**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.

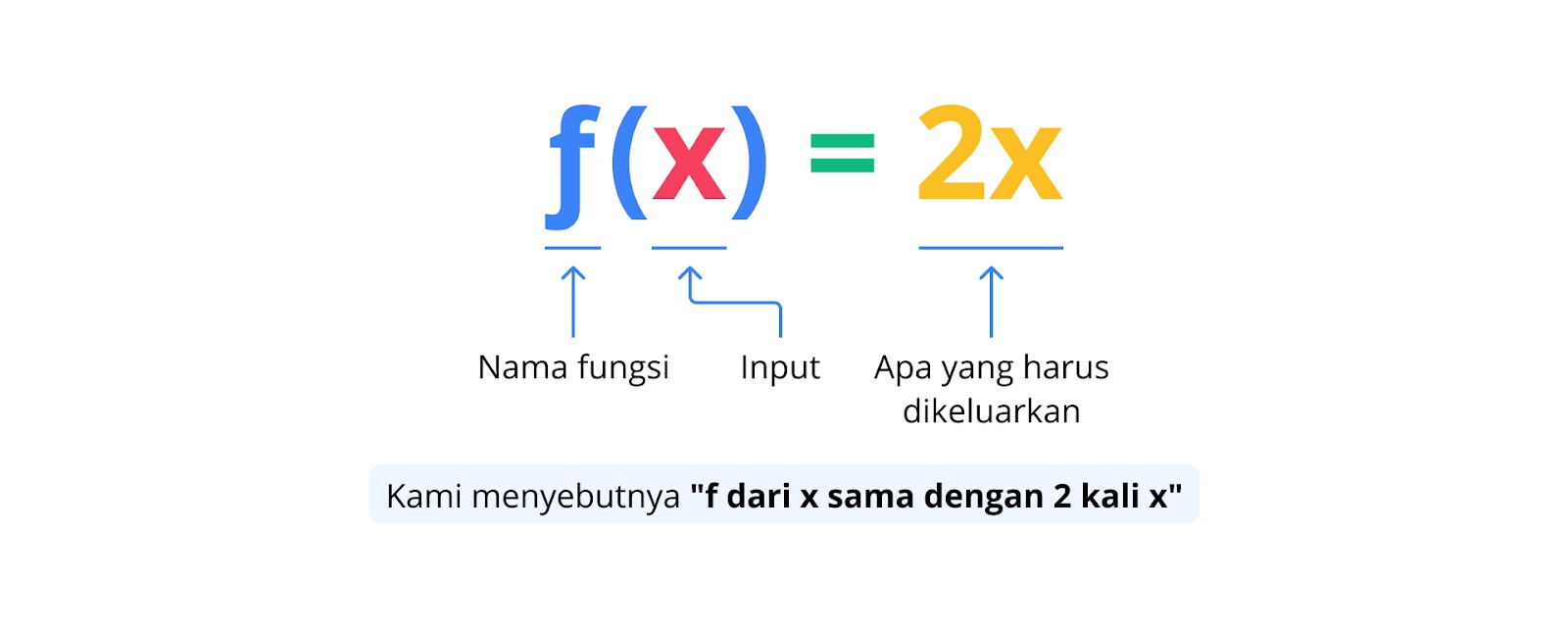
**Definisi Subprogram**

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/32973)

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/32973)

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.

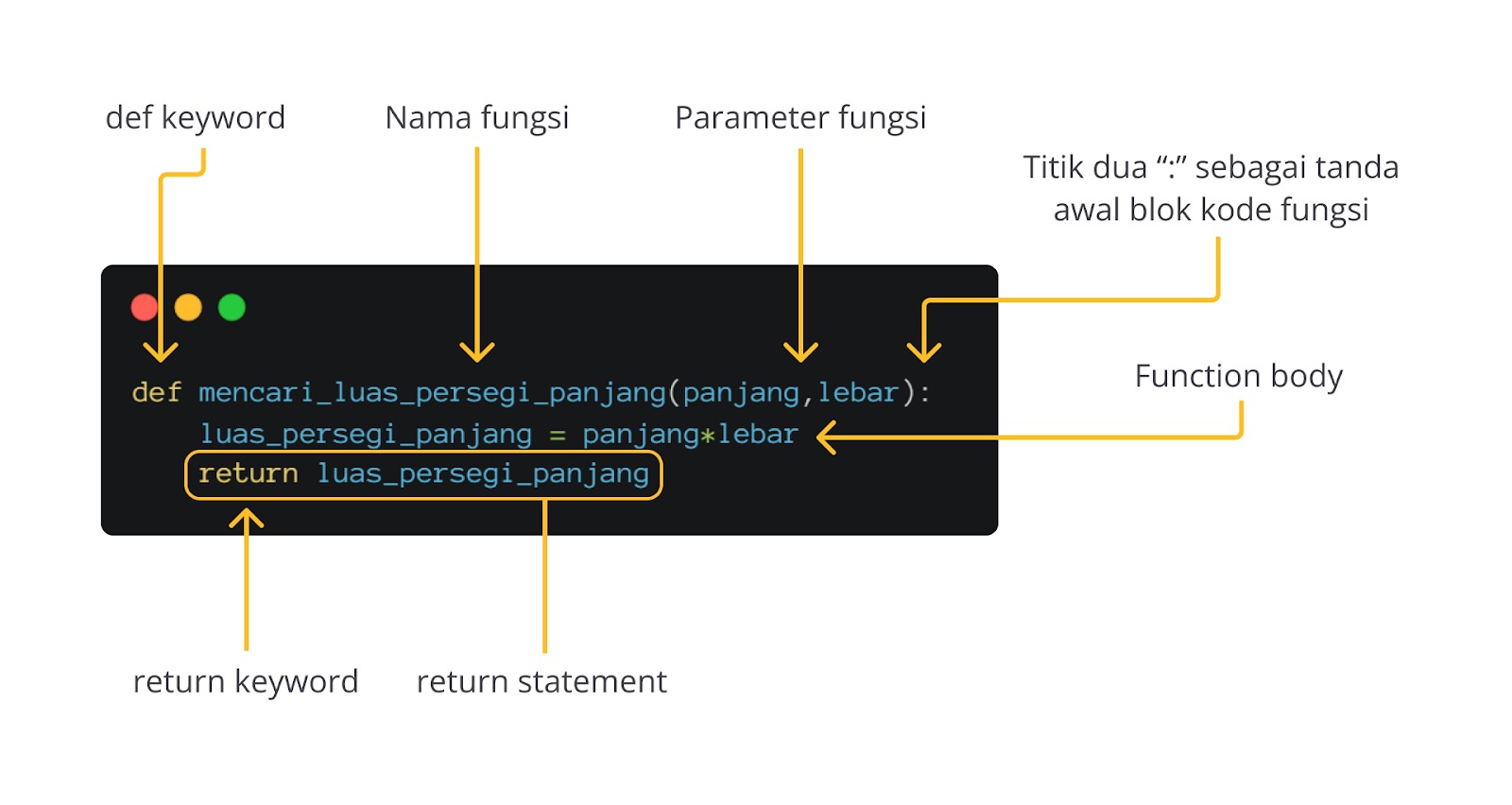
Namun, jika Anda ingin menggunakan fungsi di luar built-in functions, Anda bisa mengimpor sebuah library. Library adalah koleksi banyaknya modul yang saling terkait dan dapat digunakan berulang kali. Library dalam Python terbagi menjadi dua jenis, yakni berikut.

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 |

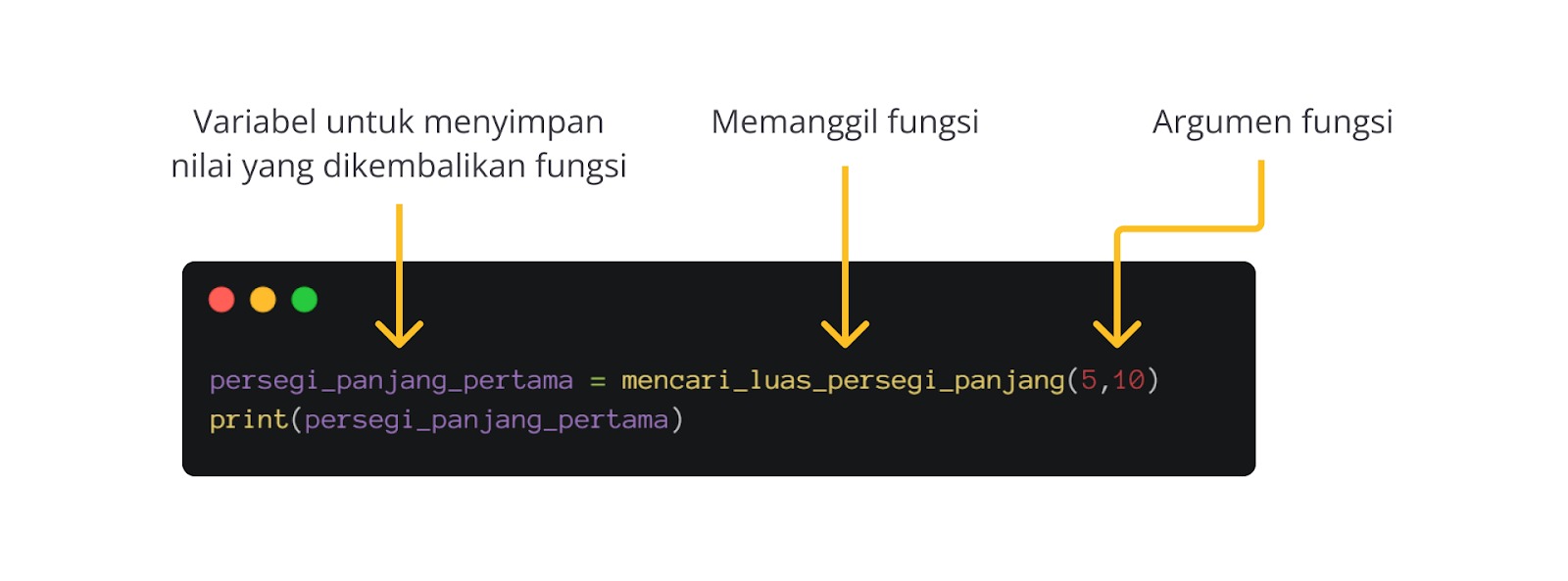
Untuk membuat fungsi sendiri (user-defined functions) dalam Python, kita dapat membuatnya dengan mengikuti struktur berikut.

[](https://www.dicoding.com/academies/86/tutorials/32973)

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/32973)

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.

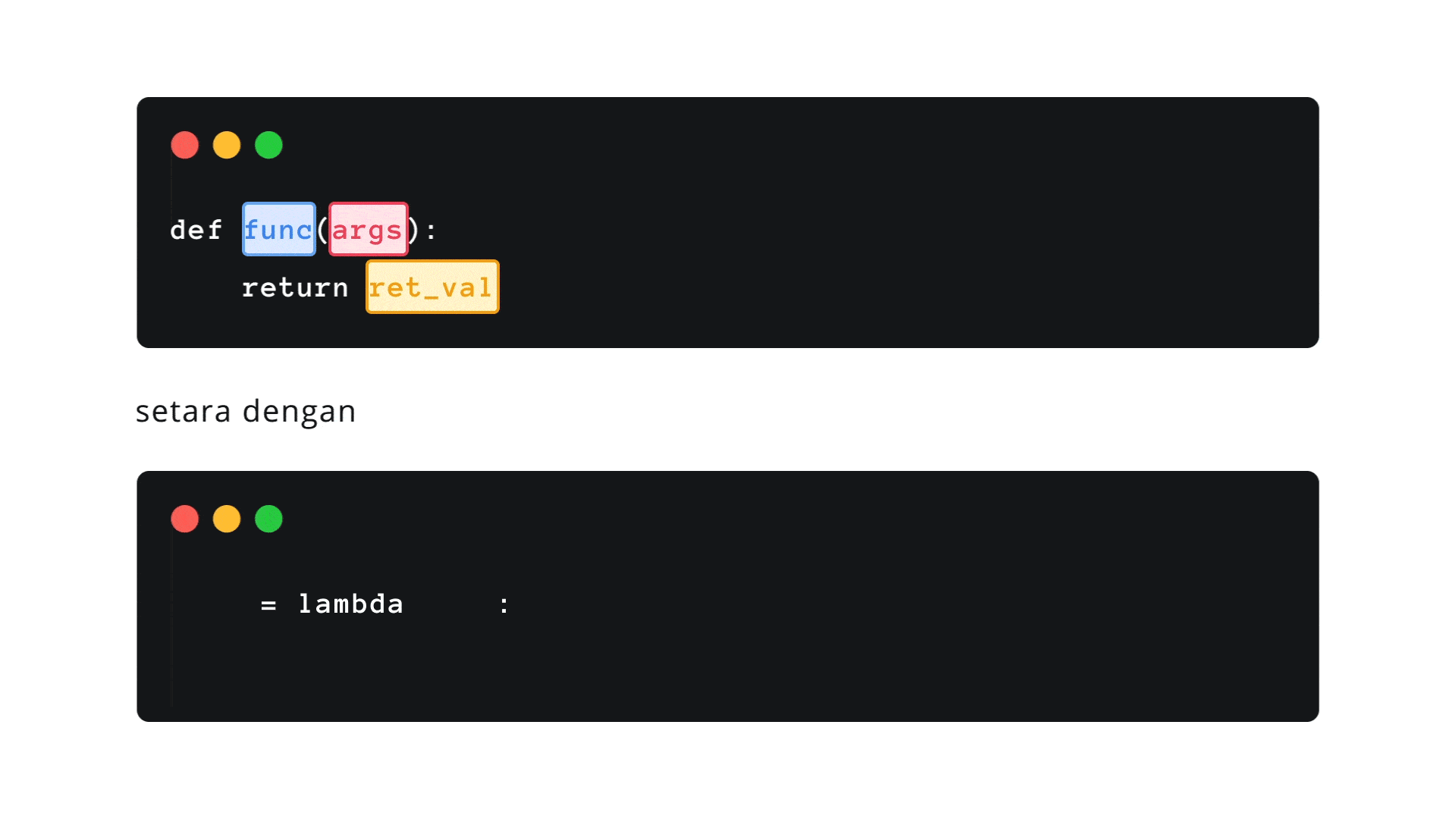
1. def mencari\_luas\_persegi\_panjang(panjang,lebar):
2. """
3. Fungsi ini digunakan untuk menghitung luas persegi panjang.
5. Args:
6. panjang (int): Panjang persegi panjang.
7. lebar (int): Lebar persegi panjang.
9. Returns:
10. int: Luas persegi panjang hasil perhitungan.
11. """
13. luas\_persegi\_panjang = 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/32973)

**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

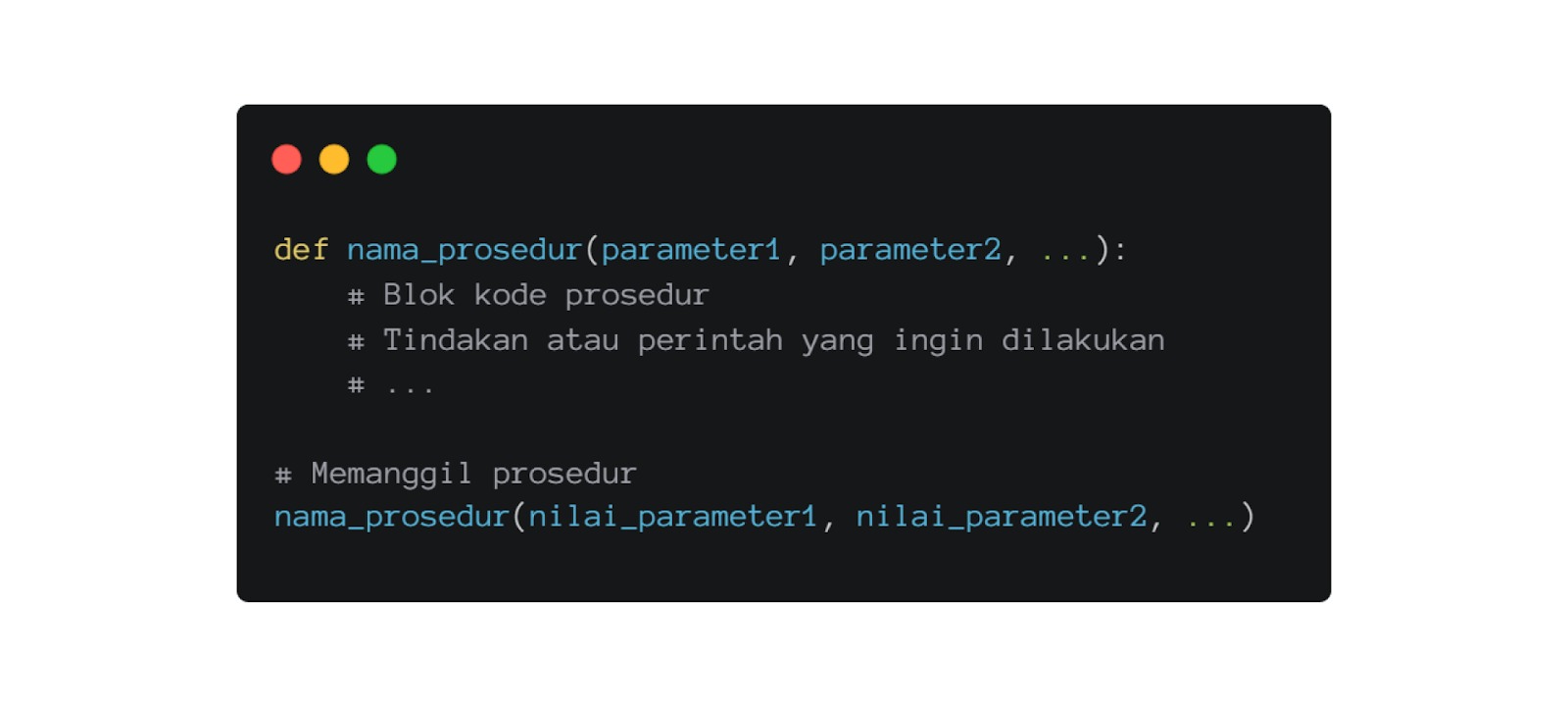
Anda juga bisa mengimpor kode, seperti fungsi, kelas, hingga variabel secara spesifik. Misalnya Anda ingin mengimpor fungsi “mencari\_luas\_persegi\_panjang” dan variabel “nama” dari modul hello. Gunakan kode di bawah ini.

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/32973)

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()