**Rangkuman Kelas**

**Rangkuman Berkenalan dengan Python**

Kita sudah berada di penghujung materi pertama. Sampai sini Anda sudah memiliki pengetahuan mendasar mengenai Python. Mari kita rangkum secara saksama.

**Pengenalan Python**

Python adalah bahasa pemrograman multifungsi yang dirilis pada tahun 1991 oleh Guido van Rossum (GvR). Beliau membuat Python sebagai bahasa pemrograman yang mudah dibaca dan dimengerti (*readable*) serta memiliki kemampuan penanganan kesalahan (*exception handling*).

Python memiliki ciri khas tersendiri sebagai salah satu pemrograman populer. Salah satu ciri khas yang paling dikenal adalah Python tidak mewajibkan penggunaan titik koma atau semi colon (;) pada setiap akhir kode programnya.

**Bersiap Membuat Kode Program di Lokal**

Untuk menjalankan program Python di lokal komputer, Anda perlu mempersiapkan dua hal, yakni menginstal Python dan menyiapkan Integrated Development Environment (IDE).

Banyak cara yang dapat dilakukan untuk melakukan instalasi Python, semuanya bergantung pada *operating system*(OS) atau sistem operasi yang Anda gunakan serta sumber yang dipilih. Umumnya, Anda dapat mengakses laman berikut untuk mengunduh Python: <https://www.python.org/downloads/>.

Integrated Development Environment (IDE) merupakan sebuah aplikasi yang menyediakan fasilitas komprehensif untuk pengembangan aplikasi bagi para programmer. IDE memiliki banyak fitur, salah satunya adalah kode editor yang mengizinkan Anda untuk membuat dan mengubah kode program.

Ada banyak sekali IDE populer untuk Python yang ramai digunakan oleh programmer. Beberapa di antaranya sebagai berikut.

1. **Visual Studio Code**  
   Visual Studio Code adalah IDE populer yang telah digunakan oleh banyak programmer. Visual Studio Code menyediakan ribuan plugin yang dapat membantu programmer untuk membuat program.
2. **PyCharm**  
   PyCharm adalah IDE yang dibuat khusus untuk pengembangan aplikasi menggunakan bahasa pemrograman Python. Dengan tujuan tersebut, banyak fitur-fitur khusus yang diberikan oleh PyCharm kepada programmers untuk mempermudah proses pengembangan aplikasi Python.
3. **Jupyter Notebook**  
   Jupyter Notebook adalah IDE berbasis web yang memungkinkan Anda untuk membuat, berbagi kode program, serta berkolaborasi dengan programmer lain. Jupyter Notebook terdiri dari beberapa sel di dalamnya. Setiap sel dapat berisi kode ataupun teks. Setiap sel tersebut dapat dijalankan satu per satu dan menampilkan hasilnya tanpa harus membangun semua kode terlebih dahulu.
4. **Google Colaboratory**  
   Google Colaboratory adalah IDE berbasis web online yang memiliki fungsi sama seperti Jupyter Notebook. Dengan Google Colaboratory, Anda tidak perlu melakukan instalasi seperti yang dilakukan ketika ingin menggunakan Jupyter Notebook.

**Menjalankan Kode Program di Lokal**

Untuk menjalankan kode program Python di lokal komputer, Anda dapat menjalankannya menggunakan tiga mode.

1. **Kode Interaktif**  
   Mode kode interaktif memungkinkan Anda menjalankan kode Python hanya berbasis terminal/command prompt. Mode ini biasanya digunakan para programmer untuk bereksplorasi dan menjalankan dua sampai tiga baris kode saja. Pastikan Anda telah menginstal Python ketika ingin menggunakan kode interaktif.
2. **Script**  
   Mode scriptadalah mode yang paling sering digunakan oleh programmer untuk membuat kode program. Sederhananya, Anda akan membuat sebuah file (disebut sebagai script) dengan ekstensi “.py”. Kemudian untuk bisa menjalankan kode di dalamnya, Anda perlu mengeksekusi file tersebut.
3. **Notebook**  
   Mode Notebook merujuk pada lingkungan pengembangan kode interaktif, seperti Jupyter Notebook atau Google Colaboratory.

**Variable dan Assignment**

Variabel merupakan lokasi dalam komputer yang digunakan untuk menyimpan nilai dengan tipe data tertentu. Ketika menuliskan variabel, Anda telah memerintahkan komputer untuk mencari dan memesan ruang kosong dalam komputer yang nantinya akan diisi oleh nilai atau data. Proses pemberian atau penetapan nilai pada sebuah variabel disebut dengan assignment. Umumnya, format assignment adalah berikut.

|  |
| --- |
| <Ruas Kiri> = <Ruas Kanan> |

Ruas kiri adalah variabel dan ruas kanan dapat berupa ekspresi/nilai/variabel yang sudah jelas nilainya.

**Input/Output dan Komentar**

Dalam membuat kode program, Anda dapat menetapkan nilai secara langsung ataupun mengizinkan pengguna menentukannya. Proses ini disebut dengan input. Untuk melakukannya, Anda dapat menggunakan perintah “input()”.

1. input()

Lalu, munculkan hasil keluarannya (output) dengan menggunakan perintah “print()”.

1. print()

Terakhir, Anda dapat memberikan komentar pada program yang dibangun untuk memberikan konteks pada kode. Komentar merupakan barisan teks yang akan diabaikan oleh Python ketika program tersebut dijalankan.

Ada dua tipe komentar dalam Python sebagai berikut.

1. **Inline Comment**  
   Inline comment merupakan satu baris komentar yang biasanya diletakkan pada baris yang sama dengan kode atau satu baris sebelum kode.
   1. # Variabel ini menyimpan nama 'Dicoding Indonesia'
   2. name = 'Dicoding Indonesia'

Dengan menggunakan tanda pagar “#”, program yang Anda bangun akan menganggap baris tersebut adalah komentar sehingga tidak akan dijalankan dan memunculkan error.

1. **Block Comment**  
   Block comment merupakan satu blok kode bertujuan untuk menjelaskan kode kompleks atau membuat dokumentasi dari sebuah fungsi atau modul. Anda dapat mengapit blok teks dengan tiga buah single quote (''') atau double quote ("") untuk menjadikannya sebagai blok komentar.
   1. """
   2. Ini adalah Block Comment,
   3. Teks ini akan diabaikan oleh Python.
   4. """
   5. print("Hello World!")

Dengan menggunakan tiga double quote ("""), program yang Anda bangun akan menganggap blok tersebut adalah komentar sehingga tidak akan dijalankan dan memunculkan error.  
  
Selain itu, berikut adalah implementasi block comment menggunakan single quote (‘’’).

* 1. '''
  2. Ini adalah Block Comment,
  3. Teks ini akan diabaikan oleh Python.
  4. '''
  5. print("Hello World!")

Kedua cara tersebut sama-sama mengarahkan program Anda untuk menganggap teks di dalamnya sebagai komentar, sehingga ketika dijalankan tidak akan memunculkan error.

**Rangkuman Berinteraksi dengan Data**

Kita sudah berada di penghujung materi Berinteraksi dengan Data. Sampai sini, Anda telah memiliki pemahaman mendasar mengenai data dalam Python. Mari kita rangkum secara saksama.

**Abstraksi Data**

Abstraksi data merupakan kemampuan Anda untuk mengerti konteks dan merepresentasikannya menjadi bentuk lain sesuai dengan konteks masalahnya.

Ketika menuliskan data dalam pemrograman, komputer tidak akan mengetahui data yang dimaksud hingga Anda mendeklarasikan tipe datanya.

**Data Typing**

Dalam pemrograman, Anda harus mengenali istilah deklarasi dan inisialisasi yang umum ditemui ketika membuat sebuah program.

**Deklarasi dan Inisialisasi**

Deklarasi merujuk pada pembuatan variabel dengan menentukan tipe data dan nama variabelnya. Contohnya seperti berikut yang merupakan deklarasi dalam pemrograman C/C++.

1. int age;
2. float salary;

Inisialisasi merujuk kepada pemberian nilai awal pada variabel yang sebelumnya telah dideklarasikan. Contohnya seperti berikut yang merupakan deklarasi dalam pemrograman C/C++.

1. int age = 17;
2. float salary = 5000000;

Dalam Python, Anda tidak diharuskan mendeklarasikan tipe data variabel. Anda dapat langsung melakukan inisialisasi variabel.

1. age = 17
2. salary = 5000000.0
4. print(type(age))
5. print(type(salary))
7. """
8. Output:
9. <class 'int'>
10. <class 'float'>
12. """

**Tipe Data**

Sebuah data memiliki tipe yang berbeda-beda. Dalam Python, Tipe data dapat dikelompokkan menjadi **tipe data primitif** dan **tipe data collection**.

**Tipe Data Primitif**

Tipe data primitif merupakan jenis data yang paling dasar dalam pemrograman. Tipe data ini menyimpan *single value*. Beberapa tipe data primitif sebagai berikut.

1. **Numbers**
   1. Integer: Bilangan bulat positif atau negatif dan tidak memiliki angka desimal. Contoh: 1; -20; 999; dan 0.
   2. Float: Bilangan riil yang dapat mewakili bilangan bulat atau bilangan desimal. Contoh: 3.14; 1; dan 4.01E+1
   3. Complex: Bilangan kompleks. **(Kita tidak akan menggunakannya di kelas ini.)**Contoh: 1+2j
2. **Boolean**  
   Boolean merupakan tipe data yang hanya bernilai TRUE atau FALSE. Tipe data ini merepresentasikan nilai kebenaran (*truth values*).

|  |
| --- |
| True False |

1. Sebenarnya, setiap variabel yang memiliki nilai bisa dievaluasi dan menghasilkan nilai *true*. Hanya ada beberapa nilai yang akan dianggap bernilai false, yakni berikut.
   1. Nilai yang sudah didefinisikan bernilai salah: **None dan False**.
   2. Angka nol dari semua tipe numerik: **0, 0.0, 0j, Decimal(0), Fraction(0,1)**.
   3. Urutan (*sequence*) dan koleksi (*collection*) yang kosong: **"", (), {}, set(), range(0)**.
2. **String**  
   String merupakan karakter yang berurutan. Ketika Anda membuat variable bernilai string maka akan diawali dengan *single quote*('') atau *double quote*("").
   1. "Dicoding Indonesia"

**Tipe Data Collection**

Selain tipe data primitif yang menyimpan *single value*, ada tipe data lain, yakni **tipe data collection**. Tipe data ini menyimpan satu atau lebih data primitif sebagai satu kelompok. Berikut adalah tipe data yang masuk ke dalam tipe data collection.

1. List  
   List merupakan jenis kumpulan data terurut (ordered sequence) dan salah satu tipe data yang sering digunakan pada Python.  
     
   Melakukan inisialisasi list pada Python cukup mudah: menggunakan kurung siku "[]" dan setiap elemennya dipisahkan dengan koma.
   1. x = [1, 2.2, "Dicoding"]

Perlu diingat bahwa nilai yang ada dalam sebuah list selalu dimulai dari indeks ke-0. Artinya, nilai "1" pada list di atas merupakan indeks ke-0.

1. **Tuple**  
   Tuple merupakan jenis dari list yang tidak dapat diubah elemennya. Umumnya, tuple digunakan untuk data yang bersifat sekali deklarasi dan dapat dieksekusi lebih cepat. Anda dapat mendeklerasikan tuple dengan menggunakan tanda kurung "()" dan setiap elemen di dalamnya dipisahkan dengan koma.
   1. x = (1, "Dicoding", 1+3j)
2. **Set**  
   Set merupakan kumpulan item bersifat unik dan tanpa urutan (unordered collection). Anda dapat melakukan inisialisasi variabel set dengan menggunakan tanda kurawal "{}" dan setiap elemennya dipisahkan dengan koma.
   1. x = {1, 2, 3 , 7, 13}
3. **Dictionary**  
   Dictionary pada Python merupakan kumpulan pasangan key-valueyang bersifat tidak berurutan. Dictionary dapat digunakan untuk menyimpan data kecil hingga besar. Pada Python, dictionary didefinisikan dengan kurawal dan tambahan definisi berikut.
   1. Setiap elemen pasangan *key-value* dipisahkan dengan koma (,).
   2. Key dan valuedipisahkan dengan titik dua (:).
   3. Key dan valuedapat berupa tipe variabel/objek apa pun.
      1. x = { 'name': 'Perseus Evans', 'age': 20, 'isMarried': False }

**Konversi antara Tipe Data**

Anda pun dapat melakukan konversi antar tipe data dengan menggunakan beberapa fungsi yang tersedia pada Python. Fungsi merupakan blok kode yang dapat dipanggil untuk melakukan tugas tertentu. Beberapa fungsi yang dapat digunakan untuk mengonversi tipe data pada Python sebagai berikut.

1. Konversi integer ke float: **float().**
2. Konversi float ke integer: **int().**
3. Konversi dari-dan-ke string: **str(), float(), int().**

**Transformasi Angka, Karakter, dan String**

Khusus tipe data string, terdapat berbagai fungsi untuk mentransformasikan tipe data string menjadi bentuk lain. Beberapa di antaranya berikut.

1. **Mengubah Huruf Besar/Kecil**  
   Mengubah string menjadi UPPERCASE atau lowercase.
   1. upper()
   2. lower()
2. **Awalan dan Akhiran**  
   Menghapus karakter *whitespace*pada suatu string.
   1. rstrip()
   2. lstrip()
   3. strip()
   4. startswith()
   5. endswith()
3. **Memisah dan Menggabung String**  
   Fungsi-fungsi untuk memisahkan dan menggabungkan string.
   1. join()
   2. split()
4. **Mengganti Elemen String**  
   Metode yang bertujuan untuk mengganti elemen string dengan elemen string lainnya.
   1. replace()
5. **Pengecekan String**  
   Fungsi-fungsi yang bertujuan untuk melakukan pengecekan pada string dan mengembalikan nilai Boolean.
   1. isupper()
   2. islower()
   3. isalpha()
   4. isalnum()
   5. isdecimal()
   6. isspace()
   7. istitle()
6. **Formatting pada String**  
   Fungsi-fungsi untuk pemformatan string.
   1. zfill()
   2. rjust()
   3. ljust()
   4. center()
7. **String Literals**  
   String literals adalah serangkaian karakter yang diapit oleh tanda kutip baik tunggal (') maupun ganda (").  
     
   String dapat ditulis mudah dalam Python dengan cara diapit oleh tanda petik tunggal. Namun, dalam kondisi tertentu, dibutuhkan petik tunggal di tengah string (misalnya struktur kepemilikan dalam Bahasa Inggris—Dicoding's Cat atau penyebutan Jum'at pada hari dalam bahasa Indonesia).  
     
   Solusinya adalah menggunakan *escape character* yang memungkinkan Anda untuk menggunakan karakter yang sebelumnya tidak bisa dimasukkan dalam string. Umumnya diawali dengan *backslash* (\) dan diikuti karakter tertentu yang diinginkan. Beberapa contoh *escape character*sebagai berikut.
   1. \' Single quote
   2. \" Double quote
   3. \t Tab
   4. \n Newline (*line break*)
   5. \\ Backslash
8. **Raw String**  
   Merupakan cara untuk mencetak string sesuai dengan apa pun input atau teks yang diberikan. Umumnya digunakan untuk regex atau beberapa implementasi lain yang sangat bergantung pada keberadaan backslash. Untuk mengimplementasikan raw strings, sisipkan huruf r sebelum pembuka string.
   1. print(r'Dicoding\tIndonesia')
   3. """
   4. Output:
   5. Dicoding\tIndonesia
   6. """

**Operasi pada List, Set, dan String**

Ada beberapa fungsi untuk melakukan operasi pada list, set, dan string. Berikut beberapa di antaranya.

1. **len()**  
   Fungsi yang bertujuan untuk menghitung panjang atau banyaknya elemen dari list, set, dan string.
   1. contoh\_list = [1, 3, 3, 5, 5, 5, 7, 7, 9]
   3. print(contoh\_list)
   4. print(len(contoh\_list))
   6. """
   7. Output:
   8. [1, 3, 3, 5, 5, 5, 7, 7, 9]
   10. 9
   11. """
2. **min() dan max()**  
   Fungsi yang digunakan untuk mengetahui nilai minimum dan maksimum dari suatu list.
   1. angka = [13, 7, 24, 5, 96, 84, 71, 11, 38]
   2. print(min(angka))
   3. print(max(angka))
   5. """
   6. Output:
   7. 5
   8. 96
   9. """
3. **count()**  
   Fungsi yang digunakan untuk mengetahui berapa kali suatu objek muncul dalam list.
   1. genap = [2, 4, 4, 6, 6, 6, 8, 10, 10]
   2. print(genap.count(6))

   5. """
   6. Output:
   7. 3
   8. """
4. **In dan Not In**  
   **In** dan **not in** merupakan operator yang diperuntukkan untuk mengetahui nilai atau objek yang ada dalam list. Anda bisa menggunakan operator ini untuk memastikan suatu nilai ada dalam list bahkan dalam string. Operatori **in** dan **not in**akan mengembalikan nilai boolean True atau False.
   1. kalimat = "Belajar Python di Dicoding sangat menyenangkan"
   2. print('Dicoding' in kalimat)
   3. print('tidak' in kalimat)
   4. print('Dicoding' not in kalimat)
   5. print('tidak' not in kalimat)
   7. """
   8. Output:
   9. True
   10. False
   11. False
   12. True
   13. """
5. **Memberikan Nilai untuk Multiple Variable**Anda dapat memberikan nilai untuk multiple variable dengan cara berikut.
   1. data = ['shirt', 'white', 'L']
   2. apparel, color, size = data
   4. print(data)
   6. """
   7. Output:
   8. ['shirt', 'white', 'L']
   9. """
6. **sort()**  
   Fungsi sort() digunakan untuk mengurutkan angka atau huruf. Berikut contoh implementasinya.
   1. kendaraan = ['motor', 'mobil', 'helikopter', 'pesawat']
   2. kendaraan.sort()
   4. print(kendaraan)
   6. """
   7. Output:
   8. ['helikopter', 'mobil', 'motor', 'pesawat']
   9. """

**Rangkuman Ekspresi**

Kita sudah berada di penghujung materi Ekspresi. Sampai sini, Anda memiliki pemahaman mendasar mengenai ekspresi yang akan sering Anda jumpai ketika membuat program Python. Mari kita rangkum secara saksama.

**Pengertian Ekspresi**

Ekspresi pada pemrograman merupakan kombinasi dari satu atau lebih variabel, konstanta, operator, dan fungsi yang bermakna untuk menghasilkan suatu nilai dalam tipe tertentu.

**Jenis-Jenis Ekspresi**

Umumnya ekspresi dibagi menjadi dua, yakni **menurut arity dari operator**dan **menurut tipe data yang dihasilkan**.

1. **Ekspresi Menurut Arity dari Operator**
   1. Biner  
      Jenis ekspresi yang memiliki dua operan. Beberapa contohnya adalah **(x + y), (x - y), (x \* y)**, dan sebagainya.
   2. Uner  
      Jenis ekspresi yang memiliki bentuk dasar operasi dengan satu operan. Contohnya adalah increment (x+=1), decrement (x-=1), dan negasi (not x).
2. **Ekspresi Menurut Tipe Data yang Dihasilkan**  
   Jenis ekspresi yang dikelompokkan berdasarkan tipe data yang dihasilkan. Berikut adalah penjelasan lebih detailnya.

| **Jenis** | **Contoh** |
| --- | --- |
| Ekspresi aritmetika:  <numerik> <operator> <numerik> = <numerik> | 2 + 2 = 4, 2 - 2 = 0 |
| Ekspresi relasional:  <numerik> <operator> <numerik> = <boolean> | 3 < 10 = True, 1 > 10 = False |
| Ekspresi logika:  <boolean> <operator> <boolean> = <boolean> | True or False = True |

**Jenis-Jenis Operator**

Selain ekspresi yang memiliki beragam jenis, operator pun memiliki berbagai jenis yang dikelompokkan menjadi operator aritmetika, operator relasional, operator logika, dan operator assignment.

1. **Operator Aritmetika**  
   Operator aritmetika merupakan jenis operator untuk melakukan operasi aritmetika. Perhatikan tabel di bawah ini untuk memahami contoh penerapan operator aritmetika. **Asumsikan x bernilai 11 dan y bernilai 5.**

| **Operator** | **Deskripsi** | **Contoh** |
| --- | --- | --- |
| Penjumlahan (+) | Menambahkan nilai dari kedua operan. | x + y = 16 |
| Pengurangan (-) | Mengurangi nilai dari kedua operan. | x - y = 6 |
| Perkalian (\*) | Mengalikan nilai dari kedua operan. | x \* y = 55 |
| Pembagian bulat (//) | Membagi nilai dari kedua operan. Jika operan adalah integer, hasil operasi adalah bilangan bulat. | x // y = 2 |
| Pembagian riil (/) | Membagi nilai dari kedua operan. Jika operan adalah float, hasil operasi adalah bilangan riil. | x / y = 2.2 |
| Modulo (%) | Sisa hasil pembagian nilai dari dua operan. | x % y = 1 |
| Pangkat (\*\*) | Memangkatkan nilai dari dua operan. | x \*\* y = 161051 |

1. **Operator Relasional**  
   Operator relasional merupakan operator perbandingan antara dua operan yang berupa integer, float, string, ataupun boolean. Hasil akhir dari operator ini adalah nilai bertipe boolean. Perhatikan tabel di bawah untuk memahami contoh penerapan operator relasional. **Asumsikan kedua variabel bertipe numerik atau float dengan x bernilai 5 dan y bernilai 10.**  
     
   Jika variabel bertipe integer atau float, berikut adalah penjelasan detailnya.

| **Operator** | **Deskripsi** | **Contoh** |
| --- | --- | --- |
| Sama dengan (==) | Menghasilkan True, jika kedua operan bukan merupakan bilangan riil dan bernilai sama. | x == y, menghasilkan False. |
| Tidak Sama dengan (!=) | Menghasilkan True, Jika kedua operan tidak bernilai sama. | x != y, menghasilkan True. |
| Lebih Besar dari (>) | Menghasilkan True, jika operan kiri lebih besar dari operan kanan. | x > y, menghasilkan False. |
| Kurang dari (<) | Menghasilkan True, jika operan kanan lebih besar dari operan kiri. | x < y, menghasilkan True. |
| Lebih Besar dari Sama dengan (>=) | Menghasilkan True, jika operan kiri lebih besar atau sama dengan operan kanan. | x >= y, menghasilkan False |
| Kurang dari Sama dengan (<=) | Menghasilkan True, jika operan kanan lebih besar atau sama dengan operan kiri. | x <= y, menghasilkan True |

1. Jika variabel adalah bertipe string, berikut penjelasan detailnya. **Asumsikan x bernilai "Dicoding" dan y bernilai "Indonesia".**

| **Operator** | **Deskripsi** | **Contoh** |
| --- | --- | --- |
| Sama dengan (==) | Menghasilkan True, jika kedua string memiliki nilai yang identik/sama persis. | x == y, menghasilkan False. |
| Tidak Sama dengan (!=) | Menghasilkan True, jika kedua string memiliki nilai yang tidak sama. | x != y, menghasilkan True. |
| Lebih Besar dari (>) | Menghasilkan True, jika **huruf pertama pada string pertama** lebih BESAR dari **huruf pertama pada string kedua** dalam urutan alfabet. | x > y, menghasilkan False. |
| Kurang dari (<) | Menghasilkan True, jika **huruf pertama pada string pertama** lebih KECIL dari **huruf pertama pada string kedua** dalam urutan alfabet. | x < y, menghasilkan True. |
| Lebih Besar dari Sama dengan (>=) | Menghasilkan True, jika **huruf pertama pada string pertama** lebih besar atau sama dengan dari **huruf pertama pada string kedua** dalam urutan alfabet. | x >= y, menghasilkan False. |
| Kurang dari Sama dengan (<=) | Menghasilkan True, jika **huruf pertama pada string pertama** lebih kecil atau sama dengan dari **huruf pertama pada string kedua** dalam urutan alfabet. | x <= y, menghasilkan True. |

1. **Operator Logika**  
   Operator logika merupakan jenis operator untuk melakukan operasi logika dengan kedua operannya bertipe boolean. Hasil akhir dari operasi ini adalah nilai bertipe boolean. Perhatikan kode di bawah ini untuk memahami contoh penerapannya, **asumsikan bahwa p bernilai True dan q bernilai False.**

| **Operator** | **Deskripsi** | **Contoh** |
| --- | --- | --- |
| "AND" atau "&" | Logika yang hanya menghasilkan True jika kedua operan bernilai True. | p and q = False, p & q = False |
| "OR" atau "|" | Logika yang menghasilkan True jika salah satu dari kedua operan bernilai True. | p or q = True, p | q = True |
| NOT | Logika yang bertujuan untuk membalikkan nilai logika dari operannya. | not q = True |

1. **Operator Assignment**  
   Operator ini bertujuan untuk melakukan proses assignment atau pemberian nilai pada suatu variabel dengan nilai tetap. Perhatikan tabel di bawah ini untuk memahami contoh penerapan operator assignment. **Asumsikan x bernilai 11 dan y bernilai 5.**

| **Operator** | **Deskripsi** | **Contoh** |
| --- | --- | --- |
| += | Menyederhanakan operasi x = x + y. | x += y, menghasilkan nilai 16. |
| -= | Menyederhanakan operasi x = x - y. | x -= y, menghasilkan nilai 6. |
| \*= | Menyederhanakan operasi x = x \* y. | x \*= y, menghasilkan nilai 55. |
| /= | Menyederhanakan operasi x = x / y. | x /= y, menghasilkan nilai 2.2. |
| %= | Menyederhanakan operasi x = x % y. | x %= y, menghasilkan nilai 1. |

**Rangkuman Aksi Sekuensial**

Kita sudah berada di penghujung materi aksi sekuensial. Sampai sejauh ini, Anda telah memiliki pemahaman mengenai aksi sekuensial Python yang menjadi ilmu mendasar dalam pemrograman, khususnya Python. Mari kita rangkum secara saksama.

**Pengenalan Aksi Sekuensial**

Aksi sekuensial adalah sederetan instruksi yang akan dijalankan oleh komputer berdasarkan urutan penulisannya. Dalam Python, kode yang Anda bangun akan berjalan sesuai dengan urutan perintahnya.

Perlu diperhatikan bahwa ada program yang akan berubah hasilnya jika urutan baris instruksinya diubah. Ada pula program yang hasilnya **tidak** akan berubah jika urutan baris instruksinya diubah.

**Python Interpreter**

Dalam Python, kode program yang Anda bangun akan ditransformasi menjadi kode yang mudah dimengerti oleh mesin menggunakan program compiler atau interpreter. Compiler merupakan program yang akan menerjemahkan bahasa pemrograman menjadi bahasa mesin sebelum dijalankan. Ini artinya program yang Anda bangun secara keseluruhan akan diubah terlebih dahulu semuanya menjadi bahasa mesin.

Hal berbeda terjadi pada interpreter, yang akan menerjemahkan bahasa Python menjadi bahasa mesin satu per satu secara langsung. Hal ini memungkinkan Anda untuk melihat hasil program segera setelah satu baris kode dieksekusi hingga selesai. Implementasi interpreter ada pada mode interaktif Python.

**Block Code**

Sebuah program pada Python dibangun berdasarkan blok-blok kode. Sebuah blok merujuk pada potongan kode program Python yang dijalankan sebagai satu unit. Kode blok dapat berupa modul, fungsi, dan kelas. Contoh kode Python yang memiliki bentuk blok adalah perulangan for.

1. for i in range(10):
2. print(i)

**Case-sensitive**

Python termasuk bahasa pemrograman *case-sensitive*. Ini artinya Python memperlakukan huruf besar dan kecil sebagai karakter yang berbeda dalam penamaan variabel, nama fungsi, atau penulisan kode secara umum. Berikut contoh penerapannya.

1. teks = "Dicoding"
2. Teks = "Indonesia"
4. print(teks)
5. print(Teks)
7. """
8. Output:
9. Dicoding
10. Indonesia
11. """

Pada program di atas, Anda membuat dua variabel dengan nama "teks" dan "Teks". Python akan menganggap bahwa variabel tersebut berbeda, walaupun bagi kita sebagai manusia, kedua hal tersebut memiliki arti yang sama.

**One-liner**

Selain membangun kode berdasarkan bloknya. Anda juga dapat membuat sebuah kode hanya dalam satu baris saja. Konsep ini dikenal sebagai *one-liner*.

*One-liner* merupakan gaya penulisan pada Python yang memungkinkan Anda untuk membuat sebuah kode hanya dalam satu baris. *One-liner* adalah salah satu keunggulan dalam Python yang susah untuk diimplementasikan dalam beberapa bahasa pemrograman yang lainnya.

Tujuan dari *one-liner* ini adalah membuat satu baris kode yang singkat dan jelas. Perlu diingat bahwa tidak semua kode blok dapat dijadikan *one-liner*, seperti deklarasi fungsi, modul, dan kelas.

Ada banyak kode program pada Python yang memiliki versi *one-liner* masing-masing. Salah satu contohnya adalah program penukaran dua variabel.

1. x = 1
2. y = 2
4. x, y = y, x

**Rangkuman Control Flow**

Kita sudah berada di penghujung materi Control Flow. Sampai sejauh ini, Anda telah memiliki pemahaman mengenai berbagai jenis kontrol dalam pemrograman, seperti percabangan, perulangan, hingga penanganan kesalahan. Mari kita rangkum secara saksama.

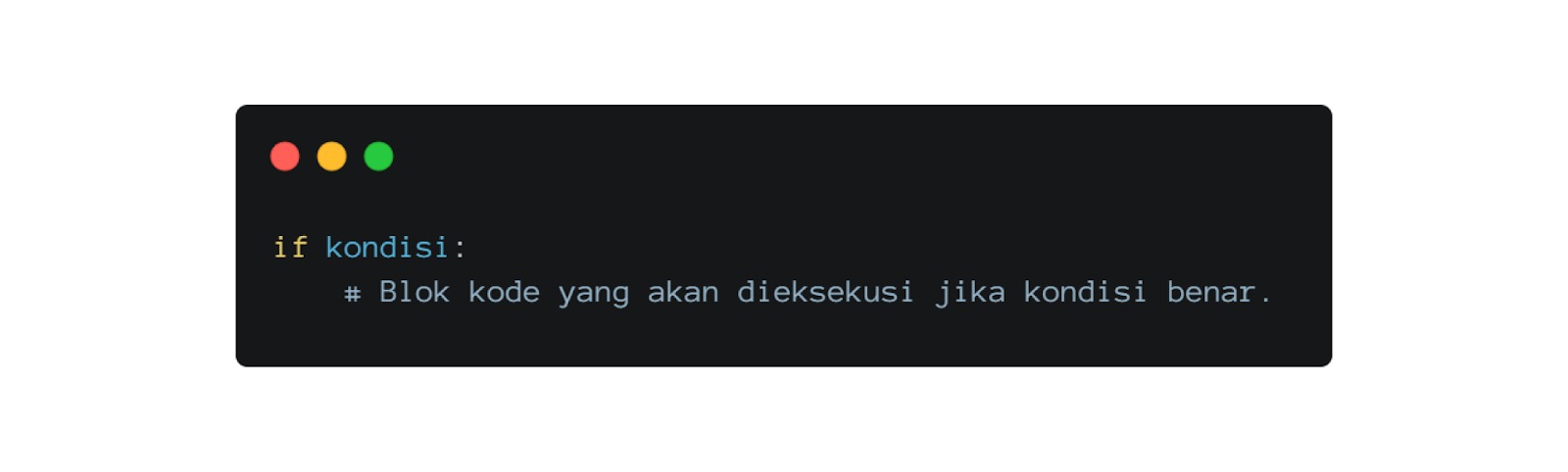
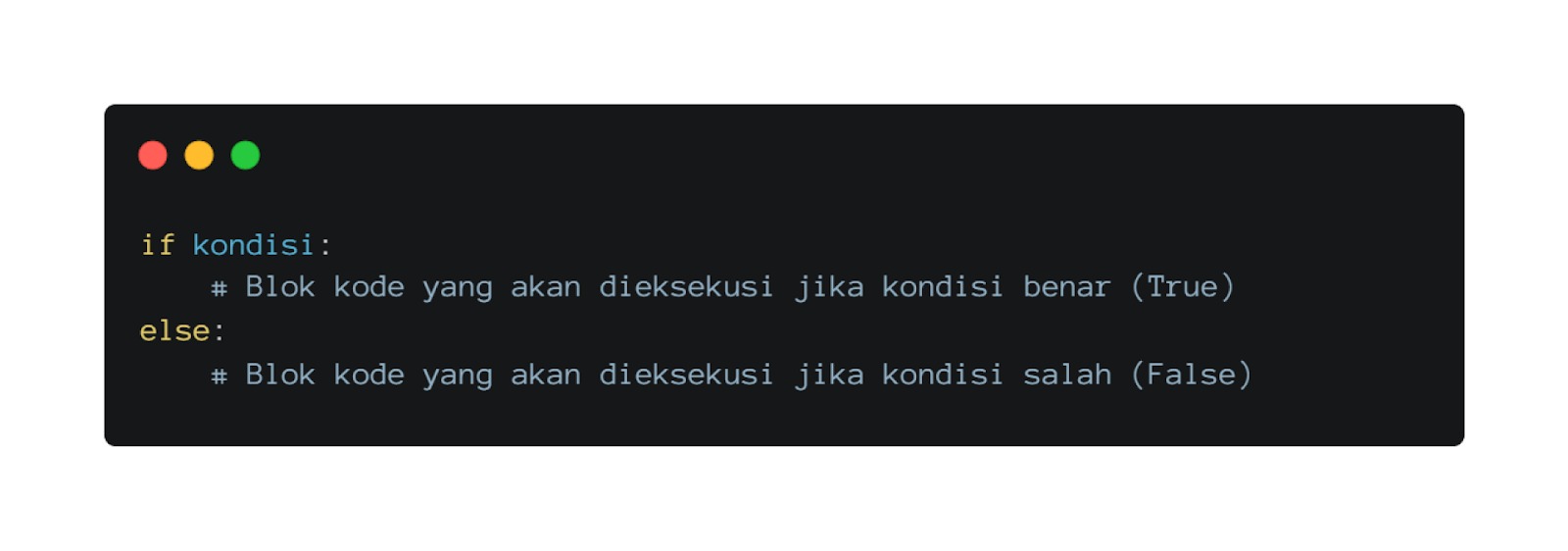
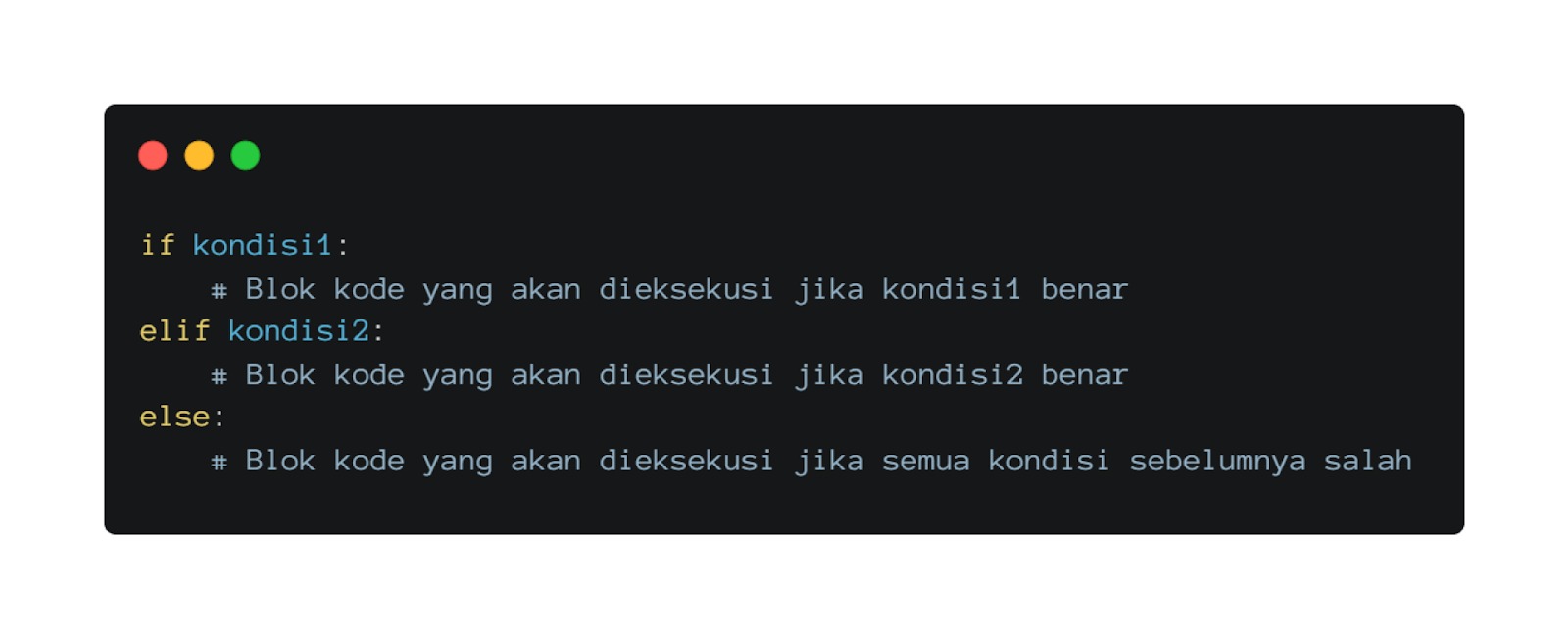
**Percabangan dan Ternary Operators**

**Percabangan**

Dalam pemrograman, sebuah kode program dapat berjalan berdasarkan kondisi tertentu. Maknanya, Anda dapat memberikan instruksi berdasarkan **"Jika-maka"**.

1. **Jika**Anda tidak menyelesaikan kelas Memulai Pemrograman dengan Python, **maka** Anda tidak lulus dari kelas Memulai Pemrograman dengan Python.
2. **Jika** variabel nama kurang dari dua, **maka** variabel tersebut tidak memenuhi kriteria kondisi.

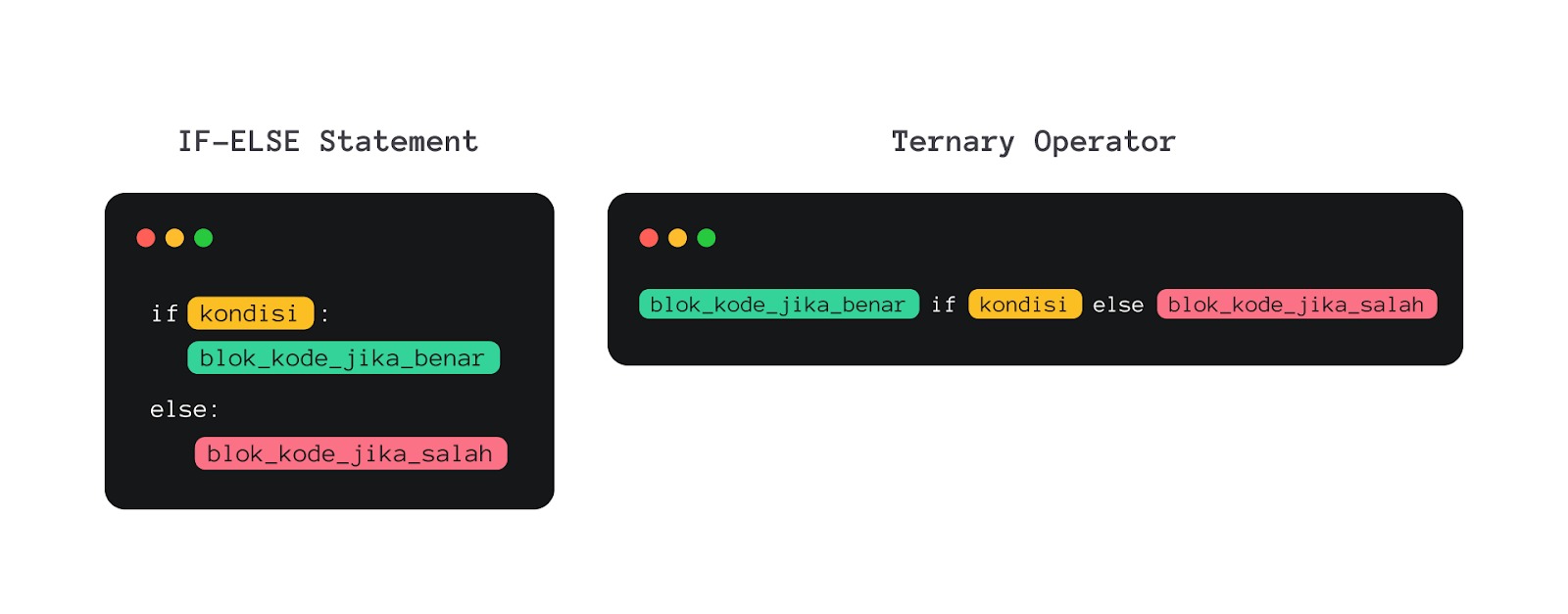
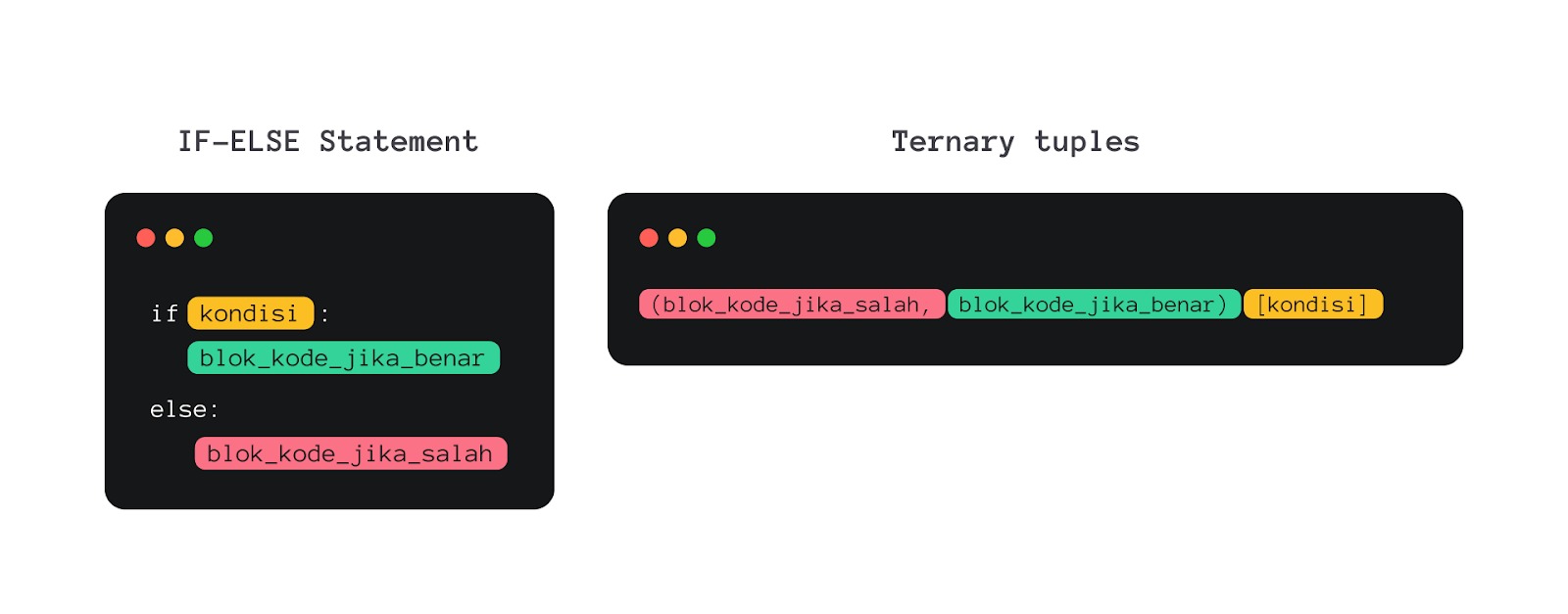
Statement atau sintaks untuk melakukan percabangan sebagai berikut.

1. **If**  
   If adalah statement Python yang akan mengecek nilai variabel di dalamnya memenuhi kriteria suatu kondisi atau tidak. Jika memenuhi kriteria, kondisi tersebut bernilai *true*. Jika tidak memenuhi kriteria, kondisi akan bernilai *false*. Jika kondisi if bernilai *true*, kode yang berada dalam blok kode if akan dieksekusi.  
     
   Struktur dari IF statement sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)Sebagai informasi tambahan, kita juga dapat menambahkan **'and'** atau **'or'**operator dalam kondisi percabangan.
2. **Else**  
   Else adalah statement yang menjadi jalan keluar saat kondisi atau hasil evaluasi if statement bernilai *false*. Ini maksudnya, program akan menjalani blok kode if terlebih dahulu dan jika hasilnya adalah false, program akan menjalankan else statement sebagai jalan keluar atau kondisi terakhir.  
     
   Jika kita gabungkan if dan else, struktur berikut dihasilkan.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)
3. **ELIF**  
   Elif merupakan kependekan dari else if dan alternatif untuk if bertingkat atau *switch case*. Elif statement berada di posisi setelah if. Anda dapat menambahkan elif statement lebih dari satu karena tidak dibatasi dan opsional.  
     
   Struktur keseluruhan percabangan jika kita gabungkan antara if, elif, dan else sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

**Ternary Operators**

Ternary operators termasuk conditional expressions pada Python. Conditional expressions adalah bentuk ekspresi yang bertujuan untuk mengevaluasi kondisi dan mengembalikan nilai berdasarkan hasil evaluasinya. Anda bisa asumsikan bahwa ternary operators ini merupakan versi one-liner dari if dan else.

Ada 2 tipe ternary sebagai berikut.

1. **Ternary Operators**  
   Ternary jenis ini adalah jenis expression yang umum digunakan. Struktur dari ternary operators sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)
2. **Ternary Tuples**  
   Jenis ternary kedua adalah jenis ternary yang menggunakan tipe data tuples.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

**Perulangan**

Perulangan adalah jenis kode program yang bertujuan untuk meningkatkan efektivitas program dalam membuat kode program berulang. Ada beberapa jenis perulangan dalam Python, yakni for, while, dan for bersarang.

**For**

For adalah sintaks dalam Python yang bersifat **definite iteration**. Definite iteration adalah sebuah proses iterasi atau perulangan yang jumlah pengulangannya ditentukan secara eksplisit sebelumnya.

 Format dari perulangan for sebagai berikut.

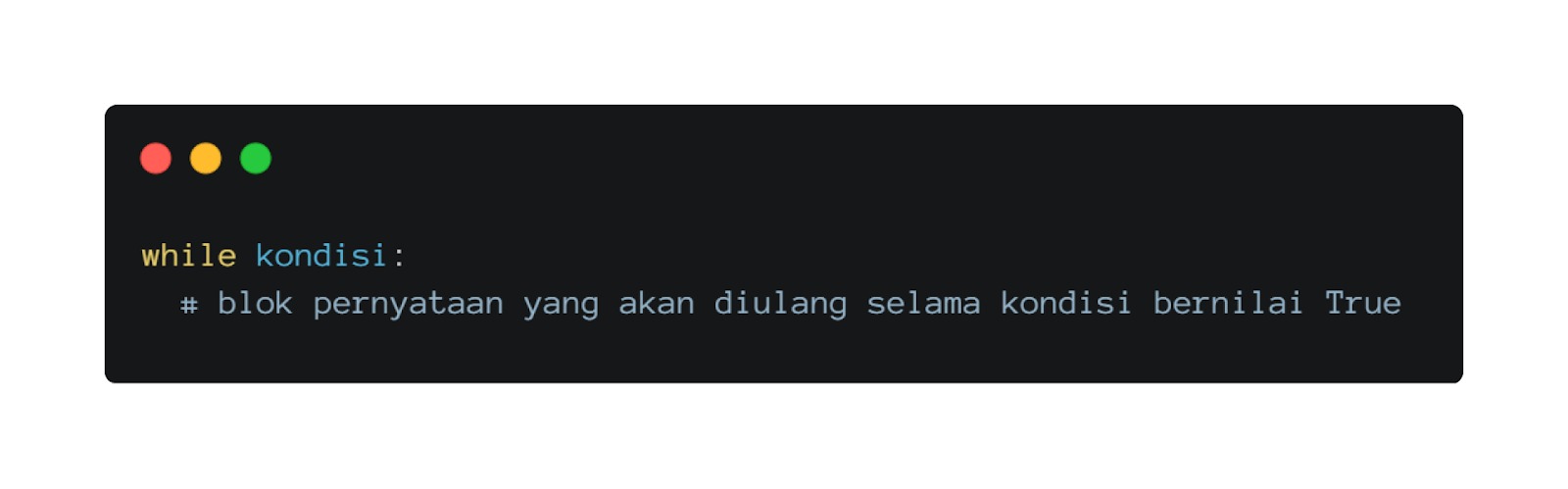
[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

[<iterable>](https://wiki.python.org/moin/Iterator)merupakan segala object dalam Python yang dapat diiterasi seperti list, tuple, hingga string. Ada pula <var> merupakan variabel yang akan mengambil elemen berikutnya dari <iterable> setiap kali iterasi berjalan.

**While**

While adalah sintaks dalam Python yang bersifat **indefinite iteration**. Indefinite iteration adalah sebuah proses iterasi yang akan berhenti ketika memenuhi kondisi tertentu.

Format dari perulangan while sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Kondisi merupakan ekspresi yang akan dievaluasi dan menghasilkan nilai true atau false. Selama hasil evaluasi bernilai true, program akan terus berjalan hingga menghasilkan nilai false.

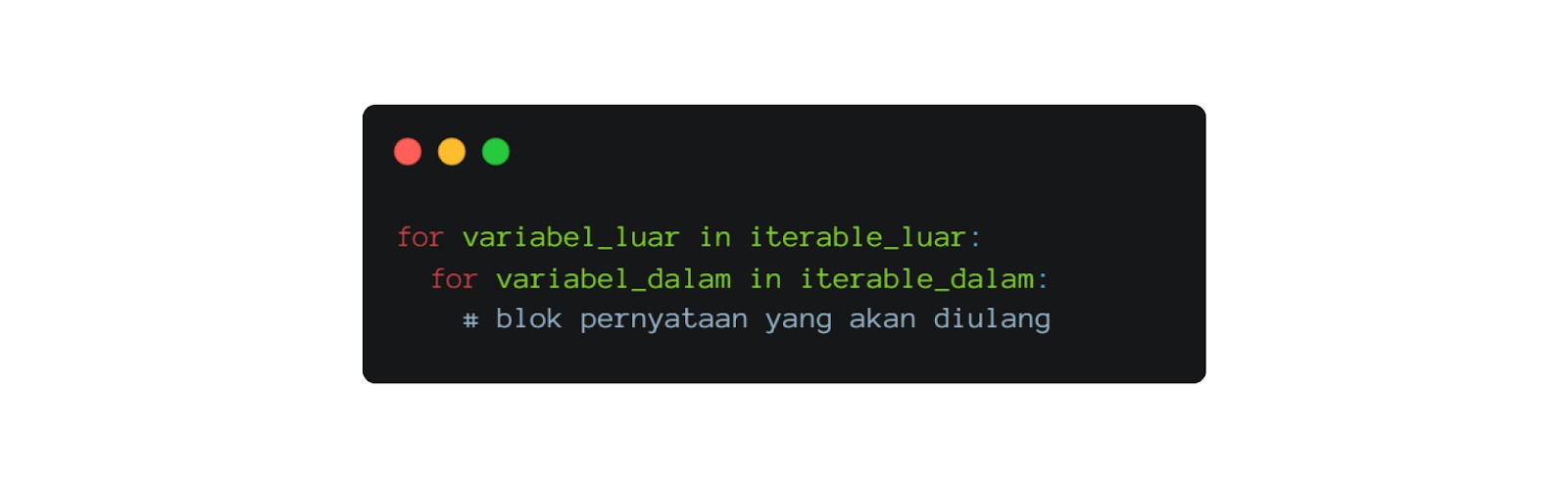
Namun, ada kondisi yang akan menyebabkan perulangan terus berjalan tanpa henti. Kondisi ini disebut dengan **infinite loop**, kebalikan dari **indefinite loop**. Kondisi **infinite loop** berarti kita melakukan perulangan yang tidak pernah memenuhi kondisi yang diinginkan. Contohnya ketika kita melakukan perulangan, tetapi tidak menambahkan increment di akhir kode.

1. counter = 1
2. while counter <= 5:
3. print(counter)

**For Bersarang**

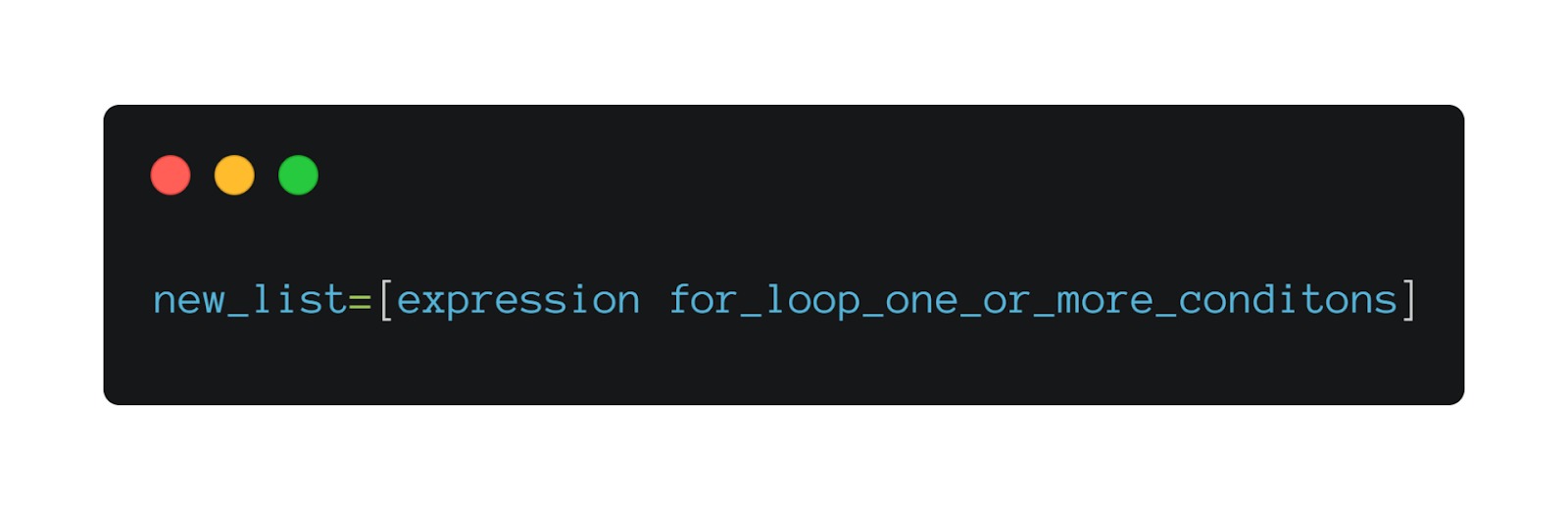
Ketika Anda membuat perulangan, sering kali menemukan perulangan dalam perulangan atau disebut sebagai nested loop.

Format dari nested loop sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Anda dapat asumsikan bahwa terdapat dua perulangan, yakni "perulangan luar" dan "perulangan dalam". Program akan melakukan "perulangan luar" terlebih dahulu, lalu akan melakukan "perulangan dalam". "variabel\_luar" akan mengambil nilai dari "iterable\_luar" sedangkan "variabel\_dalam" akan mengambil nilai dari "iterable\_dalam".

**Kontrol Perulangan**

1. **Break**  
   Break statement adalah pernyataan untuk menghentikan perulangan dan kemudian program akan otomatis keluar dari perulangan tersebut, lalu dilanjutkan dengan mengeksekusi blok perulangan selanjutnya.  
     
   Jika Anda memiliki perulangan yang bertingkat seperti for bersarang, break akan menghentikan perulangan sesuai dengan tingkatan atau letak perulangannya berada.
2. **Continue**  
   Continue statement adalah pernyataan untuk membuat iterasi berhenti, kemudian melanjutkan ke iterasi berikutnya. Continue seolah **mengabaikan**pernyataan (statement) yang berada antara continue hingga akhir blok.
3. **Else setelah For**  
   Pada Python juga dikenal else setelah for dengan fungsinya untuk perulangan yang bersifat pencarian. Else setelah for ini bisa dikatakan sebagai memberikan jalan keluar program saat pencarian tidak ditemukan.  
     
   Perlu diperhatikan oleh Anda, if dan else berkaitan walaupun berbeda blok. **Pada else setelah for, statement else tidak akan dieksekusi saat if pernah sekali saja benar.**Dengan kata lain, **break dalam if harus tidak terjadi untuk memicu else setelah for.**
4. **Else setelah While**  
   Berbeda dengan else setelah for, pada statement else setelah while,**blok statement else akan selalu dieksekusi saat kondisi pada while menjadi salah**.
5. **Pass**  
   Pass statement adalah pernyataan yang digunakan jika Anda menginginkan sebuah pernyataan atau blok pernyataan (statement), tetapi tidak ada tindakan atau program tidak melakukan apa pun.  
     
   Statement pass digunakan dalam situasi-situasi ketika Python memerlukan adanya pernyataan, tetapi tidak memiliki tindakan yang perlu dilakukan pada saat itu. Biasanya itu adalah kondisi ketika Anda membutuhkan *placeholder* untuk menunjukkan bahwa tidak ada operasi yang perlu dilakukan. Hal ini dapat membantu kita mengatur struktur kode secara rapi dan memungkinkan penambahan implementasi di kemudian hari.
6. **List Comprehension**  
   List comprehension adalah cara untuk menghasilkan list baru berdasarkan list atau *iterables* yang telah ada sebelumnya. Sintaks dasarnya sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)  
   Catatan:
   1. **new\_list**merupakan variabel yang dideklarasikan oleh Anda.
   2. **expression**merupakan ekspresi yang akan dijalankan seiring perulangan bernilai benar.
   3. **for\_loop\_one\_or\_more\_conditions**merupakan perulangan for yang Anda definisikan. Contohnya adalah "for n in angka" yang ada pada contoh sebelumnya.

**Penanganan Kesalahan (Error Handling and Exception Handling)**

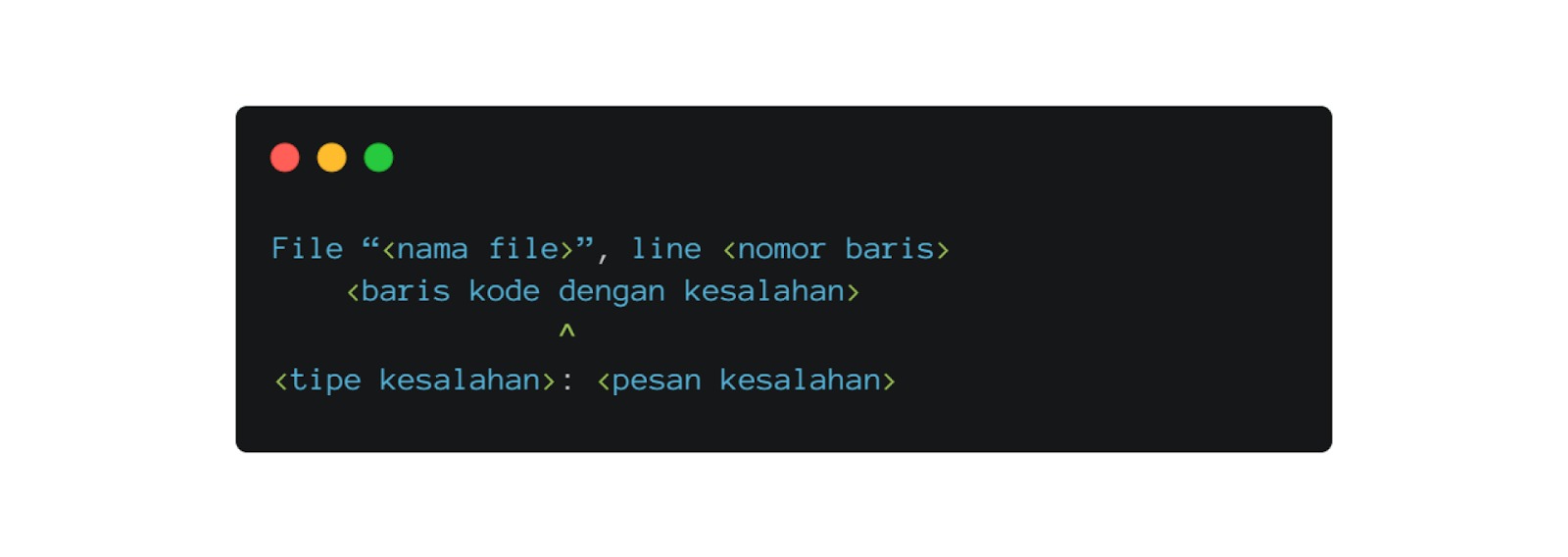
Saat Anda membuat program, sering kali menemukan setidaknya dua jenis kesalahan berdasarkan kejadiannya.

1. Kesalahan sintaks (*syntax errors*) atau sering disebut juga sebagai kesalahan penguraian (*parsing errors*).
2. Pengecualian (*exceptions*) atau sering disebut juga sebagai kesalahan saat beroperasi (*runtime* errors).

**Kesalahan Sintaks (Syntax Errors)**

Kesalahan sintaks (*syntax errors*) adalah jenis kesalahan yang terjadi ketika Python tidak mengerti perintah Anda. Ini mengakibatkan pesan kesalahan muncul sebelum program tersebut berjalan.

Secara umum struktur kesalahan sintaks sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Berikut adalah penjelasan detail satu per satu terkait poin di atas.

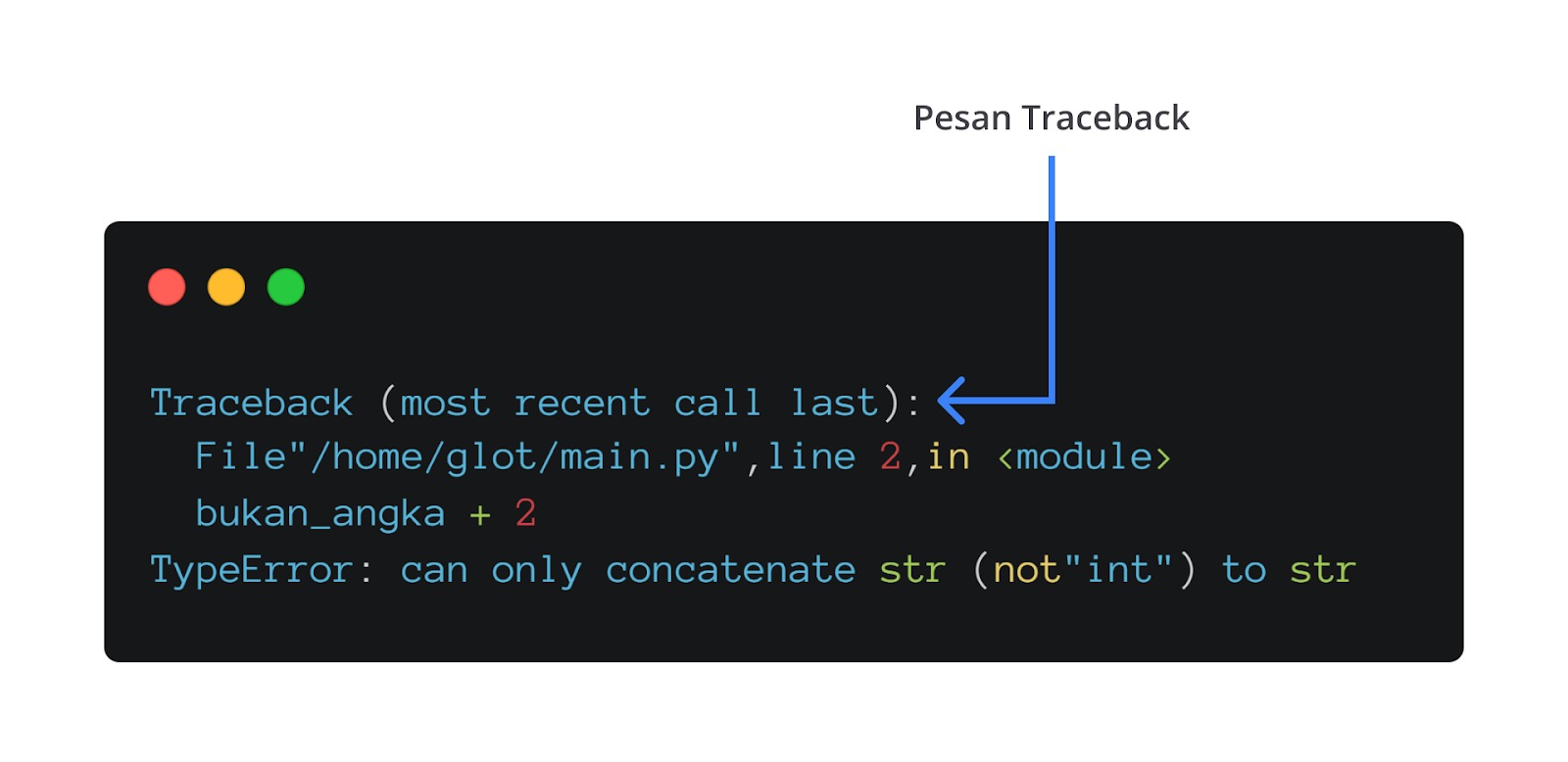
1. "<nama file>" merupakan file Python yang Anda eksekusi. Jika Anda menggunakan mode script melalui lokal komputer dan program Anda menghasilkan Error, pesan ini akan memunculkan nama script atau file Python Anda.
2. <nomor baris> merupakan nomor baris kode dalam file Anda yang mengalami kesalahan.
3. <baris kode> merupakan kode yang mengalami kesalahan dalam file Anda.
4. <tipe kesalahan> merupakan kelompok atau tipe kesalahan yang Anda alami, contohnya SyntaxErrordan IndentationError.
5. <pesan kesalahan> merupakan pesan detail kesalahan atau keterangan yang diberikan oleh program. Contohnya “invalid syntax” dan “expected an indented block”.

**Pengecualian (Exceptions)**

Pengecualian adalah kesalahan yang terjadi ketika Python mengerti perintah Anda, tetapi mendapatkan masalah saat mengikutinya. Umumnya, pengecualian bisa terjadi ketika aplikasi sudah mulai beroperasi.

Jenis kesalahan ini adalah kesalahan yang paling sering ditemui ketika Anda membuat kode program yang kompleks. Meskipun kode atau ekspresi dari Python yang Anda tulis sudah benar, ada kemungkinan terjadi kesalahan ketika perintah tersebut dieksekusi.

Secara umum, struktur pengecualian sama seperti kesalahan sintaksis. Namun, hal yang menjadi pembeda adalah pada struktur pengecualian memberikan pesan "traceback (most recent call last)".

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

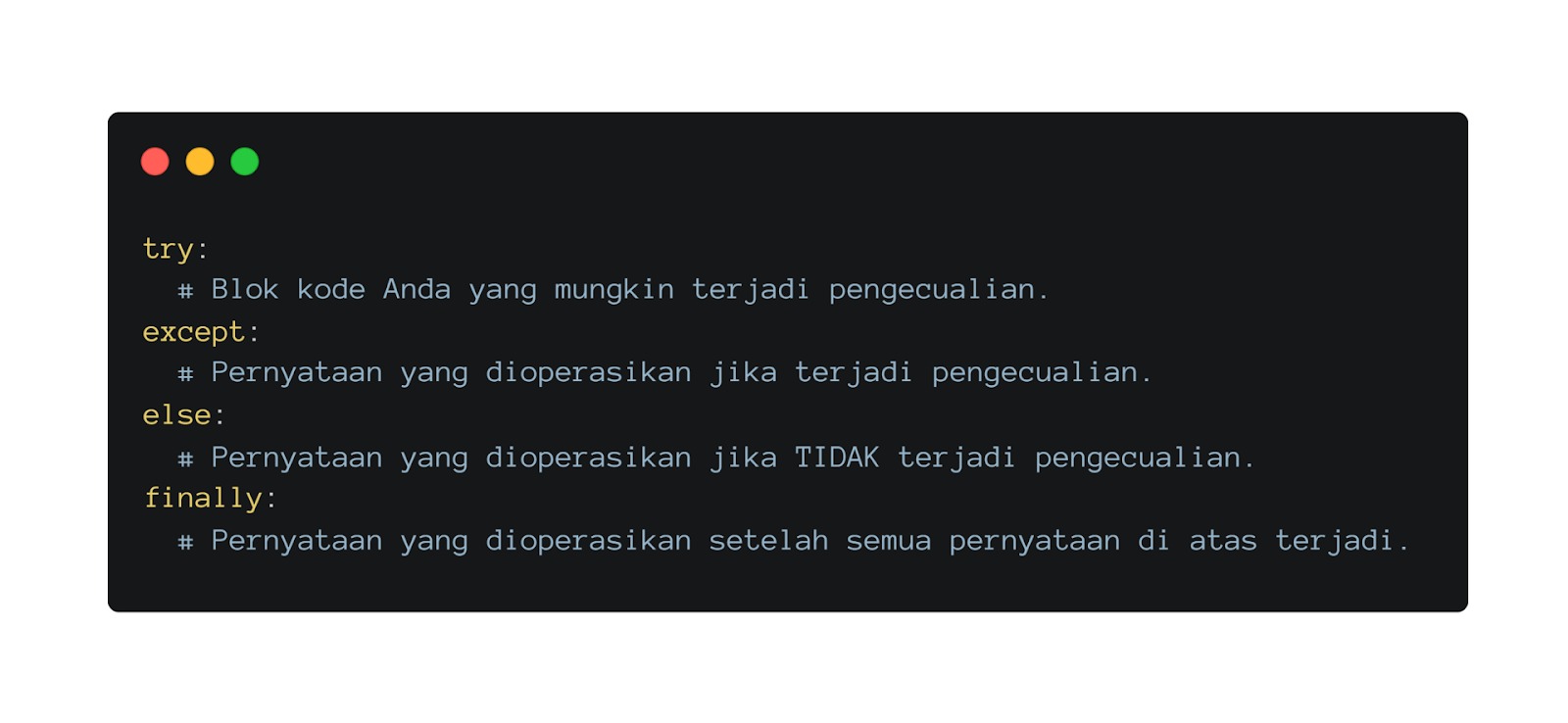
Pesan ini mengacu pada informasi yang ditampilkan ketika terjadi kesalahan atau pengecualian (*exception*). Pesan *traceback* ini menyediakan "jejak" dari kode yang dieksekusi sehingga Anda dapat melacak kembali jalur eksekusi program sebelum mencapai titik error.

**Penanganan Pengecualian**

Untuk menyelesaikan setiap masalah pengecualian, kita dapat membangun kode program dalam menangani hal tersebut.

Program Python yang Anda bangun dapat dilengkapi penanganan terhadap pengecualian dari tipe kesalahan yang Anda tentukan. Konsep ini dikenal dengan *exceptions handling* yang menggunakan pernyataan **try-except**untuk menangani pengecualian tersebut.

Secara keseluruhan, struktur lengkap dari penanganan pengecualian sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Pada try statement, program akan menjalankan blok kode yang mungkin terjadi pengecualian. Pada except statement, program akan mengeksekusi statement ini jika terjadi pengecualian. Pada else statement, program akan mengeksekusi statement ini jika **tidak** terjadi pengecualian. Pada finally statement, program akan mengeksekusi statement ini setelah semua pernyataan di atas terjadi.

**Raise Exception**

Jika sebelumnya kita menangani kesalahan yang **TIDAK DISENGAJA**, kali ini kita akan mempelajari cara menangani kesalahan yang **DISENGAJA**. Umumnya, ketika membuat kode program kita ingin membatasi program tersebut dengan kondisi tertentu.

Perlu diingat bahwa umumnya, raise digunakan bersamaan dengan if-else statement.

Berikut adalah contoh penerapan raise statement untuk menangani kesalahan atau pengecualian yang disengaja.

1. var = -1
2. if var < 0:
3. raise ValueError("Bilangan negatif tidak diperbolehkan")
4. else:
5. for i in range(var):
6. print(i+1)

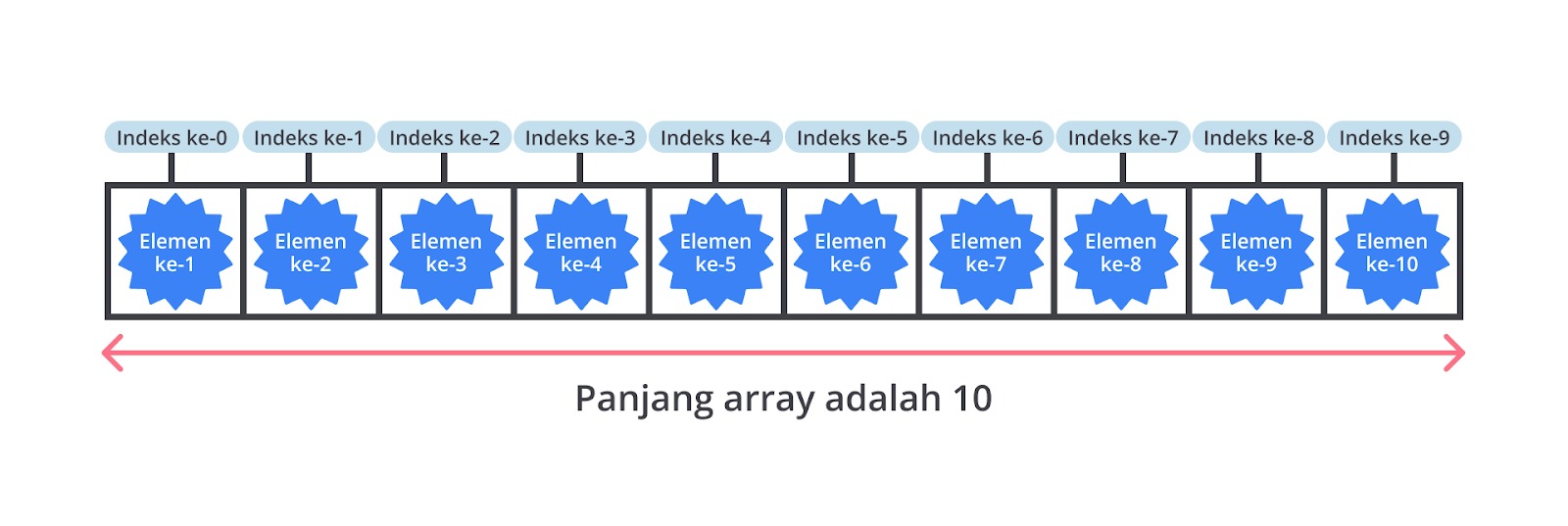
9. """
10. Traceback (most recent call last):
11. File "/home/glot/main.py", line 3, in <module>
12. raise ValueError("Bilangan negatif tidak diperbolehkan")
13. ValueError: Bilangan negatif tidak diperbolehkan

**Rangkuman Array dan Pemrosesannya**

Kita sudah berada di penghujung materi Array dan Pemrosesannya. Sampai sini, Anda telah mempelajari salah satu struktur data, yakni array. Mari kita rangkum secara saksama.

**Fundamental Array**

Array adalah salah satu jenis dari struktur data linear dan terdiri dari kumpulan elemen bertipe data sama dengan indeks yang berurutan atau linear. Struktur array adalah berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Berikut adalah penjelasan dari struktur tersebut.

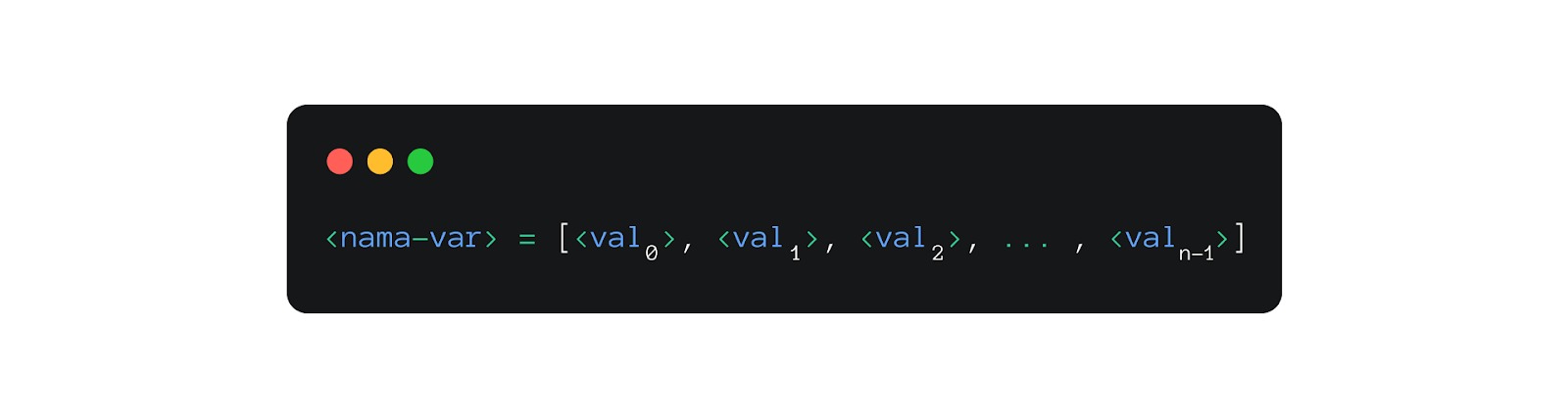
1. Indeks-indeks (indices): Kumpulan indeks yang keseluruhan dari kumpulan indeks tersebut disebut sebagai array.
2. Indeks pertama: Indeks pertama array yang selalu dimulai dari 0.
3. Element: Nilai yang berada dalam suatu indeks, contohnya jika nilai dari indeks 8 adalah string "Dicoding", kita bisa sebut sebagai "elemen ke-9 adalah string 'Dicoding'".
4. Array length: Panjang dari suatu array. Dalam gambar tersebut, panjang array adalah 10.

Dalam Python terdapat dua cara untuk menggunakan array, yakni berikut.

1. Menggunakan List
   1. x = [1, 2, 3, 4, 5]
   2. print(x)
   4. """
   5. Output:
   6. [1, 2, 3, 4, 5]
   7. """
2. Menggunakan Library Array
   1. import array
   3. x = array.array("i",[1, 2, 3, 4, 5])
   4. print(x)
   5. print(type(x))
   7. """
   8. Output:
   9. array('i', [1, 2, 3, 4, 5])
   10. <class 'array.array'>
   11. """

**Implementasi Array dengan Python**

Dalam kelas ini, kita telah mempelajari array menggunakan list Python. Secara detail, ada dua cara untuk melakukan deklarasi array menggunakan list Python, yakni berikut.

1. **Mendefinisikan Isi Array**  
   Cara pertama adalah dengan mendeklarasikan variabel array sekaligus mendefinisikan isi array. Cara ini dilakukan jika kita sudah tahu nilai yang perlu diberikan.  
     
   Berikut adalah struktur mendeklarasikan variabel array dengan mendefinisikan isi array secara langsung.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)  
   Dari struktur tersebut, **<nama-var>**merupakan nama variabel array yang dideklarasikan sebanyak **n** dengan elemen-elemennya adalah **<val0>, <val1>, <val2>, … , <valn-1>**. Perlu diingat bahwa elemen tersebut terurut berdasarkan indeks dari **0** hingga **n-1**.
2. **Mendefinisikan Nilai Default**  
   Cara kedua adalah mendeklarasikan array dengan mendefinisikan nilai default. Cara ini dilakukan jika kita tidak mengetahui nilai yang diberikan. Kita dapat memberikan nilai default terlebih dahulu sebagai upaya untuk memberikan nilai awal.  
     
   Nilai default ditentukan oleh kesepakatan bersama sesuai kebutuhan yang nilainya di luar dari rentang yang ditentukan. Misalnya tim Anda menentukan nilai dalam list harus berkisar dari 1 hingga 10. Kita bisa menyepakati "0" sebagai nilai default karena di luar jangkauan yang disepakati (1-10).  
     
   Berikut adalah struktur mendeklarasikan variabel array dengan mendefinisikan nilai default.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)  
   Berikut adalah penjelasan lebih detail terkait struktur tersebut.
   1. <nama-var> merupakan variabel yang Anda deklarasikan.
   2. <default-val> merupakan nilai default yang Anda definisikan. Umumnya, programmer akan menggunakan nilai di luar range yang telah disepakati sebagai nilai default. Misalnya jika range nilai yang disepakati seharusnya 1 hingga 10, nilai default bisa kita definisikan dengan 0.
   3. <n> merupakan ukuran panjangnya array.

**Pemrosesan Sekuensial pada Array**

Pemrosesan array merujuk pada operasi-operasi yang dilakukan pada elemen-elemen suatu array. Operasi ini melibatkan manipulasi hingga pengolahan elemen yang ada pada array. Adapun pemrosesan sekuensial adalah sebuah pemrosesan setiap elemen array yang dimulai dari elemen pada indeks terkecil hingga terbesar. Pemrosesan sekuensial lebih sering menggunakan pengulangan (loop/iterasi) dalam setiap prosesnya.

Sebab pemrosesan sekuensial melibatkan semua elemen di dalamnya, ada beberapa hal yang perlu diperhatikan.

1. Setiap elemen array diakses secara langsung melalui indeksnya (metode indexing).
2. Elemen pertama (*first element*) adalah elemen array dengan indeks terkecil yang selalu dimulai dari 0.
3. Elemen selanjutnya (*next element*) dicapai melalui suksesor indeks.
4. Kondisi berhenti dicapai jika indeks yang diproses adalah indeks terbesar yang sudah terdefinisi.
5. Suatu array tidak boleh kosong, minimal memiliki **satu** elemen di dalamnya.

Banyak sekali contoh penerapan pemrosesan sekuensial pada array, beberapa di antaranya sebagai berikut.

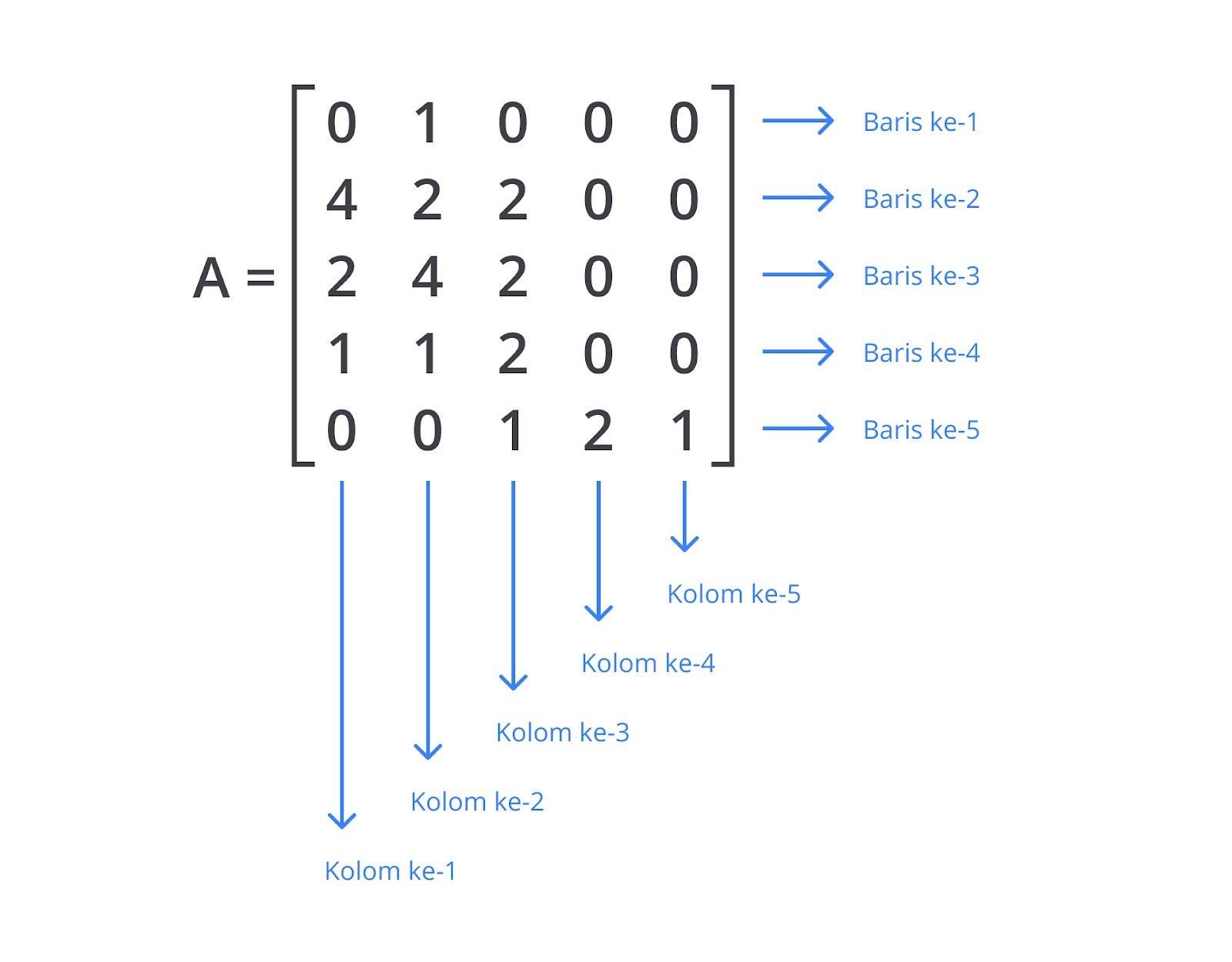
1. Mengisi array secara sekuensial.
2. Menghitung nilai rata-rata elemen array.
3. Mengalikan elemen array dengan suatu nilai.
4. Mencari nilai terbesar atau terkecil pada array.
5. Mencari indeks letak suatu nilai ditemukan pertama kali dalam array, dan sebagainya.

**Rangkuman Matriks**

Kita sudah berada di penghujung materi matriks. Sampai sejauh ini, Anda telah memahami materi matriks dalam matematika hingga penerapannya dalam pemrograman Python. Mari kita rangkum secara saksama.

**Fundamental Matriks**

Matriks dalam matematika merupakan himpunan yang terdiri dari bilangan atau elemen berdasarkan baris dan kolom. Dalam matematika, struktur matriks sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Contoh matriks dalam matematika beragam jenisnya, beberapa di antaranya sebagai berikut.

1. **Matriks Pengukuran**  
   Matriks pengukuran adalah jenis matriks dengan indeks (i, j) yang merepresentasikan suatu titik koordinat. Setiap elemen dari matriks ini merepresentasikan hasil pengukuran pada suatu titik koordinat tertentu dan merupakan bilangan real atau tipe data float.
2. **Matriks Satuan**  
   Selanjutnya adalah matriks satuan yang merupakan matriks dengan elemen bernilai hanya 0 atau 1. Setiap elemen matriks ini bertipe data *integer*.

Dalam pemrograman, matriks adalah kumpulan data yang diatur dalam **bentuk tabel dua dimensi**dengan setiap elemennya terdefinisi berdasarkan baris dan kolom. Matriks dalam pemrograman diimplementasikan menggunakan array dua dimensi. Bahkan dalam Python, matriks dapat diimplementasikan menggunakan nested list atau list di dalam list.

1. matriks = [[1, 2, 3],
2. [4, 5, 6],
3. [7, 8, 9]]
5. print(matriks)
7. """
8. Output:
9. [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
10. """

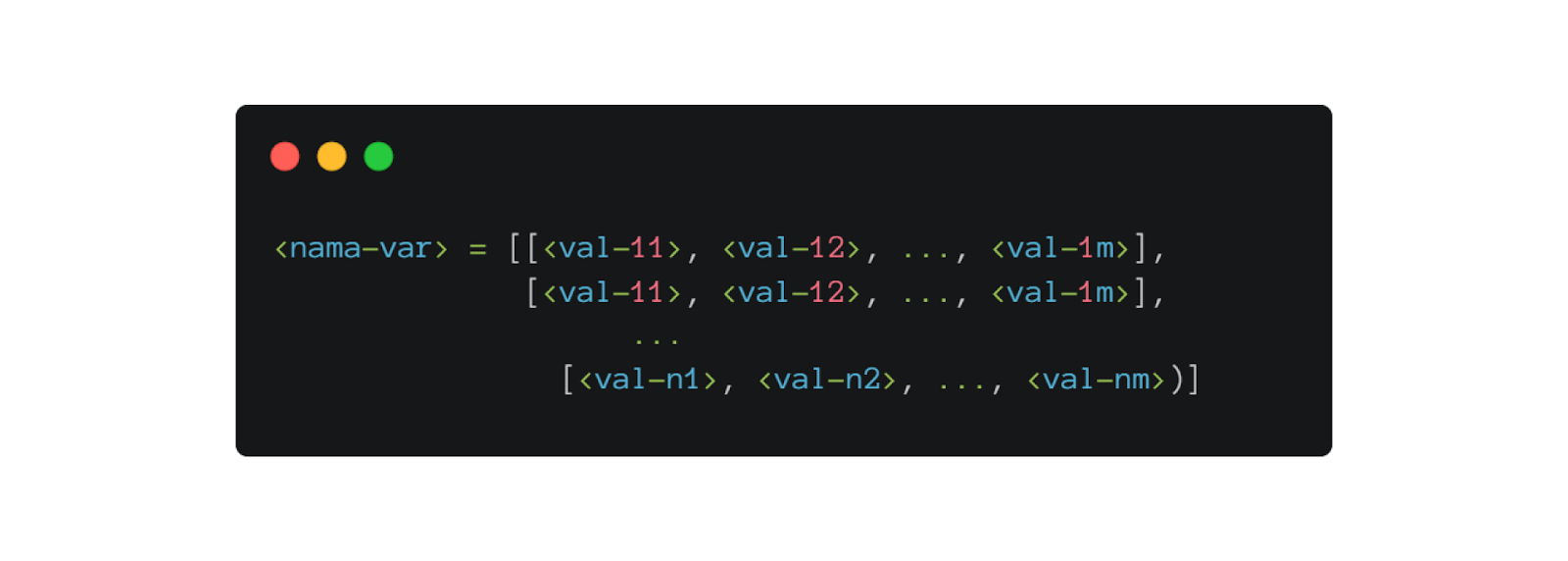
Selain itu, kita juga bisa menggunakan libraryPython untuk menerapkan matriks, seperti NumPy. Secara keseluruhan, matriks dalam pemrograman dapat kita simpulkan sebagai berikut.

1. Matriks adalah kumpulan data yang diatur dalam **bentuk tabel dua dimensi**dengan setiap elemennya terdefinisi berdasarkan baris dan kolom.
2. Setiap elemen matriks dapat diakses melalui metode indexing jika kedua indeks telah diketahui.
3. Elemen matriks dideklarasikan memiliki tipe homogen, yang artinya elemen tersebut harus memiliki tipe data yang sama. Jika elemen tersebut adalah bilangan real, seluruh elemen lainnya pun adalah bilangan real.

Namun, perlu diingat bahwa mendeklarasikan matriks menggunakan list memang praktis, tetapi sangat memakan banyak memori.

**Implementasi Matriks pada Python**

Pada materi ini, kita mempelajari cara mendeklarasikan matriks hingga mengakses setiap elemen matriks dengan metode indexing. Cara untuk mendeklarasikan matriks sebagai berikut.

1. **Deklarasi sekaligus inisialisasi nilai matriks.**  
   Cara pertama adalah mendeklarasikan matriks sekaligus menginisialisasikan nilainya dengan ukuran N baris dan M Kolom (NxM). Cara ini dilakukan jika kita telah mengetahui nilai yang perlu diberikan.  
     
   Berikut adalah struktur untuk mendeklarasikan matriks dengan menginisialisasikan nilainya sekaligus.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)
2. **Deklarasi dengan nilai default**.  
   Cara kedua adalah mendeklarasikan matriks dengan nilai default. Sebagaimana materi array, nilai default ditentukan oleh kesepakatan bersama sesuai kebutuhan dengan nilainya di luar dari rentang yang ditentukan.  
     
   Misalnya, tim Anda menentukan nilai dalam list harus berkisar dari 1 hingga 10. Kita bisa menyepakati nilai "0" sebagai nilai default karena di luar jangkauan yang disepakati (1-10). Cara kedua ini melibatkan list comprehension yang sama seperti pada materi array.  
     
   Struktur dari deklarasi dengan nilai default sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Selanjutnya, cara untuk mengakses elemen pada matriks dapat menggunakan metode indexing. Ingat bahwa matriks merupakan tabel data yang terdiri dari baris dan kolom sehingga jika Anda ingin mengakses elemen dari matriks, perlu mengetahui indeks dari baris dan kolom.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

**Operasi Matriks pada Python**

Dalam matematika ataupun pemrograman, operasi matriks dapat melibatkan dua matriks sekaligus atau pun satu matriks saja. Beberapa operasi tersebut di antaranya sebagai berikut.

1. Operasi 1 matriks.
   * Menghitung total semua elemen matriks.
   * Mengalikan elemen matriks dengan konstanta.
   * Transpose matriks.
   * Inverse matriks.
   * Menentukan determinan, dan sebagainya.
2. Operasi 2 matriks.
   * Menambahkan dua matriks.
   * Mengalikan dua matriks.
   * Pembagian dua matriks, dan sebagainya.

**Rangkuman Subprogram**

Kita sudah berada di penghujung materi Subprogram. Sampai sejauh ini, Anda diharapkan paham untuk mengimplementasikan subprogram dalam setiap program yang Anda bangun. Mari kita rangkum secara saksama.

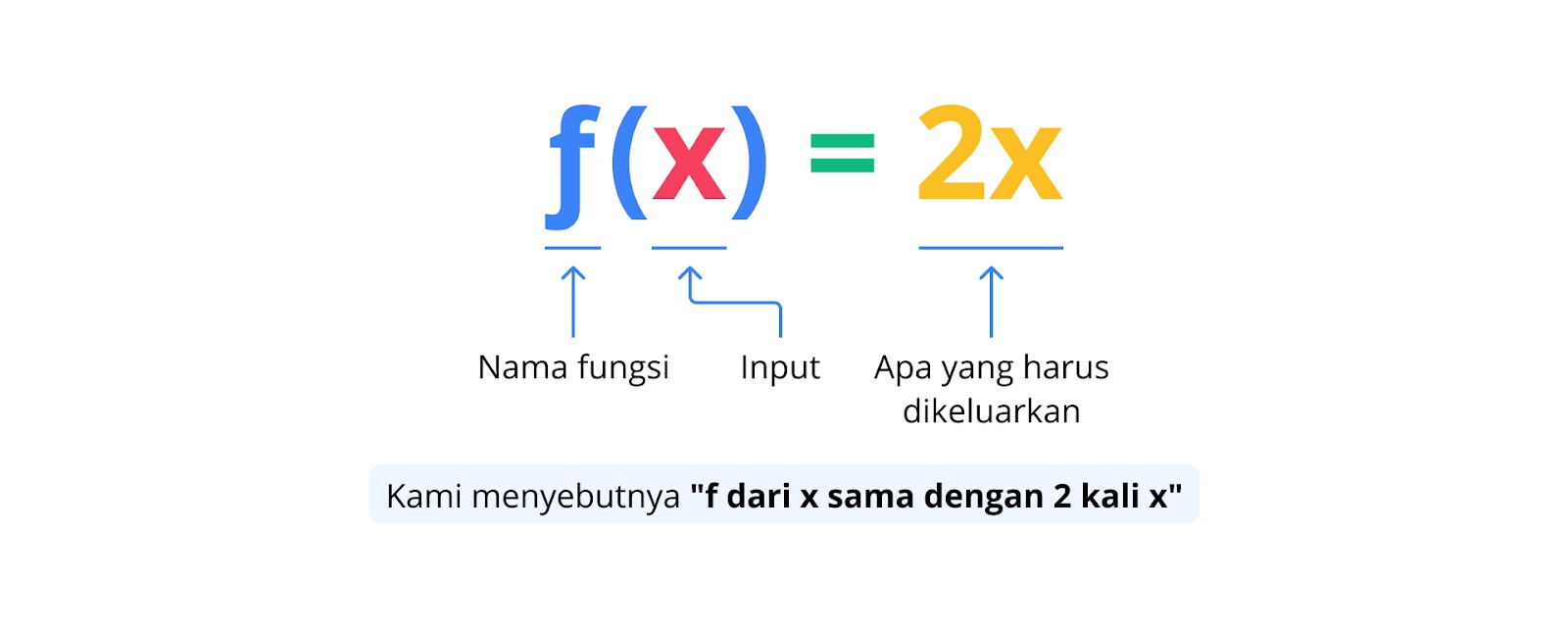
**Definisi Subprogram**

Subprogram adalah serangkaian instruksi dirancang untuk melakukan operasi yang sering digunakan dalam suatu program. Subprogram yang sering digunakan terdiri dari dua jenis, yakni berikut.

1. **Fungsi**  
   Fungsi adalah blok kode yang dapat menerima input, melakukan pemrosesan, dan mengembalikan output. Hasil atau output tersebut dinyatakan dalam sebuah tipe data yang eksplisit, artinya fungsi yang dibuat dapat ditentukan untuk mengembalikan tipe data integer, string, atau lainnya.
2. **Prosedur**  
   Prosedur adalah deretan instruksi yang jelas keadaan awal (*initial state*) dan keadaan akhirnya (*final state*). Prosedur mirip dengan program secara umum, tetapi memiliki cakupan yang kecil dan terbatas.

**Fungsi**

Fungsi dalam pemrograman sebenarnya didasari oleh konsep pemetaan (asosiasi) dan fungsi dalam matematika. Fungsi pada matematika merupakan pemetaan antara dua himpunan nilai, yaitu **domain**dan **range**. Kita bisa bayangkan fungsi sebagai sebuah mesin yang memiliki input(**domain**)dan output(**range**). Outputtersebut pasti terkait dengan input, bagaimana pun kondisinya. Berikut adalah notasi fungsi yang sering dijumpai dalam matematika.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Dari gambar tersebut, f merupakan nama fungsi, x adalah input, dan 2x adalah hal yang harus dikeluarkan oleh fungsi tersebut (output).

Dalam pemrograman, fungsi dapat diumpamakan seperti merakit isi **black box.**

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Selayaknya **black box**, kita tidak perlu tahu tentang hal yang terjadi di dalam kotak (fungsi) tersebut. Kita hanya perlu fokus pada keadaan awal yang merupakan himpunan nilai yang terdefinisi sebagai input (domain) dan keadaan akhir yang merupakan himpunan nilai yang terdefinisi sebagai output (range).

Fungsi terbagi menjadi dua jenis, yakni berikut.

1. **Built-in Functions**  
   Built-in functions atau dalam bahasa Indonesia berarti fungsi bawaan adalah kumpulan fungsi yang sudah terintegrasi dengan bahasa pemrograman Python sehingga tidak perlu mengimpor **modul** atau **library** tambahan. Fungsi bawaan ini menyediakan fungsi-fungsi inti dan merupakan dasar dari bahasa Python. Contoh dari fungsi bawaan adalah print(), len(), type(), range(), dan sebagainya.
2. **User-defined Functions**  
   User-defined functions atau dalam bahasa Indonesia berarti fungsi yang didefinisikan pengguna adalah jenis fungsi yang kita definisikan sendiri untuk melakukan tugas spesifik tertentu. Contoh dari **user-defined functions**adalah fungsi yang telah kita buat di awal materi ini tentang mencari luas persegi panjang.

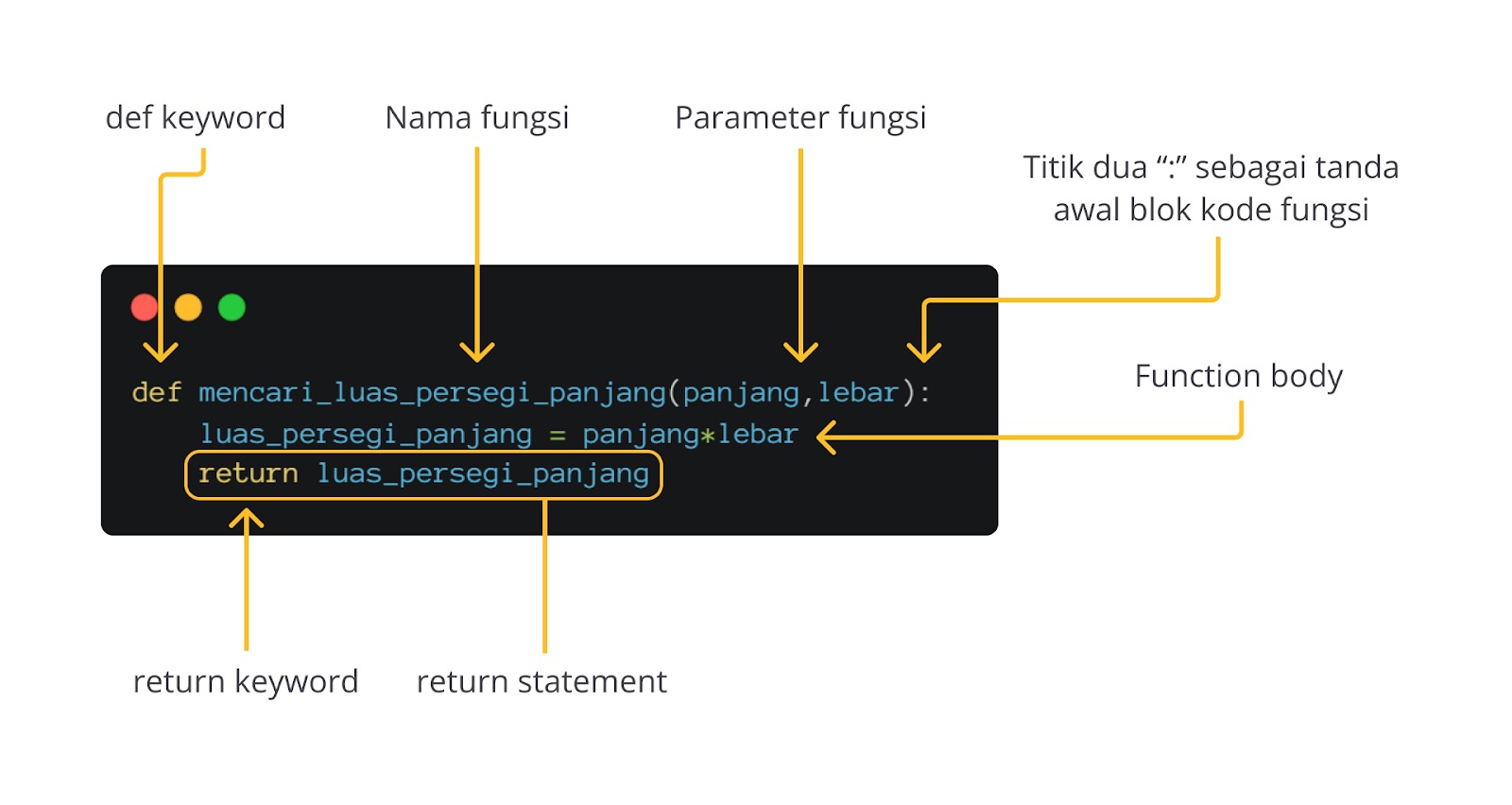
Namun, jika Anda ingin menggunakan fungsi di luar built-in functions, Anda bisa mengimpor sebuah library. Library adalah koleksi banyaknya modul yang saling terkait dan dapat digunakan berulang kali. Library dalam Python terbagi menjadi dua jenis, yakni berikut.

1. **Python Standard Library**  
   Python Standard Library adalah jenis library yang telah terpasang secara otomatis ketika Anda melakukan instalasi Python. Python Standard Library berisi kumpulan modul dan paket yang disertakan secara default oleh Python. Paket (package) merupakan sebuah direktori yang berisi satu atau lebih modul yang terkait dan saling berhubungan.
2. **External Library**  
   Jika sebelumnya impor library tidak perlu dilakukan untuk Python Standard, berbeda halnya dengan external library yang mengharuskan Anda mengimpor library untuk bisa menggunakannya. External Library adalah jenis library yang dikembangkan oleh individu atau organisasi di luar tim inti pengembang Python.

Keterkaitan antara fungsi, modul, package, dan library dapat dilihat pada tabel berikut.

| **Nama** | **Definisi** | **Contoh** |
| --- | --- | --- |
| Fungsi | Blok kode yang dapat digunakan kembali untuk mengeksekusi fungsionalitas tertentu saat dipanggil. | print(), len(), mencari\_luas\_persegi\_panjang() |
| Built-in functions | Kumpulan fungsi yang sudah terintegrasi dengan bahasa pemrograman Python sehingga tidak perlu mengimpor **modul** atau **library** tambahan. | print(), len(), range() |
| User-defined functions | Jenis fungsi yang kita definisikan sendiri untuk melakukan tugas spesifik tertentu. | mencari\_luas\_persegi\_panjang() |
| Modul | File berisi kode Python berupa **fungsi**, kelas, dan sebagainya. | Math, dan semua file yang kita buat sendiri dengan ekstensi ".py" (main.py, var.py, dan sebagainya) |
| Package | Sebuah direktori berisi satu atau lebih **modul** yang terkait dan saling berhubungan. | NumPy, Pandas |
| Library | Koleksi dari banyaknya **modul dan paket**yang saling terkait dan dapat digunakan berulang kali. | Matplotlib, TensorFlow, Beautiful Soup |

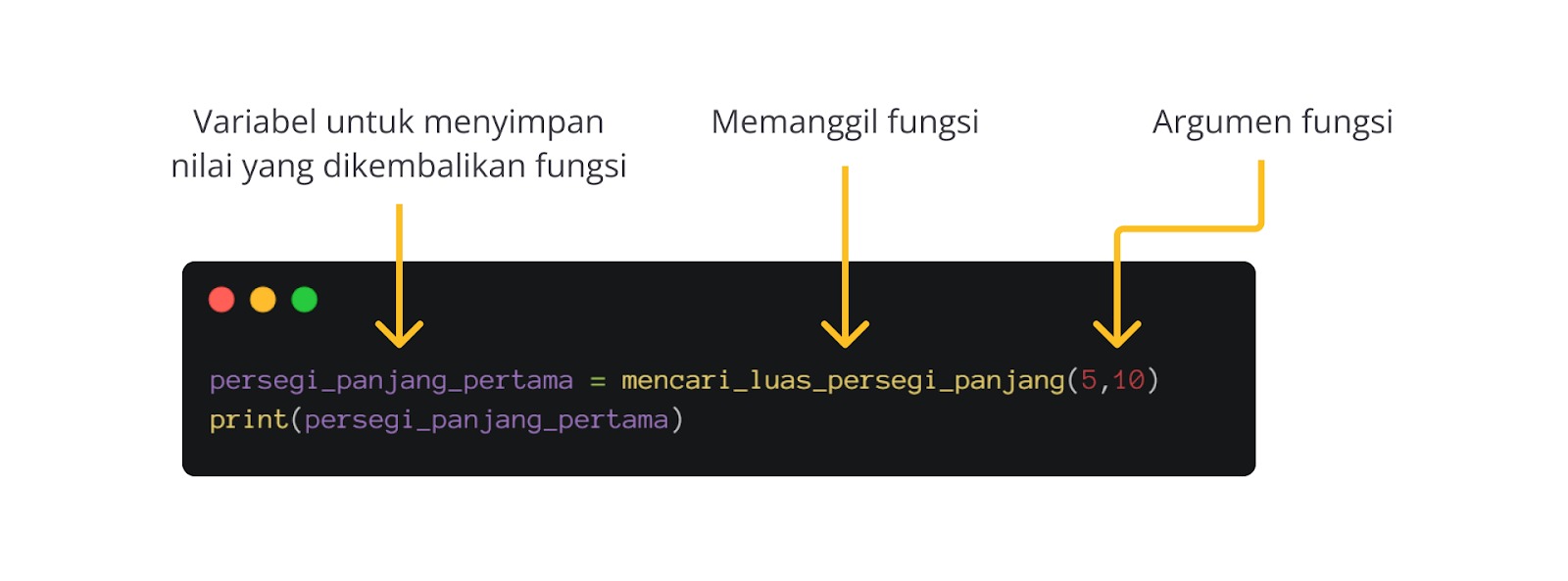
Untuk membuat fungsi sendiri (user-defined functions) dalam Python, kita dapat membuatnya dengan mengikuti struktur berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Fungsi di atas memiliki beberapa elemen yang dapat diikuti, yakni berikut.

1. Def: Keyword dari Python untuk membuat fungsi.
2. Nama fungsi: Nama yang Anda deklarasikan untuk fungsi yang akan dibuat.
3. Parameter fungsi: Variabel yang digunakan untuk menyimpan nilai dari argumen.
4. Setiap fungsi harus diakhiri dengan titik dua ":" untuk menandakan awal blok kode fungsi.
5. Setelah titik dua ":", di bawahnya kita mendefinisikan blok kode yang ingin dieksekusi.
6. Terakhir, kita menggunakan **return keyword**yang merupakan bagian dari **return statement**. Return statement bertujuan untuk mengembalikan nilai atau hasil eksekusi fungsi tersebut.

Untuk memanggil fungsi yang telah dibuat tersebut, kita dapat mengikuti struktur di bawah ini.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Dengan catatan sebagai berikut.

1. Nama fungsi, tentu Anda harus menyebutkan nama fungsi yang ingin digunakan. Namun ingat, gunakan kurung tutup "()" untuk memanggilnya.
2. Argumen bisa dikatakan sebagai nilai yang diberikan kepada fungsi. Nantinya, nilai tersebut akan disimpan dalam parameter fungsi.

Terakhir, untuk membuat fungsi lebih mudah dipahami oleh programmer lain, kita bisa membuat **dokumentasi berupa docstring**. Docstringmerupakan akronim dari documentation string, bertujuan untuk membuat dokumentasi terhadap fungsi yang dibuat. Umumnya, dokumentasi yang dijelaskan berupa argumen, return, deskripsi fungsi, dan sebagainya.

Contohnya sebagai berikut.

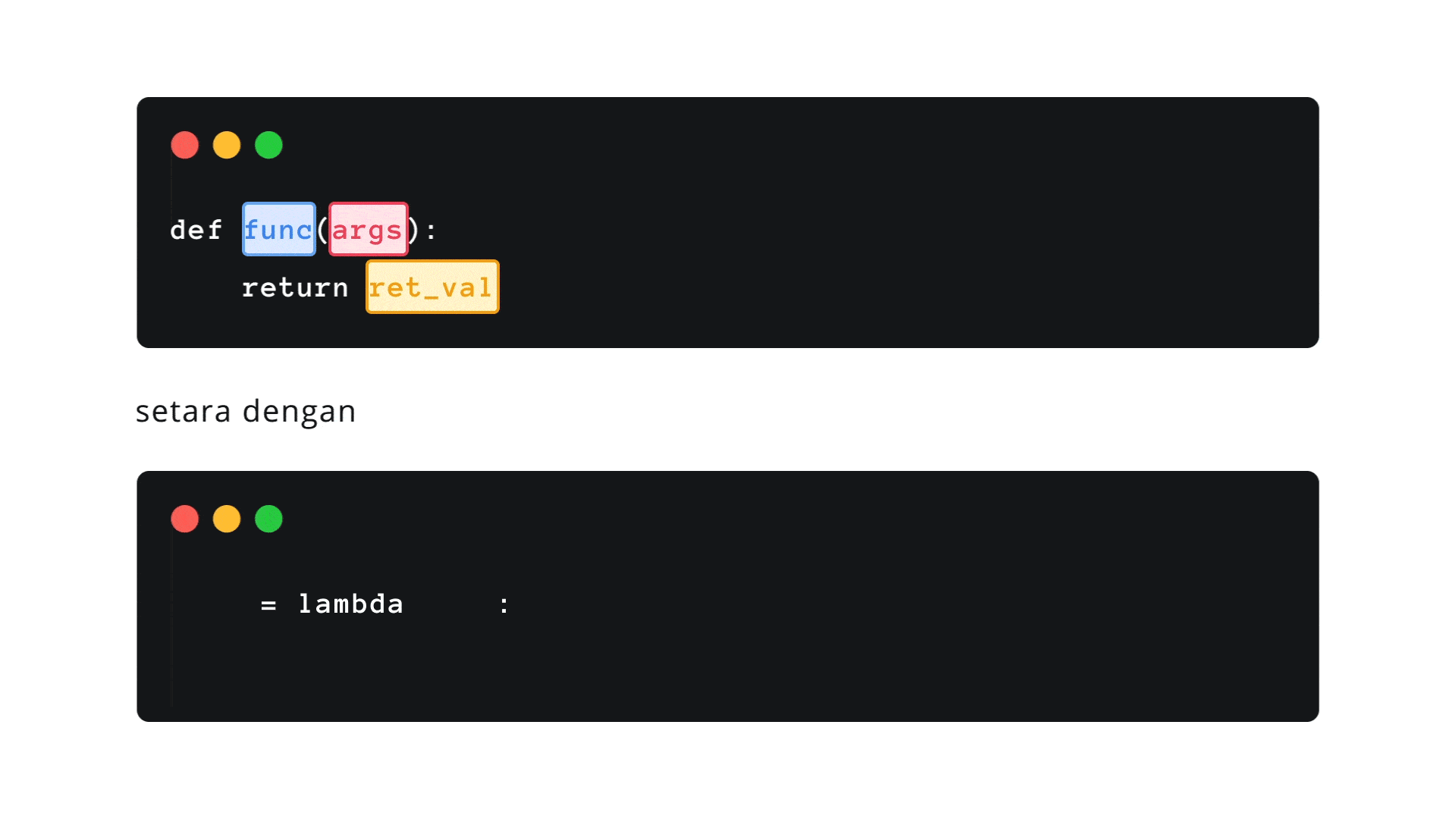
1. def mencari\_luas\_persegi\_panjang(panjang,lebar):
2. """
3. Fungsi ini digunakan untuk menghitung luas persegi panjang.
5. Args:
6. panjang (int): Panjang persegi panjang.
7. lebar (int): Lebar persegi panjang.
9. Returns:
10. int: Luas persegi panjang hasil perhitungan.
11. """
13. luas\_persegi\_panjang = panjang\*lebar
14. return luas\_persegi\_panjang
16. persegi\_panjang\_pertama = mencari\_luas\_persegi\_panjang(5,10)
17. print(persegi\_panjang\_pertama)

Pada kode di atas, kita mendefinisikan docstring dengan memberikan blok komentar dengan tiga double quote (""") tepat di bawah "def" keyword. Elemen yang kita masukkan dalam docstring tersebut adalah deskripsi untuk menjelaskan tujuan fungsi yang dibuat, argumen untuk menjelaskan argumen yang dapat diterima oleh fungsi tersebut, dan return untuk menjelaskan nilai yang akan dikembalikan oleh fungsi.

Argumen dan parameter pada fungsi memiliki beragam jenisnya. Secara umum, berikut adalah jenis-jenis dari argumen dan parameter.

1. **Keyword Argument**  
   Keyword Argument adalah jenis argumen yang disertai dengan nama parameter (*identifier*) dan secara eksplisit disebutkan.
   1. def mencari\_luas\_persegi\_panjang(panjang,lebar):
   2. luas\_persegi\_panjang = panjang\*lebar
   3. return luas\_persegi\_panjang
   5. persegi\_panjang\_pertama = mencari\_luas\_persegi\_panjang(panjang=5, lebar=10)
2. **Positional Argument**  
   Kebalikan dari keyword adalah positional, artinya Anda tidak menyebutkan nama parameter (*identifier*) secara eksplisit.
   1. def mencari\_luas\_persegi\_panjang(panjang,lebar):
   2. luas\_persegi\_panjang = panjang\*lebar
   3. return luas\_persegi\_panjang
   5. persegi\_panjang\_pertama = mencari\_luas\_persegi\_panjang(5,10)
3. **Positional-or-Keyword**  
   Jenis ini adalah parameter default dalam Python. Dengan jenis ini, kita dapat menggunakan positionalmaupun keyword argument ketika memanggil fungsi.
   1. def greeting(nama, pesan):
   2. return "Halo, " + nama + "! " + pesan
   4. print(greeting("Dicoding", "Selamat pagi!"))
   5. print(greeting(pesan="Selamat sore!", nama="Dicoding"))
4. **Positional-Only**  
   Parameter ini hanya dapat dimanfaatkan menggunakan jenis argumen posisi saat pemanggilan fungsi. Parameter ini ditentukan menggunakan sintaks "/".
   1. def penjumlahan(a, b, /):
   2. return a + b
   4. print(penjumlahan(8, 50))
5. **Keyword-Only**  
   Parameter ini kebalikan dari yang sebelumnya. Kita harus menggunakan keyword argument untuk memanggil fungsi dengan jenis parameter ini. Parameter ini ditentukan dengan sintaks "\*" (asterisk).
   1. def greeting(\*, nama, pesan):
   2. return "Halo, " + nama + "! " + pesan
   4. print(greeting(pesan="Selamat sore!",nama="Dicoding"))
6. **Var-Positional**  
   Parameter ini menampung jumlah argumen posisi yang bervariasi saat pemanggilan fungsi. Parameter ini ditentukan dengan menggunakan sintaks **\*args**.
   1. def hitung\_total(\*args):
   2. print(type(args))
   3. total = sum(args)
   4. return total
   6. print(hitung\_total(1, 2, 3))
7. **Var-Keyword**  
   Parameter ini dapat menampung jumlah keyword argument yang bervariasi saat pemanggilan fungsi. Parameter ini ditentukan dengan menggunakan sintaks **\*\*kwargs** yang berperan sebagai dictionary (seperti tipe datanya). Argumen pada pemanggil fungsi akan berperan sebagai **value**dan parameter (identifier) berperan sebagai **key.**
   1. def cetak\_info(\*\*kwargs):
   2. info = ""
   3. for key, value in kwargs.items():
   4. info += key + ': ' + value + ", "
   5. return info
   7. print(cetak\_info(nama="Dicoding", usia="17", pekerjaan="Python Programmer"))

Selain fungsi yang didefinisikan menggunakan def keyword, kita juga bisa membuat versi one-liner dari fungsi tersebut. Konsep ini disebut dengan fungsi anonim atau juga dikenal sebagai lambda expression. Keterkaitan antara fungsi menggunakan def keyword dengan fungsi anonim dapat dilihat pada gambar bergerak (GIF) berikut.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

**Nama fungsi (func)** setara dengan nama variabel yang digunakan untuk menyimpan ekspresi lambda, **args**adalah argumen yang kita butuhkan untuk dioperasikan, dan **ret\_val** merupakan nilai yang kita kembalikan (return).

Terakhir, kita dapat mengimpor file berisi fungsi dari satu file ke file yang lain. Hal ini karena setiap file berekstensi **.py**yang kita buat, dikenal juga sebagai modul oleh Python. Untuk mengimpor fungsi yang diinginkan dari file yang telah ditentukan, Anda hanya perlu menggunakan pernyataan impor. Misalnya, jika Anda memiliki fungsi dalam file **hello.py**yang ingin diimpor ke file utama bernama **main.py**, gunakan kode berikut dalam main.py.

1. import hello

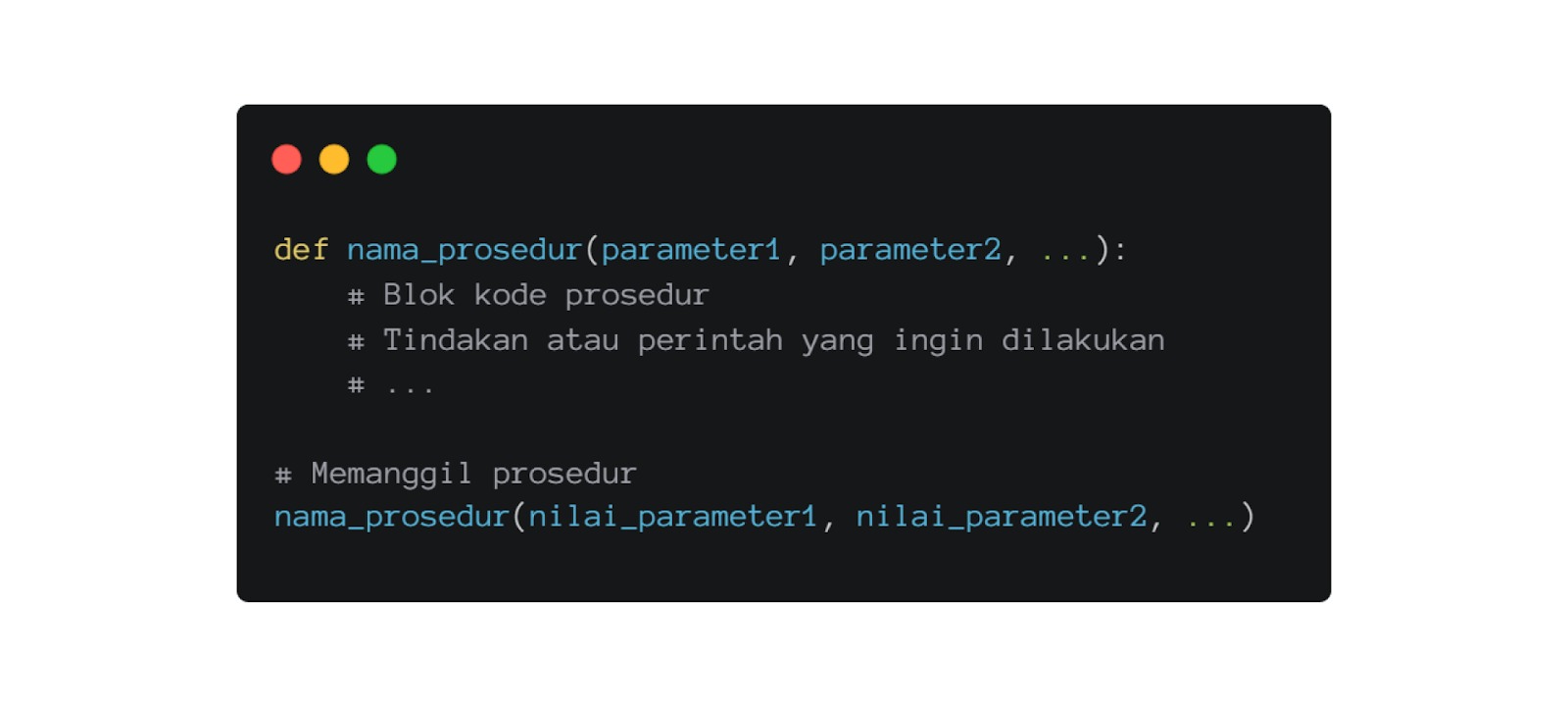
Anda juga bisa mengimpor kode, seperti fungsi, kelas, hingga variabel secara spesifik. Misalnya Anda ingin mengimpor fungsi “mencari\_luas\_persegi\_panjang” dan variabel “nama” dari modul hello. Gunakan kode di bawah ini.

1. from hello import mencari\_luas\_persegi\_panjang, nama

**Prosedur**

Dalam KBBI, kata [prosedur](https://kbbi.kemdikbud.go.id/entri/prosedur) memiliki makna sebagai tahap kegiatan untuk menyelesaikan suatu aktivitas. Hal ini sama seperti prosedur sebagai subprogram yang merupakan pengelompokan instruksi-instruksi yang sering dipakai dalam program.

Berbeda dengan fungsi, prosedur tidak mengharuskan adanya parameter input atau output dan dapat dipandang sebagai fungsi yang tidak menghasilkan nilai. Dalam Python, prosedur didefinisikan dengan **return tanpa ekspresi atau nilai**yang dihasilkan di akhir fungsi.

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Secara konsep, gambar di atas merupakan kerangka dasar prosedur pada Python. Sekilas memang sangat mirip dengan fungsi, hanya saja kita tidak mendefinisikan **return**dan bahkan **return value**.

Untuk memanggil prosedur, caranya serupa seperti Anda memanggil fungsi. Cukup mendefinisikan satu baris instruksi, seperti "greeting()". Untuk pemberian argumen dan parameter pada prosedur, kita dapat memakai cara yang sama seperti pada fungsi yang telah dijelaskan sebelumnya.

1. def greeting(name):
2. print("Halo " + name + ", Selamat Datang!")
4. greeting("Dicoding Indonesia")

Anda juga bisa membuat prosedur tanpa memiliki parameter input sehingga hanya memiliki body kode saja. Contohnya, kita membuat prosedur greeting tanpa parameter name dan ia hanya akan menampilkan pesan “Halo Selamat Datang!”.

1. def greeting():
2. print("Halo Selamat Datang!")
4. greeting()

**Rangkuman Object-Oriented Programming (OOP)**

Kita sudah berada di penghujung materi Object-Oriented Programming (OOP). Sampai sejauh ini, Anda diharapkan paham untuk mengimplementasikan konsep OOP ke dalam setiap program yang Anda bangun. Mari kita rangkum secara saksama.

**Duck Typing**

Duck typing merupakan konsep yang menjelaskan bahwa sebuah tipe atau class dari sebuah object tidak lebih penting daripada methodyang menjadi perilakunya. Konsep ini berbunyi "**if it walks like a duck and it quacks like a duck, then it must be a duck**" yang artinya jika sesuatu berjalan seperti bebek dan bersuara seperti bebek, kemungkinan besar ia adalah bebek.

Python ingin memberikan keleluasaan terhadap para developernya untuk tidak perlu mencemaskan tipe atau kelas (class) dari sebuah objek (object), yang lebih penting adalah kemampuan melakukan operasinya (method).

**Class, Object, dan Method**

Object-oriented programming adalah paradigma pemrograman berorientasi pada pengorganisasian kode menjadi objek-objek yang memiliki atribut dan perilaku (method). Objek adalah turunan dari class dan kelas merupakan cetakan yang memungkinkan kita dapat membuat banyak objek berdasarkan cetakan tersebut.

Method adalah perilaku atau tindakan yang dapat dilakukan oleh objek atau kelas. Sebagaimana halnya maju, mundur, berbelok, dan berhenti pada contoh sebelumnya. Atribut adalah variabel yang menjadi identitas dari objek atau kelas, seperti warna dan kecepatan pada contoh sebelumnya.

Mari sederhanakan dengan tabel berikut.

| **Nama** | **Deskripsi** | **Contoh** |
| --- | --- | --- |
| Class (Kelas) | Cetakan (blueprint) untuk membuat objek-objek dengan karakteristik dan perilaku yang serupa. | Mobil; Manusia. |
| Object (Objek) | Turunan atau perwujudan dari kelas. | Mobil Dicoding; Budi, Herman, Asep. |
| Perilaku (Method) | Perilaku atau tindakan yang dapat dilakukan oleh objek atau kelas. | Maju, mundur, berbelok, berhenti. |
| Atribut | Variabel yang menjadi identitas dari objek atau kelas. | Warna, kecepatan, merek. |

**Class**

Pembuatan class dalam Python mirip seperti fungsi, yakni perlu menggunakan keyword untuk bisa membuatnya. Keyword atau kata kunci untuk membuat kelas dalam Python adalah "class".

1. class Mobil:
2. pass

**Object (Objek)**

Untuk memanggil kelas yang telah dibuat, kita membuat sebuah objek. Berdasarkan KBBI dari kemendikbud, [objek](https://kbbi.kemdikbud.go.id/entri/objek) merupakan benda, hal, dan sebagainya yang dijadikan sasaran untuk diteliti, diperhatikan, dan sebagainya. Keterkaitan antara objek dan class sangat erat. Contohnya, jika Anda membuat kelas bernama manusia, objeknya adalah manusia dengan nama yang berbeda.

Anda bisa umpamakan kelas adalah bentuk abstrak dari objek layaknya cetakan atau *blueprint*. Saat kelas diwujudkan menjadi bentuk yang lebih nyata, proses ini disebut sebagai instansiasi. Itulah sebabnya objek disebut juga sebagai instance atau **instance of the class**.

1. class Mobil:
2. # Atribut
3. warna = "Merah"
5. mobil\_1 = Mobil()

**Atribut**

Dalam Python, ada dua jenis atribut kelas yang dapat dibagi, yaitu **atribut kelas**dan **atribut objek atau instance**. Atribut kelas adalah jenis atribut yang secara otomatis terdefinisi dan menjadi bawaan kelas ketika instance dibuat berdasarkan kelas tersebut. Anda dapat menganggapnya sebagai nilai default atau bawaan dari kelas. Jika Anda membuat beberapa objek berdasarkan kelas yang memiliki jenis atribut ini, setiap objek akan memiliki atribut yang sama dengan nilai yang sama.

Namun, perlu diperhatikan bahwa jenis atribut kelas memiliki kelemahan, yaitu ketika nilai atribut kelas diubah, perubahan tersebut akan memengaruhi semua objek yang dibuat berdasarkan kelas tersebut.

1. **Atribut Kelas**  
   Atribut kelas adalah jenis atribut yang melekat pada kelas sehingga menjadi bawaan ketika membuat sebuah instance.
   1. class Mobil:
   2. # Atribut kelas
   3. warna = "Merah"
   5. mobil1 = Mobil()
   6. print(mobil1.warna)
2. **Atribut Instance**  
   Atribut instance atau atribut objek adalah atribut yang terkait dengan instance atau objek itu sendiri, bukan kelas.
   1. class Mobil:
   2. # Atribut Instance
   3. def \_\_init\_\_(self):
   4. self.warna = 'Merah'
   6. mobil\_1 = Mobil()
   7. print(mobil\_1.warna)

Untuk membuat atribut instance, kita perlu membuat atribut tersebut melalui class constructor.

**Class Constructor**

Pembangun kelas atau class constructor adalah sebuah **fungsi khusus**dalam Python yang digunakan untuk menentukan nilai atau kondisi awal dari suatu kelas. Dengan fungsi ini, saat kita melakukan proses instansiasi atau pembuatan objek baru, hal pertama yang dilakukan adalah memanggilnya terlebih dahulu.

Tidak hanya untuk membuat atribut, kita dapat menambahkan parameter lain dalam class constructor.

1. class Mobil:
2. def \_\_init\_\_(self, warna, merek, kecepatan):
3. self.warna = warna
4. self.merek = merek
5. self.kecepatan = kecepatan
7. mobil\_1 = Mobil('Merah', 'DicodingCar', 160)
9. print(mobil\_1.warna)
10. print(mobil\_1.merek)
11. print(mobil\_1.kecepatan)
13. """
14. Output:
15. Merah
16. DicodingCar
17. 160
18. """

**Method**

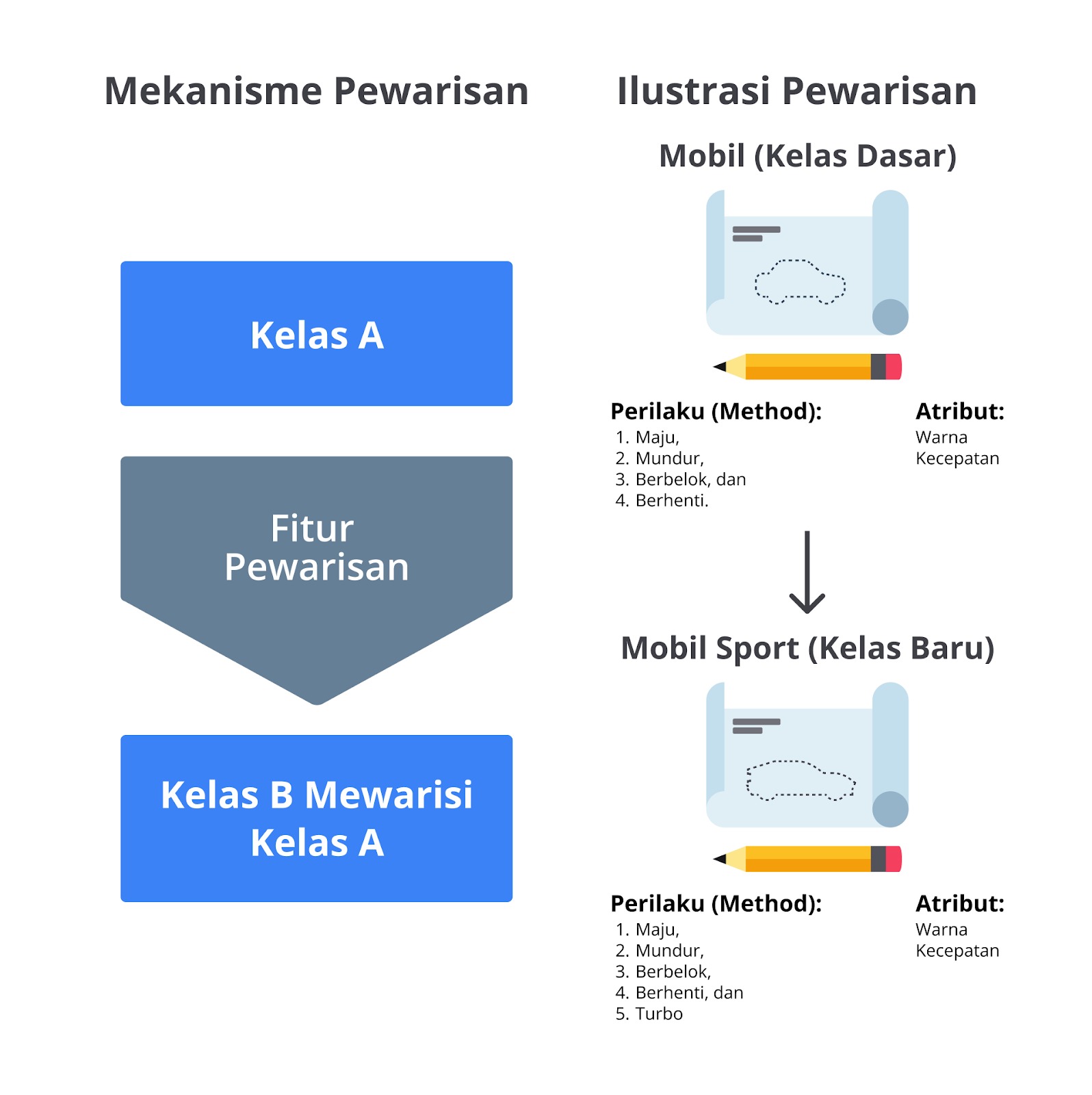
Method merupakan perilaku atau tindakan yang dapat dilakukan oleh objek atau kelas. Dalam pembuatan metode, sebenarnya kita membuat fungsi di dalam kelas itu sendiri. Dengan kata lain, kita menggunakan kata kunci **"def"** atau membuat fungsi sebagai suatu metode. Method sendiri dibagi menjadi tiga jenis.

1. **Metode dari Object (Object Method)**  
   Jenis pertama adalah method yang melekat terhadap objek. Ciri dari jenis metode ini adalah **adanya parameter *self***yang merujuk pada objek saat ini yang dibuat.
   1. class Mobil:
   2. def \_\_init\_\_(self, warna, merek, kecepatan):
   3. self.warna = warna
   4. self.merek = merek
   5. self.kecepatan = kecepatan
   7. def tambah\_kecepatan(self):
   8. self.kecepatan += 10
2. **Metode secara Statis(Static Method)**  
   Static method adalah fungsi atau method pada sebuah kelas yang bersifat statis. Artinya metode atau fungsi ini bersifat independen dan tidak terikat pada instance kelas. Metode ini dapat dianggap seperti kita membuat fungsi seperti biasa, tetapi didefinisikan dalam kelas. Jadi, ini menjadi perilaku untuk kelas tersebut. Untuk membuat static method, Anda bisa menambahkan dekorator **@staticmethod**tepat sebelum mendefinisikan fungsi atau metode.
   1. class Mobil:
   2. def \_\_init\_\_(self, merek):
   3. self.merek = merek
   5. @staticmethod
   6. def intro\_mobil():
   7. print("Ini adalah metode dari kelas Mobil")
   9. Mobil.intro\_mobil()
   10. mobil\_1 = Mobil("DicodingCar")
   11. mobil\_1.intro\_mobil()
3. **Metode dari Class(Class Method)**  
   Metode terakhir adalah class method yang termasuk jenis metode cukup spesial dalam Python. Jika object method identik dengan parameter *self* yang merujuk pada objek, **class method**juga memerlukan sebuah parameter yang merujuk pada kelas.
   1. class Mobil:
   2. def \_\_init\_\_(self, merek):
   3. self.merek = merek
   5. @classmethod
   6. def intro\_mobil(cls):
   7. print("Ini adalah metode dari kelas Mobil")
   9. Mobil.intro\_mobil()
   10. mobil\_1 = Mobil("DicodingCar")
   11. mobil\_1.intro\_mobil()

**Inheritance**

Berikut adalah rangkuman materi terkait inheritance atau dalam bahasa Indonesia disebut pewarisan.

**Mekanisme Pewarisan**

[](https://www.dicoding.com/academies/86/tutorials/33363?from=33358)

Untuk melakukan pewarisan, anggap kita memiliki "kelas A" sebagai induk atau kelas dasar. Dari kelas A tersebut, kita membuat kelas baru bernama "kelas B" sebagai kelas turunan dari kelas yang didapatkan (kelas A). Ketika kelas B mewarisi kelas A, secara otomatis kelas ini memiliki fitur-fitur yang dimiliki oleh kelas A tersebut, dalam hal ini atribut-atribut dan metode-metode.

Jika kelas B memiliki nama metode yang sama dengan kelas A, metode tersebut akan menimpa metode yang diwariskan oleh kelas A.

1. class Mobil:
2. def \_\_init\_\_(self, warna, merek, kecepatan):
3. self.warna = warna
4. self.merek = merek
5. self.kecepatan = kecepatan
7. def tambah\_kecepatan(self):
8. self.kecepatan += 10

11. class MobilSport(Mobil):
12. def turbo(self):
13. self.kecepatan += 50

**Override**

Ketika kita membuat metode baru di kelas turunan (kelas baru) dengan nama yang sama seperti metode di kelas induk, itu akan menyebabkan metode baru menimpa (override) metode pada kelas induk.

1. class Mobil:
2. def \_\_init\_\_(self, warna, merek, kecepatan):
3. self.warna = warna
4. self.merek = merek
5. self.kecepatan = kecepatan
7. def tambah\_kecepatan(self):     # tambah\_kecepatan
8. self.kecepatan += 10
10. class MobilSport(Mobil):
11. def turbo(self):
12. self.kecepatan += 50
14. def tambah\_kecepatan(self):     # tambah\_kecepatan
15. self.kecepatan += 20

**Super**

Terakhir adalah super, yakni fungsi yang digunakan untuk menggunakan metode atau atribut dari kelas induk, tetapi tidak ingin menuliskan ulang semua kode. Nama super sebenarnya merujuk pada kelas induk yang disebut juga sebagai **super class**. Kita bisa memanfaatkan konsep ini untuk menghindari kode berulang dan memanfaatkan fungsi yang sudah ada pada kelas induk (super class).

1. class Mobil:
2. def \_\_init\_\_(self, warna, merek, kecepatan):
3. self.warna = warna
4. self.merek = merek
5. self.kecepatan = kecepatan
7. def tambah\_kecepatan(self):
8. self.kecepatan += 10

11. class MobilSport(Mobil):
12. def turbo(self):
13. self.kecepatan += 50
15. def tambah\_kecepatan(self):
16. super().tambah\_kecepatan()     # Super
17. print("Kecepatan Anda meningkat! Hati-Hati!")

**Rangkuman Style Guide pada Python**

Kita sudah berada di penghujung materi style guide pada Python. Sampai sejauh ini, Anda diharapkan paham untuk memahami cara membangun kode Python yang lebih baik dan benar sesuai panduan yang telah diberikan oleh Python melalui PEP8. Mari kita rangkum secara saksama.

**Pengecekan Style Guide PEP8**

Saat membangun program pada Python, seringkali kode yang dibuat 'cukup berantakan' sehingga kita perlu mengecek bahwa kode tersebut sudah rapi dan benar dengan mengacu pada panduan yang telah diberikan oleh Python, yaitu PEP 8.

PEP atau *Python Enhancement Proposals* adalah panduan yang telah menjadi acuan untuk perkembangan Python. Salah satu panduan tersebut membahas mengenai arahan gaya penulisan (style guide) yang baik dan benar ketika Anda ingin membangun kode menggunakan Python. Panduan tersebut adalah PEP8 yang berjudul "[Style Guide for Python Code](https://peps.python.org/pep-0008/)".

Tujuan dari panduan ini agar kode Anda lebih mudah dibaca dan dipahami oleh programmer lain serta menghindari kemungkinan kesalahan yang akan muncul.

**Lint**

Lint atau linting adalah proses pengecekan kode atas kemungkinan terjadi kesalahan (error), termasuk dalam proses ini adalah mengecek kesesuaian terhadap arahan gaya penulisan kode (style guide) PEP8. Aplikasi yang digunakan untuk proses ini disebut linter. Integrasi linter dengan editor kode Anda akan membuat efisien dalam menulis kode Python. Pertimbangan ini karena keluaran atau output dari aplikasi linter hanya berupa teks singkat berupa keterangan dan kode Error atau Warning atau Kesalahan Konvensi Penamaan (Naming Conventions).

Dengan lint atau linting akan meminimalkan kode Anda mengalami error, salah satunya karena kesalahan indentasi di Python. Sebelum kode Anda diproses oleh interpreter Python dengan IndentationError, lint akan memberitahukannya lebih dahulu ke Anda. Berikut adalah tiga jenis aplikasi linter.

1. **Pycodestyle** (sebelumnya bernama pep8)  
   [Pycodestyle](https://github.com/PyCQA/pycodestyle) adalah aplikasi open source (berlisensi MIT/Expat) untuk membantu mengecek kode terkait gaya penulisan kode dengan konvensi PEP8.
2. **Pylint**  
   [Pylint](https://github.com/PyCQA/pylint) adalah aplikasi open source (berlisensi GPL v2) untuk melakukan analisis kode Python, mengecek untuk kesalahan (error) pemrograman, memaksakan standar penulisan kode dengan mengecek penulisan kode yang tidak baik, serta memberikan saran untuk refactoring sederhana.
3. **Flake8**  
   [Flake8](https://github.com/PyCQA/flake8) adalah aplikasi open source (berlisensi MIT) yang membungkus sejumlah kemampuan aplikasi lain, seperti pycodestyle, pyflakes, dan sejumlah (skrip/fitur) lainnya.

**Memformat Kode**

Jika proses lint atau linting hanya melakukan pengecekan, kali ini adalah arahan gaya penulisan kode agar bisa sesuai dengan PEP8. Kita akan kembali menggunakan beberapa aplikasi yang nantinya akan diinstal.

Proses memformat kode akan sama dengan cara melakukan proses linting, yaitu kita akan mengeksekusi script. Perbedaannya adalah output yang dihasilkan. Jika proses linting menghasilkan pesan dengan menunjukkan baris dan kode yang mengalami kesalahan, proses memformat kode akan memberikan pesan berupa kode yang telah diperbaiki. Ini artinya Anda tidak perlu mengubah kode secara manual. Berikut adalah tiga jenis aplikasi untuk memformat kode.

1. **black**  
   [black](https://github.com/psf/black) adalah proyek open source yang dikembangkan di repository Python Software Foundation (PSF) dengan lisensi MIT. Untuk mendapatkan gambaran, versi online (tidak resmi) ada di [https://black.now.sh](https://black.now.sh/).
2. **YAPF (Yet Another Python Formatter)**  
   [YAPF](https://github.com/google/yapf) adalah proyek open source yang dikembangkan di repository Google dengan lisensi Apache.
3. **autopep8**  
   [autopep8](https://github.com/hhatto/autopep8) adalah proyek open source (berlisensi MIT) yang termasuk paling awal untuk memformat kode dengan bantuan lint pycodestyle.

**Style Guide Statement Gabungan**

Setelah mengetahui aplikasi untuk pengecekan dan memformat kode, kali ini kita akan belajar cara membuat kode yang baik dan benar.

**Statement Gabungan**

Saat Anda membuat program dengan banyak statement, usahakan untuk tidak menggabungkan >1 statement pada baris yang sama.

**Disarankan** seperti ini.

1. if foo == 'blah':
2. do\_blah\_thing()
3. do\_one()
4. do\_two()
5. do\_three()

**Tidak** disarankan seperti ini.

1. if foo == 'blah': do\_blah\_thing()
2. do\_one(); do\_two(); do\_three()

Anda diperbolehkan untuk membuat sebuah konten/isi dari if/for/while yang cukup pendek untuk diletakkan dalam satu baris (program tetap berjalan). Namun, pastikan tidak melakukannya jika if/for/while Anda bertingkat atau bersifat multi clause, misalnya if-else, try-finally, dan sebagainya.

**Tidak** disarankan seperti ini.

1. if foo == 'blah': do\_blah\_thing()
2. for x in lst: total += x
3. while t < 10: t = delay()

**Sangat tidak** disarankan seperti ini.

1. if foo == 'blah': do\_blah\_thing()
2. else: do\_non\_blah\_thing()
3. try: something()
4. finally: cleanup()
5. do\_one(); do\_two(); do\_three(long, argument,
6. list, like, this)
7. if foo == 'blah': one(); two(); three()

**Penggunaan Trailing Commas**

Koma di bagian akhir (trailing commas) umumnya bersifat opsional, satu statement ketika ia bersifat wajib adalah saat kita membuat variabel menggunakan tipe tuple dengan satu elemen. Hal ini umumnya diperjelas dengan kurung untuk menghindari penghapusan atau pembersihan.

**Disarankan** seperti ini.

1. FILES = ('setup.cfg',)

**Tidak** disarankan seperti ini.

1. FILES = 'setup.cfg',

Tidak umum jika Anda meletakkan trailing comma pada baris tempat Anda menutup kurung/kurawal/siku seperti di bawah ini, kecuali dalam tuple dengan satu elemen seperti yang dijelaskan di atas.

**Disarankan** seperti ini.

1. FILES = [
2. 'setup.cfg',
3. 'tox.ini',
4. ]
5. initialize(FILES,
6. error=True,
7. )

**Tidak** disarankan seperti ini.

1. FILES = ['setup.cfg', 'tox.ini',]
2. initialize(FILES, error=True,)

**Anotasi Fungsi**

Anotasi fungsi adalah fitur yang memungkinkan kita untuk menambahkan informasi tambahan tentang parameter dan return value dari sebuah fungsi. Jika sebelumnya kita belajar menambahkan informasi terkait fungsi dengan menambahkan docstring, anotasi fungsi lebih spesifik untuk menjelaskan parameter dan return value.

Penggunaan anotasi fungsi sebaiknya menggunakan aturan baku untuk titik dua (:) dan menggunakan spasi untuk penggunaan arah panah atau arrow (->). Hal ini disebut sebagai type hints yang merujuk pada [PEP 484](https://www.python.org/dev/peps/pep-0484).

1. Yes:
2. def munge(input: str):  # Menambahkan informasi parameter bertipe string
3. pass
4. def munge() -> str:   # Menambahkan informasi return value bertipe string
5. pass
7. No:
8. def munge(input:str):  # Menambahkan informasi parameter bertipe string
9. pass
10. def munge()->str:   # Menambahkan informasi return value bertipe string
11. pass

Jika kita membuat fungsi yang menggabungkan anotasi dengan nilai parameter, sebaiknya tetap menggunakan spasi sebelum dan sesudah tanda sama dengan (=). Namun, ketika membuat fungsi biasa tanpa adanya anotasi, sebaiknya tidak menggunakan spasi sebelum dan sesudah tanda sama dengan (=).

1. Yes:
2. def LuasPersegiPanjang(panjang:int = 2, lebar=None):
3. pass
5. No:
6. def LuasPersegiPanjang(panjang: int=2, lebar = None):
7. pass

**Style Guide Prinsip Penamaan pada Python**

Penamaan pada Python mencakup banyak hal, seperti penamaan fungsi, kelas, dan sebagainya. Berikut adalah beberapa rekomendasi untuk penamaan yang meliputi overriding, penamaan deskriptif, fungsi, method, dan sebagainya. Namun, Anda juga dapat memilih mempertahankan styling yang sudah digunakan sebelumnya untuk menjaga konsistensi internal tim atau perusahaan. Ini karena konsistensi internal lebih diutamakan.

**Prinsip Overriding**

Nama yang dilihat oleh user publik sebaiknya merefleksikan penggunaan/fungsinya dan bukan implementasinya.

**Penamaan Deskriptif**

Penamaan deskriptif adalah cara untuk memberikan nama yang informatif, jelas, dan sesuai dengan tujuan dari elemen kode. Penamaan deskriptif ini meliputi variabel, fungsi, kelas, hingga konstanta.

Ada berbagai cara penamaan yang umum digunakan dalam Python. Pemilihan cara penamaan ini penting untuk menjaga konsistensi dan kejelasan kode. Penamaan ini juga merujuk pada PEP8 mengenai [Naming Conventions](https://peps.python.org/pep-0008/#naming-conventions) dan [Naming Styles](https://peps.python.org/pep-0008/#descriptive-naming-styles).

Berikut adalah beberapa cara penamaan yang umum:

1. Satu karakter huruf kecil: b
2. Satu karakter huruf besar: B
3. Huruf kecil: hurufkecil
4. Huruf kecil dengan pemisah kata garis bawah: huruf\_kecil\_dengan\_pemisah\_kata\_garis\_bawah
5. HURUF BESAR: HURUFBESAR
6. HURUF BESAR dengan pemisah garis bawah: HURUF\_BESAR\_DENGAN\_PEMISAH\_GARIS\_BAWAH
7. Huruf Besar di Awal Kata (CapWords, CamelCase): HurufBesarDiAwalKata (pastikan semua singkatan/akronim dituliskan dengan huruf besar, misalnya HTTPServerError, bukan HttpServerError)
8. Huruf Campuran: hurufCampuran (mirip dengan CapWords, hanya berbeda di karakter paling awal)
9. Huruf Besar di Awal Kata dengan Garis Bawah: Huruf\_Besar\_Di\_Awal\_Kata\_Dengan\_Garis\_Bawah

Python tidak menyarankan atau lebih tepatnya tidak dibutuhkan jika Anda membuat sebuah fungsi yang diawali huruf atau frasa, seperti 'f' jika fungsinya 'f\_mean()',  'r' jika fungsinya 'r\_name()', dan sebagainya.

Selain penggunaan huruf atau frasa yang tidak direkomendasikan, berikut adalah beberapa bentuk penamaan khusus yang umum ditemukan dalam penamaan fungsi. Ini juga bisa Anda terapkan pada penamaan variabel dan kelas.

1. \_diawali\_sebuah\_garis\_bawah: penamaan ini dapat digunakan untuk penggunaan internal lemah yang merujuk pada penggunaannya dengan lingkup tertentu.
2. diakhiri\_sebuah\_garis bawah\_: penamaan ini digunakan untuk mengatasi redundan dengan *keyword*/*reserved words* di Python.
3. \_\_diawali\_dua\_garis bawah: menegaskan bahwa sebuah objek adalah bagian dari kelas tertentu.
4. \_\_diawali\_dan\_diakhiri\_dua\_garis bawah\_\_: Objek atau atribut tertentu yang diciptakan Python untuk digunakan dalam program. Contohnya adalah  \_\_init\_\_, \_\_import\_\_ or \_\_file\_\_.

**Hal-Hal yang Harus Diperhatikan dalam Penamaan**

Berikut adalah hal-hal yang harus diperhatikan dalam penamaan fungsi, method, hingga penamaan kelas.

1. **Nama yang Dihindari**  
   Hindari karakter l (huruf L kecil), O (huruf o besar) atau I (huruf i besar) sebagai nama variabel satu karakter karena mereka sulit dibedakan dengan angka satu dan nol.
2. **ASCII Compatibility**  
   Merujuk pada [PEP 3131](https://www.python.org/dev/peps/pep-3131), suatu identifiers yang digunakan dalam Python Standard Library harus kompatibel dengan kode ASCII.
3. **Nama Paket dan Nama Modul**  
   Penamaan modul sebaiknya pendek atau singkat, menggunakan huruf kecil, dan opsional garis bawah (\_) untuk meningkatkan keterbacaan. Nama paket juga sebaiknya singkat, menggunakan huruf kecil, dan hindari garis bawah(\_).
4. **Nama Kelas**  
   Saat menamai kelas, gunakan CamelCase atau CapWords. Pastikan semua akronim (misal HTTP) ditulis keseluruhan dengan huruf besar.
5. **Penulisan Tipe Variabel**  
   Untuk penamaan variabel, umumnya menggunakan CamelCase atau CapWords.
6. **Nama Exception**  
   Untuk pengecualian (exception), Anda juga menerapkan konvensi penamaan kelas pada exception karena ia seharusnya bertipe kelas. Bedanya, tambahkan "Error" atau nama deskriptif lain pada nama exception Anda.
7. **Nama Variabel Global**  
   Dalam variabel global, penamaannya bisa mengikuti fungsi/modul yang bersifat publik. Anda bisa menggunakan garis bawah untuk menghindari variabel tersebut diimpor jika ia termasuk modul non-publik.
8. **Nama Fungsi, Parameter, dan Variabel**  
   Nama fungsi, parameter, dan variabel sebaiknya menggunakan huruf kecil dengan pemisahan menggunakan garis bawah untuk meningkatkan keterbacaan. mixedCase dapat digunakan jika ada dependensi dengan pustaka menggunakan style tertentu.
9. **Argumen Fungsi dan Method**  
   Dalam pembuatan fungsi dan method pada suatu kelas, ada beberapa hal yang perlu dipertimbangkan..
   1. Gunakan self sebagai argumen pertama jika Anda membuat instance method.
   2. Gunakan cls sebagai argumen pertama ketika Anda membuat class method.
10. **Nama Method dan Variabel Instance**  
    Saat membuat method dan variabel dalam suatu kelas, gunakan standar penamaan fungsi, yaitu gunakan huruf kecil dengan pemisah kata garis bawah untuk meningkatkan keterbacaan. Tambahkan garis bawah sebagai awalan untuk method non-publik dan variabel internal pada fungsi.
11. **Konstanta**  
    Dalam memberikan nama variabel bertipe konstanta, umumnya didefinisikan pada bagian atas modul dengan huruf besar, misalnya 'PI = 3,14'  atau  'TOTAL = 4.14213'.
12. **Selalu Persiapkan untuk Inheritance**  
    Saat membangun metode dan variabel dalam sebuah kelas, sebaiknya Anda dapat langsung mengetahui atribut pada metode dan variabel tersebut, publik atau non-publik. Jika Anda ragu, jadikan atributnya non-publik. Sebab, lebih mudah menjadikan sebuah variabel/method bersifat non-publik menjadi publik, dibandingkan sebaliknya.  
      
    Variabel atau method yang bersifat non-publik adalah suatu variabel atau method yang hanya digunakan untuk lingkup tertentu dan tidak diakses secara langsung di luar.  
      
    Method/Variabel publik dipersiapkan untuk pihak eksternal menggunakan kelas Anda. Anda juga otomatis berkomitmen untuk menghindari adanya *incompatible backward changes*atau suatu kode yang tidak dapat berjalan kembali setelah adanya perubahan.  
      
    Sebaliknya, method/variabel dengan atribut non-publik hanya digunakan oleh Anda sebagai developer. Itu juga tidak memberikan garansi kepada siapa pun bahwa Anda takkan mengubah atau menghapusnya. Di sini kita tidak menggunakan atribut private karena dalam Python tidak ada atribut yang benar-benar private.  
      
    Saat mendeklarasikan variabel/method tersebut, ikuti panduan Pythonic berikut.
    1. Atribut publik tidak menggunakan awalan garis bawah.
    2. Jika nama sebuah method/variabel publik sama dengan *reserved keyword*, tambahkan akhiran garis bawah. Hindari menyingkat atau mengurangi huruf.
    3. Pada data publik bersifat simpel, hindari nama yang terlalu panjang. Cukup dengan nama atribut sependek mungkin. Ingatlah bahwa pada masa depan Anda akan mungkin mengembangkan skema atau data ini sehingga nama sependek apa pun mungkin akan menguntungkan Anda.
    4. Jika Anda berniat untuk mewariskan atau membuat subclass dari kelas dan menginginkan sebuah variabel hanya digunakan di kelas utama saja, tambahkan awalan dua garis bawah. Ini akan memudahkan Anda karena Python mengenalinya sebagai konvensi kelas, untuk menghindari kemungkinan kesamaan nama atau implementasi.

**Rangkuman Unit Testing**

Kita sudah berada di penghujung materi Unit Testing. Sampai sejauh ini, Anda diharapkan telah paham terkait tes unit menggunakan library unittest pada Python. Sekarang, mari kita rangkum secara saksama.

**Pengantar Unit Testing**

Saat membangun aplikasi atau program yang lebih kompleks, aplikasi tersebut akan memunculkan dependensi, yaitu satu atau lebih fungsi digunakan oleh fungsi lain. Bahkan, ketika kita mulai membangun aplikasi dengan rekan kita, kita membuat fungsi yang digunakan oleh rekan ataupun sebaliknya. Dependensi tersebut tentu perlu dipastikan bahwa fungsionalitasnya dapat berjalan dengan baik dan tidak terganggu dengan adanya perubahan atau fungsi baru yang dibuat.

Di sinilah kita butuh pengujian (tes) untuk fungsi-fungsi tersebut. Pengujian sebenarnya dapat dibedakan menjadi dua tipe, yaitu **manual** dan **otomatis**.

1. Manual testing adalah proses pengujian yang dilakukan oleh seseorang yang ditunjuk sebagai tester (penguji).
2. Testing otomatis merupakan hal yang sebaliknya. Ini adalah pengujian yang dilakukan secara otomatis terhadap kode-kode yang kita bangun berdasarkan rencana pengujian (test plan).

Tidak hanya sekadar manual dan otomatis, ada juga **unit testing** dan **integration testing**.

1. Integration testing adalah pengujian yang bertujuan untuk menguji suatu sistem sebagai satu kesatuan.
2. Unit testing adalah pengujian yang lebih spesifik dan fokus terhadap bagian-bagian kecil.

Dalam Python, untuk melakukan unit testing dapat menggunakan library **unittest**. Layaknya sebuah framework pengujian, unittest mendukung beberapa hal esensial berikut.

* Pengujian secara otomatis.
* Kode awal proses (setup) dan akhir proses (shutdown) yang dapat digunakan ulang.
* Penyatuan sejumlah pengujian dalam sebuah koleksi.
* Terpisahnya framework pengujian dari framework pelaporan (*reporting*).

Library unittest mendukung sejumlah konsep penting yang berorientasi objek, antara lain berikut.

* Test fixture merepresentasikan **persiapan**yang dibutuhkan untuk melakukan satu pengujian atau lebih serta proses pembersihannya (cleanup).
* Test case adalah sebuah unit dari pengujian ketika ia mengecek sejumlah respons dari sebagian kelompok masukan. unittest menyediakan basis class dan TestCase yang akan digunakan untuk membuat kasus pengujian baru.
* Test suite adalah sebuah koleksi dari kasus-kasus pengujian, koleksi dari test suite itu sendiri, atau gabungan keduanya.
* Test runner adalah komponen yang akan mengatur (*orchestrates*) eksekusi dari pengujian-pengujian dan menyediakan keluaran untuk pengguna.

**Penerapan Unit Test dengan Library unittest**

Unit melakukan unit testing, Anda bisa menggunakan library bawaan dari Python, yaitu **unittest**.

1. import unittest

Berikut adalah salah satu penerapan unit testing dengan menggunakan library unittest.

1. import unittest
3. def koneksi\_ke\_db():
4. print("[terhubung ke db]")
6. def putus\_koneksi\_db(db):
7. print("[tidak terhubung ke db {}]".format(db))
9. class User:
10. username = ""
11. aktif = False
13. def \_\_init\_\_(self, db, username):  # using db sample
14. self.username = username
16. def set\_aktif(self):
17. self.aktif = True

20. class TestUser(unittest.TestCase):
21. # Test Case 1
22. def test\_user\_default\_not\_active(self):
23. db = koneksi\_ke\_db()
24. dicoding = User(db, "dicoding")
25. self.assertFalse(dicoding.aktif)  # tidak aktif secara default
26. putus\_koneksi\_db(db)
28. # Test Case 2
29. def test\_user\_is\_active(self):
30. db = koneksi\_ke\_db()
31. dicoding = User(db, "dicoding")
32. dicoding.set\_aktif()  # aktifkan user baru
33. self.assertTrue(dicoding.aktif)
34. putus\_koneksi\_db(db)
36. if \_\_name\_\_ == "\_\_main\_\_":
37. # Test Runner
38. unittest.main()

**Rangkuman Library Populer pada Python**

Kita sudah berada di penghujung materi Library Populer pada Python. Sampai sejauh ini, Anda diharapkan telah memahami library populer pada Python yang beragam dan cara penggunaannya. Sekarang, mari kita rangkum secara saksama.

**Pengenalan Library**

Library adalah koleksi dari banyaknya modul dan paket yang saling terkait dan dapat digunakan berulang kali. Paket atau package adalah sebuah direktori yang berisi satu atau lebih modul yang terkait dan saling berhubungan. Jumlah library Python sangat banyak yang terbagi menjadi **Python Standard Library**dan **Python External Library**.

Python Standard Library adalah jenis library yang telah terinstal secara otomatis ketika kita melakukan instalasi Python. Anda tidak perlu melakukan instalasi kembali jika ingin menggunakannya. Beberapa contoh Python Standard Library adalah “os”, “datetime”, “re”, serta lainnya yang dapat Anda baca lebih lengkap pada laman [berikut](https://docs.python.org/3/library/).

Python External Library adalah kumpulan kode yang telah dikembangkan oleh orang lain atau komunitas dan disediakan dalam bentuk paket atau modul yang dapat diimpor. Jenis library ini mengharuskan Anda untuk melakukan instalasi agar dapat digunakan. External library ini dikembangkan oleh individu atau organisasi di luar tim inti pengembang Python.

Untuk melakukan instalasi library eksternal, Anda dapat melakukan beberapa cara, seperti menggunakan PIP dan conda.

**PIP**

PIP adalah package manager resmi dari Python yang dapat digunakan untuk mengunduh, menginstal, menghapus, dan mengelola package Python dari Python Package Index (PyPI) dan repositori lainnya. [PyPI](https://pypi.org/) merupakan repositori online yang menyediakan ribuan package dari Python yang siap digunakan oleh para pengembang.

Selain mengelola paket, Anda juga bisa membuat lingkungan virtual dalam Python menggunakan PIP. Hanya saja, pip cenderung difokuskan untuk mengelola instalasi paket dibanding lingkungan virtual.

Kabar baiknya, pip biasanya telah terpasang secara otomatis jika Anda menggunakan Python 2 untuk versi 2.7.9 ke atas atau Python 3 untuk versi 3.4 ke atas. Maka dari itu, silakan periksa bahwa pip telah terpasang dengan menjalankan perintah berikut.

|  |
| --- |
| pip --version |

Jika lokal komputer Anda belum memiliki pip, Anda bisa mengikuti langkah-langkah berikut.

1. Unduh file berikut: <https://bootstrap.pypa.io/get-pip.py>.
2. Buka terminal dan buka folder tempat Anda menyimpan file yang telah diunduh. Kemudian jalankan perintah berikut.

|  |
| --- |
| python get-pip.py |

Sekarang, Anda telah menyiapkan pip sebagai package manager yang mendukung instalasi package dan library Python. Untuk melakukan instalasi, Anda bisa mengikuti perintah berikut.

|  |
| --- |
| pip install <nama-package> |

Ganti <nama-package> dengan nama package atau library yang ingin Anda unduh. Untuk menghapus package, Anda bisa mengikuti perintah berikut.

|  |
| --- |
| pip uninstall <nama-package> |

Silakan ganti <nama-package> dengan package atau library yang ingin Anda hapus.

**Conda**

Conda adalah package manager dan environment manager untuk Python. Conda memungkinkan kita untuk membuat dan mengelola lingkungan (environment) terisolasi atau terpisah satu sama lain. Dengan terisolasinya setiap lingkungan tersebut, menguntungkan kita untuk mencegah konflik yang terjadi antar proyek.

Dengan adanya lingkungan yang terisolasi, Anda bisa menyesuaikan semua library, modul, hingga versi Python sesuai dengan kebutuhan masing-masing proyek. Conda sendiri dapat diinstal melalui Anaconda dan Miniconda.

Anaconda adalah sistem distribusi perangkat lunak yang di dalamnya mencakup Conda. Ketika menginstal Anaconda, Anda pun dapat menggunakan beberapa library dan plugin tambahan melalui Anaconda tersebut. Miniconda adalah versi ringan dari Anaconda. Miniconda hanya berisi conda dan beberapa package dasar yang diperlukan untuk menjalankannya.

Conda sendiri hadir dalam dua bentuk utama: “conda” sebagai package dan environment manager serta “conda-forge” sebagai sebuah repositori berisi ribuan paket yang disediakan oleh komunitas conda.

Kelebihan Conda adalah sifat tidak terbatasnya pada bahasa pemrograman Python. Conda mendukung paket-paket dalam bahasa pemrograman lain, seperti R.

**Library Text Processing**

Library text processing bertujuan untuk melakukan pemrosesan teks dan menyederhanakan serta mempercepat tugas-tugas pemrosesan teks. Beberapa library populer terkait pemrosesan teks adalah berikut.

1. String  
   [String](https://docs.python.org/3/library/string.html)adalah salah satu modul bawaan Python yang tidak perlu dideklarasikan. Pada modul string ada fungsi-fungsi yang dapat dioperasikan pada variabel bertipe string seperti di bawah.
   * upper(): Ubah setiap huruf dalam string menjadi huruf kapital.
   * lower(): Ubah setiap huruf dalam string menjadi huruf kecil.
   * split(): Pisahkan teks berdasarkan delimiter (karakter pemisah).
   * title(): Jadikan setiap awal kata kapital.
   * zfill(): Tambahkan nol di awal string sebanyak nilai yang ada pada parameter.
2. Regex  
   [Regex](https://docs.python.org/3/library/re.html) atau regular expression adalah sebuah cara untuk mencari teks berdasarkan pola tertentu. Umpamanya, ketika ingin mencari sebuah kata dalam kamus, misalnya arti dari kata parsing, kita akan mencari kata tersebut di halaman yang memiliki kata dengan awalan p, lalu pa. Regex bekerja dengan konsep yang sama. Pada regex, kita mencari sebuah kata atau kumpulan kata dengan memberikan pola yang diinginkan.

**Library Matematika**

Library yang dapat digunakan untuk permasalahan matematika adalah library [math](https://docs.python.org/3/library/math.html) yang termasuk salah satu modul bawaan Python. Anda hanya perlu melakukan impor untuk modul math. Anda hanya perlu melakukan impor untuk modul math. Berikut contoh penerapannya.

1. import math
3. print(math.sqrt(25))  # Output: 5.0
4. print(math.pi)  # Output: 3.141592653589793

**Library Parser**

Library parser pada Python menyediakan fasilitas untuk menguraikan kode Python menjadi struktur data yang dapat diproses dan dianalisis. Anda dapat menggunakan [Getopt](https://docs.python.org/3.8/library/getopt.html) atau [ArgParse](https://docs.python.org/3.8/library/argparse.html).

Argument parser bermanfaat jika kita ingin membuat program atau skrip kecil yang langsung menerima parameter pada saat pemanggilan program. Hal ini biasa digunakan dalam pemanggilan aplikasi atau skrip di CLI/terminal \*nix-based, misalnya Linux dan MacOS. Contoh penerapannya berikut.

1. import argparse
3. parser = argparse.ArgumentParser()
4. parser.add\_argument('-o', '--output', action='store\_true', help="tampilkan output")
5. args = parser.parse\_args()
7. if args.output:
8. print("Halo, ini merupakan sebuah output dari panggildicoding.py")

**Library Pengolahan Data**

Library pengolahan data bertujuan untuk membantu dalam manipulasi, analisis, dan pemrosesan data. Library ini menyediakan berbagai fungsi dan metode yang memudahkan pengguna untuk melakukan operasi pengolahan data dengan lebih efisien dan cepat.

Tujuan dari library ini untuk menyederhanakan tugas-tugas kompleks yang berkaitan dengan pengolahan data. Jadi, Anda tidak perlu mengimplementasikan semuanya dari awal. Berikut adalah beberapa library populer yang digunakan untuk pengolahan data.

1. **Pandas**  
   [Pandas](https://pandas.pydata.org/) adalah library populer yang digunakan untuk pengelolaan dan analisis data. Library ini menyediakan struktur data dan alat untuk membantu pengguna dalam melakukan manipulasi, pembersihan, transformasi, dan analisis data dengan mudah dan efisien.
2. **NumPy**  
   Library [NumPy](https://numpy.org/) adalah package fundamental yang sering digunakan untuk scientific computing pada Python. Library ini menyediakan objek array multidimensi, berbagai jenis objek lainnya, seperti masked array dan matrix, dan sebagainya.
3. **Matplotlib**  
   Selanjutnya adalah [matplotlib](https://matplotlib.org/) yang merupakan library untuk melakukan visualisasi data. Matplotlib termasuk jenis library eksternal sehingga Anda perlu melakukan instalasi matplotlib terlebih dahulu.
4. **Seaborn**  
   Terakhir adalah library [seaborn](https://seaborn.pydata.org/) yang termasuk jenis library dengan tujuan untuk visualisasi data sama seperti matplotlib. Bahkan library seaborn dibangun berdasarkan pada library matplotlib.

**Library File Management**

Library file management adalah kumpulan library yang dirancang untuk membantu pengguna dalam mengelola serta berinteraksi dengan berkas dan direktori pada sistem file. Beberapa library file management adalah berikut.

1. OS  
   Modul [OS](https://docs.python.org/3/library/os.html) pada Python berguna untuk fungsi-fungsi yang berkaitan dengan sistem operasi, misalnya open(), path(), getcwd(), dan fungsi lainnya. Modul ini memungkinkan Anda untuk memanfaatkan fungsi yang sama dan mengeksekusi fungsi terkait OS yang mungkin berbeda di setiap sistem operasi. Ada beberapa fitur yang hanya bekerja pada sistem operasi tertentu.
2. JSON  
   Untuk serialization dengan bahasa lain, umumnya kita menggunakan [JSON](https://docs.python.org/3/library/json.html)(JavaScript Object Notation) yang memiliki beberapa perbedaan karakteristik dengan pickle, yakni berikut.
   * JSON adalah format text-serialization dan umumnya menggunakan Unicode atau UTF-8. Sementara pickle bersifat binary serialization.
   * JSON dapat dibaca dengan mudah oleh manusia, sementara pickle tidak.
   * JSON dapat dioperasikan dan digunakan di luar ekosistem Python. Pickle adalah Python-specific.
   * JSON secara default hanya dapat merepresentasikan subset dari built-in type pada Python.
   * Pickle dapat merepresentasikan hampir (jika tidak seluruh) tipe Python dan secara default melakukan kompresi data.  
       
     Sebagaimana yang telah disebutkan sebelumnya, JSON adalah format text yang ditujukan untuk serialization. Agar data dapat dengan mudah ditransmisikan antar berbagai sumber tanpa khawatir bentuknya kacau, menggunakan JSON adalah salah satu pilihan yang tepat.  
       
     JSON memiliki format yang hampir mirip dengan dictionary tempat data disimpan dengan format key & value pair.
3. **Pickle**  
   Jika Anda memiliki sebuah list yang ingin disimpan atau ditransmisikan tanpa khawatir bentuknya akan rusak atau kacau, fungsi dari library pickle dapat dimanfaatkan. Pickle termasuk fungsi Object Serialization pada Python. Pickling adalah istilah untuk mengubah objek menjadi byte stream, sedangkan unpickling adalah perlakuan sebaliknya.

**Library Web Scraping**

Library web scraping adalah jenis library untuk membantu pengguna mengumpulkan data dari halaman web. Proses ini disebut sebagai “web scraping” atau “web crawling”. Anda bisa menggunakan fungsi dan metode pada library ini untuk mengekstraksi informasi dari situs web dan menyimpannya dalam format yang dapat diakses dan digunakan dalam analisis atau aplikasi lainnya.

1. **Beautifulsoup**  
   Beautifulsoupadalah library untuk mengambil data dari halaman web dan mengekstrak informasi yang diperlukan.
2. **Urllib**  
   Urllibadalah library bawaan dari Python yang bertujuan untuk scraping konten dari sebuah website. Penggunaan urllib berbeda dengan beautifulsoup. Bisa dikatakan bahwa cara penggunaan urllib sedikit kompleks dibandingkan beautifulsoup.

**Library Machine Learning**

Selanjutnya adalah library yang digunakan untuk melakukan pemelajaran mesin. Anda dapat menggunakan library berikut untuk membantu Anda menyelesaikan permasalahan machine learning. Berikut adalah beberapa library populer untuk machine learning.

1. **Scikit-learn**  
   Pertama adalah scikit-learn yang menyediakan berbagai algoritma pemelajaran mesin siap pakai untuk membantu dalam pengembangan model pemelajaran mesin, pemrosesan data, dan evaluasi kinerja model.
2. **TensorFlow**  
   Selanjutnya adalah TensorFlow yang termasuk salah satu library paling populer terkait machine learning. Dengan menggunakan TensorFlow, Anda bisa mengembangkan machine learning hingga tahap deployment.
3. **PyTorch**  
   Terakhir ada PyTorch, yakni library machine learning yang dikembangkan oleh Facebook’s AI Research lab (FAIR). PyTorch menyediakan alat dan kerangka kerja yang kuat untuk mengembangkan model pemelajaran mesin, terutama dalam konteks jaringan saraf tiruan (neural networks).

**Library Web Development**

Terakhir, ada library yang bertujuan untuk pengembangan aplikasi web. Sebagaimana yang sudah dijelaskan dalam materi-materi sebelumnya, Python dapat digunakan untuk pengembangan aplikasi web pada sisi server. Berikut adalah library yang dapat digunakan untuk membantu Anda mengembangkan web.

1. **Django**  
   Django adalah high-level Python web framework yang mendukung pengembangan secara cepat, bersih, serta pragmatis.
2. **Flask**  
   Flask adalah web framework untuk Python yang ditujukan untuk membangun aplikasi web. Flask dirancang dengan tujuan menjadi ringan, fleksibel, dan sederhana.
3. **Fast API**  
   FastAPI adalah web framework untuk Python yang tujuannya merancang dan membangun API dengan cepat, efisien, dan aman. FastAPI memberikan kinerja yang tinggi, sintaks yang intuitif, serta dukungan otomatisasi dokumentasi yang kuat. Jadi, ia cocok untuk pengembangan mikroservis, layanan web responsif, dan sebagainya.