UNIVERSITAS
PENDIDIKAN INDONESIA

Rekayasa
Perangkat
Lunak

Python Programming

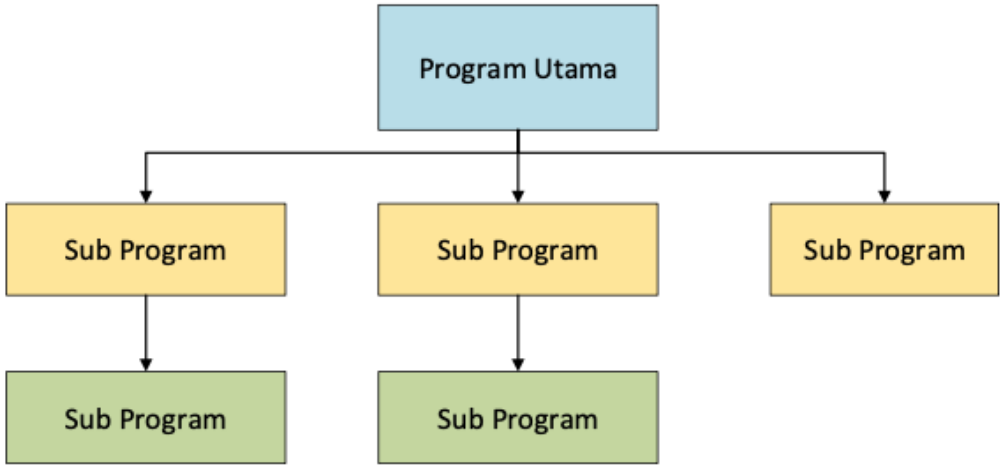
Function

01

Function dan Procedure

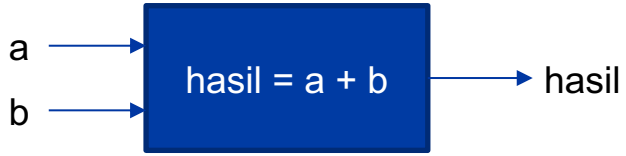
Konsep Dasar

- Fungsi dan Prosedur adalah bagian kecil yang berada dalam program utama yang digunakan untuk menyelesaikan masalah khusus dengan parameter yang diberikan.



Function

- Fungsi merupakan proses diantara sebuah *input* dan *output*



- Fungsi atau *function* di Python merupakan melakukan organisasi kode agar kode bisa digunakan tanpa perlu dideklarasikan kembali.
- Fungsi adalah blok kode yang dapat menerima argumen (input) dan **mengembalikan hasil (output).**
- Fungsi biasanya digunakan untuk melakukan komputasi atau perhitungan berdasarkan argumen yang diberikan dan mengembalikan hasil dari perhitungan tersebut.
- Fungsi menggunakan pernyataan **"return"** untuk mengembalikan nilai ke pemanggilnya.

Procedure

- Prosedur adalah blok kode yang juga dapat menerima argumen (input), tetapi **tidak mengembalikan nilai.**
- Prosedur digunakan untuk mengeksekusi tindakan tertentu tanpa perlu mengembalikan hasil yang dapat digunakan dalam ekspresi lain.
- Biasanya, prosedur hanya menggunakan pernyataan "print" atau melakukan perubahan pada variabel atau objek, tetapi tidak mengembalikan nilai.
- Pada Python, penulisan sintak fungsi dan prosedur Adak ada perbedaan
- Python tidak memiliki perbedaan formal antara fungsi dan prosedur seperti bahasa pemrograman lainnya, tetapi kita dapat membedakannya berdasarkan apakah mereka mengembalikan nilai atau tidak.

Function dan Procedure

```python
# Contoh Function
def penjumlahan(a, b):
    hasil = a + b
    return hasil

penjumlahan(2, 3)
```

✓ 0.0s

```
5
```

```python
# Contoh Procedure
def greeting(nama):
    print(f"Halo, {nama}!")

greeting("Sasa")
```

✓ 0.0s

```
Halo, Sasa!
```

Jangkauan Variabel

- Jangkauan variabel meliputi bagian-bagian program dimana sebuah variabel masih bisa diakses.
- **Variabel lokal:** variable yang dideklarasikan di dalam fungsi.
- **Variable global:** variabel yang dideklarasikan di luar fungsi dan ditempatkan di atas semua fungsi dalam suatu program.
- Jangkauan pada variable global meliputi seluruh program.
- Variabel yang dideklarasikan secara global, dapat dideklarasikan kembali (redeclared) di subprogram
- Namun disarankan tidak terlalu banyak menggunakan variabel global karena memungkinkan terjadinya error jika program semakin besar, sulit melacak kesalahan, dan data tidak terjaga dengan baik.

Contoh Jangkauan Variabel

```python
# Contoh Global Variabel

globalVar = 10

def globalFunction():
    print(f"Nilai dari global variabel adalah: {globalVar}")

globalFunction()
print(globalVar)
```

✓ 0.0s

```
Nilai dari global variabel adalah: 10
10
```

```python
# Contoh Lokal Variabel

def lokalFunction():
    lokalVar = 5
    print(f"Nilai dari lokal variabel adalah: {lokalVar}")

lokalFunction()
# print(lokalVar)    akan menghasilkan error
```

✓ 0.0s

```
Nilai dari lokal variabel adalah: 5
```

Membuat Function

- Berikut adalah contoh simple untuk membuat *function* di Python
- Untuk memaggil function kita menggunakan `def nama_function` tersebut diikuti dengan tanda kurung buka tutup ()

```
1 def funcPertama(): # membuat function
2     print("Perintah ini menggunakan function")
3
4 funcPertama() # memanggil function
```

```
Perintah ini menggunakan function
```

- Posisi parameter akan sesuai dengan urutan saat function dibuat
- Untuk memanggil parameter tersebut juga harus dilakukan sesuai dengan urutannya

Membuat Function

- Dengan function kita tidak perlu mendeklarasikan kode berulang kali
- Argumen adalah informasi yang diteruskan ke dalam fungsi yang ditentukan setelah nama fungsi dan ditulis dalam tanda kurung ()
- Pada sebuah function, kita bisa menambahkan sebanyak apapun argument yang diperlukan
- Setiap argument dipisahkan oleh tanda koma
- nama merupakan argument yang digunakan untuk function greeting

```
1  def greeting(nama):
2      print(f"Halo, {nama}!")
3
4  greeting("Shafira")
5  greeting("Novia")
```

```
Halo, Shafira!
Halo, Novia!
```

Arguments

- Karena susunan argument harus sesuai dengan yang dideklarasikan saat pertama membuat function, maka kita harus memperhatikannya
- Pada contoh berikut susunan antara (5, 2) dan (2,5) akan menghasilkan nilai yang berbeda hal ini disebabkan karena dalam function pengurangan, argument pertama (a) yang dikurang oleh argument kedua (b)

```
1  def pengurangan(a, b):
2      print(a - b)
3
4  pengurangan(5, 2)
5  pengurangan(2, 5)
```

```
3
-3
```

Arguments

- Baik argument maupun parameter memiliki arti yang sama, yaitu informasi yang diteruskan ke dalam suatu function
- Dari sudut pandang function, parameter adalah variable yang tercantum dalam tanda kurung dalam definisi function.
- Sedangkan argument merupakan nilai yang dikirim ke function saat dipanggil

```
1  def kelilingSegitiga(a, b, c):
2      print(a + b + c)
3
4  kelilingSegitiga(1, 2, 3)
```

```
6
```

Arguments

- Sebuah function yang dipanggil harus sesuai jumlah argument-nya. Tidak boleh lebih dan tidak boleh kurang
- Jika jumlah argument tidak sesuai, maka akan menghasilkan *error*

```
1 def kelilingSegitiga(a, b, c):
2     print(a + b + c)
3
4 kelilingSegitiga(1, 2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 4
      1 def kelilingSegitiga(a, b, c):
      2     print(a + b + c)
----> 4 kelilingSegitiga(1, 2)

TypeError: kelilingSegitiga() missing 1 required positional argument: 'c'
```

Default Parameter Value

- Default parameter adalah cara untuk membuat nilai dari suatu parameter agar ketika kita memanggil function tanpa argument, function tersebut akan menggunakan nilai bawaannya *(default value)*

```python
def greeting(nama = "Kuro"):
    print(f"Halo, {nama}!")

greeting()
greeting("Shafira")
greeting("Novia")
greeting("Bella")
```

```
Halo, Kuro!
Halo, Shafira!
Halo, Novia!
Halo, Bella!
```

Keyword Arguments

- Penulisan argument juga bisa dibuat dengan format key = value
- Apabila menggunakan format ini, maka susunan penulisan argument tidak akan dipermasalahkan lagi

```python
def pengurangan(a, b):
    print(f"Hasilnya adalah {a - b}")

# menggunakan keyword arguments
pengurangan(b = 2, a = 5)

# tanpa keyword arguments
pengurangan(2, 5)
```

```
Hasilnya adalah 3
Hasilnya adalah -3
```

Arbitrary Arguments

- Apabila jumlah argument yang harus diteruskan di suatu function tidak/belum diketahui, maka kita dapat menambahkan tanda bintang (*) sebelum nama parameter
- Ini disebut dengan arbitrary arguments atau disingkat sebagai *args

```python
def funcArgs(*angka):
    print(f"Angka terakhir yang dimasukkan yaitu {angka[-1]}")

funcArgs(1, 2, 8, 9, 4, 5)
```

```
Angka terakhir yang dimasukkan yaitu 5
```

```python
def funcArgs(*angka):
    print(f"Angka yang dimasukkan yaitu {angka[:]}")

funcArgs(1, 2, 8, 9, 4, 5)
```

```
Angka yang dimasukkan yaitu (1, 2, 8, 9, 4, 5)
```

Arbitrary Keyword Arguments

- Apabila jumlah keyword argument yang harus diteruskan di suatu function tidak/belum diketahui, maka kita dapat menambahkan dua tanda bintang (**) sebelum nama parameter
- Ini disebut dengan arbitrary keyword arguments atau disingkat sebagai **kwargs

```
1  def funcKwargs(**angka):
2      print(f"Angka ketiga yaitu {angka['ketiga']}")
3
4  funcKwargs(pertama = 1, kedua = 2, ketiga = 3, keempat = 4)
```

✓ 0.0s

```
Angka ketiga yaitu 3
```

Argument from List

- Kita juga bisa menggunakan sebuah list untuk menjadi argument dari suatu function
- Berikut merupakan contoh aplikasi dari membuat argument dari list

```
1  def namaBuah(buah):
2      for i in buah:
3          print(i)
4
5  buah = ["apel", "anggur", "jeruk", "mangga"]
6
7  namaBuah(buah)
```

```
apel
anggur
jeruk
mangga
```

Return

- Return merupakan pernyataan yang akan mengeksekusi program keluar dari function saat itu dan mengembalikan nilai tertentu
- Apabila kita ingin membuat function yang mengembalikan nilai, maka kita berlu membuat statement return

```
1  def luasPersegi(sisi):
2      sisi * sisi
3
4  # tanpa statement return
5  print(luasPersegi(5))
```

```
None
```

```
1  def luasPersegi(sisi):
2      return sisi * sisi
3
4  # dengan statement return
5  print(luasPersegi(5))
```

```
25
```

```
1  def luasPersegi(sisi):
2      luas = sisi * sisi
3      return luas
4
5  # dengan statement return
6  print(luasPersegi(5))
```

```
25
```

Pass

- Pass digunakan apabila function belum selesai atau kosong dan untuk menghindari error

```
  1  def kosong():
```

0.0s

```
  Cell In[19], line 1
ref='  vscode-notebook-cell:?execution_count=19&line=0'>0</a>;31m    def kosong():
ref='  vscode-notebook-cell:?execution_count=19&line=0'>0</a>m                ^
ref='  vscode-notebook-cell:?execution_count=19&line=0'>0</a>;31mIndentationError: expected an indented block
```

```
  1  def kosong():
  2      pass
```

0.0s

02

Studi Kasus

Studi Kasus

Buatlah program menggunakan bahasa pemrograman Python untuk:

1. Deret Fibonacci dengan menggunakan fungsi
2. Menghitung volume tabung yang dibuat dalam sebuah fungsi
3. Menghitung fungsi nilai total dan nilai rata-ratanya berdasarkan nilai yang diinputkan dari nilai total dimana banyaknya angka tidak dideklarasikan. Contoh:

   Input: 2, 3, 5, dan 10

   Total: 2 + 3 + 5 + 10 = 20

   Rata-rata = 20 / 5 = 4

Studi Kasus

4. Pak Dedi merupakan seorang Kepala Laboratorium Komputer di SMAN 2 Harapan. Dia ingin membuat sebuah menu login yang dapat memberi kesempatan user untuk memasukkan password kembali ketika dia salah sebanyak 3 kali. Ketika user terus memasukkan password yang salah, maka user tersebut akan keluar dari menu login tersebut. Pak budi juga menambahkan bahwa password ini harus sudah ada dalam sistem yang di buat, sehingga sistem itu hanya mengecek password saja tanpa memperdulikan username.

   Login dengan 3 kesempatan Admin

   username : Daspro2023

   password : Latihan

Studi Kasus

5. Aira diminta guru Olahraganya untuk menghitung waktu teman-temannya mulai berlari dan selesai berlari. Ia juga diminta untuk menghitung selisih waktu yang dia buat dan menuliskan hasilnya ke dalam format Jam - Menit - Detik. Bantulah Aira untuk menyelesaikan masalah tersebut agar dapat dilakukan dengan cepat.

   Hitung selisih jam - menit - detik .

   Input mulai:

   Jam: 08

   Menit: 10

   Detik: 20

   Input selesai:

   Jam: 09

   Menit: 15

   Detik: 30

   Hitung selisih:

   Output: 1 jam - 5 menit - 10 detik

UNIVERSITAS
PENDIDIKAN INDONESIA

Rekayasa
Perangkat
Lunak

Terimakasih