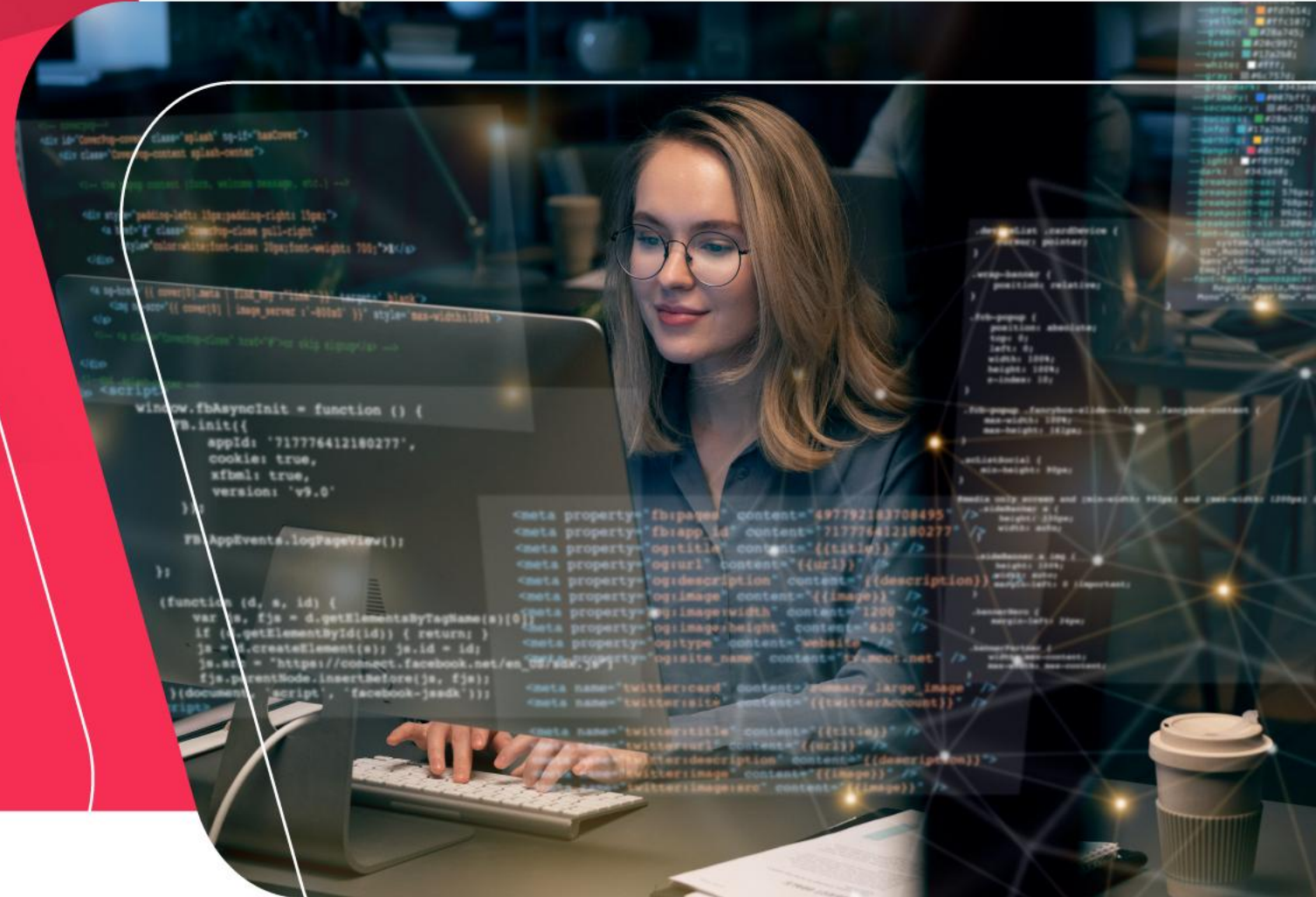


# Python for Data Science

Day 03



Arithmetic

# Math Operation

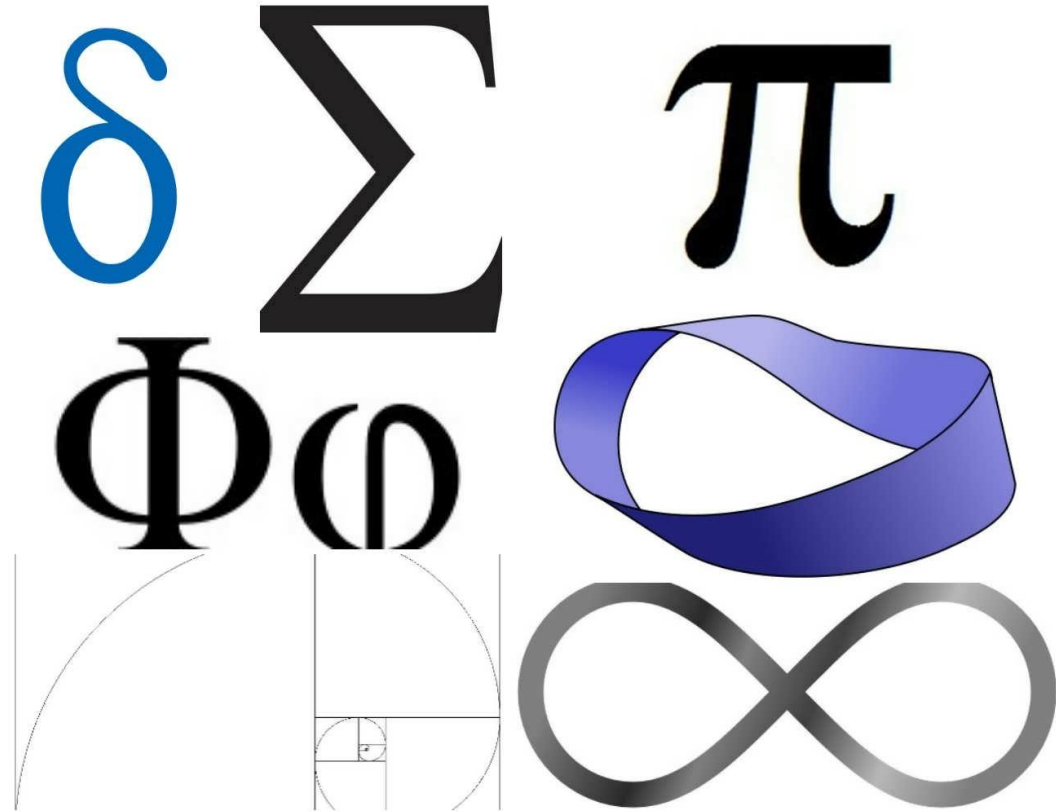
Addition ( + )

Subtraction ( - )

Multiply ( \* )

Division ( / )

Modulus ( % )



# Math Operation

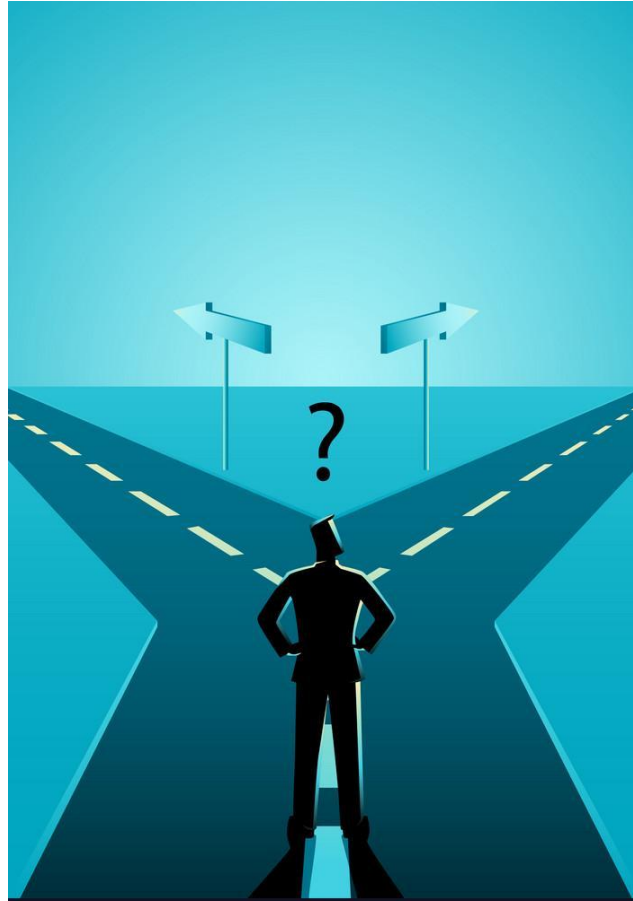
Operation	a	b	Code	c
Addition	6	2	<code>c = a + b</code>	8
Subtraction			<code>c = a - b</code>	4
Multiply			<code>c = a * b</code>	12
Division			<code>c = a / b</code>	3
Modulus			<code>c = a % b</code>	0

# Math Operation

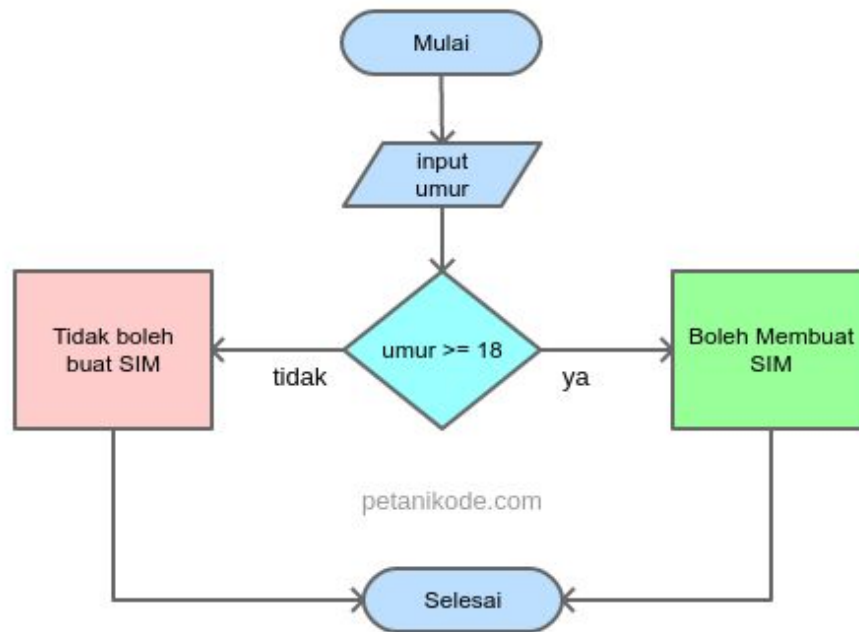
Operation	a	b	Code	c
Greater Than	6	2	<code>a &gt; b</code>	True
Greater Equal Than			<code>a &gt;= b</code>	True
Less Than			<code>a &lt; b</code>	False
Less Equal Than			<code>a &lt;= b</code>	False
Equal			<code>a == b</code>	False
Not Equal			<code>a != b</code>	True

# Conditional If

# What If?



# What If?



Conditional If can be used to choose path of workflow.

Example:

1. Exam
2. License Age Eligibility



# Example



```
condition.py > ...  
1  ### Struktur Percabangan If ###  
2  lulus = "tidak"  
3  if lulus == "tidak":  
4      print("kamu harus ikut remidial")  
5
```



```
condition.py > ...  
1  ### Struktur Percabangan If ###  
2  lulus = "tidak"  
3  if lulus == "tidak":  
4  print("kamu harus ikut remidial")  
5
```

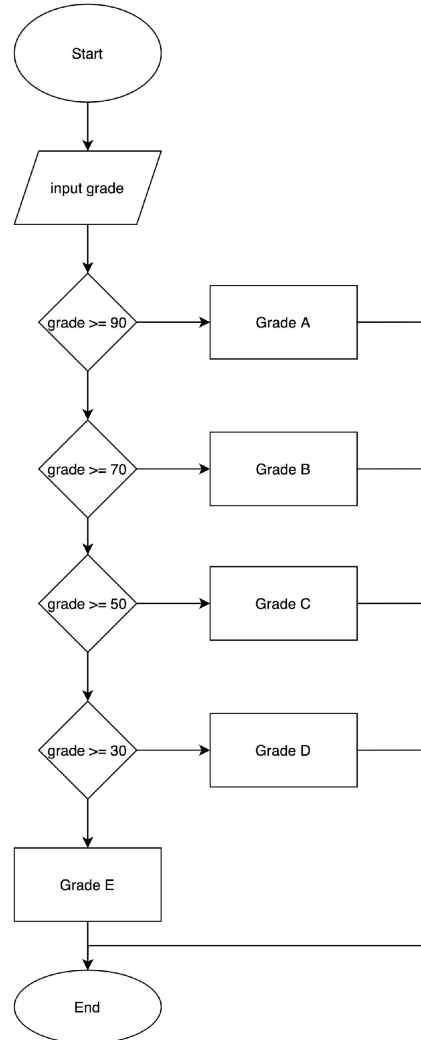
- “Jika **lulus == "tidak"** maka cetak teks "kamu harus ikut remidi"”
- Kita menggunakan operator relasi sama dengan (**==**) untuk membandingkan isi variabel **lulus**. Sedangkan tanda titik-dua (**:**) adalah tanda untuk memulai blok kode *If*.
- Penulisan blok *If*, harus diberikan indentasi **tab** atau **spasi 2x**.

# Another Example

```
6  # cek_umur.py
7  umur = input("Berapa umur kamu: ")
8
9  if umur >= 18:
10     print("Kamu boleh membuat SIM")
11 else:
12     print("Kamu belum boleh membuat SIM")
13
```

- Selain blok *If*, terdapat juga blok *Else* yang akan dieksekusi apabila kondisi **umur >= 18** salah (**False**).

# Nested If



- Percabangan *If/Elif/Else* di gunakan apabila terdapat lebih dari dua pilihan keputusan.
- Kata kunci **elif** artinya *Else if*, fungsinya untuk membuat kondisi/logika tambahan apabila kondisi pertama salah.

# Example

# Looping

# Looping

- Perulangan dalam bahasa pemrograman berfungsi menyuruh komputer melakukan sesuatu secara berulang-ulang. Terdapat dua jenis perulangan dalam bahasa pemrograman python, yaitu perulangan dengan **for** dan **while**.
- Perulangan **for** disebut *counted loop* (perulangan yang terhitung), sementara perulangan **while** disebut *uncounted loop* (perulangan yang tak terhitung). Perbedaannya adalah perulangan **for** biasanya digunakan untuk mengulangi kode yang sudah diketahui banyak perulangannya. Sementara **while** untuk perulangan yang memiliki syarat dan tidak tentu berapa banyak perulangannya.

# Looping - For

```
perulangan.py > ...  
1  ### Perulangan For ###  
2  for indek in range(banyak_perulangan):  
3      # jalankan kode ini  
4      # jalankan juga kode ini  
5  #kode ini tidak akan diulang karena berada di luar for
```

- Variabel **indek** berfungsi untuk menampung indeks, dan fungsi **range()** berfungsi untuk membuat list dengan nilai dari range.

# Looping – Example For

```
7  # file: perulanganFor.py
8  ulang = 10
9
10 for i in range(ulang):
11     print "Perulangan ke-" + str(i)

14 # berkas: perulanganFor.py
15 item = ['kopi', 'nasi', 'teh', 'jeruk']
16
17 for isi in item:
18     print isi
```

- Pertama kita menentukan banyak perulangannya sebanyak 10x
- Variabel **i** berfungsi untuk menampung indeks, dan fungsi **range()** berfungsi untuk membuat list dengan range dari 0-10. Fungsi **str()** berfungsi merubah tipe data integer ke string.



# Looping – Example While

```
20  ### Perulangan While ###
21  while(True):
22      # jalankan kode ini
23      # kode ini berada di luar perulangan while
24
25  # berkas: perulanganWhile.py
26  jawab = 'ya'
27  hitung = 0
28
29  while(True):
30      hitung += 1
31      jawab = raw_input("Ulang lagi tidak? ")
32      if jawab == 'tidak':
33          break
34
35  print "Total perulangan: " + str(hitung)
```

- Pertama menentukan variabel untuk menghitung, dan menentukan kapan perulangan berhenti. kalau pengguna menjawab tidak maka perulangan akan terhenti.
- Melakukan perulangan dengan *while*, kemudian menambah satu variabel **hitung** setiap kali mengulang. lalu menanyakan kepada pengguna, apakah mau berhenti mengulang atau tidak?
- Setelah selesai mengulang, cetak berapa kali perulangan tersebut terjadi

# Function

# fungsi

```
fungsi.py > ...  
1  ### Fungsi ###  
2  def nama_fungsi():  
3      print "Hello ini Fungsi"
```

The diagram illustrates the structure of a Python function definition. It shows the code `def hello():` followed by two indented lines: `print "Hello"` and `print "world"`. Annotations include: an arrow pointing to `hello()` labeled "nama fungsi"; an arrow pointing to the indented lines labeled "body/isi fungsi"; a red double-headed arrow between the `def` keyword and the first indented line labeled "indentasi"; and a red dashed box enclosing the two indented lines.

- Pada pembuatan program yang kompleks dan memiliki banyak fitur, kita diharuskan menggunakan fungsi.
- Dengan fungsi, kita dapat memecah program besar menjadi sub program yang lebih sederhana.
- Masing-masing fitur pada program dapat kita buat dalam satu fungsi. Pada saat kita membutuhkan fitur tersebut, kita tinggal panggil fungsinya saja.

# Contoh fungsi

```
5  # Membuat Fungsi
6  def salam():
7      print "Hello, Selamat Pagi"
8
9  ## Pemanggilan Fungsi
10 salam()
```

```
5  # Membuat Fungsi
6  def salam():
7      print "Hello, Selamat Pagi"
8
9  ## Pemanggilan Fungsi
10 salam()
11 salam()
12 salam()
```

```
Hello, Selamat Pagi
Hello, Selamat Pagi
Hello, Selamat Pagi
```

- **Salam()** dibawah code, berfungsi untuk memanggil/mengambil isi fungsi bernama **salam()**.
- Intinya apapun yang ada di dalam fungsi, ketika dipanggil itulah yang akan dilakukan.
- fungsi juga dapat dipanggil pada fungsi lain, bahkan bisa memanggil dirinya sendiri. Fungsi yang memanggil dirinya sendiri, disebut fungsi rekursif.

# Fungsi dengan parameter

```
def fungsi(parameter):  
    print parameter
```

ini parameter  
↓

- Sekarang, bagaimana kalau kita ingin memberikan input nilai ke dalam fungsi? Kita bisa memanfaatkan parameter.
- Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi.

# Contoh fungsi dengan parameter

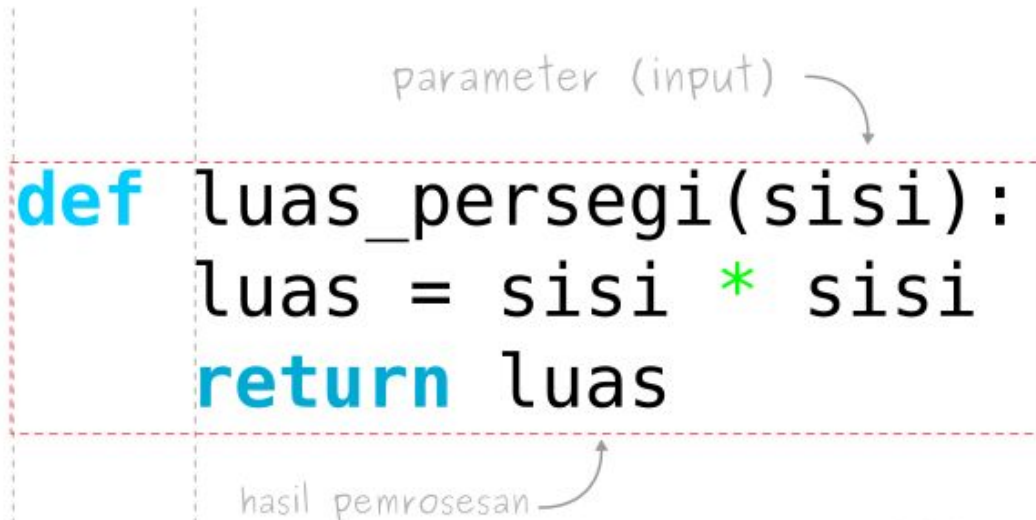
```
14  ### Fungsi Dengan Parameter ###
15  def salam(ucapan):
16      print(ucapan)
17      salam("Selamat siang")
18
```

```
19  # Membuat fungsi dengan parameter
20  def luas_segitiga(alas, tinggi):
21      luas = (alas * tinggi) / 2
22      print "Luas segitiga: %f" % luas
23
24  # Pemanggilan fungsi
25  luas_segitiga(4, 6)
```

```
Luas segitiga: 12.000000
```

- kita membuat fungsi dengan parameter **ucapan**.
- **"Selamat siang"** adalah nilai parameter yang kita berikan. Jika parameter ada 2, kita bisa menggunakan tanda koma (,) untuk memisahkannya.

# Fungsi yang mengembalikan nilai



```
def luas_persegi(sisi):  
    luas = sisi * sisi  
    return luas
```

The diagram shows a Python function definition for calculating the area of a square. The function is enclosed in a dashed red box. An arrow points from the text "parameter (input)" to the parameter "sisi" in the function signature. Another arrow points from the text "hasil pemrosesan" to the "return" statement.

- Fungsi yang tidak mengembalikan nilai biasanya disebut dengan prosedur.
- Namun, kadang kita butuh hasil proses dari fungsi untuk digunakan pada proses berikutnya. Maka fungsi harus mengembalikan nilai dari hasil pemrosesannya.
- Cara mengembalikan nilai adalah menggunakan kata kunci **return** lalu diikuti dengan nilai atau variabel yang akan dikembalikan.

# Contoh fungsi yang mengembalikan nilai

```
27 ##### Fungsi Yang Mengembalikan Nilai ###
28 def luas_persegi(sisi):
29     luas = sisi * sisi
30     return luas
31
32 # pemanggilan fungsi
33 print "Luas persegi: %d" % luas_persegi(6)
```

Luas persegi: 36

```
35 # rumus: sisi x sisi
36 def luas_persegi(sisi):
37     luas = sisi * sisi
38     return luas
39
40
41 # rumus: sisi x sisi x sisi
42 def volume_persegi(sisi):
43     volume = luas_persegi(sisi) * sisi
```

- Apa bedanya dengan fungsi `luas_segitiga()` yang tadi?
- Pada fungsi `luas_segitiga()` kita melakukan `print` dari hasil pemrosesan secara langsung di dalam fungsinya. Sedangkan fungsi `luas_persegi()`, kita melakukan `print` pada saat pemanggilannya.
- Jadi, fungsi `luas_persegi()` akan bernilai sesuai dengan hasil yang dikembalikan. Sehingga kita dapat memanfaatkannya untuk pemrosesan berikutnya.
- Pada contoh di samping, kita melakukan pemanggilan fungsi `luas_persegi()` untuk menghitung volume persegi.



# Variabel global dan lokal pada python

- Saat kita menggunakan fungsi, maka kita juga harus mengetahui yang namanya variabel Global dan Lokal.
- Variabel Global adalah variabel yang bisa diakses dari semua fungsi, sedangkan variabel lokal hanya bisa diakses di dalam fungsi tempat ia berada saja.
- Pada Python, urutan pengaksesan variabel (*scope*) dikenal dengan sebutan LGB (Local, Global, dan Build-in). Jadi program python mulai mencari variabel lokal terlebih dahulu, kalau ada maka itu yang digunakan. Variabel Build-in adalah variabel yang sudah ada di dalam Python.

# Contoh variabel global dan lokal

```
45  ### Variabel global dan lokal pada python ###
46  # ini variabel global
47  nama = "Cimiko"
48  versi = "1.0.0"
49
50  def help():
51      # ini variabel lokal
52      nama = "My Program"
53      versi = "1.0.2"
54      # mengakses variabel lokal
55      print "Nama: %s" % nama
56      print "Versi: %s" % versi
57
58
59  # mengakses variabel global
60  print "Nama: %s" % nama
61  print "Versi: %s" % versi
62
63  # memanggil fungsi help()
64  help()
```

- Perhatikanlah variabel **nama** yang berada di dalam fungsi **help()** dan diluar fungsi **help()**.
- Variabel **nama** yang berada di dalam fungsi **help()** adalah variabel lokal.
- Jadi, saat kita memanggil fungsi **help()** maka nilai yang akan tampil adalah nilai yang ada di dalam fungsi **help()**. Python mulai mencari dari lokal, ke global, dan build-in.
- Kalau di tiga tempat itu tidak ditemukan, maka biasanya akan terjadi **NameError** atau variabel tidak ditemukan.

```
Nama: Cimiko
Versi: 1.0.0
Nama: My Program
Versi: 1.0.2
```