menentukan". Kapan pun Anda ingin membuat sebuah

fungsi, berarti Anda juga menentukan nama dari fungsi tersebut. Itulah mengapa di dalam bahasa pemrograman Python memakai istilah **`def`** tersebut.

Kemudian, Anda berikan spasi setelah **`def`** tersebut dan memberikan sebuah nama. Misalkan Anda ingin sebuah fungsi yang dapat menampilkan kalimat sapaan di kolom **Console**, Anda dapat memberikan nama fungsi Anda **`sapa`**. Lalu, Anda berikan kurung buka dan kurung tutup setelah nama tersebut. Anda kemudian mengakhirinya dengan tanda titik dua seperti yang diperlihatkan pada `Kode Program 130`.

`Kode Program 130`

```
1  def sapa():
```

Sekarang Anda mungkin sudah menyadari bahwa di dalam bahasa pemrograman Python, setiap kali Anda melihat sebuah tanda titik dua di akhir sebuah baris kode, biasanya ketika Anda menekan tombol Enter pada *keyboard* Anda untuk beralih ke baris berikutnya, secara otomatis kursor Anda akan digeser ke arah kanan beberapa spasi karena bahasa pemrograman Python mengetahui bahwa Anda ingin melakukan sesuatu terhadap sebuah kondisi yang diberikan oleh baris kode dengan tanda titik dua tersebut. Sederhananya, segala sesuatu yang digeser ke arah kanan menjadi bagian dari baris kode di atasnya yang memiliki posisi lebih ke arah kiri.

Sebagai contoh, dengan fungsi **`sapa()`** tersebut, apabila Anda ingin membuat sebuah kalimat sapaan kepada seseorang yang menjalankan kode program Anda di pagi hari, Anda dapat mengetikkan **`print('Selamat pagi.')`** seperti yang diperlihatkan pada `Kode Program 131`. Baris kedua kode tersebut menjadi sebuah fungsi bernama **`sapa()`** yang akan menampilkan kalimat sapaan di kolom **Console** berupa **`Selamat pagi`**.

Kode Program 131

```
1  def sapa():
2    print('Selamat pagi.')
```

Ketika Anda menjalankannya dengan menekan tombol **"► Run"**, Anda tidak menemukan apa-apa di dalam kolom **Console**. Anda mungkin merasa aneh. Anda telah memiliki sebuah fungsi untuk menampilkan kalimat sapa. Seharusnya kalimat sapa itu muncul pada kolom **Console**. Apa yang terjadi?

Jadi, Anda sebenarnya baru memberikan nama saja untuk fungsi Anda tersebut. Anda baru memberikan nama **`sapa()`** untuk baris **`print('Selamat pagi.')`**. Apabila Anda ingin menjalankan fungsi **`sapa()`** tersebut, Anda harus memanggilnya. Ini dinamakan memanggil fungsi.

Untuk lebih mudahnya, bayangkan bahwa semua orang di lingkungan Anda memiliki nama mereka masing-masing. Untuk memanggil seseorang, tentunya Anda harus memanggil nama orang tersebut (atau sebutan umum seperti Pak, Bu, dan sebagainya). Begitu juga dengan fungsi. Fungsi yang Anda buat di atas sebenarnya sudah ada di dalam program dan sudah Anda tentukan bahwa fungsi tersebut memiliki nama **`sapa()`**.

Jadi, pada baris berikutnya, misal pada baris keempat (penulis memberikan satu baris kosong di antara penamaan fungsi dan pemanggilan fungsi untuk kemudahan dalam pembacaan baris-baris kode), Anda dapat mengetikkan **`sapa()`** seperti yang diperlihatkan pada `Kode Program 132`. Kemudian, apabila Anda kembali menjalankan kode program tersebut, Anda akan menemukan bahwa kalimat sapaan **Selamat pagi.** sekarang ditampilkan pada kolom **Console** seperti yang diperlihatkan pada `Console 87`.

Kode Program 132

```
1  def sapa():
2   print('Selamat pagi.')
3
4  sapa()
```

Console 87

```
Selamat pagi.
>
```

Hal yang paling menarik dari fungsi adalah Anda dapat memanggilnya berkali-kali. Untuk itu, misalnya Anda memanggil fungsi **`sapa()`** sebanyak tiga kali seperti yang diperlihatkan pada `Kode Program 133`, Anda akan melihat bahwa di kolom **Console**, kalimat sapaan **Selamat pagi.** pun akan ditampilkan sebanyak tiga kali juga seperti yang diperlihatkan pada `Console 88`.

Kode Program 133

```
1  def sapa():
2   print('Selamat pagi.')
3
4  sapa()
5  sapa()
6  sapa()
```

Console 88

```
Selamat pagi.
Selamat pagi.
Selamat pagi.
>
```

Jadi, sebanyak apa pun Anda memanggil sebuah fungsi, Anda akan menjalankan baris-baris kode yang ada di dalam fungsi tersebut. Walaupun pada contoh ini Anda hanya memiliki satu baris kode di dalam sebuah fungsi, Anda dapat menyertakan sebanyak mungkin baris kode di dalam fungsi Anda sesuai dengan kebutuhan Anda.

Sebagai contoh, Anda dapat menambahkan baris kode di dalam fungsi **`sapa()`** Anda seperti **`print('Salam sehat.')`** sebagaimana diperlihatkan pada `Kode Program 134`. Ketika Anda menjalankan kode program Anda, kedua baris kode di dalam fungsi **`sapa()`** tersebut akan ditampilkan sebanyak tiga kali di dalam kolom **Console** seperti yang diperlihatkan pada `Console 89` karena kedua baris tersebut sekarang merupakan bagian dari fungsi **`sapa()`** yang dicirikan oleh posisi kedua baris tersebut yang lebih menjorok ke arah kanan dibandingkan baris kode **`def sapa():`**-nya.

Kode Program 134

```
1  def sapa():
2   print('Selamat pagi.')
3   print('Salam sehat.')
4
5  sapa()
6  sapa()
7  sapa()
```

Console 89

```
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
>
```

Dengan melihat contoh tersebut, mungkin Anda berpikir untuk menggabungkan ketiga baris terakhir pada `Kode Program 134` tersebut ke dalam sebuah ***for loop*** sebagaimana yang telah Anda pelajari sebelumnya. Misalnya Anda ingin memanggil fungsi **`sapa()`** tersebut sebanyak sepuluh kali. Alih-alih Anda mengetikkan sepuluh kali baris **`sapa()`** di dalam kode program Anda, Anda dapat menggunakan ***for loop*** untuk menggantikan tiga baris terakhir pada `Kode Program 134` tersebut seperti yang diperlihatkan pada `Kode Program 135`. Di kolom **Console** dapat terlihat bahwa fungsi **`sapa()`** akan dijalankan

sebanyak sepuluh kali sesuai dengan parameter yang diberikan di baris ***for loop***-nya seperti yang diperlihatkan pada `Console 90`.

Kode Program 135

```
1  def sapa():
2   print('Selamat pagi.')
3   print('Salam sehat.')
4
5  for x in range(10):
6   sapa()
```

Console 90

```
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
Selamat pagi.
Salam sehat.
>
```

Makin kompleks proyek bahasa pemrograman Python yang akan Anda buat, fungsi-fungsi seperti ini sangat bermanfaat untuk mengorganisasi kode program Anda. Anda dapat memberikan nama pada satu atau lebih baris kode Anda dan Anda dapat menjalankannya beberapa kali di beberapa baris atau tempat di proyek bahasa pemrograman Python Anda.

Tantangan 12:
Buatlah sebuah fungsi dengan nama salam_kenal yang menampilkan dua baris yang menyapa pengguna yang menjalankan program dan menyapa pengguna tersebut dengan kalimat yang sesuai!

Contoh solusi Tantangan 12:

Kode Program 136

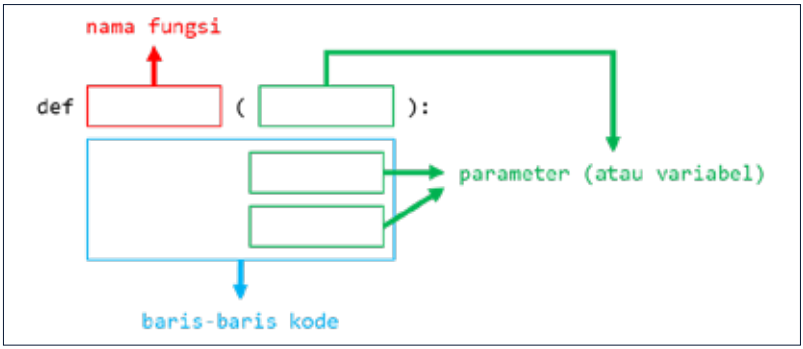
```
1  def salam_kenal():
2   print('Selamat datang Bapak dan Ibu sekalian.')
3   print('Perkenalkan nama saya Budhi Gustiandi.')
4
5  salam_kenal()
```

Console 91

```
Selamat datang Bapak dan Ibu sekalian.
Perkenalkan nama saya Budhi Gustiandi.
>
```

B. Parameter

Mari kita lanjutkan pembelajaran mengenai fungsi dengan membahas **parameter**. **Parameter** adalah cara untuk memberi masukan ke dalam sebuah fungsi. Mengapa **parameter** ini berguna? Mari kita lihat kembali tantangan yang sudah kita kerjakan pada bagian sebelumnya di mana kita menampilkan kalimat salam di dalam kolom **Console**. Misalnya, Anda ingin membuat sebuah fungsi di mana ketika Anda menjalankannya, Anda dapat memasukkan nama orang yang baru dan menampilkannya di dalam kolom **Console** tanpa harus mengubah baris-baris kode di dalam fungsi tersebut. Anda dapat melakukannya dengan menggunakan **parameter**. **Parameter** dimasukkan di dalam tanda kurung yang mengikuti sebuah fungsi. **Parameter** tersebut sebenarnya berbentuk variabel yang akan dimasukkan ke dalam sebuah fungsi seperti yang diperlihatkan pada Gambar 6.2.



Gambar 6.2 Ilustrasi Fungsi yang Memiliki Parameter

Jadi, dalam contoh ini, apabila Anda ingin menampilkan kalimat salam berbeda sesuai dengan nama orang yang memberi salam, Anda harus membuat sebuah **parameter** untuk dimasukkan ke dalam fungsi tersebut. Misalkan **parameter** yang ingin Anda masukkan Anda beri nama **`pemberi_salam`**, maka fungsi yang ditampilkan pada baris-baris kode di atas akan menjadi seperti yang diperlihatkan pada `Kode Program 137`.

Kode Program 137

```
1  def salam_kenal(pemberi_salam):
2   print('Selamat datang Bapak dan Ibu sekalian.')
3   print('Perkenalkan nama saya Budhi Gustiandi.')
4
5  salam_kenal()
```

Parameter `pemberi_salam` tersebut akan menjadi variabel yang dapat digunakan di dalam fungsi. Untuk itu, Anda pun harus mengubah baris ketiga pada kode program tersebut dengan menggantikan ***string*** '**Budhi Gustiandi**' menjadi nama dari **parameter** yang telah Anda buat, yaitu **pemberi_salam** seperti yang diperlihatkan pada `Kode Program 138.`

Kode Program 138

```
1 def salam_kenal(pemberi_salam):
2  print('Selamat datang Bapak dan Ibu sekalian.')
3  print(f'Perkenalkan nama saya {pemberi_salam}.')
4
5 salam_kenal()
```

Namun, pada saat Anda menjalankan kode program tersebut, Anda akan mendapatkan pesan kesalahan yang menyatakan bahwa pada baris kelima kode Anda, Anda berusaha memanggil fungsi yang telah Anda buat bernama **`salam_kenal()`**, tetapi Anda tidak memasukkan **parameter** apa pun ke dalamnya seperti yang diperlihatkan pada `Console 92`. Untuk itu, karena Anda telah memutuskan membuat fungsi dengan satu **parameter** di dalamnya, ketika Anda ingin memanggil fungsi tersebut, Anda pun harus menyertakan satu **parameter** di dalamnya.

Console 92

```
Traceback (most recent call last):
 File "main.py", line 5, in <module>
 salam_kenal()
TypeError: salam_kenal() missing 1 required positional
argument: 'pemberi_salam'
>
```

Anda dapat memasukkan nama orang di antara tanda kurung ketika Anda memanggil fungsi yang telah Anda buat tersebut seperti yang diperlihatkan pada `Kode Program 139`. Ketika Anda kembali menjalankan kode program tersebut, Anda akan melihat bahwa di kolom **Console** menampilkan kalimat salam dengan normal dan ***string*** 'Budhi' yang dimasukkan sebagai **parameter** ketika Anda memanggil fungsi **`salam_kenal()`** akan menggantikan variabel **`nama_orang`** di baris kode yang menggunakan variabel tersebut seperti yang diperlihatkan pada Console 93.

Kode Program 139

```
1 def salam_kenal(pemberi_salam):
2  print('Selamat datang Bapak dan Ibu sekalian.')
3  print(f'Perkenalkan nama saya {pemberi_salam}.')
4
5 salam_kenal('Budhi')
```

Console 93

```
Selamat datang Bapak dan Ibu sekalian.
Perkenalkan nama saya Budhi.
>
```

Saat Anda ingin mengubah nama orang yang menyapa, Anda tidak perlu lagi mengubah baris-baris kode di dalam fungsi tersebut. Anda cukup memasukkan nama orang tersebut di dalam tanda kurung ketika Anda memanggil fungsi tersebut. Misal kali ini Anda ingin memberi kalimat salam dari seseorang bernama Ratih, Anda cukup memasukkan ***string* 'Ratih'** ke dalam tanda kurung tersebut seperti yang diperlihatkan pada `Kode Program 140`. Ketika Anda menjalankan kode program tersebut, Anda akan melihat sekarang kalimat salam yang ditujukan oleh '**`Ratih`**' alih-alih oleh '**`Budhi`**' seperti yang diperlihatkan pada `Console 94` tanpa Anda harus mengubah baris-baris kode yang berada di dalam fungsi yang Anda panggil tersebut. Anda dapat mencobanya dengan menggunakan nama-nama orang yang Anda kenal atau dengan menggunakan nama Anda sendiri untuk memastikan bahwa fungsi tersebut telah berjalan dengan baik.

Kode Program 140

```
1 def salam_kenal(pemberi_salam):
2  print('Selamat datang Bapak dan Ibu sekalian. ')
3  print(f'Perkenalkan nama saya {pemberi_salam}.')
4
5 salam_kenal('Ratih')
```

Console 94

```
Selamat datang Bapak dan Ibu sekalian.
Perkenalkan nama saya Ratih.
>
```

Selain teks, Anda juga dapat memasukkan banyak jenis **parameter** berbeda ke dalam fungsi yang Anda buat agar fungsi tersebut dapat berjalan sesuai dengan kebutuhan Anda. Anda juga tidak hanya dapat memasukkan satu **parameter** saja, tetapi dapat disesuaikan dengan kebutuhan Anda. Misalnya, Anda ingin membuat sebuah fungsi yang Anda namakan **`penjumlahan()`**. Fungsi tersebut tentunya membutuhkan sekurang-kurangnya dua **parameter** yang dapat digunakan sebagai masukan ke dalam fungsi tersebut. Jadi, Anda dapat membuat dua **parameter** untuk fungsi tersebut, misalnya **`bilangan_1`** dan **`bilangan_2`** seperti yang diperlihatkan pada Kode Program 141. Anda pisahkan **parameter** yang satu dengan yang lainnya dengan menggunakan tanda baca koma di dalam tanda baca kurung yang menyertai fungsi tersebut.

Kode Program 141

```
1 def penjumlahan(bilangan_1, bilangan_2):
```

Kemudian, di dalam fungsi tersebut, Anda akan menambahkan baris-baris kode yang akan menggunakan **parameter `bilangan_1`** dan **`bilangan_2`** yang menjadi masukan ke dalam fungsi seperti yang diperlihatkan pada Kode Program 142. Jangan lupa, pada saat memanggil fungsi yang telah Anda buat tersebut, Anda harus menyertakan dua **parameter** di dalam tanda baca kurung yang mengikuti fungsi tersebut agar tidak muncul pesan kesalahan. Pada contoh tersebut Anda memanggil fungsi **`penjumlahan()`** dengan memasukkan **parameter 7** dan **8**. Angka 7 ini akan menggantikan **parameter `bilangan_1`** dan angka 8 ini akan menggantikan parameter **`bilangan_2`** pada fungsi yang telah Anda buat. Kemudian,

angka-angka tersebut akan diteruskan ke dalam baris-baris kode yang berada di dalam fungsi tersebut sehingga pada saat Anda menjalankan fungsi tersebut maka pada kolom **Console** akan ditampilkan bilangan **15** yang merupakan jumlah dari **7** dan **8** seperti yang diperlihatkan pada `Kode Program 142`. Jumlah ini diperoleh dari baris kode penjumlahan yang diperlihatkan pada baris keempat dan diteruskan ke baris kelima pada baris-baris kode tersebut.

Kode Program 142

```
1 def penjumlahan(bilangan_1, bilangan_2):
2  print(f'Bilangan pertama adalah {bilangan_1}.')
3  print(f'Bilangan kedua adalah {bilangan_2}.')
4  jumlah = bilangan_1 + bilangan_2
5  print(f'Jumlah kedua bilangan adalah: {jumlah}.')
6
7 penjumlahan(7, 8)
```

Console 95

```
Bilangan pertama adalah 7.
Bilangan kedua adalah 8.
Jumlah kedua bilangan adalah: 15.
>
```

Cobalah untuk mengganti **parameter-parameter** yang Anda gunakan pada saat memanggil fungsi tersebut. Misalnya gantilah angka **7** dengan angka **8** dan angka **8** dengan angka **2** seperti yang diperlihatkan pada `Kode Program 143`. Pada saat Anda menjalankannya, di dalam kolom **Console** akan ditampilkan angka **10** sebagai penjumlahan dari angka **8** dan **2** seperti yang diperlihatkan pada `Console 96`. Di samping itu, perhatikan bahwa Anda tidak perlu lagi mengubah-ubah baris-baris kode yang ada di dalam fungsi tersebut.

Kode Program 143

```
1 def penjumlahan(bilangan_1, bilangan_2):
2  print(f'Bilangan pertama adalah {bilangan_1}.')
3  print(f'Bilangan kedua adalah {bilangan_2}.')
4  jumlah = bilangan_1 + bilangan_2
5  print(f'Jumlah kedua bilangan adalah: {jumlah}.')
6
7 penjumlahan(8, 2)
```

Console 96

```
Bilangan pertama adalah 8.
Bilangan kedua adalah 2.
Jumlah kedua bilangan adalah: 10.
>
```

Sampai dengan bagian ini, Anda sebenarnya telah beberapa kali menggunakan fungsi yang menggunakan **parameter**, salah satunya adalah fungsi **`print()`**. Perhatikan bahwa fungsi tersebut membutuhkan **parameter** agar Anda dapat menampilkan sesuatu pada kolom **Console**. Apabila Anda tidak memasukkan **parameter** ke dalam fungsi **`print()`**, fungsi tersebut tetap berjalan dengan memberikan baris kosong untuk ditampilkan di dalam kolom ***Console***. Hal tersebut dikarenakan ketika tidak ada **parameter** yang digunakan pada saat memanggil fungsi **`print()`**, fungsi tersebut seolah-olah menampilkan sebuah ***string*** kosong sebagai **parameter**-nya atau dengan kata lain seolah-olah Anda memberikan perintah **`print(‘ ‘)`** di dalam baris kode Anda.

Tantangan 13:
Buatlah sebuah fungsi dengan nama data_siswa yang memiliki parameter-parameter nama_siswa dan umur_siswa. Kemudian, fungsi tersebut akan menampilkan kalimat yang memberikan informasi mengenai nama siswa dan umur siswa di dalam kolom **Console**.

Contoh solusi Tantangan 13:

Kode Program 144

```
1 def data_siswa(nama_siswa, umur_siswa):
2  print(f'Nama siswa: {nama_siswa}.')
3  print(f'Umur siswa: {umur_siswa}.')
4
5 data_siswa('Pradya', 13)
```

Console 97

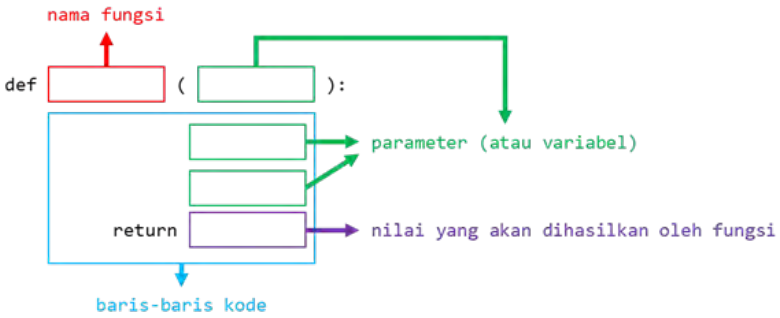
```
Nama siswa: Pradya.
Umur siswa: 13.
>
```

Catatan: Fungsi dapat menggunakan bermacam-macam **parameter**. Pada contoh solusi Tantangan 13, dapat terlihat bahwa fungsi tersebut menggunakan **parameter** pertama dalam bentuk ***string*** atau teks dan **parameter** kedua dalam bentuk ***integer*** atau bilangan bulat.

C. *Return*

Bagian ini merupakan bagian terakhir dari pelajaran mengenai fungsi. Terdapat satu lagi konsep terakhir mengenai fungsi, yaitu bagaimana cara mengembalikan informasi yang dihasilkan oleh sebuah fungsi.

Jadi, sebagaimana pada bagian sebelumnya, Anda dapat memberikan informasi ke dalam sebuah fungsi melalui **parameter**. Anda juga dapat mengeluarkan informasi dari sebuah fungsi dengan mengembalikan informasi tersebut ke baris-baris kode yang membutuhkannya. Apabila fungsi yang Anda buat dapat menghasilkan sebuah nilai (baik itu nilai numerik, *string*, atau *boolean*), Anda dapat mengeluarkan nilai tersebut melalui ***return*** yang dimasukkan di dalam baris-baris kode di dalam fungsi Anda. Ilustrasi fungsi Anda akan menjadi seperti yang diperlihatkan pada Gambar 6.3.



Gambar 6.3 Ilustrasi Fungsi yang Memiliki Parameter dan Return

Mari kita langsung membuat sebuah contoh fungsi yang dapat membuat angka yang menjadi masukannya menjadi tiga kali lipat. Pertama, Anda membuat fungsinya terlebih dahulu, misalkan Anda beri nama **`tiga_kali_lipat`** seperti yang diperlihatkan pada `Kode Program 145`. Di dalam tanda kurung yang mengikuti fungsi tersebut, Anda masukkan **parameter** berupa bilangan seperti yang diperlihatkan pada `Kode Program 146`.

Kode Program 145

```
1  def tiga_kali_lipat():
```

Kode Program 146

```
1  def tiga_kali_lipat(bilangan):
```

Anda ingin ketika seseorang memasukkan sebuah bilangan ke dalam fungsi, fungsi tersebut akan mengalikannya dengan bilangan tiga sehingga fungsi menghasilkan sebuah bilangan baru dengan besaran tiga kali lipat dari bilangan yang dimasukkan tersebut. Untuk itu, baris kode yang Anda masukkan di dalam fungsi tersebut adalah seperti yang diperlihatkan pada `Kode Program 147`. Anda pun kemudian mencoba memanggil fungsi tersebut dengan memberikan **parameter 7** ke dalamnya seperti yang diperlihatkan pada `Kode Program 148`.

Kode Program 147

```
1  def tiga_kali_lipat(bilangan):
2    bilangan * 3
```

Kode Program 148

```
1  def tiga_kali_lipat(bilangan):
2    bilangan * 3
3
4  tiga_kali_lipat(7)
```

Namun, ketika Anda menjalankan kode program Anda, tidak ada yang ditampilkan di dalam kolom **Console**. Tentu Anda sudah menduganya karena di dalam baris-baris kode Anda tidak ada baris kode yang mengandung **`print()`**. Anda coba modifikasi kode program Anda menjadi seperti yang diperlihatkan pada `Kode Program 149`. Kini, ketika Anda menjalankan kode program Anda, pada kolom **Console** akan ditampilkan hasil perkalian bilangan yang Anda masukkan sebagai **parameter**, yaitu **7** dengan bilangan **3** sehingga pada kolom **Console** akan ditampilkan bilangan **21** seperti yang diperlihatkan pada `Console 98`.

Kode Program 149

```
1  def tiga_kali_lipat(bilangan):
2    print(bilangan * 3)
3
4  tiga_kali_lipat(7)
```

Console 98

```
21
>
```

Akan tetapi, Anda tidak ingin hanya menampilkan hasil perkalian tersebut di dalam kolom **Console**. Anda ingin menyimpan hasil perkalian tersebut dan dapat mengeluarkannya dari fungsi agar dapat digunakan sebagai masukan pada fungsi lainnya. Anda dapat melakukannya dengan menggunakan fungsi ***`return`***. Anda

modifikasi baris kedua kode yang diperlihatkan pada baris-baris kode di atas menjadi **`return bilangan * 3`** seperti yang diperlihatkan pada `Kode Program 150`. Dalam contoh ini, Anda berusaha untuk mengeluarkan hasil dari fungsi **`tiga_kali_lipat()`** dalam bentuk ***integer*** agar dapat digunakan di baris kode yang lain. Anda juga dapat menggunakan fungsi ***return*** tersebut untuk mengeluarkan hasil dalam bentuk apa pun, yaitu ***float***, ***string***, ***boolean***, dan lain-lain.

Kode Program 150

```
1  def tiga_kali_lipat(bilangan):
2   return bilangan * 3
3
4  tiga_kali_lipat(7)
```

Pada saat Anda menjalankan kode program tersebut, tidak ada hasil yang ditampilkan di dalam kolom **Console**. Namun, sebenarnya hasil dari fungsi tersebut telah tersimpan di dalam kode program Anda. Contohnya, apabila Anda mengubah baris keempat kode program di atas menjadi **`print(tiga_kali_lipat(7))`** seperti yang diperlihatkan pada `Kode Program 151`, ketika Anda menjalankan kode program tersebut, Anda akan melihat hasil dari fungsi **`tiga_kali_lipat(7)`** tersebut ditampilkan di kolom **Console** seperti yang diperlihatkan pada `Console 98`. Ini berarti nilai dari fungsi **`tiga_kali_lipat(7)`** diubah oleh kode program Anda menjadi bilangan **21**. Kemudian, bilangan **21** inilah yang ditampilkan di dalam kolom **Console** dengan menggunakan fungsi **`print()`**.

Kode Program 151

```
1  def tiga_kali_lipat(bilangan):
2   return bilangan * 3
3
4  print(tiga_kali_lipat(7))
```

Biasanya, di dalam pembuatan sebuah kode program, keluaran dari fungsi yang memiliki fungsi ***return*** di dalamnya akan dimasukkan ke dalam sebuah variabel agar dapat digunakan oleh fungsi lain yang ada di dalam kode program tersebut. Misalnya, Anda ingin menyimpan hasil dari fungsi **`tiga_kali_lipat()`** ke dalam sebuah variabel yang Anda beri nama **`hasil_perkalian`** maka Anda dapat mengubah baris keempat pada `Kode Program 151` menjadi **`hasil_perkalian = tiga_kali_lipat(7)`** seperti yang diperlihatkan pada `Kode Program 152`.

Kode Program 152

```
1  def tiga_kali_lipat(bilangan):
2   return bilangan * 3
3
4  hasil_perkalian = tiga_kali_lipat(7)
5
6  print(hasil_perkalian)
```

Kemudian, ketika Anda mencoba menampilkan isi dari variabel **`hasil_perkalian`** tersebut dengan menggunakan fungsi **`print()`** seperti pada baris keenam pada kode program di atas, akan ditampilkan bilangan **`21`** di kolom **Console** yang merupakan hasil dari fungsi **`tiga_kali_lipat(7)`**. Di sinilah fungsi ***return*** dapat terlihat dengan jelas.

Tantangan 14:
Buatlah sebuah ***`list`*** yang terdiri dari minimal tiga teks di dalamnya. Kemudian buatlah sebuah fungsi yang mengubah masing-masing teks di dalam ***`list`*** tersebut menjadi huruf kapital semuanya. Jangan lupa untuk menggunakan fungsi ***return*** sebagaimana Anda pelajari pada bagian ini. Tampilkan masing-masing keluaran dari fungsi tersebut di dalam kolom **Console.**

Catatan: Untuk membuat huruf menjadi kapital, Anda dapat menggunakan fungsi **`upper()`.**

Contoh solusi Tantangan 14:

Kode Program 153

```
1  daftar_nama = ['Budhi', 'Ratih', 'Pradya']
2
3  def kapital(teks):
4   return teks.upper()
5
6  for nama in daftar_nama:
7   teks_kapital = kapital(nama)
8   print(teks_kapital)
```

Console 99

```
Console
BUDHI
RATIH
PRADYA
>
```

Catatan: Untuk menampilkan masing-masing isi dari ***list***, Anda dapat menggunakan ***for loop*** seperti yang telah Anda pelajari di bagian sebelumnya.

D. Komentar

Pada bagian ini, Anda akan belajar tentang komentar atau lebih tepatnya cara untuk memberikan catatan atau ide di dalam kode program Anda. Baris-baris yang mengandung komentar tidak akan dijalankan sebagai kode oleh program karena program akan mengabaikannya. Namun, baris-baris tersebut sangat berguna bagi Anda yang membuat kode program atau orang lain yang membaca kode program buatan Anda nantinya. Bisa jadi suatu saat Anda membuat sebuah program dan setelah sekian lama Anda ingin memperbaharui kode program Anda tersebut, tetapi Anda lupa mengapa Anda menulis baris-baris kodenya seperti itu. Bisa jadi juga Anda akan bekerja sama di dalam sebuah tim dalam membuat sebuah kode program yang kompleks dan saling terkait satu sama lainnya sehingga masing-masing anggota di tim Anda

akan mengerti alur dari baris-baris kode yang Anda buat dan Anda dapat mengerti baris-baris kode yang dibuat oleh anggota-anggota lain di tim Anda.

Komentar dapat dibuat dengan cara mengetikkan tanda pagar (#) di depan baris kode yang ingin Anda jadikan komentar. Sebagai contoh, apabila Anda ingin memberikan komentar pada bagian awal `Kode Program 153` yang telah Anda buat seperti pada contoh solusi Tantangan 14 dengan menyebutkan bahwa langkah pertama dari tantangan yang coba Anda selesaikan adalah "**`Buatlah sebuah list yang terdiri dari minimal tiga teks di dalamnya`**," Anda dapat memberikan komentar tersebut seperti yang diperlihatkan pada baris pertama `Kode Program 154`. Pada saat Anda menjalankan kode tersebut, program tidak akan membaca baris pertama dari baris kode yang telah Anda beri komentar tersebut sehingga keluarannya akan tetap seperti sebelumnya.

Kode Program 154

```
1  # Buatlah sebuah list yang terdiri dari minimal tiga teks
   di dalamnya
2  daftar_nama = ['Budhi', 'Ratih', 'Pradya']
3
4  def kapital(teks):
5   return teks.upper()
6
7  for nama in daftar_nama:
8   teks_kapital = kapital(nama)
9   print(teks_kapital)
```

Selain di awal baris, komentar juga dapat ditempatkan tepat di samping baris kode yang Anda buat seperti yang diperlihatkan pada `Kode Program 155`. Pada baris kelima kode program tersebut, dapat terlihat bahwa komentar diawali dengan tanda pagar (#) untuk menunjukkan bahwa itu bukan bagian dari yang harus dijalankan oleh program. Sama seperti sebelumnya, pada saat Anda menjalankan

baris-baris kode tersebut, program akan mengabaikan komentarnya dan keluaran yang ditampilkan di dalam kolom **Console** akan tetap seperti sebelumnya.

Kode Program 155

```
1  # Buatlah sebuah list yang terdiri dari minimal tiga
   teks di dalamnya
2  daftar_nama = ['Budhi', 'Ratih', 'Pradya']
3
4  def kapital(teks):
5   return teks.upper() # fungsi untuk membuat teks kapital
6
7  for nama in daftar_nama:
8   teks_kapital = kapital(nama)
9   print(teks_kapital)
```

Beberapa pemrogram bahkan menggunakan komentar untuk merencanakan baris-baris kodenya sebelum benar-benar menulis baris-baris kode tersebut atau bahkan memberi tanda baris-baris kode mana saja yang sedang dikerjakan di dalam sebuah program yang kompleks. Sayangnya, bahasa pemrograman Python tidak memiliki cara singkat untuk memberikan komentar yang berlaku untuk beberapa baris sehingga Anda hanya dapat menggunakan tanda pagar tersebut untuk memberikan komentar yang berlaku untuk satu baris kode saja.

Masih ingatkah bahwa pada bagian-bagian sebelumnya Anda telah membuat sebuah fungsi dengan satu **parameter** atau lebih? Nah, komentar juga dapat digunakan untuk menjelaskan batasan-batasan parameter apa saja yang dapat menjadi masukan dari sebuah fungsi, bahkan dapat menjelaskan seperti apa keluaran dari sebuah fungsi agar dapat digunakan di dalam fungsi yang lainnya.

E. Input

Pada bagian ini, Anda akan mempelajari tentang fungsi **input** atau bagaimana program Anda dapat menerima masukan dari pengguna. Selebihnya, program Anda pun dapat bereaksi atau membuat keputusan terhadap masukan yang diberikan tersebut. Dengan kata lain, Anda akan mempelajari cara menambah fitur agar para pengguna program Anda dapat berinteraksi dengan program Anda.

Mari kita buat fungsi **input** pertama untuk program Anda. Pertama-tama, tuliskan **`input ()`** di dalam kode program Anda seperti yang diperlihatkan pada `Kode Program 156`. Kemudian, di dalam tanda kurungnya, ketikkan teks yang ingin Anda tampilkan kepada pengguna di kolom **Console**, misalnya **`'Masukkan nama lengkap Anda: '`** seperti yang diperlihatkan pada `Kode Program 157`. Pada saat Anda menjalankan kode program tersebut, kolom **Console** akan menampilkan teks yang Anda masukkan di dalam tanda kurung tersebut dan pengguna dapat memberi masukan ke dalam program Anda seperti yang diperlihatkan pada `Console 100`. Penulis mencoba memasukkan nama lengkap penulis ke dalam program tersebut dan menekan tombol **enter** seperti yang diperlihatkan pada `Console 101`.

Kode Program 156

```
1  input()
```

Kode Program 157

main.py

```
1  input('Masukkan nama lengkap Anda: ')
```

Console 100

```
Masukkan nama lengkap Anda:
```

Console 101

```
Masukkan nama lengkap Anda: Budhi Gustiandi
>
```

Kursor pada kolom **Console** akan diarahkan ke baris berikutnya. Tidak ada yang terjadi kemudian di dalam program tersebut, tetapi dapat dipastikan bahwa program telah berjalan dengan normal karena tidak ada pesan kesalahan yang muncul di kolom **Console**. Alasan mengapa tidak ada yang terjadi adalah karena memang program belum ditulis untuk melakukan apa-apa terhadap masukan yang diberikan oleh pengguna.

Agar program dapat berinteraksi dengan masukan dari pengguna, masukan dari pengguna tersebut biasanya disimpan di dalam sebuah variabel di dalam kode program. Pada contoh tersebut, Anda dapat menyimpan masukan yang diberikan oleh pengguna ke dalam variabel **`nama_lengkap`** dengan cara mengubah baris kode program menjadi **`nama_lengkap = input('Masukkan nama lengkap Anda: ')`** seperti yang diperlihatkan pada Kode Program 158. Dengan cara ini, apa pun yang pengguna ketikkan di dalam kolom **Console** setelah program dijalankan maka akan disimpan ke dalam variabel **`nama_lengkap`** tersebut.

Kode Program 158

```
1  nama_lengkap = input('Masukkan nama lengkap Anda: ')
```

Setelah masukan tersimpan ke dalam sebuah variabel, seperti yang telah dijelaskan pada bagian-bagian sebelumnya, Anda dapat menggunakan variabel tersebut dan mengolah isinya untuk dapat digunakan di dalam kode program Anda. Misalnya, apabila Anda ingin menampilkan kembali apa yang pengguna ketikkan di dalam kolom **Console**, Anda dapat menggunakan fungsi **`print()`** dengan parameter isi dari variabel tersebut seperti yang diperlihatkan pada Kode Program 159.

Kode Program 159

```
1  nama_lengkap = input('Masukkan nama lengkap Anda: ')
2
3  print(f'Nama lengkap Anda adalah {nama_lengkap}.')
```

Console 102

```
Masukkan nama lengkap Anda: Budhi Gustiandi
Nama lengkap Anda adalah Budhi Gustiandi.
>
```

Dari tampilan **Console** tersebut terlihat bahwa ketika penulis mengetikkan nama lengkap penulis ke dalam program dan menekan tombol **enter**, program akan mengeluarkan pesan sesuai dengan nama lengkap yang telah penulis masukkan tersebut. Anda dapat mencobanya dengan menggunakan nama lengkap Anda sendiri.

Contoh lain misalnya Anda ingin menjadikan seluruh huruf yang dimasukkan oleh pengguna menjadi huruf kapital. Anda dapat menggunakan fungsi **`.upper()`** terhadap isi dari variabel yang telah menyimpan teks yang dimasukkan oleh pengguna seperti yang diperlihatkan pada `Kode Program 160`.

Kode Program 160

```
1 nama_lengkap = input('Masukkan nama lengkap Anda: ')
2
3 nama_lengkap_kapital = nama_lengkap.upper()
4 print(f'Nama lengkap Anda adalah {nama_lengkap_kapital}.')
```

Console 103

```
Masukkan nama lengkap Anda: Budhi Gustiandi
Nama lengkap Anda adalah BUDHI GUSTIANDI.
>
```

Dari contoh tersebut dapat terlihat bahwa meskipun penulis memasukkan nama lengkap penulis tidak dalam bentuk huruf kapital, program dapat mengolahnya untuk ditampilkan sebagai huruf kapital di kolom **Console**. Hal ini dikarenakan sebelum program menampilkan teks yang dimasukkan oleh pengguna, program dapat terlebih dahulu mengolah teks tersebut.

Salah satu hal yang perlu Anda perhatikan terkait dengan penulisan teks yang ingin ditampilkan di kolom **Console** kepada pengguna adalah jangan lupa untuk memberikan spasi agar pengguna dapat dengan mudah membaca tampilan program yang dijalankan tersebut. Hal tersebut sebenarnya telah diperlihatkan pada contoh-contoh sebelumnya. Ketika penulis memberikan contoh teks di dalam fungsi **input**, penulis selalu memberikan spasi (bahkan tanda titik dua) untuk memperlihatkan batas antara teks yang ditampilkan oleh program dan teks yang akan dimasukkan oleh pengguna.

Hal lain yang ingin penulis tekankan adalah bahwa masukan dari pengguna pada baris kode dengan fungsi **input** selalu memiliki jenis data ***string*** atau teks. Jadi, walaupun pengguna memasukkan angka di kolom **Console** untuk berinteraksi dengan program, program akan menganggapnya sebagai sebuah teks.

Misalnya, Anda ingin membuat sebuah program yang ingin menampilkan bilangan tiga kali lipat dari bilangan yang diberikan oleh pengguna seperti yang diperlihatkan pada `Kode Program 161`. Pada saat pengguna memasukkan bilangan **7** di kolom **Console**, alih-alih menampilkan hasil 21, program menampilkan tiga kali bilangan 7 secara berderet (**777**) di kolom **Console** seperti yang diperlihatkan pada `Console 104`. Hal ini dikarenakan bilangan yang dimasukkan oleh pengguna tersebut diperlakukan sebagai jenis data teks atau ***string*** oleh program. Perkalian terhadap sebuah ***string*** atau teks sama dengan ***string*** atau teks tersebut dideretkan sesuai dengan pengalinya.

Kode Program 161

```
1  bilangan = input('Masukkan bilangan yang ingin Anda
   kalikan tiga: ')
2
3  bilangan_tiga_kali_lipat = bilangan * 3
4  print(bilangan_tiga_kali_lipat)
```

Console 104

```
Masukkan bilangan yang ingin Anda kalikan tiga: 7
777
>
```

Untuk mengubah jenis data ***string*** atau teks menjadi jenis data numerik, Anda dapat menggunakan fungsi **int()**(apabila Anda ingin mengubahnya menjadi ***integer*** seperti untuk contoh di atas) dan memasukkan teks yang ingin Anda ubah tersebut ke dalam tanda kurung di dalam fungsi **int()** tersebut. Contohnya adalah seperti yang diperlihatkan pada pada `Kode Program 162`. Pada kode program tersebut, bilangan yang dimasukkan oleh pengguna disimpan sebagai teks di dalam variabel **`bilangan_dalam_teks`**. Setelah itu, program mengubah teks di dalam variabel tersebut menjadi bilangan ***integer*** dengan menggunakan fungsi **`int(bilangan_dalam_teks)`** dan menyimpan hasilnya di dalam variabel **`bilangan_dalam_integer`**. Variabel **`bilangan_dalam_integer`** inilah yang kemudian diolah lebih lanjut oleh program (dalam hal ini dikalikan dengan bilangan tiga) untuk kemudian hasilnya disimpan di dalam variabel **`bilangan_tiga_kali_lipat`**. Terakhir, program menampilkan hasil perkalian tersebut kembali ke kolom **Console** agar dapat dilihat oleh pengguna dengan menggunakan fungsi **`print()`**. Pada saat program tersebut dijalankan, dapat terlihat kini bilangan **`7`** yang dimasukkan oleh penulis dikembalikan ke kolom **Console** oleh program dengan menampilkan bilangan **`21`** yang merupakan hasil perkalian dari 7 dengan 3 seperti yang diperlihatkan pada `Console 105`. Jadi, jangan lupa untuk mengonversi jenis data ***string*** atau teks ke dalam jenis data ***integer*** atau ***float*** apabila terdapat baris program Anda yang ingin mengolah teks tersebut sebagai numerik.

Kode Program 162

```
1 bilangan_dalam_teks = input('Masukkan bilangan yang
  ingin Anda kalikan tiga: ')
2
3 bilangan_dalam_integer = int(bilangan_dalam_teks)
4 bilangan_tiga_kali_lipat = bilangan_dalam_integer * 3
5 print(bilangan_tiga_kali_lipat)
```

Console 105

```
Masukkan bilangan yang ingin Anda kalikan tiga: 7
21
>
```

Tantangan 15:
Buat sebuah kode program yang menerima masukan dari pengguna berupa nama lengkap seperti yang ditampilkan di awal bagian ini kemudian tambahkan pilihan kepada pengguna apakah pengguna ingin menampilkan teks yang dimasukkan menjadi huruf besar semua atau menjadi huruf kecil semua.

Contoh solusi Tantangan 15:

Kode Program 163

```
1  nama_lengkap = input('Masukkan nama lengkap Anda: ')
2  pilihan = input('''
3  Bagaimana nama lengkap Anda ingin ditampilkan?
4  1) Huruf kapital
5  2) Huruf kecil
6  Pilihan Anda: ''')
7
8  if pilihan == '1':
9  nama_lengkap = nama_lengkap.upper()
10  print('Anda memilih tampilan huruf kapital.')
11 elif pilihan == '2':
12  nama_lengkap = nama_lengkap.lower()
13  print('Anda memilih tampilan huruf kecil.')
14 else:
15  print('Anda tidak memilih pilihan yang valid.')
16
17 print(f'Nama lengkap Anda adalah {nama_lengkap}.')
```

Catatan: Pada contoh solusi tersebut, ketika program dijalankan, pengguna diminta untuk memasukkan nama lengkap terlebih dahulu. Setelah itu, pengguna akan diminta untuk memasukkan angka 1 atau 2 untuk memilih bagaimana nama lengkapnya akan ditampilkan pada keluaran program di kolom **Console** seperti yang diperlihatkan pada Console 106 sampai dengan Console 108.

Console 106

```
Masukkan nama lengkap Anda: Budhi Gustiandi
Bagaimana nama lengkap Anda ingin ditampilkan?
1) Huruf kapital
2) Huruf kecil
Pilihan Anda: 1
Anda memilih tampilan huruf kapital.
Nama lengkap Anda adalah BUDHI GUSTIANDI.
>
```

Console 107

```
Masukkan nama lengkap Anda: Budhi Gustiandi
Bagaimana nama lengkap Anda ingin ditampilkan?
1) Huruf kapital
2) Huruf kecil
Pilihan Anda: 2
Anda memilih tampilan huruf kecil.
Nama lengkap Anda adalah budhi gustiandi.
>
```

Console 108

```
Masukkan nama lengkap Anda: Budhi Gustiandi
Bagaimana nama lengkap Anda ingin ditampilkan?
1) Huruf kapital
2) Huruf kecil
Pilihan Anda: A
Anda tidak memilih pilihan yang valid.
Nama lengkap Anda adalah Budhi Gustiandi.
>
```

Apabila Anda tidak ingin mengubah terlebih dahulu pilihan angka 1 atau 2 dari jenis data teks atau ***string*** menjadi jenis data numerik (***integer***) seperti pada `Kode Program 163`, ketika Anda

akan menggunakan variabel **`pilihan`** yang digunakan untuk menyimpan masukan dari pengguna, Anda harus memperlakukannya sebagai jenis data ***string*** atau teks seperti yang diperlihatkan oleh baris kedelapan dan kesebelas pada baris-baris kode tersebut. Untuk mengatasi pengguna yang memasukkan pilihan di luar pilihan yang disediakan, Anda juga dapat menggunakan ***else statement*** seperti yang diperlihatkan baris ke-14 pada kode program tersebut.

Proyek 3: Permainan Tebak Angka

Bab 7

Selamat! Anda berhasil sampai di proyek akhir dari pembelajaran Anda. Anda telah belajar banyak mengenai bahasa pemrograman Python dan akan sangat berarti apabila Anda dapat menerapkan semuanya di dalam proyek akhir ini.

Jadi, Anda akan membuat sebuah program permainan tebak angka. Idenya adalah program Anda akan memilih sebuah bilangan secara acak, misalnya di antara 1 dan 100, kemudian pengguna akan mencoba menebak bilangan apa yang telah dipilih oleh program Anda tersebut. Setiap kali pengguna menebak bilangan tersebut, apabila belum tepat, program Anda akan memberitahukan kepada pengguna bahwa bilangan yang ditebaknya terlalu kecil atau terlalu besar. Ketika pengguna berhasil menebak bilangan tersebut, program akan memberi informasi kepada pengguna pada tebakan ke berapa pengguna berhasil menebak bilangan tersebut.

Untuk membuat proyek yang cukup kompleks seperti ini, Anda dapat melakukannya dengan membuat langkah-langkah di dalam kode program Anda terlebih dahulu dengan menggunakan komentar sebelum Anda menuliskan baris-baris kode dalam bahasa pemrograman Python seperti yang diperlihatkan pada `Kode Program 164`. Dengan menggunakan komentar-komentar tersebut, gambaran alur program secara keseluruhan dapat Anda lihat secara jelas dan Anda dapat mengerjakan baris-baris kodenya bagian per bagian sesuai dengan penjelasan tersebut.

Kode Program 164

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  # Program meminta pengguna untuk menebak bilangan tersebut
3  # Pengguna menebak bilangan yang telah ditentukan program
4  # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
5  # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
6  # Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
```

Anda dapat memulai untuk menulis baris-baris kode untuk menerapkan langkah pertama dari program Anda seperti yang diperlihatkan pada baris pertama `Kode Program 164`, yaitu "Program menentukan sebuah bilangan antara 1 dan 100 secara acak." Seperti telah Anda pelajari di bagian sebelumnya, untuk menentukan bilangan secara acak, Anda perlu mengimpor modul **random**. Anda dapat melakukannya dengan mengetikkan **`import random`** seperti yang diperlihatkan pada `Kode Program 165`.

Kode Program 165

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  import random
3
4  # Program meminta pengguna untuk menebak bilangan tersebut
5  # Pengguna menebak bilangan yang telah ditentukan program
6  # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
7  # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
8  # Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
```

Setelah itu, Anda dapat menggunakan fungsi **`random.randint(1, 100)`** dan menyimpannya di dalam sebuah variabel bernama **`bilangan`**. Kodenya seperti diperlihatkan baris keempat pada `Kode Program 166`.

Kode Program 166

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  import random
3
4  bilangan = random.randint(1, 100)
5
6  # Program meminta pengguna untuk menebak bilangan tersebut
7  # Pengguna menebak bilangan yang telah ditentukan program
8  # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
9  # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
10 # Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
```

Untuk memastikan bahwa tidak ada baris kode yang Anda tuliskan salah, Anda dapat mengujinya dengan menggunakan fungsi **`print(bilangan)`**. Fungsi tersebut menampilkan bilangan acak di dalam kolom **Console** setiap kali Anda menjalankan program tersebut seperti yang diperlihatkan pada baris keenam `Kode Program 167`.

Kode Program 167

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  import random
3
4  bilangan = random.randint(1, 100)
5
6  print(bilangan)
7
8  # Program meminta pengguna untuk menebak bilangan tersebut
9  # Pengguna menebak bilangan yang telah ditentukan program
10 # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
11 # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
12 # Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
```

Jalankan program Anda beberapa kali untuk memastikan tidak ada pesan kesalahan yang ditampilkan di kolom **Console** dan bilangan yang ditampilkan secara acak adalah bilangan di antara 1 dan 100. Apabila sudah benar, Anda dapat menghapus fungsi **`print()`** tersebut karena nanti pengguna akan dapat melihat berapa bilangan yang telah dipilih oleh program Anda dan dapat langsung menebaknya. Anda telah menyelesaikan langkah pertama dari program permainan tebak angka Anda.

Langkah selanjutnya yang harus dikerjakan adalah membuat baris-baris kode untuk komentar "Program meminta pengguna untuk menebak bilangan tersebut" dan "Pengguna menebak bilangan yang telah ditentukan program" seperti yang diperlihatkan pada baris kedelapan dan kesembilan kode program tersebut. Pada langkah ini Anda dapat menggunakan fungsi **`input()`** seperti yang telah dipelajari pada bagian sebelumnya. Jangan lupa untuk memakai fungsi **`int()`** untuk mengubah masukan dari pengguna dari format teks menjadi format ***integer*** sebagaimana yang diperlukan oleh program Anda. Kemudian, simpanlah bilangan yang ditebak oleh pengguna tersebut di dalam sebuah variabel, misalnya **`tebakan`**, agar dapat dibandingkan dengan bilangan yang telah ditentukan oleh program di langkah sebelumnya. Contoh baris kode dan tampilan kolom **Console** setelah programnya dijalankan adalah seperti yang diperlihatkan pada Kode Program 168.

Kode Program 168

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  import random
3
4  bilangan = random.randint(1, 100)
5
6  # Program meminta pengguna untuk menebak bilangan tersebut
7  # Pengguna menebak bilangan yang telah ditentukan program
8  tebakan = int(input('Berapakah bilangan yang telah dipilih? Tebakan Anda: '))
9
10 # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
11 # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
12 # Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
```

Console 109

```
Berapakah bilangan yang telah dipilih? Tebakan Anda: _
```

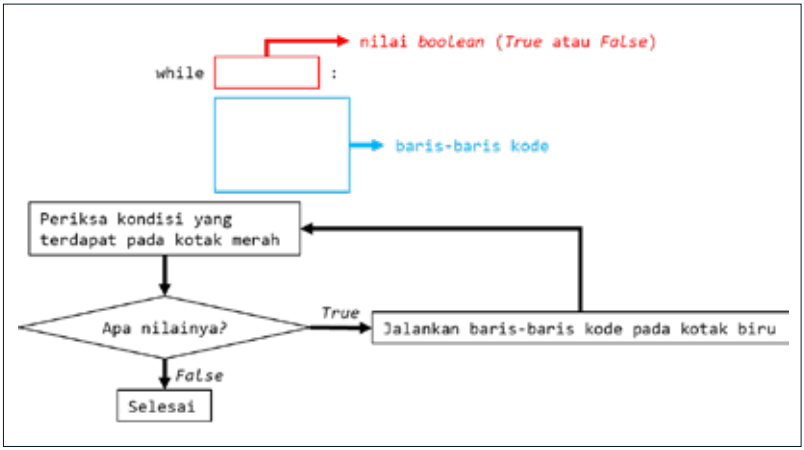
Langkah berikutnya adalah "Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya" seperti yang diperlihatkan baris kesepuluh `Kode Program 168`. Anda mungkin berpikir untuk menggunakan ***if statement*** secara langsung seperti yang telah Anda pelajari sebelumnya. Namun, perlu diingat bahwa pada langkah berikutnya, "Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut ...." seperti yang diperlihatkan baris ke-11 `Kode Program 168`. Hal ini berarti bahwa Anda perlu menggunakan perulangan agar program terus berjalan sampai tebakan pengguna benar.

A. *While Loop*

Pada bagian sebelumnya Anda telah mempelajari tentang ***for loop*** dan mungkin Anda berpikir bahwa ***for loop*** dapat digunakan untuk mengatasi hal tersebut. Namun, perlu diingat bahwa ***for loop*** memiliki kelemahan tersendiri, yaitu Anda harus tahu persis berapa kali perulangan tersebut harus berjalan, sementara di dalam permainan tebak angka yang Anda buat, Anda tidak dapat memastikan berapa kali pengguna dapat melakukan tebakan sampai bilangan yang telah ditentukan oleh program dapat ditebak dengan benar. Karena bilangan yang disediakan oleh program permainan tebak angka Anda memiliki rentang 1 sampai dengan 100, jumlah tebakan pengguna juga berkisar dari satu kali tebakan langsung benar atau bahkan baru di tebakan ke-100, pengguna baru dapat menebak bilangan tersebut dengan benar.

Di sinilah penulis ingin mengenalkan sebuah konsep baru kepada Anda yang disebut dengan ***while loop***. Pada dasarnya ***while loop*** merupakan kombinasi dari ***if statement*** dan ***for loop***. ***While loop*** sangat berguna ketika Anda berada di dalam sebuah situasi ingin melakukan

perulangan di dalam kode program Anda, tetapi Anda tidak tahu berapa kali program harus melakukan perulangan tersebut. ***While loop*** akan dijalankan selama kondisi yang dipersyaratkannya bernilai ***True*** dan akan dihentikan ketika kondisi yang dipersyaratkannya bernilai ***False***. Diagram alir dari ***while loop*** secara sederhana seperti yang diperlihatkan pada Gambar 7.1.



Gambar 7.1 Diagram Alir *While Loop*

Pada program permainan tebak angka Anda, kondisi yang dipersyaratkan agar perulangan terus berjalan adalah ketika tebakan pengguna masih tidak sama dengan bilangan yang telah ditentukan oleh program. Ketika tebakan pengguna sudah tepat sama dengan bilangan yang telah ditentukan oleh program, perulangan akan dihentikan dan program tidak akan meminta pengguna untuk melakukan tebakan lagi. Cara membuat ***while loop*** cukup sederhana. Anda tinggal mengetikkan **while**, spasi, kemudian kondisi yang bernilai ***True***, dan tanda baca titik dua. Dari kondisi tersebut, Anda dapat membuat baris kode ***while loop*** seperti yang diperlihatkan pada `Kode Program 169`.

Kode Program 169

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  import random
3
4  bilangan = random.randint(1, 10)
5
6  # Program meminta pengguna untuk menebak bilangan tersebut
7  # Pengguna menebak bilangan yang telah ditentukan program
8  tebakan = int(input(‘Berapakah bilangan yang telah dipilih?
   Tebakan Anda: ))
9
10 # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
11 # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
12 while tebakan != bilangan:
13     tebakan = int(input('Berapakah bilangan yang telah dipilih? Tebakan Anda:))
14
15 # Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
16 print('Anda berhasil menebak dengan benar!')
```

Di sini kondisi **`tebakan != bilangan`** berarti bahwa selama tebakan yang diberikan oleh pengguna belum sama dengan bilangan yang telah ditentukan oleh program. Perhatikan pula bahwa penulis sedikit memodifikasi baris ke-4 kode program Anda untuk mempermudah dalam pengujian program Anda tersebut dengan membatasi bilangan acak yang digunakan hanya dari 1 sampai dengan 10 saja. Di dalam ***while loop***, Anda dapat menyalin baris kedelapan kode program Anda karena itulah yang akan dilakukan oleh program selama tebakan dari pengguna belum benar. Pada saat tebakan dari pengguna sudah benar, kondisi dari ***while loop*** akan bernilai ***False***

dan program tidak akan menjalankan ***while loop*** tersebut dan akan beralih ke baris kode ke-16 yang berada di bawahnya (dalam contoh ini fungsi **`print()`** yang menampilkan pesan kepada pengguna bahwa pengguna telah berhasil menebak dengan benar). Ketika Anda menjalankan kode program tersebut, dapat terlihat bahwa selama tebakan yang diberikan pengguna masih salah, program akan terus meminta pengguna untuk menebak bilangan yang telah ditentukan. Setelah beberapa kali melakukan tebakan, dapat terlihat bahwa tebakan dari pengguna sudah benar dan program pun akan selesai berjalan seperti yang diperlihatkan pada `Console 110`.

`Console 110`

```
Berapakah bilangan yang telah dipilih? Tebakan Anda: 7
Berapakah bilangan yang telah dipilih? Tebakan Anda: 3
Berapakah bilangan yang telah dipilih? Tebakan Anda: 5
Berapakah bilangan yang telah dipilih? Tebakan Anda: 4
Berapakah bilangan yang telah dipilih? Tebakan Anda: 8
Berapakah bilangan yang telah dipilih? Tebakan Anda: 9
Berapakah bilangan yang telah dipilih? Tebakan Anda: 1
Berapakah bilangan yang telah dipilih? Tebakan Anda: 2
Anda telah berhasil menebak dengan benar.
>
```

Bagaimana apabila pengguna memasukkan selain angka di kolom **Console**? Program akan menampilkan pesan kesalahan yang memberi tahu pengguna bahwa pengguna seharusnya memasukkan angka seperti yang diperlihatkan pada `Console 111`.

`Console 111`

```
Berapakah bilangan yang telah dipilih? Tebakan Anda: a
Traceback (most recent call last):
 File "main.py", line 8, in <module>
  Tebakan = int(input('Berapakah bilangan yang telah
dipilih? Tebakan Anda: '))
ValueError: invalid literal for int() with base 10: 'a'
>
```

Untuk mengatasi hal tersebut, Anda dapat melakukan teknik validasi masukan. Namun, hal tersebut merupakan pembahasan yang lebih mendalam dan di luar lingkup dari buku ini. Untuk sementara, Anda dapat mengasumsikan bahwa pengguna hanya memasukkan angka di kolom **Console** tersebut.

Langkah terakhir di bagian ini adalah Anda ingin program menampilkan jumlah tebakan yang sudah dilakukan oleh pengguna sampai tebakannya benar seperti yang dipersyaratkan baris ke-15 pada `Kode Program 169`. Untuk memenuhi hal tersebut, Anda dapat membuat sebuah variabel yang akan menghitung jumlah tebakan yang dilakukan oleh pengguna. Misalnya variabel tersebut bernama **`jumlah_tebakan`**. Anda dapat mengetikkan setelah Anda membuat variabel **`tebakan`** seperti yang diperlihatkan pada `Kode Program 170`.

Kode Program 170

```
1  # Program menentukan sebuah bilangan antara 1 dan 100 secara acak
2  import random
3
4  bilangan = random.randint(1, 10)
5
6  # Program meminta pengguna untuk menebak bilangan tersebut
7  # Pengguna menebak bilangan yang telah ditentukan program
8  tebakan = int(input('Berapakah bilangan yang telah dipilih? Tebakan Anda: ))
9  jumlah_tebakan = 1
10
11 # Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
12 # Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
```

```
13 while tebakan != bilangan:
14     jumlah_tebakan += 1
15     tebakan = int(input('Berapakah bilangan yang telah
   dipilih? Tebakan Anda: ))
16
17 # Apabila tebakan pengguna benar, program akan menampilkan
   berapa kali pengguna telah melakukan tebakan
18 print(f'Anda berhasil menebak dengan benar setelah
   {jumlah_tebakan} kali menebak.')
```

Variabel **`jumlah_tebakan`** diinisiasi dengan angka 1 karena walaupun misalnya ada pengguna yang berhasil menebak bilangan yang ditentukan oleh program pada pertama kali tebakannya, pengguna telah melakukan tebakan sebanyak 1 kali. Nilai dari variabel **`jumlah_tebakan`** ini dinaikkan sebanyak 1 kali setiap kali pengguna melakukan tebakan yang salah sehingga di dalam ***while loop***, ditambahkan baris kode **`jumlah_tebakan += 1`** seperti yang diperlihatkan baris ke-14 pada kode program tersebut. Nilai akhir dari variabel **jumlah_tebakan** tersebut pada akhirnya ditampilkan kembali kepada pengguna di kolom **Console** setelah pengguna berhasil menebak bilangan yang telah ditentukan oleh program dengan benar seperti yang diperlihatkan baris ke-18 `Kode Program 169`. Ketika program dijalankan, dapat terlihat pada gambar tersebut contoh seorang pengguna yang berhasil menebak bilangan setelah lima kali menebak seperti yang diperlihatkan pada `Console 112`.

Console 112

```
Berapakah bilangan yang telah dipilih? Tebakan Anda: 3
Berapakah bilangan yang telah dipilih? Tebakan Anda: 5
Berapakah bilangan yang telah dipilih? Tebakan Anda: 4
Berapakah bilangan yang telah dipilih? Tebakan Anda: 8
Berapakah bilangan yang telah dipilih? Tebakan Anda: 9
Anda telah berhasil menebak dengan benar setelah 5 kali
menebak.
>
```

Sampai dengan bagian ini, program Anda telah berjalan dengan baik. Namun, ada beberapa fitur yang belum dibahas, yaitu perilaku program untuk memberi tahu kepada pengguna bahwa tebakannya masih terlalu kecil atau terlalu besar seperti persyaratan yang diberikan baris ke-12 pada `Kode Program 170` sehingga pengguna mendapat gambaran ke arah mana harus menebak apabila bilangan yang dipilih secara acaknya tidak hanya 10 seperti pada contoh sebelumnya. Hal tersebut akan dibahas pada bagian berikutnya.

B. Lebih Besar, Lebih Kecil, dan Membuat Program Lebih Interaktif

Anda akan terus mengembangkan program permainan tebak angka Anda pada bagian ini. Anda akan memulainya dengan menambahkan fitur yang memberitahukan kepada pengguna bahwa tebakannya terlalu kecil atau terlalu besar sehingga yang perlu Anda lakukan adalah menggunakan ***if statement*** di dalam ***while loop*** seperti yang diperlihatkan pada `Kode Program 171`. Anda dapat menempatkan ***if statement*** tersebut sebelum maupun sesudah baris yang menaikkan nilai variabel **`jumlah_tebakan`**. Pada contoh ini, penulis menempatkannya setelah nilai dari variabel tersebut dinaikkan.

Kode Program 171

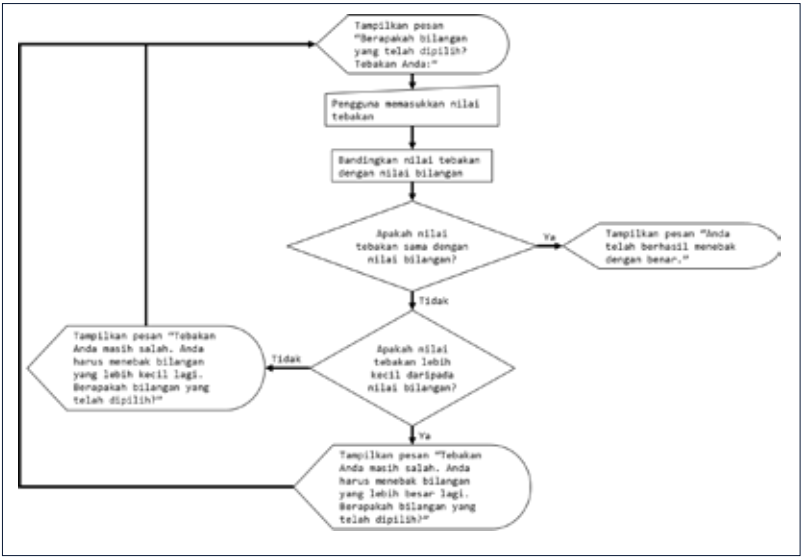
```
1  # Program menentukan sebuah bilangan antara 1 dan 100
   secara acak
2  import random
3
4  bilangan = random.randint(1, 100)
5
6  # Program meminta pengguna untuk menebak bilangan
   tersebut
7  # Pengguna menebak bilangan yang telah ditentukan
   program
```

```
8  tebakan = int(input('Berapakah bilangan yang telah
   dipilih? Tebakan Anda: ))
9  jumlah_tebakan = 1
10
11 # Program membandingkan tebakan pengguna dengan bilangan
   yang telah ditentukan sebelumnya
12 # Apabila tebakan pengguna salah, program akan kembali
   meminta pengguna untuk menebak bilangan tersebut dengan
   memberikan petunjuk bahwa tebakannya terlalu kecil atau
   terlalu besar
13 while tebakan != bilangan:
14  jumlah_tebakan += 1
15  if tebakan < bilangan:
16   tebakan = int(input('''
17     Tebakan Anda masih salah.
18     Anda harus menebak bilangan yang lebih besar lagi.
19     Berapakah bilangan yang telah dipilih?
20     Tebakan Anda: '''))
21  else:
22   tebakan = int(input('''
23     Tebakan Anda masih salah.
24     Anda harus menebak bilangan yang lebih kecil lagi.
25     Berapakah bilangan yang telah dipilih?
26     Tebakan Anda: '''))
27
28 # Apabila tebakan pengguna benar, program akan menampilkan
   berapa kali pengguna telah melakukan tebakan
29 print(f'Anda berhasil menebak dengan benar setelah
   {jumlah_tebakan} kali menebak.')
```

Baris ke-15 pada `Kode Program 171` memiliki arti bahwa jika bilangan yang dipilih oleh pengguna lebih kecil daripada bilangan yang telah ditentukan oleh program, program akan menjalankan baris-baris kode ke-16 sampai ke-20. Baris-baris kode di dalam ***if statement*** ini sebenarnya serupa dengan baris kode ke-15 pada kode program sebelumnya. Namun, di sini penulis memodifikasi informasi yang ditampilkan di kolom **Console** kepada pengguna agar pengguna mengetahui bahwa bilangan yang dipilihnya masih lebih kecil dan program meminta pengguna untuk menebak bilangan yang lebih besar lagi.

Akan tetapi, baris ke-21 kode program tersebut yang menggunakan ***else statement*** memiliki arti sebaliknya, yaitu jika bilangan yang dipilih oleh pengguna lebih besar daripada bilangan yang telah ditentukan oleh program, program akan menjalankan baris-baris kode ke-22 sampai ke-26. Di sini, program akan memberitahukan kepada pengguna di kolom **Console** bahwa pengguna harus menebak bilangan yang lebih kecil lagi dari bilangan yang telah ditebak oleh pengguna sebelumnya. Diagram alur dari program setelah penambahan ***if statement*** adalah seperti yang diperlihatkan pada Gambar 7.2. Selain menggunakan **`else:`** pada contoh tersebut, Anda juga dapat menggunakan **`elif tebakan > bilangan:`**. Jangan lupa apa pun yang Anda pilih, posisi awal baris kodenya harus sejajar dengan posisi ***if statement*** yang Anda gunakan pada baris ke-15 pada kode program tersebut.

Perhatikan juga bahwa baris ke-4 kode program tersebut penulis ubah kembali menjadi menghasilkan bilangan bulat acak antara 1 dan 100. Pada saat Anda menjalankan program tersebut, contoh tampilan pada kolom **Console** akan seperti yang diperlihatkan pada `Console 113`. Dari tampilan kolom **Console** tersebut dapat terlihat bahwa program telah berinteraksi dengan pengguna dengan memberitahukan kepada pengguna bahwa tebakannya masih terlalu besar atau kecil sampai tebakan pengguna sama dengan bilangan yang telah ditentukan oleh program. Program juga menginformasikan kepada pengguna jumlah tebakan yang telah dilakukan sampai pengguna tersebut menebak bilangan dengan benar.



Gambar 7.2 Diagram Alur Program setelah Penambahan *If Statement* di dalam *While Loop*

Console 113

```
Berapakah bilangan yang telah dipilih?
Tebakan Anda: 50

Tebakan Anda masih salah.
Anda harus menebak bilangan yang lebih besar lagi.
Berapakah bilangan yang telah dipilih?
Tebakan Anda: 75

Tebakan Anda masih salah.
Anda harus menebak bilangan yang lebih besar lagi.
Berapakah bilangan yang telah dipilih?
Tebakan Anda: 83

Tebakan Anda masih salah.
Anda harus menebak bilangan yang lebih kecil lagi.
Berapakah bilangan yang telah dipilih?
Tebakan Anda: 80
```

```
Tebakan Anda masih salah.
Anda harus menebak bilangan yang lebih besar lagi.
Berapakah bilangan yang telah dipilih?
Tebakan Anda: 81

Tebakan Anda masih salah.
Anda harus menebak bilangan yang lebih besar lagi.
Berapakah bilangan yang telah dipilih?
Tebakan Anda: 82
Anda telah berhasil menebak dengan benar setelah 5 kali
menebak.
>
```

Sampai dengan bagian ini sebenarnya program Anda telah selesai dan siap untuk digunakan. Namun, ada beberapa hal yang penulis ingin sampaikan agar para pengguna program Anda tahu apa yang harus mereka lakukan ketika mereka menjalankan program Anda.

Pertama, Anda disarankan untuk memberikan informasi singkat terkait dengan program pada saat program dijalankan. Dalam hal program permainan tebak angka ini, Anda dapat memberikan informasi kepada pengguna terkait dengan batasan bilangan yang dipilih secara acak oleh program seperti yang diperlihatkan pada `Kode Program 172`. Pada tampilan kolom **Console** ketika kode program tersebut dijalankan seperti yang diperlihatkan pada `Console 114`, program akan memberikan ucapan selamat datang secara singkat dan kemudian program memberitahukan kepada pengguna bahwa bilangan yang harus ditebak adalah berada di dalam rentang 1 sampai 100 sehingga pengguna tidak akan tiba-tiba menebak bilangan di luar rentang tersebut (misal 1.000 atau 1.000.000).

Kode Program 172

```
print('Selamat datang di dalam permainan tebak angka.')
print('Pada permainan ini Anda diminta untuk menebak bilangan di antara 1 dan 100.')
# Program menentukan sebuah bilangan antara 1 dan 100 secara acak
import random

bilangan = random.randint(1, 100)

# Program meminta pengguna untuk menebak bilangan tersebut
# Pengguna menebak bilangan yang telah ditentukan program
tebakan = int(input('Berapakah bilangan yang telah dipilih? Tebakan Anda: '))
jumlah_tebakan = 1

# Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
# Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
while tebakan != bilangan:
 jumlah_tebakan += 1
 if tebakan < bilangan:
  tebakan = int(input('''
    Tebakan Anda masih salah.
    Anda harus menebak bilangan yang lebih besar lagi.
    Berapakah bilangan yang telah dipilih?
    Tebakan Anda: '''))
 else:
  tebakan = int(input('''
    Tebakan Anda masih salah.
    Anda harus menebak bilangan yang lebih kecil lagi.
    Berapakah bilangan yang telah dipilih?
    Tebakan Anda: '''))

# Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
print(f'Anda berhasil menebak dengan benar setelah {jumlah_tebakan} kali menebak.')
```

Console 114

```
Selamat datang di dalam permainan tebak angka.
Pada permainan ini Anda diminta untuk menebak bilangan di
antara 1 dan 100.
Berapakah bilangan yang telah dipilih?
Tebakan Anda:
```

Kedua, Anda juga dapat memberikan sedikit jeda waktu setiap kali program menampilkan pesan di kolom **Console** sehingga pengguna dapat merasakan seolah-olah program Anda tersebut "hidup". Jeda waktu dapat digunakan dengan menggunakan modul **`time`** yang tersedia di dalam bahasa pemrograman Python. Oleh karena itu, Anda harus mengimpornya terlebih dahulu ke dalam kode program Anda dengan menggunakan perintah **`import time`**. Untuk menggunakannya, Anda dapat mengetikkan perintah **`time.sleep()`** dan memberikan bilangan ***integer*** di dalam tanda kurungnya. Misalkan, Anda ingin memberi jeda selama 3 detik, Anda dapat mengetikkan **`time.sleep(3)`** pada baris kode Anda. Contoh penggunaan jeda waktu di dalam program permainan tebak angka yang telah Anda buat adalah seperti yang diperlihatkan pada Kode Program 173.

Kode Program 173

```
1  import time
2
3  print('Selamat datang di dalam permainan tebak angka.')
4  time.sleep(2)
5  print('Pada permainan ini Anda diminta untuk menebak
   bilangan di antara 1 dan 100.')
6  time.sleep(3)
7
8  # Program menentukan sebuah bilangan antara 1 dan 100
   secara acak
9  print('Program akan memilih bilangan di antara 1 dan 100
   secara acak.')
```

```
time.sleep(3)
print('Memilih bilangan...')
time.sleep(2)

import random

bilangan = random.randint(1, 100)

# Program meminta pengguna untuk menebak bilangan tersebut
# Pengguna menebak bilangan yang telah ditentukan program
tebakan = int(input('Berapakah bilangan yang telah dipilih? Tebakan Anda: '))
jumlah_tebakan = 1

# Program membandingkan tebakan pengguna dengan bilangan yang telah ditentukan sebelumnya
# Apabila tebakan pengguna salah, program akan kembali meminta pengguna untuk menebak bilangan tersebut dengan memberikan petunjuk bahwa tebakannya terlalu kecil atau terlalu besar
while tebakan != bilangan:
 jumlah_tebakan += 1
 if tebakan < bilangan:
  tebakan = int(input('''
    Tebakan Anda masih salah.
    Anda harus menebak bilangan yang lebih besar lagi.
    Berapakah bilangan yang telah dipilih?
    Tebakan Anda: '''))
 else:
  tebakan = int(input('''
    Tebakan Anda masih salah.
    Anda harus menebak bilangan yang lebih kecil lagi.
    Berapakah bilangan yang telah dipilih?
    Tebakan Anda: '''))

# Apabila tebakan pengguna benar, program akan menampilkan berapa kali pengguna telah melakukan tebakan
print(f'Anda berhasil menebak dengan benar setelah {jumlah_tebakan} kali menebak.')
```

Coba ketikkan contoh tersebut ke dalam bagian awal program Anda dan jalankan program tersebut. Anda akan merasakan bahwa program Anda seakan-akan menjadi lebih "hidup" karena Anda melihat program Anda tersebut "berpikir". Memang Anda juga akan merasakan program Anda tersebut berjalan lebih lambat. Jadi, sebenarnya fungsi jeda waktu tersebut hanyalah pilihan tambahan dari fitur-fitur di dalam program Anda.

Selamat! Anda telah berhasil menyelesaikan proyek pemrograman dengan menggunakan bahasa Python ketiga Anda yang merupakan proyek terakhir di dalam buku ini. Anda pasti merasa sangat bangga dengan pencapaian Anda tersebut.

Bab 8 Penutup

Selamat! Anda telah beranjak dari yang sebelumnya benar-benar pemula menjadi seseorang yang dapat membuat beberapa proyek pemrograman dengan menggunakan bahasa Python. Apabila Anda ingin terus mengembangkan kemampuan Anda di dalam penguasaan bahasa pemrograman Python, kini Anda dapat mencoba untuk memasang berbagai aplikasi untuk mendukung pemrograman dengan menggunakan bahasa Python pada komputer Anda, baik itu aplikasi yang disediakan oleh pengembang resmi bahasa pemrograman Python maupun aplikasi-aplikasi pihak ketiga yang biasanya banyak digunakan oleh para pemrogram sehingga Anda kini dapat membuat program Anda sendiri pada komputer Anda, tidak harus menggunakan peramban (*browser*) saja.

Penulis sengaja tidak menyebutkan dan menjelaskan aplikasi-aplikasi tersebut lebih lanjut agar Anda dapat berusaha mencari sendiri informasi terkait hal tersebut. Anggap saja ini bagian dari tantangan

yang biasa penulis berikan di setiap akhir pembahasan sebuah konsep di dalam buku ini.

Seperti yang penulis katakan pada bagian awal buku ini, Python merupakan salah satu bahasa pemrograman yang paling populer saat ini. Banyak sekali aplikasi-aplikasi populer yang dibangun dengan menggunakan bahasa pemrograman Python, mulai dari aplikasi-aplikasi berbasis situs web hingga aplikasi-aplikasi berbasis *mobile* atau yang sering Anda gunakan di dalam telepon genggam Anda.

Penulis berharap, setelah menyelesaikan buku ini, Anda dapat bergabung dengan komunitas-komunitas pemrogram. Anda dapat belajar lebih mendalam jargon-jargon yang ada di dalam bahasa pemrograman Python di sana. Anda juga dapat menjalin pertemanan dengan para pemrogram tersebut, menanyakan informasi yang masih belum jelas ketika Anda mempelajari bahasa pemrograman Python, bahkan mendapatkan tantangan proyek-proyek pemrograman yang berkaitan dengan bahasa pemrograman Python. Jangan lupa untuk secara rutin mengunjungi halaman situs resmi pengembangan bahasa pemrograman Python di www.python.org untuk memperoleh informasi-informasi terkini terkait bahasa pemrograman Python tersebut.

Terakhir, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada Anda yang telah menyelesaikan pembelajaran sampai akhir. Penulis berharap Anda dapat menikmati buku ini sebagaimana penulis menikmati ketika menyusunnya.

Daftar Pustaka

divotomezove. (2021, 25 September). *Sampah plastik, sampah, mendaur ulang* [Foto]. Pixabay. https://pixabay.com/id/illustrations/sampah-plastik-sampah-mendaur-ulang-6644673/.

Enterprise, J. (2016). *Trik cepat menguasai pemrograman Python: Membantu Anda mempelajari pemrograman Python secara cepat dan mudah*. Elex Media Komputindo.

Enterprise, J. (2017). *Otodidak pemrograman Python: Referensi paling pas untuk menguasai Python*. Elex Media Komputindo.

Enterprise, J. (2019). *Python untuk programmer pemula: Buku bergizi untuk programmer pemula yang ingin mempelajari Python*. Elex Media Komputindo.

fawaz-qureshi. (2023, 6 Juli). *Kaca, jus, buah mangga* [Foto]. Pixabay. https://pixabay.com/id/illustrations/kaca-jus-buah-mangga-jus-mangga-8108589/

Huda, A., & Ardi, N. (2020). *Dasar-dasar pemrograman berbasis Python*. UNP Press.

JUD. (2016). *Pemrograman Python untuk pemula: Mengupas langkah demi langkah mengenal, menguasai & menggunakan pemrograman Python bagi para pemula*. CV Jubilee Solusi Enterprise.

Mastrodomenico, R. (2022). *The Python book*. John Wiley and Sons Ltd.

mojzagrebinfo. (2014, 30 Juli). *Tape, nature, apel [Foto]*. Pixabay. https://pixabay.com/id/photos/tape-apel-gelas-air-biru-diet-403590/

OpenClipart-Vectors. (2013, 13 Juli). *Mangkuk anjing, baki, mangkuk* [Foto]. Pixabay. https://pixabay.com/id/vectors/mangkuk-anjing-baki-mangkuk-merah-162044/

Penelope883. (2022, 10 Juni). *Kopi, coffecup, cangkir* [Foto]. Pixabay. https://pixabay.com/id/photos/kopi-cangkir-minum-pembakaran-7250123/

PDPics. (2014, 15 Juli). *Dictionary, words, grammar image* [Foto]. Pixabay. https://pixabay.com/photos/dictionary-words-grammar-abc-390055/

Python. (2023). *Browse the docs online or download a copy of your own*. https://www.python.org/doc/

Ragabz. (2021, 15 Januari). *Kue, susu, chocolate chip cookie* [Foto]. Pixabay. https://pixabay.com/id/photos/kue-susu-chocolate-chip-cookie-5893040/

Rangkuti, Y. M., Al Idrus, S. I., & Tarigan, D. D. (2021). *Pengantar pemrograman Python: Dilengkapi 200++ tutorial based on Pycharm* (R. R. Rerung (ed.)). Media Sains Indonesia.

Rastati, R. (2022). *Metaverse, membangun kehidupan dalam dunia virtual: Menjanjikan tapi juga potensial bermasalah*. The Conversation. https://theconversation.com/metaverse-membangun-kehidupan-dalam-dunia-virtual-menjanjikan-tapi-juga-potensial-bermasalah-174547

Supardi, Y., & Dede. (2020). *Semua bisa menjadi programmer Python: Case study*. Elex Media Komputindo.

Tentang Penulis

Budhi Gustiandi setelah menyelesaikan jenjang pendidikan S-1 di Teknik Elektro Institut Teknologi Bandung (ITB), melanjutkan pendidikan S-2 di The University of Melbourne, Australia. Penulis mengambil program Master of Information Systems. Kembali ke tanah air, penulis bekerja sebagai pegawai negeri sipil (PNS) di Pusat Teknologi dan Data Penginderaan Jauh (Pustekdata), Lembaga Penerbangan dan Antariksa Nasional (LAPAN), yang kemudian bergabung di Badan Riset dan Inovasi Nasional (BRIN). Hampir semua kegiatan yang penulis lakukan di institusi berkaitan dengan pemrograman, terutama yang berkaitan dengan proses akuisisi dan pengolahan data satelit pengindraan jauh yang diterima di Indonesia. Tidak kurang dari 50 publikasi di bidang pemrograman dan implementasinya di bidang pengindraan jauh telah penulis hasilkan, baik sendiri maupun bersama-sama tim, di lingkup nasional dan internasional. Untuk korespondensi, penulis dapat dihubungi melalui budhigustiandi@gmail.com.

Indeks

Salah satu permasalahan yang ditemui oleh para pemrogram pemula dalam mempelajari sebuah bahasa pemrograman baru adalah aspek teknis yang harus mereka persiapkan. Sebut saja, misalnya, aplikasi-aplikasi apa saja yang harus mereka pasang atau instal di komputer untuk menunjang pembelajaran mereka dan jargon-jargon apa saja yang harus mereka ketahui dalam memahami alur pembelajaran pada sebuah buku. Terkadang hal-hal "kecil" seperti inilah yang membuat para pembaca menjadi tidak betah dan menyerah dalam mempelajari sebuah bahasa pemrograman. Tentu saja hal-hal tersebut tidak akan Anda temui di dalam buku ini.

Anda akan dipandu untuk mempelajari bahasa pemrograman Python dengan pendekatan penjelasan istilah dan implementasi melalui analogi-analogi di dunia nyata. Tantangan-tantangan kecil disajikan di setiap langkahnya untuk memastikan Anda memahami setiap konsep yang diberikan dan siap untuk menerapkannya dalam proyek-proyek yang akan Anda bangun seiring Anda belajar. Mampukah Anda menguasai dasar-dasar dari bahasa pemrograman Python tanpa ada pengalaman membuat sebuah program sebelumnya? Dengan buku ini, Anda pasti bisa!

BRIN Publishing
The Legacy of Knowledge

Diterbitkan oleh:
Penerbit BRIN, Anggota Ikapi
Gedung B.J. Habibie Lantai 8,
Jl. M.H. Thamrin No. 8,
Jakarta Pusat 10340
E-mail: penerbit@brin.go.id
Website: penerbit.brin.go.id

DOI: 10.55981/brin.656

ISBN 978-623-8372-21-8