## Pengantar Fungsi dan Prosedur

[](https://www.dicoding.com/academies/620/tutorials/35043)

Dalam materi sebelumnya, Anda sudah memahami prinsip dasar mengatur alur program. Materi tersebut membawa kita untuk mencerna dan menganalisis cara kerja suatu pengondisian dan mengulang suatu blok program supaya program dapat berjalan sesuai alur yang diharapkan.

Pada materi kali ini, kita akan membahas tentang fungsi dan prosedur secara mendalam dengan pendekatan pemrograman prosedural yang komprehensif. Untuk itu, Anda perlu belajar beberapa materi mengenai hal berikut:

* penerapan fungsi dalam menjalankan blok program dan mengembalikan nilai;
* penerapan prosedur dalam mengelola blok program; serta
* penerapan tingkat lanjut dalam menjalankan fungsi dan prosedur.

Sebagaimana biasanya, kita akan senam otak terlebih dahulu sebelum memulai pembahasan tentang materi ini. Untuk pengingat, senam otak ini memberikan kita perspektif baru dalam memahami materi. Selain itu, kita dapat melatih otak kita untuk menyelesaikan permasalahan dalam kehidupan sehari-hari. Sudah mulai penasaran? Yuk, simak cerita Bapak Fajar dalam menjalani kegiatannya di area pembangunan.

Bapak Fajar sedang melakukan inspeksi di area pembangunan beserta rekan kerjanya. Hal ini perlu dilakukan agar pekerja tetap menjalankan tugasnya sesuai dengan rencana. Masih ingatkah daftar inspeksi yang dilakukan oleh Bapak? Ya, berikut adalah daftar inspeksi selama proses pembangunan berlangsung.

* Pemahaman terhadap rencana dan desain.
* Ketersediaan bahan dan peralatan.
* Pengawasan pekerjaan.
* Kualitas konstruksi.
* Keselamatan kerja.
* Koordinasi dengan pemilik proyek.

[](https://www.dicoding.com/academies/620/tutorials/35043)

Setiap jenis inspeksi memiliki spesifikasi dan tujuan yang berbeda-beda. Dengan kata lain, instruksi yang dijalankan pada setiap inspeksi pun berbeda. Serangkaian instruksi tersebut sangat spesifik dikerjakan pada satu inspeksi tertentu dan tidak bisa disamaratakan dengan yang lainnya.

Misalnya, Bapak Fajar sedang melakukan inspeksi kualitas konstruksi. Langkah atau prosedur yang dilakukan akan berfokus pada pemeriksaan kekuatan dan kestabilan struktur bangunan. Berbeda cerita ketika Bapak Fajar sedang melakukan inspeksi keselamatan kerja. Beliau akan memeriksa peralatan dan perlengkapan yang digunakan oleh pekerja untuk memastikan keamanan mereka saat menjalankan tugas.

[](https://www.dicoding.com/academies/620/tutorials/35043)

Tahapan yang dilakukan oleh Bapak Fajar saat inspeksi berlangsung merupakan penerapan dari prosedur. Mengapa demikian? Setiap prosedur memiliki tugas dan tanggung jawab yang spesifik. Ia tersusun oleh serangkaian instruksi yang memberikan efek neto positif untuk mencapai hasil sesuai dengan keinginan.

Bapak Fajar tidak dapat menjadikannya satu kesatuan aksi inspeksi karena tugas dan tujuannya berbeda. Aksi yang dilalui Bapak dalam menjalankan inspeksi pun sangat banyak. Jadi, Bapak Fajar mengelompokkan aksi inspeksi tersebut menjadi beberapa label, seperti “Kualitas Konstruksi”, “Keselamatan Kerja”, dll. Dalam satu label inspeksi memiliki instruksi yang lebih kecil dan terorganisasi. Serangkaian aksi tersebut bertugas menyelesaikan satu permasalahan spesifik.

Itulah alasan inspeksi yang dilakukan oleh Bapak Fajar merupakan penerapan dari prosedur. Inspeksi tersusun atas serangkaian aksi dengan tugas dan tujuan yang spesifik. Ia dapat dijalankan kembali apabila Bapak membutuhkan inspeksi dadakan ataupun terjadwal.

Nah, apakah Anda sudah terbayang tentang materi yang akan dibahas? Penjelasan di atas masih menggambarkan secara umum tentang pengertian dari prosedur. Materi berikutnya akan membahas lebih lengkap terkait fungsi dan prosedur beserta penerapannya dalam bahasa pemrograman Python. Tunggu apa lagi, yuk, segera ke materi berikutnya!

## Fungsi

Saat bertugas di area pembangunan, Bapak Fajar diminta untuk melakukan survei lapangan oleh kepala proyek. Hal ini bertujuan untuk memastikan area pembangunan dapat dibangun sesuai rencana. Kepala proyek berkata kepada Bapak Fajar, “Lakukan survei di area tersebut dan pastikan kita bisa melakukan renovasi rumah atau tidak.”

Setelah proses survei telah dilakukan, Bapak Fajar mengabari ke kepala proyek bahwa kita bisa menjalankan renovasi rumah berdasarkan hasil survei. Alhasil, kepala proyek memutuskan untuk realisasi renovasi dan mempersiapkan kebutuhan pekerja.

[](https://www.dicoding.com/academies/620/tutorials/35048?from=35043)

### Definisi Fungsi

Dari proses perintah yang diajukan oleh kepala proyek kepada Bapak Fajar merupakan salah satu konsep dasar fungsi. Apa hubungannya perintah kepala proyek dengan fungsi? Ternyata, ada keterkaitan antar keduanya. Saat kepala proyek meminta melakukan survei dan Bapak kembali membawa hasil, ada proses yang dilakukan Bapak bernama “survei” dan mengembalikan nilai, yaitu “bisa dilakukan renovasi”.

**Fungsi** dapat diartikan sebagai suatu proses instruksi yang dibungkus dengan nama/label dan mengembalikan nilai kembali. Di dalamnya ada serangkaian aksi yang memiliki tujuan spesifik. Pada kurun waktu tertentu, fungsi dapat dipanggil dan dijalankan berkali-kali tergantung dengan kebutuhan. Itulah alasan perintah kepala proyek adalah penerapan dari fungsi.

Berdasarkan penjelasan di atas, fungsi memiliki beberapa sifat. Berikut adalah sifat dari sebuah fungsi.

* **Modular**  
  Fungsi bersifat modular karena serangkaian aksi terpisah dengan tugas utama. Hal ini memungkinkan kita untuk mengelompokkan sebagian instruksi dengan tujuan yang sama dan spesifik.
* **Reusable**  
  Fungsi bersifat reusable karena ia dapat digunakan kembali dalam berbagai waktu tertentu. Hal ini mengurangi duplikasi kode sehingga meningkatkan efisiensi dalam proses pengembangan program.

Ciri khas dari sebuah fungsi ada tiga komponen, yaitu fungsi harus memiliki **tujuan khusus**, menjalankan **serangkaian instruksi** tertentu, dan **menghasilkan** **output**. Selagi ketiga komponen tersebut dimiliki pada suatu instruksi, ia dapat dikatakan sebagai fungsi.

[](https://www.dicoding.com/academies/620/tutorials/35048?from=35043)

* **Tujuan Khusus**  
  Fungsi harus memiliki tugas yang spesifik dan terbatas supaya dapat dijelaskan dalam satu atau beberapa kalimat. Hal ini bertujuan agar orang lain dapat memahami dengan mudah. Jadi, instruksi yang berjalan harus fokus dan tidak terlalu melebar dari tujuan tersebut.
* **Serangkaian Instruksi**  
  Kita perlu menjalankan serangkaian instruksi untuk menunjang tujuan tersebut. Instruksi tersebut menjadi esensial agar kita mampu memanipulasi dan mengolah data saat fungsi itu dipanggil. Berdasarkan tujuan khusus, Anda tidak boleh menjalankan instruksi yang panjang sehingga fungsi dapat terbaca dengan jelas.
* **Menghasilkan Output**  
  Saat Anda memanipulasi dan mengolah data, hasilnya perlu diteruskan ke pihak tertentu. Dengan begitu, ia dapat dieksekusi dan diproses ke instruksi berikutnya.

### Parameter dalam Fungsi

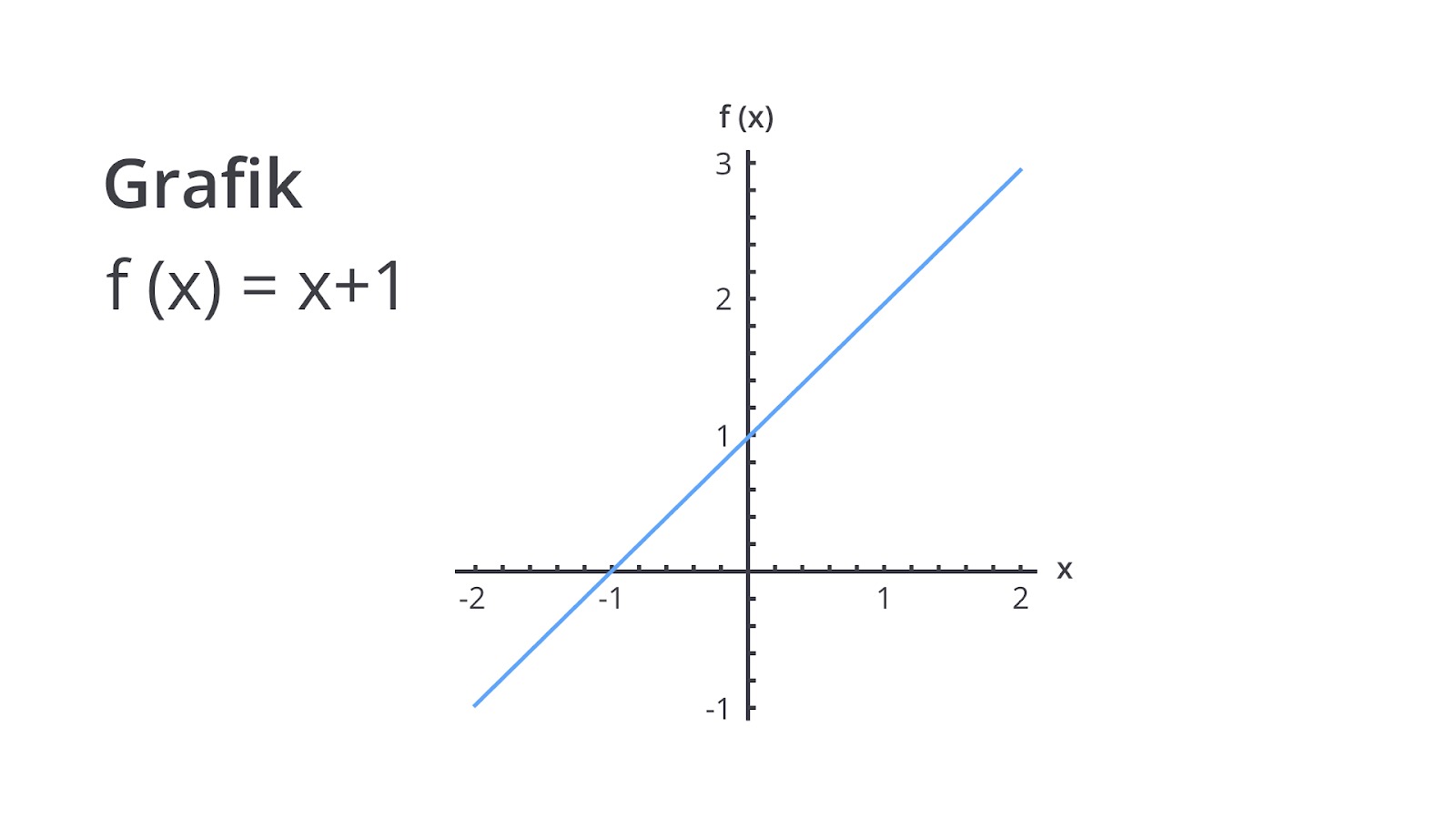
Adakalanya, fungsi dapat terlibat dengan lingkungan sekitarnya. Bayangkan bila Bapak Fajar melakukan survei tanpa alat atau Ibu Fika sedang menerima pesanan katering dalam jumlah besar tanpa adanya bantuan dari orang lain. Pekerjaan mereka tidak dapat terlaksana dengan baik. Alhasil, fungsi “survei” dan “mempersiapkan katering” tidak akan berjalan dengan semestinya.

Interferensi dapat terjadi dari dalam dan luar fungsi. Apabila interferensi dari dalam, aspek atau komponen memang dibutuhkan dari dalam fungsi dan sudah tersedia di dalamnya. Sebaliknya, ada sebuah fungsi yang membutuhkan komponen penting, tetapi belum tersedia di dalamnya. Fungsi hanya menyediakan syarat atau spesifikasi komponen yang sesuai kriteria untuk menjalankannya.

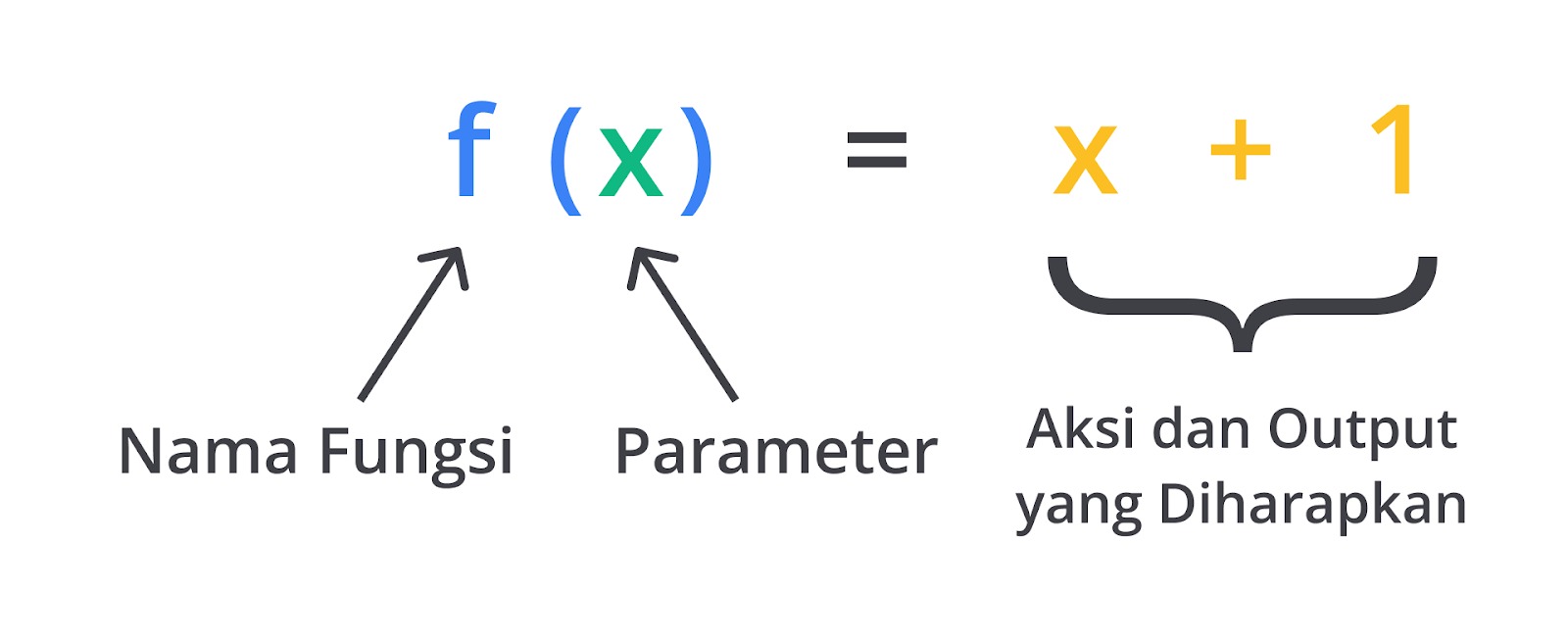
Itulah alasan fungsi memiliki komponen lain, yaitu parameter. Parameter ini adalah komponen opsional untuk membantu jalannya suatu fungsi dengan tujuan menghasilkan output yang diharapkan. Parameter harus memiliki spesifikasi yang relevan sesuai permintaan fungsi. Jika tidak, fungsi tidak dapat berjalan dengan baik.

|  |
| --- |
| Ketika Bapak Fajar sedang berada di area renovasi rumah, ternyata ada material yang belum tersedia untuk menunjang jalannya proyek. Beliau berinisiatif menuliskan spesifikasi material yang relevan berdasarkan kebutuhan renovasi. Dengan begitu, Bapak mendapatkan material sesuai dengan keperluan dan renovasi rumah dapat berjalan seperti yang direncanakan.  [Bapak Fajar sedang menulis kebutuhan material yang dibutuhkan saat renovasi rumah.](https://www.dicoding.com/academies/620/tutorials/35048?from=35043)  Rencana “pembelian material” yang dilakukan oleh Bapak Fajar merupakan konsep fungsi juga. Ia memiliki tujuan jelas, ada serangkaian aksi untuk menyelesaikan renovasi rumah, dan menghasilkan rumah yang selesai sesuai dengan rencana.  Namun, ada komponen tambahan yang mendukung fungsi tersebut, yaitu material. Ia belum tersedia dalam perencanaan renovasi rumah. Apabila material tersebut tiada atau tak sesuai dengan spesifikasi, renovasi rumah akan terhambat dan tidak berjalan sesuai rencana. Material ini dianggap sebagai parameter untuk menyokong jalannya renovasi rumah. |

Saat Bima belajar mata pelajaran matematika, dia sadar bahwa ada materi yang menyerupai konsep fungsi ini. Materi tersebut adalah **fungsi**. Dia belajar banyak hal, mulai dari fungsi linear, fungsi aljabar, fungsi invers, dll. Di balik beragam fungsi tersebut, ada hal yang serupa dengan konsep fungsi secara umum.

[](https://www.dicoding.com/academies/620/tutorials/35048?from=35043)

Tenang, kita tidak belajar matematika secara mendalam. Akan tetapi, perhatikan fungsi dari persamaan gambar di atas. Ada fungsi bernama **f**memiliki parameter **x**. Fungsi tersebut menghasilkan nilai keluaran **x + 1**. Berikut adalah penjabaran dari fungsi tersebut.

[](https://www.dicoding.com/academies/620/tutorials/35048?from=35043)

Parameter xberfungsi untuk menyelesaikan persamaan **f(x)**. Apabila kita memasukkan variabel selain x, apakah persamaan dapat terselesaikan? Jelas tidak. Kita perlu memberikan nilai sesuai kriteria yang dibutuhkan dalam persamaan tersebut.

### Penggunaan Fungsi di Python

Mendeklarasikan fungsi dalam bahasa pemrograman Python serupa dengan menuliskan fungsi matematika. Hanya saja, ada kata kunci **def** sebelum nama fungsi, aksi dibungkus layaknya blok program pada umumnya, dan output fungsi diberi kata kunci **return**. Berikut adalah contoh deklarasi dan pemanggilan fungsi pada bahasa pemrograman Python.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35048?from=35043#run1-editor1)

1

# deklarasi fungsi

2

def f(x):

3

return x + 1

4

​

5

# pemanggilan fungsi

6

y = f(4)

7

print(y)

8

​

9

# Output: 5

 Input  Reset

 Jalankan

Kita bedah satu per satu kode di atas. Kata kunci def adalah penanda bahwa kode tersebut merupakan sebuah fungsi. Lalu, bagaimana fungsi **f(x)** dipanggil? Anda hanya memanggil nama fungsinya diikuti dengan argumen untuk menggantikan nilai x. Sebab fungsi mengembalikan suatu nilai, Anda dapat melakukan proses assignment ke variabel tertentu, dalam contoh ini adalah variabel y.

### Argumen dan Parameter

Mungkin, Anda sering tertukar dengan istilah parameter dan argumen. Kedua istilah tersebut memiliki konsep yang berbeda, tetapi saling terikat satu dengan lainnya. Berikut adalah perbedaan antara argumen dan parameter.

* **Argumen**  
  Argumen adalah nilai atau ekspresi yang dikirim ke fungsi saat fungsi dipanggil. Argumen ini diberikan ke parameter dan diteruskan ke dalam fungsi untuk menjalankan aksi tertentu.
* **Parameter**  
  Parameter ini dapat diibaratkan sebagai spesifikasi atau syarat untuk menjalankan fungsi. Ia sebagai wadah untuk menyimpan nilai dari luar fungsi. Ketika fungsi dipanggil, parameter harus diisi sebuah nilai supaya fungsi dapat bekerja dengan benar.

Berdasarkan kedua penjelasan tersebut, kita dapat menyimpulkan bahwa ketika fungsi memiliki parameter, ia harus diberi suatu argumen agar dapat bekerja dengan benar.

Argumen dapat diisi saat pemanggilan fungsi berlangsung. Namun, adakalanya, kita tidak perlu memasukkan argumen saat fungsi dipanggil. Mengapa bisa demikian? Iya, kita dapat memanfaatkan parameter opsional sehingga argumen tidak selalu didefinisikan setiap fungsi dipanggil. Berikut adalah contoh penerapan fungsi yang menggunakan parameter opsional.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35048?from=35043#run2-editor1)

1

def bayar\_makanan(jumlah, harga\_per\_porsi=10000):

2

total\_bayar = jumlah \* harga\_per\_porsi

3

return total\_bayar

4

​

5

total = bayar\_makanan(2)

6

print(total)

7

​

8

# Output: 20000

 Input  Reset

 Jalankan

Saat Bima membeli mi ayam seharga 10000, pemanggilan fungsi **bayar\_makanan** tidak perlu menyisipkan argumen pada parameter harga\_per\_porsi. Secara default, parameter tersebut bernilai 10000. Akan tetapi, apabila Bima membeli nasi goreng seharga 13000, Anda dapat mengubah nilainya dengan memberikan argumen setelah parameter **jumlah**. Jadi, kode akan menjadi seperti berikut.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35048?from=35043#run3-editor1)

1

def bayar\_makanan(jumlah, harga\_per\_porsi=10000):

2

total\_bayar = jumlah \* harga\_per\_porsi

3

return total\_bayar

4

​

5

total = bayar\_makanan(2, 13000)

6

print(total)

7

​

8

# Output: 26000

 Input  Reset

 Jalankan

Selain itu, Anda juga dapat memberikan lebih dari satu parameter pada suatu fungsi. Lalu, bagaimana cara memanggil fungsinya? Coba perhatikan kode berikut.

1. def menghitung\_luas\_trapesium(sisi\_atas, sisi\_bawah, tinggi):
2. luas\_trapesium = 0.5 \* (sisi\_atas + sisi\_atas) \* tinggi
3. return luas\_trapesium

Fungsi di atas dapat dipanggil dengan dua cara.

* **Positional Argument**  
  Kita perlu meletakkan argumen berdasarkan posisi atau urutan parameter fungsi. Apabila tidak sesuai, fungsi akan menghasilkan nilai yang berbeda. Jadi, pemanggilan fungsi menghitung\_luas\_trapesiumsebagai berikut.
  1. def menghitung\_luas\_trapesium(sisi\_atas, sisi\_bawah, tinggi):
  2. luas\_trapesium = 0.5 \* (sisi\_atas + sisi\_atas) \* tinggi
  3. return luas\_trapesium
  5. y = **menghitung\_luas\_trapesium(3, 2, 4)**
  6. print(y)
  8. # Output: 12.0
* **Keyword Argument**  
  Kita tidak perlu meletakkan argumen sesuai urutan. Kita panggil nama parameter dan diikuti dengan nilai argumennya. Jadi, pemanggilan fungsi menghitung\_luas\_trapesiumsebagai berikut.
  1. def menghitung\_luas\_trapesium(sisi\_atas, sisi\_bawah, tinggi):
  2. luas\_trapesium = 0.5 \* (sisi\_atas + sisi\_atas) \* tinggi
  3. return luas\_trapesium
  5. y = **menghitung\_luas\_trapesium(**
  6. **tinggi = 4,**
  7. **sisi\_atas = 3,**
  8. **sisi\_bawah = 2,**
  9. **)**
  10. print(y)
  12. # Output: 12.0

Namun, perlu diingat bahwa semakin banyak parameter pada suatu fungsi, Anda semakin sulit untuk mengaturnya. Ada beberapa alasan kita perlu mempertimbangkan pembuatan fungsi dengan parameter yang banyak.

* **Pemanggilan fungsi yang rumit.**  
  Saat fungsi dipanggil, kita perlu menyediakan argumen untuk setiap parameter yang diminta olehnya. Hal ini dapat membuat pemanggilan fungsi semakin kompleks dan memerlukan perhatian lebih dalam menyusun argumen.
* **Sulit memahami fungsinya.**  
  Saat fungsi memiliki parameter yang panjang, kita akan semakin sulit untuk memahami konteks. Walaupun sudah memberikan nama parameter yang deskriptif, kita akan susah untuk menguraikan argumen untuk tiap parameter.
* **Salah mengirim argumen.**  
  Panjangnya penulisan kode saat memanggil fungsi membuat kita harus sangat berhati-hati dalam menyusun argumen. Kesalahan peletakan argumen membuat output fungsi tidak sesuai dengan yang diharapkan.
* **Fleksibilitas yang rendah.**  
  Sebuah fungsi yang memiliki banyak parameter menunjukkan bahwa ada banyak proses yang terjadi di dalamnya. Biasanya, tujuan adanya fungsi yang terlalu besar. Alhasil, fungsi menjalankan aksi yang banyak dan kompleks.

Lalu, bagaimana cara kita meminimalkan permasalahan pada banyaknya parameter dalam fungsi? Python memiliki lima cara untuk mendeklarasikan parameter sesuai dengan kebutuhan. Anda dapat memanfaatkan satu dari kelima cara tersebut untuk memanggil fungsi yang memiliki banyak parameter. Berikut adalah kelima cara menyusun parameter dalam fungsi.

1. **Positional-or-Keyword**  
   Cara ini dapat Anda gunakan ketika ingin membuat fungsi yang bisa diisi oleh argumen sesuai urutan atau berdasarkan kata kunci parameter. Berikut contohnya.
   1. def **greeting(nama, pesan)**:
   2. return f"Halo, nama saya {nama}. {pesan}"
   4. print(greeting("Bima", "Selamat pagi!"))
   5. print(greeting(pesan="Selamat datang!", nama="Bapak Fajar"))
   7. """
   8. Output:
   9. Halo, nama saya Bima. Selamat pagi!
   10. Halo, nama saya Bapak Fajar. Selamat datang!
   11. """
2. **Positional-Only**  
   Cara ini dapat Anda pakai ketika ingin memanggil fungsi dengan argumen berdasarkan urutan atau susunan parameter. Anda harus menambahkan karakter **slash**“/” di akhir parameter. Dengan cara ini, Anda tidak bisa memanggil fungsi menggunakan kata kunci dari parameter. Hal tersebut akan menyebabkan kode error. Berikut contoh kodenya.
   1. def **greeting(nama, pesan, /)**:
   2. return f"Halo, nama saya {nama}. {pesan}"
   4. print(greeting("Bima", "Selamat pagi!"))
   6. """
   7. Output:
   8. Halo, nama saya Bima. Selamat pagi!
   9. """
3. **Keyword-Only**  
   Anda dapat menggunakan cara ini saat ingin memanggil fungsi dengan kata kunci dari parameter. Anda perlu menambahkan karakter **asterisk** “\*” di awal parameter. Berbanding terbalik dengan **positional-only**, Anda tidak dapat menggunakan konsep memanggil fungsi dengan argumen berdasarkan urutannya. Berikut contoh kodenya.
   1. def **greeting(\*, nama, pesan)**:
   2. return f"Halo, nama saya {nama}. {pesan}"
   4. print(greeting(pesan="Selamat datang!", nama="Bapak Fajar"))
   6. """
   7. Output:
   8. Halo, nama saya Bapak Fajar. Selamat datang!
   9. """
4. **Var-positional**  
   Konsep ini menampung beberapa argumen menjadi satu nilai dan dapat dicacah layaknya sebuah data **list**. Ia disimpan dalam satu variabel bernama **args** yang melambangkan arguments. Fungsi dengan parameter berjenis ini memerlukan satu parameter diikuti dengan karakter asterisk“\*” di depan parameter. Berikut penerapannya.
   1. def **greeting(\*args)**:
   2. return f"Halo, nama saya {args[0]}. {args[1]}"
   4. print(greeting("Bapak Fajar", "Selamat datang!"))
   6. """
   7. Output:
   8. Halo, nama saya Bapak Fajar. Selamat datang!
   9. """
5. **Var-keyword**  
   Layaknya var-positional, cara ini menyimpan beberapa argumen menjadi satu nilai, tetapi dapat dicacah seperti data **dictionary**. Ia disimpan dalam satu variabel bernama **kwargs** yang melambangkan **keyword arguments**. Saat menerapkan konsep ini, Anda perlu menambahkan dua karakter asterisk “\*” di depan parameter. Berikut contoh kodenya.
   1. def **greeting(\*\*kwargs)**:
   2. return f"Halo, nama saya {kwargs['nama']}. {kwargs['pesan']}"
   4. print(greeting(pesan="Selamat pagi!", nama="Bima"))
   6. """
   7. Output:
   8. Halo, nama saya Bima. Selamat pagi!
   9. """

Wah, materinya sangat seru, bukan? Anda sudah belajar konsep fungsi secara dasar dan memahami perbedaan beserta jenis-jenis argumen dan parameter di dalamnya. Apabila ingin mengeksplorasi materi tentang fungsi, Anda dapat mengunjungi beberapa tautan berikut.

* [Defining Functions](https://docs.python.org/3/tutorial/controlflow.html#defining-functions).
* [More on Defining Function](https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions).
* [Glossary of Parameter](https://docs.python.org/3/glossary.html#term-parameter).
* [Glossary of Argument](https://docs.python.org/3/glossary.html#term-argument).

Materi berikutnya tidak kalah seru, yaitu memahami prosedur. Apa sih bedanya fungsi dan prosedur? Apakah keduanya memiliki fungsi yang berbeda? Eits, sabar, ya. Kita akan memahami lebih lengkapnya pada materi Prosedur. Yuk, pindah ke materi berikutnya.

## Prosedur

Pada materi sebelumnya, kita telah belajar konsep dasar hingga mendalam mengenai fungsi. Mulai dari pengertian, cara deklarasi fungsi, mengenal perbedaan argumen dan parameter, serta jenisnya sudah dijelaskan secara rinci. Namun, ada hal yang perlu Anda ketahui selain fungsi. Apakah itu? Ya, Anda perlu memahami konsep prosedur.

### Definisi Prosedur

Lalu, apa itu prosedur? Berdasarkan KBBI, [prosedur](https://kbbi.kemdikbud.go.id/entri/prosedur) adalah tahap kegiatan dalam menyelesaikan suatu aktivitas. Ada serangkaian aksi yang berjalan dan bertujuan untuk memecahkan suatu permasalahan. Setiap langkah harus dilakukan dengan urutan yang benar supaya tujuannya tercapai.

Pada pemrograman, prosedur dapat diartikan sebagai blok program yang menjalankan serangkaian instruksi untuk menjalankan tugas tertentu. Serangkaian aksi tersebut dikelompokkan menjadi satu blok bagian yang ditugaskan untuk menyelesaikan suatu permasalahan, baik persoalan kecil maupun besar. Dengan adanya prosedur, kita dapat mengorganisasi program dengan lebih terstruktur dan mudah dibaca oleh orang lain.

Berlandaskan pengertian di atas, apakah ada hubungannya dengan fungsi? Tentu saja ada. Secara konsep, prosedur dan fungsi memiliki serangkaian aksi khusus untuk mencapai tujuan tertentu. Namun, perbedaan dari keduanya adalah fungsi memiliki nilai kembali atau menghasilkan output, sedangkan prosedur tidak.

|  |
| --- |
| Dalam proses inspeksi “Keselamatan Kerja”, Bapak Fajar menyusun rangkaian aktivitas untuk memastikan seluruh proses konstruksi berlangsung dengan aman dan sesuai standar keselamatan. Inspeksi ini mampu mengidentifikasi dan mengatasi potensi resiko kecelakaan pada pekerja atau orang lain di area pembangunan.  Inspeksi Keselamatan Kerja ini meliputi berbagai aktivitas, yaitu memeriksa hal berikut:   * penggunaan alat pelindung diri; * kondisi peralatan dan perlengkapan; * keamanan jalur evakuasi; * tanda peringatan dan papan informasi; dll.   Semua aksi tersebut dilakukan secara rutin dan teratur agar semua aspek keselamatan terpenuhi.  [Bapak Fajar sedang melakukan inspeksi keselamatan kerja di area pembangunan rumah.](https://www.dicoding.com/academies/620/tutorials/35053?from=35048)  Aksi inspeksi yang dilakukan Bapak Fajar merupakan salah satu contoh prosedur dalam kegiatan sehari-hari. Inspeksi dibungkus dengan nama “Keselamatan Kerja” yang memiliki tujuan khusus, yaitu memastikan proses renovasi berjalan dengan aman. Inspeksi tersebut tidak menghasilkan nilai apa pun atau tidak ada proses **assignment**. Hanya kegiatan rutin yang dilakukan secara berulang.  Konsep tersebut menggambarkan prosedur dalam pemrograman. Ia hanya berisi serangkaian aksi yang mampu menyelesaikan permasalahan tanpa adanya proses pemberian hasil akhir kepada objek tertentu. Ia tidak menghasilkan output sehingga aksi berjalan secara mandiri. |

Layaknya fungsi, prosedur juga dapat memiliki parameter yang membantu jalannya serangkaian aksi. Parameter ini sifatnya opsional dan tidak harus dimiliki oleh prosedur. Apabila prosedur memiliki syarat untuk menjalankan aksi tanpa ada interferensi dari luar, ia tidak memerlukan parameter tambahan. Sebaliknya, prosedur yang memerlukan syarat dan belum tersedia di dalamnya, ia membutuhkan parameter untuk menopang syarat tersebut.

### Penggunaan Prosedur di Python

Prosedur dan fungsi memiliki struktur penulisan kode yang sama. Ia memiliki kata kunci **def** untuk melambangkan kode tersebut merupaka prosedur atau fungsi. Namun, pembeda di antara keduanya adalah tidak adanya hasil yang dikembalikan. Artinya, kata kunci **return** dapat Anda hapus dari kode pada baris akhir prosedur. Berikut adalah contoh deklarasi dan pemanggilan prosedur pada bahasa pemrograman Python.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35053?from=35048#run1-editor1)

1

# deklarasi prosedur

2

def eatSomething(food):

3

print(f"Saya sedang makan {food}.")

4

​

5

# pemanggilan prosedur

6

eatSomething("Mi Ayam")

7

​

8

# Output: Saya sedang makan Mi Ayam.

 Input  Reset

 Jalankan

Prosedur **eatSomething** memiliki parameter food. Prosedur dapat memanfaatkannya untuk menjalankan aksi mencetak teks yang dinamis. Argumen fungsi dapat berubah bergantung pada kebutuhan pengguna.

### Prosedur Tidak Mengembalikan Output

Mungkin, sebagian orang bertanya, apakah prosedur tersebut mengembalikan nilai tertentu? Bagaimana kita mengetahui bahwa prosedur tersebut mengembalikan nilai? Kita dapat memanfaatkan proses **assignment**, perintah **type**, dan **print** untuk mengetahui nilai kembali dari suatu prosedur. Berikut adalah proses untuk mengetahui nilai kembali dari prosedur.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35053?from=35048#run2-editor1)

1

# deklarasi prosedur

2

def eatSomething(food):

3

print(f"Saya sedang makan {food}")

4

​

5

# pengecekan nilai kembali dari prosedur

6

valueOfProcedure = eatSomething("Mi Ayam")

7

print(f"Fungsi eatSomething mengembalikan nilai {valueOfProcedure}.")

8

print(f"Ia merupakan objek bertipe {type(valueOfProcedure)}.")

9

​

10

"""

11

Output:

12

Saya sedang makan Mi Ayam

13

Fungsi eatSomething mengembalikan nilai None.

14

Ia merupakan objek bertipe <class 'NoneType'>.

15

"""

 Input  Reset

 Jalankan

Perhatikan potongan kode di atas. Ternyata, program di atas tetap mengembalikan nilai konstanta, yaitu [**None**](https://docs.python.org/3/library/constants.html#None). Ia merupakan suatu objek bertipe NoneType. Nilai None ini merepresentasikan suatu objek yang tidak bernilai apa pun. Apabila disandingkan dengan bahasa pemrograman lain, objek None setara dengan **null** yang menyatakan tidak ada nilainya.

Melihat pernyataan tersebut, kita dapat memanipulasi prosedur menjadi fungsi dengan memanfaatkan kata kunci return di akhir barisnya. Jadi, prosedur eatSomething akan menjadi seperti berikut.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35053?from=35048#run3-editor1)

1

# deklarasi prosedur

2

def eatSomething(food):

3

print(f"Saya sedang makan {food}.")

4

return

5

​

6

# pemanggilan prosedur

7

eatSomething("Mi Ayam")

8

​

9

# Output: Saya sedang makan Mi Ayam.

 Input  Reset

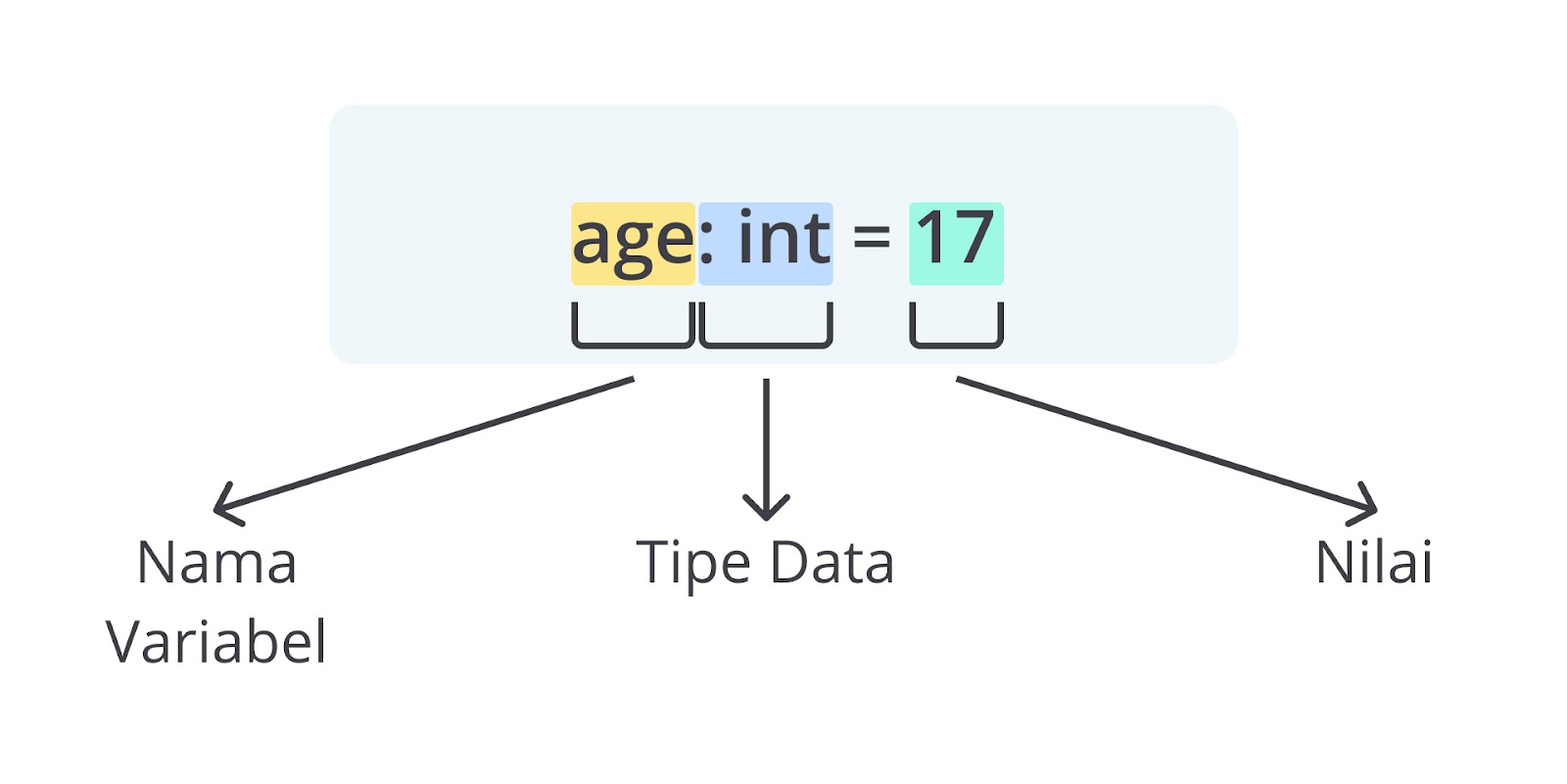
 Jalankan

Baris kode di atas tetap mengembalikan nilai None dengan catatan tidak ada nilai atau variabel yang mengikuti kata kunci return. Dengan begitu, kode di atas tetap dianggap sebagai prosedur karena tidak mengembalikan nilai apa pun.

Konsepnya, prosedur dan fungsi memiliki struktur kode yang serupa. Namun, hal yang perlu Anda perhatikan adalah prosedur tidak mengembalikan nilai apa pun. Penggunaan return atau tidak menjadi hal yang opsional dalam prosedur. Dengan demikian, prosedur dapat menjalankan serangkaian aksi dan mengembalikan nilai None.

### Type Annotation pada Variabel

Kita ulas sedikit cara kerja anotasi pada variabel. Anotasi pada tipe data dapat didefinisikan dalam proses assignment seperti berikut.

[](https://www.dicoding.com/academies/620/tutorials/35058?from=35053)

Proses pembuatan anotasi pada proses assignment tersusun atas tiga komponen, yaitu nama variabel, tipe data, dan nilai. Anda dapat menyisipkan tipe data setelah nama variabel diikuti dengan tanda colon “:”, kemudian lakukanlah proses assignment.

### Pentingnya Anotasi pada Fungsi atau Prosedur

Lalu, bagaimana dengan pembuatan fungsi atau prosedur? Adakah anotasi di dalamnya? Tentu ada. Fungsi atau prosedur memiliki satu komponen utama dan opsional, yaitu menghasilkan output dan memiliki parameter. Kedua komponen tersebut beririsan dengan tipe data pada suatu variabel.

Maksudnya bagaimana? Saat fungsi menghasilkan suatu output, ia akan mengembalikan nilai dengan tipe data tertentu. Saat fungsi memiliki parameter, ia akan melewatkan suatu nilai dengan tipe data tertentu. Kedua komponen tersebut masih mengandung unsur tipe data yang perlu disampaikan dengan jelas. Itulah salah satu alasan sebuah fungsi atau prosedur perlu diberi suatu anotasi untuk memperjelas nilai yang terkandung di dalamnya.

Perhatikan potongan kode berikut.

1. def bayar\_makanan(jumlah, harga\_per\_porsi=10000):
2. total\_bayar: int = jumlah \* harga\_per\_porsi
3. return total\_bayar
5. total: int = bayar\_makanan(2, 13000)
6. print(total)
8. # Output: 26000

Kode di atas memiliki sebuah fungsi bernama **bayar\_makanan** yang berguna untuk menghitung total harga makanan dalam nilai rupiah. Ia memiliki dua nilai parameter, yaitu jumlah dan harga\_per\_porsi, serta satu nilai kembali, yaitu total\_bayar. Nilai tersebut mengandung tipe data integer yang mengikat supaya bisa dieksekusi pada blok program fungsi.

Kita tidak menginginkan fungsi mengalami error karena argumen bernilai string, bukan? Coba kita ubah kedua argumen menjadi string. Jadi, kode akan seperti berikut.

1. def bayar\_makanan(jumlah, harga\_per\_porsi=10000):
2. total\_bayar: int = jumlah \* harga\_per\_porsi
3. return total\_bayar
5. total: int  = bayar\_makanan("Mi ayam bakso", "Nasi goreng")
6. print(total)

Berdasarkan proses assignment argumen pada fungsi, proses di atas tidak salah dan tidak ada error. Namun, apakah blok program pada fungsi mampu menjalankan aksi dengan dua nilai tersebut? Tentu tidak. Program akan menghasilkan error karena tidak bisa mengalikan dua nilai string.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35058?from=35053#run1-editor1)

1

def bayar\_makanan(jumlah, harga\_per\_porsi=10000):

2

total\_bayar: int = jumlah \* harga\_per\_porsi

3

return total\_bayar

4

​

5

total: int = bayar\_makanan("Nasi goreng", "Mi ayam bakso")

6

print(total)

7

​

8

'''

9

Output:

10

Traceback (most recent call last):

11

File "main.py", line 5, in <module>

12

File "main.py", line 2, in bayar\_makanan

13

TypeError: can't multiply sequence by non-int of type 'str'

14

'''

 Input  Reset

 Jalankan

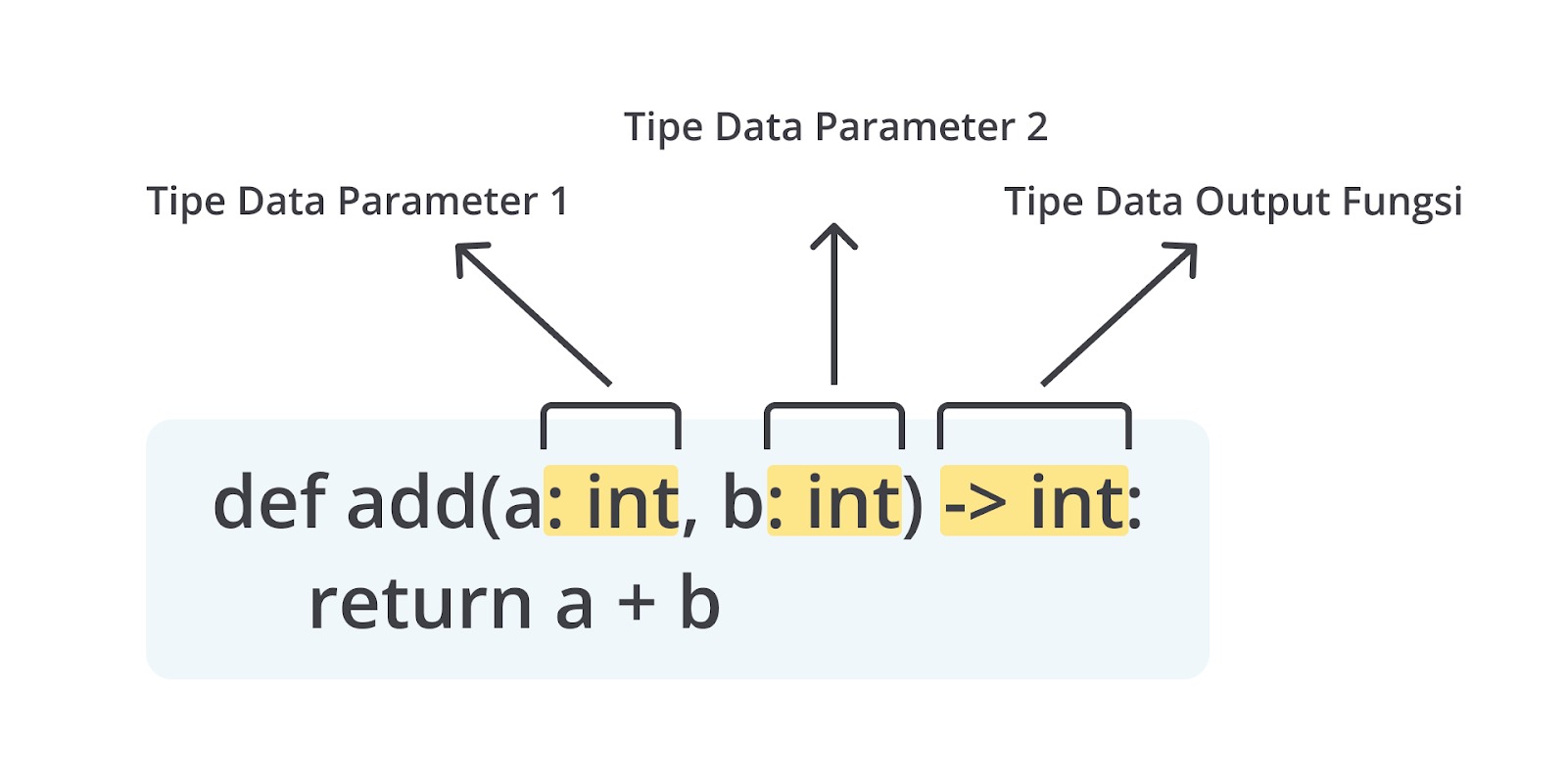
Oleh karena itu, kita perlu menjelaskan secara eksplisit tipe data pada nilai parameter dan output fungsi supaya program tidak mengalami error seperti di atas.

### Pemberian Anotasi pada Fungsi atau Prosedur

Bagaimana cara menambahkan anotasi pada fungsi? Parameter pada fungsi bertugas menyimpan nilai yang akan diteruskan untuk menjalankan aksi tertentu. Parameter dapat diibaratkan sebagai variabel yang harus terpenuhi saat fungsi dipanggil. Jadi, Anda dapat menuliskan anotasi pada parameter layaknya variabel.

Untuk output fungsi, pendefinisian tipe datanya sedikit berbeda. Ia diletakkan pada akhir setelah tanda kurung dan sebelum titik dua. Secara penulisan, ia menggunakan arrow “->” untuk menentukan tipe data yang diharapkan sebagai nilai kembali dari fungsi.

Alhasil, struktur penulisan anotasi pada fungsi dapat diilustrasikan seperti gambar berikut.

[](https://www.dicoding.com/academies/620/tutorials/35058?from=35053)

Berdasarkan ilustrasi di atas, fungsi bayar\_makanan dapat berubah menjadi seperti berikut.

1. def bayar\_makanan(jumlah: **int**, harga\_per\_porsi: **int** = 10000) **-> int**:
2. total\_bayar: int = jumlah \* harga\_per\_porsi
3. return total\_bayar
5. total: int  = bayar\_makanan(2, 13000)
6. print(total)
8. # Output: 26000

Masih ingatkah Anda dengan library mypy? Ya, library yang membantu para pengembang aplikasi Python untuk memeriksa variabel dan fungsi agar sesuai dengan kaidah penulisan. Anda bisa memeriksa kode di atas sudah benar atau tidak. Gunakan perintah mypy diikuti dengan nama file-nya.

$ **mypy main.py**

Success: no issues found in 1 source file

Apa yang terjadi apabila argumen tidak bernilai integer, tetapi string atau boolean? Sudah jelas, mypy akan mendeteksi kesalahan argumen karena tipe data tidak sesuai dengan yang diminta. Fungsi meminta sebuah nilai integer, tetapi ia mendapatkan nilai lain.

1. def bayar\_makanan(jumlah: int, harga\_per\_porsi: int = 10000) -> int:
2. total\_bayar: int = jumlah \* harga\_per\_porsi
3. return total\_bayar
5. total: int  = bayar\_makanan(**"Mi ayam bakso", "Nasi goreng"**)
6. print(total)
8. '''
9. Output Terminal:
10. $ **mypy main.py**
11. main.py:5: error: Argument 1 to "**bayar\_makanan**" has incompatible type "**str**"; expected "**int**"  [arg-type]
12. main.py:5: error: Argument 2 to "**bayar\_makanan**" has incompatible type "**str**"; expected "**int**"  [arg-type]
13. Found 2 errors in 1 file (checked 1 source file)
14. '''

Lalu, bagaimana cara memberikan anotasi pada prosedur? Secara tidak langsung, Anda sudah mampu membuat anotasi pada prosedur. Prosedur hanyalah sebuah fungsi tanpa mengembalikan nilai apa pun. Jadi, Anda tidak perlu memberikan anotasi pada nilai kembali seperti kode berikut.

1. def eatSomething(food: str):     # tidak perlu anotasi pada prosedur
2. print(f"Saya sedang makan {food}.")
3. return
5. eatSomething("Mi Ayam")
7. # Output: Saya sedang makan Mi Ayam.

Ingat, prosedur tidak mengembalikan output apa pun. Akan tetapi, kita bisa memberikan anotasi objek None pada output prosedur. Alhasil, kodenya akan seperti berikut.

1. def eatSomething(food: str) **-> None**:     # diberikan anotasi None pada prosedur
2. print(f"Saya sedang makan {food}.")
3. return
5. eatSomething("Mi Ayam")
7. # Output: Saya sedang makan Mi Ayam.

Kedua cara di atas tetap benar. Anda memberikan anotasi atau tidak, prosedur tetap terindikasi mengembalikan objek None.

### Melihat Daftar Anotasi

Setelah membuat anotasi pada seluruh variabel atau fungsi, kita dapat memeriksa dan menampilkannya melalui program. Bagaimana caranya? Python menyediakan atribut **\_\_annotations\_\_** bawaan yang berisi dictionary. Di dalamnya terdapat informasi tipe data yang telah kita beri anotasi, mulai dari variabel dalam program, parameter, hingga nilai kembali pada fungsi.

Cara pemanggilannya cukup sederhana. Anda dapat memanggil atribut \_\_annotations\_\_ seperti berikut.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35058?from=35053#run2-editor1)

1

def bayar\_makanan(jumlah: int, harga\_per\_porsi: int = 10000) -> int:

2

total\_bayar: int = jumlah \* harga\_per\_porsi

3

return total\_bayar

4

​

5

total: int = bayar\_makanan(2, 13000)

6

​

7

print(\_\_annotations\_\_)

8

print(bayar\_makanan.\_\_annotations\_\_)

9

​

10

'''

11

Output:

12

{'total': <class 'int'>}

13

{'jumlah': <class 'int'>, 'harga\_per\_porsi': <class 'int'>, 'return': <class 'int'>}

14

'''

 Input  Reset

 Jalankan

Ketika mencetak isi atribut anotasi, Anda dapat melihat seluruh tipe data variabel yang tersedia pada keseluruhan program. Berbeda cerita ketika ingin mencetak isi atribut anotasi fungsi, Anda hanya dapat memeriksa tipe data parameter dan output fungsi saja, tidak untuk variabel dalam fungsi.

Ingat, atribut tersebut hanya berlaku pada variabel dan fungsi yang sudah diberi anotasi saja, ya. Anda tidak dapat memeriksa tipe data variabel yang belum tercatat.

Coba perhatikan kode berikut.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35058?from=35053#run3-editor1)

1

def bayar\_makanan(jumlah: int, harga\_per\_porsi: int = 10000) -> int:

2

total\_bayar: int = jumlah \* harga\_per\_porsi

3

return total\_bayar

4

​

5

total = bayar\_makanan(2, 13000) # anotasi variabel total dihilangkan

6

​

7

print(\_\_annotations\_\_)

8

print(bayar\_makanan.\_\_annotations\_\_)

9

​

10

'''

11

Output:

12

{}

13

{'jumlah': <class 'int'>, 'harga\_per\_porsi': <class 'int'>, 'return': <class 'int'>}

14

'''

 Input  Reset

 Jalankan

Ketika menghilangkan anotasi pada variabel total, kita tidak dapat melihat tipe datanya. Atribut **\_\_annotation\_\_** hanya menyimpan informasi tipe data pada variabel yang beranotasi.

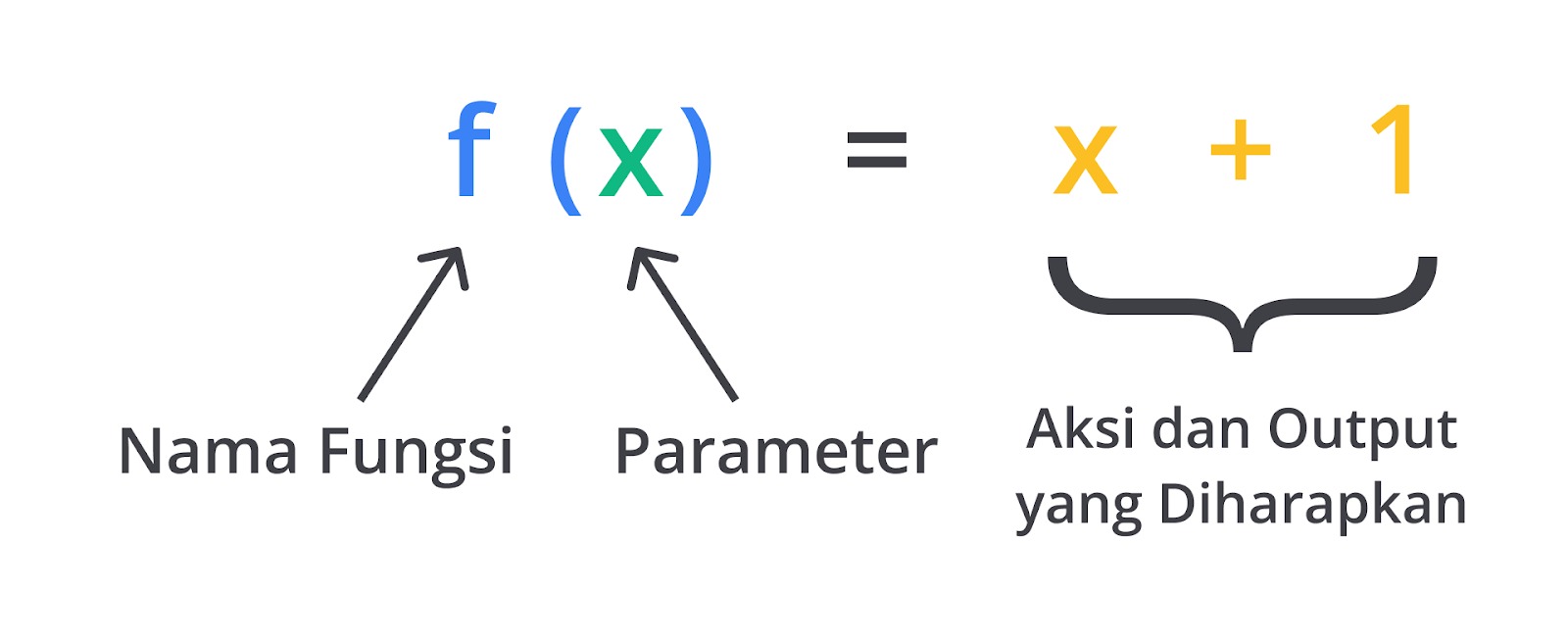
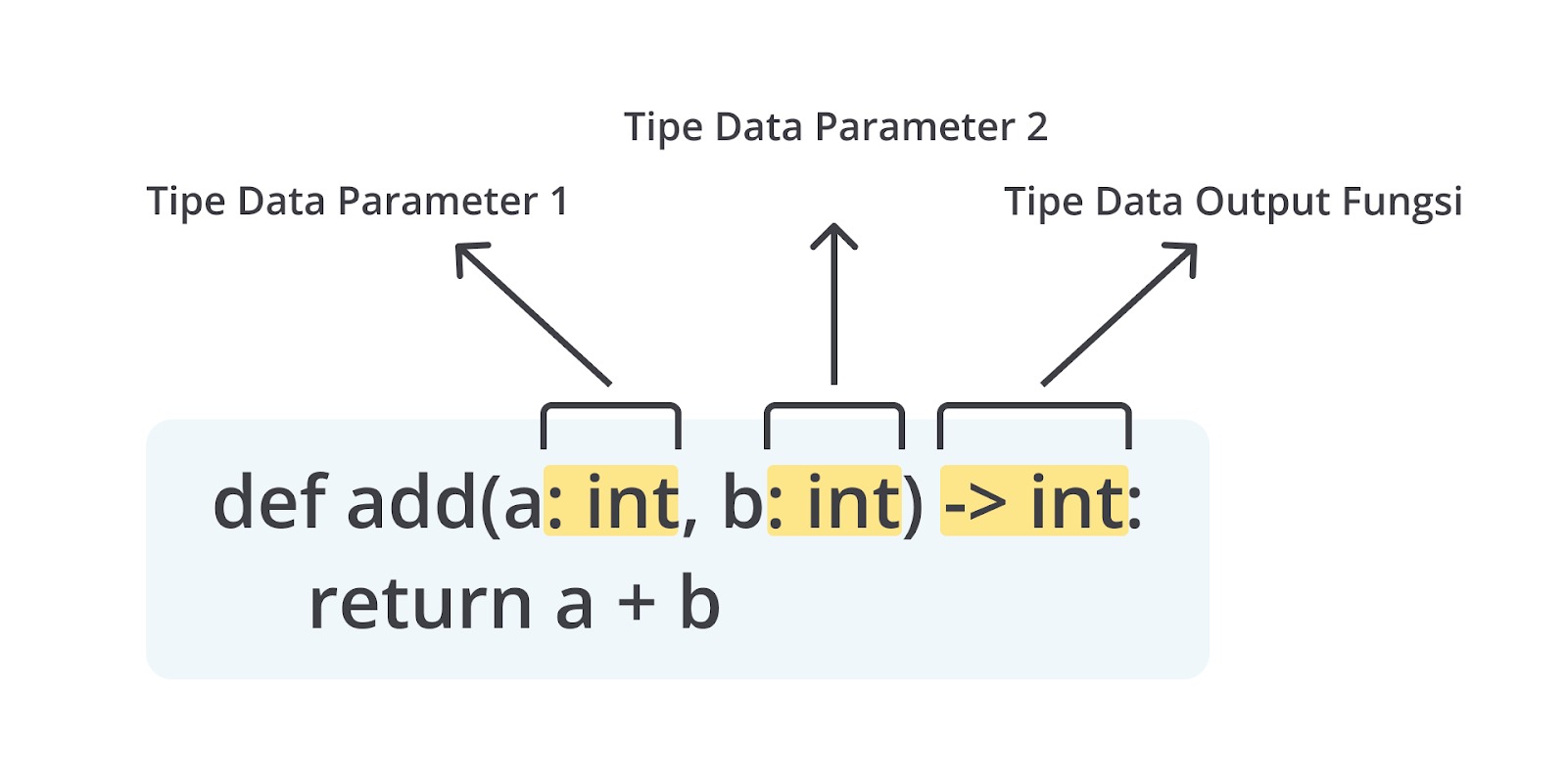
Materinya sangat insightful, bukan? Type annotation ini sangat membantu Anda untuk menyimpan informasi tipe data. Namun, penting untuk diingat bahwa informasi ini hanya informatif dan tidak memengaruhi perilaku eksekusi kode. Akan lebih baik apabila Anda menerapkan type annotation pada keseluruhan kode program.

Apabila Anda ingin mengeksplorasi materi terkait type annotation lainnya, silakan kunjungi beberapa tautan berikut.

* [Function Annotations](https://docs.python.org/3/tutorial/controlflow.html#function-annotations).
* [Python is a Nutshell, 4th Edition: Type Annotations](https://learning.oreilly.com/library/view/python-in-a/9781098113544/ch05.html).
* [Robust Python: Type Annotations](https://learning.oreilly.com/library/view/robust-python/9781098100650/ch03.html).

## Rangkuman Berenang dalam Lautan Fungsi dan Prosedur

Selamat! Anda sudah berada di penghujung materi terkait Berenang dalam Lautan Fungsi dan Prosedur. Sejauh ini, Anda telah berlatih cara membuat dan memahami konsep fungsi dan prosedur dalam menyelesaikan permasalahan. Berikut adalah ringkasan materi yang sudah Anda pelajari.

* Fungsi dapat diartikan sebagai suatu proses instruksi yang dibungkus dengan nama/label dan mengembalikan nilai kembali. Di dalamnya terdapat serangkaian aksi dengan tujuan yang spesifik.
* Dalam kurun waktu tertentu, fungsi dapat dipanggil dan dijalankan berkali-kali bergantung pada kebutuhan.
* Ciri khas dari sebuah fungsi ada tiga komponen, yaitu fungsi harus memiliki **tujuan khusus**, menjalankan **serangkaian instruksi** tertentu, dan **menghasilkanoutput**. Selagi ketiga komponen tersebut dimiliki pada suatu instruksi, ia dapat dikatakan sebagai fungsi.  
  [](https://www.dicoding.com/academies/620/tutorials/35078)
* Interferensi dapat terjadi dari dalam dan luar fungsi. Apabila interferensi dari dalam, aspek atau komponen memang dibutuhkan dari dalam fungsi dan sudah tersedia di dalamnya. Sebaliknya, ada sebuah fungsi yang membutuhkan komponen penting, tetapi belum tersedia di dalamnya.
* Ada komponen opsional dari sebuah fungsi, yaitu parameter. Ia membantu jalannya suatu fungsi untuk menghasilkan output yang diharapkan.
* Parameter harus memiliki spesifikasi yang relevan sesuai permintaan fungsi. Jika tidak, fungsi tidak dapat berjalan dengan baik.
* Fungsi dengan parameter dapat diibaratkan sebagai fungsi pada matematika. Ia memiliki nama fungsi, parameter, dan output yang diharapkan.  
  [](https://www.dicoding.com/academies/620/tutorials/35078)
* Mendeklarasikan fungsi dalam bahasa pemrograman Python serupa dengan menuliskan fungsi matematika. Hanya saja, ada kata kunci **def** sebelum nama fungsi, aksi dibungkus layaknya blok program pada umumnya, dan output fungsi diberi kata kunci **return**.
  1. # deklarasi fungsi
  2. def f(x):
  3. return x + 1
  5. # pemanggilan fungsi
  6. y = f(4)
  7. print(y)
  9. # Output: 5
* Kata kunci **def** adalah penanda bahwa kode tersebut merupakan sebuah fungsi.
* Mungkin, Anda sering tertukar dengan istilah parameter dan argumen. Kedua istilah tersebut memiliki konsep yang berbeda, tetapi saling terikat satu dengan lainnya.
  1. Argumen adalah nilai atau ekspresi yang dikirim ke fungsi saat fungsi dipanggil. Argumen ini diberikan ke parameter dan diteruskan ke dalam fungsi untuk menjalankan aksi tertentu.
  2. Parameter ini dapat diibaratkan sebagai spesifikasi atau syarat untuk menjalankan fungsi. Ia sebagai wadah untuk menyimpan nilai dari luar fungsi. Ketika fungsi dipanggil, parameter harus diisi sebuah nilai supaya fungsi dapat bekerja dengan benar.
* Ketika fungsi memiliki parameter, ia harus diberi suatu argumen saat dipanggil agar dapat bekerja dengan benar.
* Argumen dapat diisi saat pemanggilan fungsi berlangsung. Namun, adakalanya, kita tidak perlu memasukkan argumen saat fungsi dipanggil. Kita dapat memanfaatkan parameter opsional sehingga argumen tidak selalu didefinisikan setiap fungsi dipanggil.
  1. def bayar\_makanan(jumlah, harga\_per\_porsi=10000):
  2. total\_bayar = jumlah \* harga\_per\_porsi
  3. return total\_bayar
  5. total = bayar\_makanan(2)
  6. print(total)
  8. # Output: 20000
* Fungsi yang memiliki parameter dapat dipanggil dengan dua cara.
  1. **Positional argument**: meletakkan argumen berdasarkan posisi atau urutan parameter fungsi.
  2. **Keyword argument**: meletakkan argumen sesuai nama parameter dan diikuti dengan nilai argumennya.
* Ada beberapa alasan yang perlu kita pertimbangkan dalam pembuatan fungsi dengan banyak parameter. Berikut alasannya.
  1. Pemanggilan fungsi yang rumit.
  2. Sulit memahami fungsinya.
  3. Memungkinkan salah mengirim argumen.
  4. Tingkat fleksibilitas rendah.
* Python memiliki cara untuk meminimalkan permasalahan parameter yang banyak. Berikut caranya.
  1. **Positional-or-Keyword**: membuat fungsi yang bisa diisi argumen sesuai urutan atau berdasarkan kata kunci parameter.
  2. **Positional-Only**: membuat fungsi yang hanya diisi argumen berdasarkan urutan parameter.
  3. **Keyword-Only**: membuat fungsi yang hanya diisi argumen berdasarkan kata kunci parameter.
  4. **Var-positional**: membuat fungsi yang hanya diisi satu argumen dan dapat dicacah layaknya sebuah data list.
  5. **Var-keyword**: membuat fungsi yang hanya diisi satu argumen dan dapat dicacah layaknya sebuah data dictionary.
* Pengertian prosedur menurut KBBI adalah tahap kegiatan dalam menyelesaikan suatu aktivitas.
* Dalam pemrograman, prosedur dapat diartikan sebagai blok program yang menjalankan serangkaian instruksi untuk menjalankan tugas tertentu.
* Prosedur dan fungsi memiliki perbedaan yang mendasar, yaitu fungsi memiliki nilai kembali atau menghasilkan output, sedangkan prosedur tidak.
* Layaknya fungsi, prosedur juga dapat memiliki parameter yang membantu jalannya serangkaian aksi. Parameter ini sifatnya opsional dan tidak harus dimiliki oleh prosedur.
* Berikut adalah contoh deklarasi dan pemanggilan prosedur pada bahasa pemrograman Python.
  1. # deklarasi prosedur
  2. def eatSomething(food):
  3. print(f"Saya sedang makan {food}.")
  5. # pemanggilan prosedur
  6. eatSomething("Mi Ayam")
  8. # Output: Saya sedang makan Mi Ayam.
* Nyatanya, prosedur pada Python tetap mengembalikan nilai berupa objek None. Ia merupakan objek bertipe data NoneType yang tidak bernilai apa pun.
* Alasan pemberian anotasi pada fungsi atau prosedur sebagai berikut.
  1. Untuk mengatasi salah pemberian argumen pada parameter yang ada.
  2. Untuk mengetahui nilai kembali suatu fungsi.
* Struktur penulisan anotasi pada fungsi atau prosedur dapat diilustrasikan seperti gambar berikut.  
  [](https://www.dicoding.com/academies/620/tutorials/35078)
* Pemberian anotasi pada parameter fungsi dapat dilakukan layaknya pembuatan anotasi pada variabel menggunakan titik dua/colon “:” diikuti dengan tipe datanya.
* Pemberian anotasi pada pengembalian fungsi dapat diletakkan di akhir setelah tanda kurung dan sebelum titik dua. Gunakanlah arrow “->” untuk menentukan tipe data yang diharapkan sebagai nilai kembali dari fungsi.
* Anda dapat melihat daftar anotasi yang sudah dibuat dengan menggunakan atribut **\_\_annotations\_\_** bawaan Python yang berisi dictionary atau informasi tipe data.
* Untuk mengakses anotasi variabel di luar fungsi, cetaklah isi atribut \_\_annotations\_\_ pada perintah print, sedangkan untuk anotasi dalam fungsi, cetaklah isinya dengan memanggil nama fungsi dan method \_\_annotations\_\_.
  1. def bayar\_makanan(jumlah: int, harga\_per\_porsi: int = 10000) -> int:
  2. total\_bayar: int  = jumlah \* harga\_per\_porsi
  3. return total\_bayar
  5. total: int  = bayar\_makanan(2, 13000)
  7. print(**\_\_annotations\_\_**)
  8. print(**bayar\_makanan.\_\_annotations\_\_**)
  10. '''
  11. Output:
  12. {'total': <class 'int'>}
  13. {'jumlah': <class 'int'>, 'harga\_per\_porsi': <class 'int'>, 'return': <class 'int'>}
  14. '''