FUNGSI

Dasar-Dasar Pemrograman

Semester Gasal 2020/2021

Tujuan

- Memahami konsep dasar dan manfaat fungsi dalam pemrograman
- Memahami konsep variabel lokal
- Memahami penggunaan argumen/parameter pada fungsi
- Memahami penggunaan *keyword* `return` pada fungsi
- Memahami penggunaan *default arguments* dan *keyword arguments*

Fungsi

Definisi Fungsi

- Fungsi (*method*) adalah sebuah blok kode yang melakukan suatu tujuan tertentu
  - Contoh: `print()`, `input()`, `eval()`, `int()`, `float()`, `len()`
- Manfaat fungsi
  - Membuat program lebih modular
  - Menyederhanakan program
  - Memudahkan pemeliharaan program
- Struktur fungsi pada Python

```
def nama_fungsi(parameters):
    ...
    ...
    ...
    return result
```

Contoh 1

```
def print_hello():
    print("Hello...")
    print("Selamat datang...")
    print("Apa kabar?")

print_hello()
```

- Pada contoh tersebut, program akan mencetak pesan "Hello…", "Selamat datang…", dan "Apa kabar?"

Contoh 2

```
def print_persegi_panjang():
    print('*' * 15)
    print('*', ' ' * 13, '*')
    print('*', ' ' * 13, '*')
    print('*' * 15)

print_persegi_panjang()
```

- Pada contoh tersebut, program akan mencetak sebuah persegi panjang yang sisinya dibatasi oleh karakter `'*'`

Variabel Lokal

Variabel Lokal

- Suatu fungsi dapat memiliki variabel di dalamnya
- Variabel di dalam fungsi bersifat lokal, artinya hanya dapat digunakan di dalam fungsi tersebut dan tidak dapat diakses dari luar fungsi
- Contoh:

```
def print_hello():
    nama = input("Masukkan nama Anda: ")
    print("Hello,", nama)
    print("Selamat datang...")
    print("Apa kabar?")

print_hello()
```

Variabel nama adalah variabel lokal yang hanya bisa diakses di dalam fungsi.

Contoh 3

```
def func1():
    x = 100
    y = x * 5
    print('x =', x, ' --- y =', y)

def func2():
    x = 1000
    y = x / 2
    print('x =', x, ' --- y =', y)

func1()
func2()
```

- Variabel x dan variabel y pada `func1()` berbeda dengan variabel x dan variabel y pada `func2()`
- Variabel x dan variabel y pada `func1()` tidak dapat diakses dari `func2()`, begitu pula sebaliknya

Argumen / Parameter Fungsi

Argumen/Parameter Fungsi

- Fungsi bisa menerima nilai yang dapat diolah di dalam fungsi tersebut
- Suatu nilai dikirim ke dalam fungsi melalui argumen/parameter
- Contoh: `print("Hello")`
  - `print` adalah sebuah fungsi dan `"Hello"` adalah nilai yang dikirim ke dalam fungsi `print`
  - `"Hello"` merupakan sebuah argumen/parameter
- Argumen/parameter bisa berupa variabel

```
pesan = "Hello"
print(pesan)
```

- Suatu fungsi bisa memiliki banyak parameter

```
def nama_fungsi(param1, param2, …):
    ...
    ...
    ...
    return result
```

Contoh 4

```
def print_hello(n):
    for i in range(n):
        print("Tamu", i + 1)
        nama = input("Masukkan nama Anda: ")
        print("Hello,", nama)
        print("Selamat datang...")
        print("Apa kabar?\n")

print_hello(3)
```

- Pada contoh tersebut, perulangan `for` yang ada di dalam fungsi `print_hello()` akan dijalankan sebanyak `n` kali

Return Value

Return untuk mengembalikan nilai

- Nilai hasil pengolahan suatu fungsi dapat dikembalikan untuk diterima oleh bagian program lain yang memanggil fungsi tersebut
- Contoh:

```
def tambah(a, b, c):
    hasil = a + b + c
    return hasil

x = tambah(5, 7, 9)
print(x)
```

Fungsi `tambah()` mengembalikan hasil penjumlahan tiga bilangan.

Perhatikan bahwa fungsi `tambah()` hanya melakukan perhitungan, tidak melakukan aksi seperti mencetak hasil perhitungan.
Bagian yang mencetak hasil perhitungan dilakukan di luar fungsi.

Fungsi mengembalikan beberapa nilai

- Suatu fungsi dapat mengembalikan beberapa nilai sekaligus
- Contoh:

```
def solve(a, b):
    hasil1 = a + b
    hasil2 = a - b
    return hasil1, hasil2

x, y = solve(10, 7)
print(x)
print(y)
```

Fungsi `solve()` melakukan dua perhitungan, yaitu menghitung penjumlahan dua bilangan dan menghitung pengurangan dua bilangan.

Saat dipanggil, sesuai urutan nilai yang dikembalikan, `x` akan berasosiasi dengan `hasil1` dan `y` akan berasosiasi dengan `hasil2`.

Fungsi mengembalikan nilai dengan alternatif

- Jika fungsi memiliki alternatif nilai kembalian, return dapat dilakukan di beberapa bagian alternatif fungsi tersebut
  - Pastikan setiap alternatif fungsi tersebut sudah memiliki return masing-masing
- Contoh:

```
def max(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

terbesar = max(5, -3, 8)
print("Bilangan terbesar =", terbesar)
```

Fungsi `max()` mengembalikan bilangan terbesar di antara tiga bilangan.

Contoh 5

```
def is_ganjil(n):
    if n % 2 == 1:
        return True
    else:
        return False

x = 9
if is_ganjil(x):
    print("ganjil")
else:
    print("genap")
```

```
def is_ganjil(n):
    hasil = n % 2 == 1
    return hasil

x = 9
if is_ganjil(x):
    print("ganjil")
else:
    print("genap")
```

```
def is_ganjil(n):
    return n % 2 == 1

x = 9
if is_ganjil(x):
    print("ganjil")
else:
    print("genap")
```

- Fungsi `is_ganjil()` mengembalikan `True` jika n merupakan bilangan ganjil dan mengembalikan `False` jika n bukan bilangan ganjil
- Ketiga penulisan fungsi `is_ganjil()` di atas melakukan hal yang sama

Return untuk menghentikan fungsi

- Return dapat digunakan untuk menghentikan fungsi lebih awal
  - Konsepnya mirip dengan percabangan
  - Untuk beberapa kasus, penggunaan return untuk menghentikan fungsi dapat membantu penulisan fungsi menjadi lebih sederhana dan mudah dibaca
- Contoh

```
def cetak_bilangan(n):
    if n < 0 or n > 10:
        return

    for i in range(n):
        print(i)
```

Fungsi `cetak_bilangan()` akan mencetak bilangan dari `0` – n hanya jika nilai n berada pada range `1` - `10`.

Default Arguments & Keyword Arguments

Parameter default

- Suatu fungsi bisa memiliki parameter dengan nilai default
  - Nilai default membuat parameter tersebut bersifat optional, dapat digunakan ataupun tidak
  - Jika tidak digunakan, nilai parameter tersebut akan diambil dari nilai default yang ditentukan
- Dalam deklarasi fungsi, parameter default harus ditulis di belakang, setelah semua parameter non-default

```
def max(a, b, c=-999):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

terbesar = max(5, 9)
print("Bilangan terbesar =", terbesar)
```

Fungsi `max()` memiliki parameter `c` yang nilainya default sebesar `-999`.

Dengan asumsi nilai yang dibandingkan tidak kurang dari `-999`, fungsi `max()` dapat digunakan untuk mengembalikan nilai terbesar di antara dua bilangan.

Parameter dengan keyword

- Terkadang, suatu fungsi memiliki banyak parameter sehingga sulit menghafal urutannya.
- Sebagai alternatif, parameter fungsi dapat dipanggil berdasarkan nama (*keyword*)
  - Parameter yang dipanggil berdasarkan keyword harus disebut setelah parameter non-keyword
  - Parameter yang dipanggil dengan keyword dapat ditukar urutannya
  - Penting untuk memberi nama parameter yang representatif
- Contoh

```
def cetak(teks, batas, kapital):
    hasil = teks[:batas]
    if kapital:
        hasil = hasil.upper()
    print(hasil)

cetak("Selamat datang", kapital=True, batas=7)
```

Fungsi `cetak()` mencetak suatu teks sampai dari awal sampai batas tertentu, dan diubah menjadi huruf kapital semua jika diinginkan.

Parameter fungsi `cetak()`:
`teks`: teks yang akan dicetak
`batas`: indeks batas teks yang akan dicetak (`int`)
`kapital`: apakah teks ingin dicetak kapital atau tidak (`bool`)

Selamat Belajar ...!!