

Minggu Ke 9

Fungsi

Fungsi

Fungsi adalah grup/blok program untuk melakukan tugas tertentu yang berulang. Fungsi membuat kode program menjadi reusable, artinya hanya di definisikan sekali saja, dan kemudian bisa digunakan berulang kali dari tempat lain di dalam program.

Fungsi memecah keseluruhan program menjadi bagian – bagian yang lebih kecil . Dengan semakin besarnya program, maka fungsi akan membuatnya menjadi lebih mudah diorganisir dan dimanage.

Sejauh ini, kita sudah menggunakan beberapa fungsi, misalnya fungsi `print()`, `type()`, dan sebagainya. Fungsi tersebut adalah fungsi bawaan dari Python. Kita bisa membuat fungsi kita sendiri sesuai kebutuhan.

Mendefinisikan Fungsi

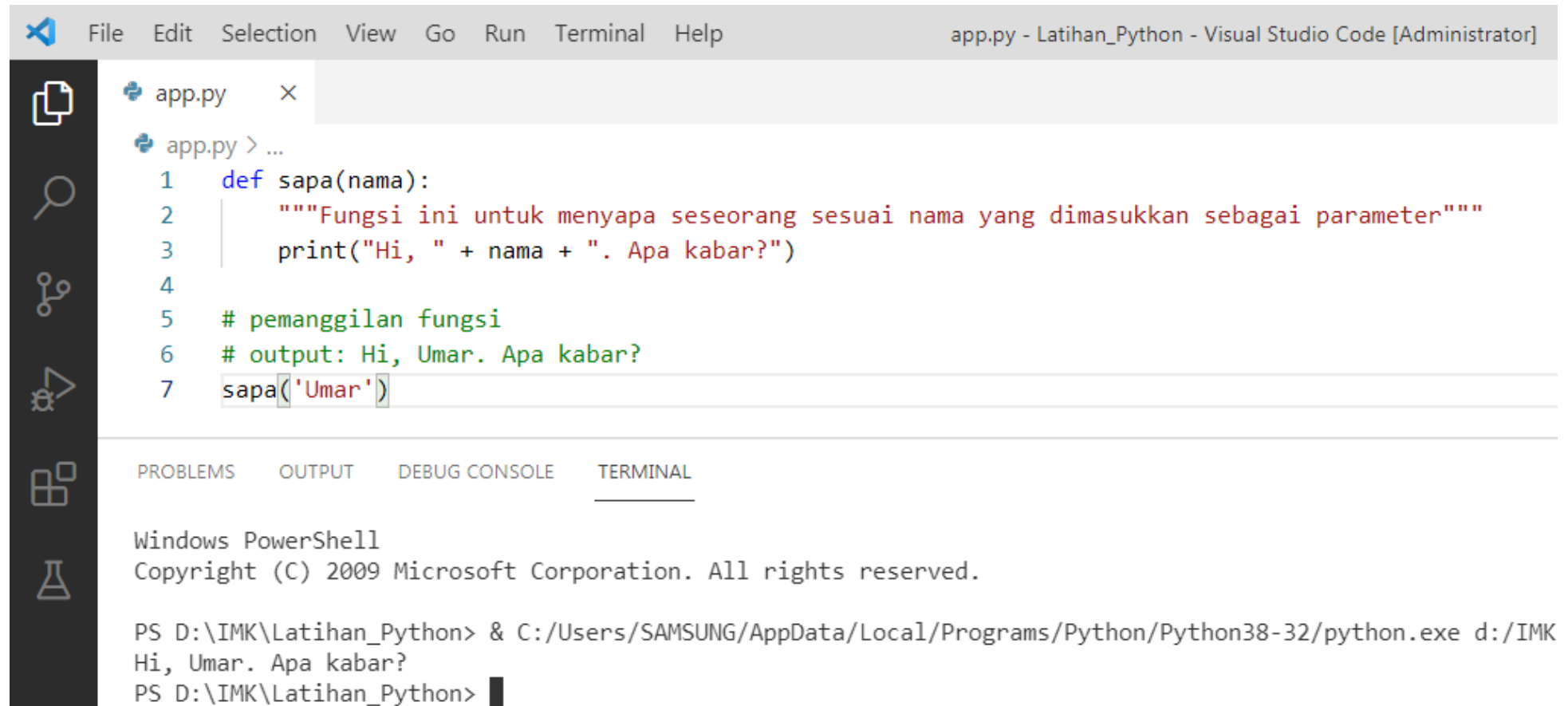
Berikut adalah sintaks yang digunakan untuk membuat fungsi:

```
def function_name(parameters):  
    """function_docstring"""  
    statement(s)  
  
    return [expression]
```

Penjelasannya dari sintaks fungsi di atas:

1. Kata kunci `def` diikuti oleh `function_name` (nama fungsi), tanda kurung dan tanda titik dua (`:`) menandai header (kepala) fungsi.
2. Parameter / argumen adalah input dari luar yang akan diproses di dalam tubuh fungsi.
3. "`function_docstring`" bersifat opsional, yaitu sebagai string yang digunakan untuk dokumentasi atau penjelasan fungsi. "`function_docstring`" diletakkan paling atas setelah baris `def`.
4. Setelah itu diletakkan baris – baris pernyataan (statements). Jangan lupa indentasi untuk menandai blok fungsi.
5. `return` bersifat opsional. Gunanya adalah untuk mengembalikan suatu nilai expression dari fungsi.

Berikut adalah contoh fungsi untuk menyapa seseorang.



```

File Edit Selection View Go Run Terminal Help
app.py - Latihan_Python - Visual Studio Code [Administrator]

app.py x
app.py > ...
1 def sapa(nama):
2     """Fungsi ini untuk menyapa seseorang sesuai nama yang dimasukkan sebagai parameter"""
3     print("Hi, " + nama + ". Apa kabar?")
4
5 # pemanggilan fungsi
6 # output: Hi, Umar. Apa kabar?
7 sapa('Umar')

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs/Python/Python38-32/python.exe d:/IMK
Hi, Umar. Apa kabar?
PS D:\IMK\Latihan_Python>

```

Memanggil Fungsi

Bila fungsi sudah didefinisikan, maka ia sudah bisa dipanggil dari tempat lain di dalam program. Untuk memanggil fungsi caranya adalah dengan mengetikkan nama fungsi berikut paramaternya. Untuk fungsi di atas, kita bisa melakukannya seperti contoh berikut:

```
>>> sapa('Galih')
```

```
Hi, Galih. Apa kabar?
```

```
>>> sapa('Ratna') Hi, Ratna. Apa  
kabar?
```

Docstring

Docstring adalah singkatan dari documentation string. Ini berfungsi sebagai dokumentasi atau keterangan singkat tentang fungsi yang kita buat. Meskipun bersifat opsional, menuliskan docstring adalah kebiasaan yang baik. Untuk contoh di atas kita menuliskan docstring. Cara mengaksesnya adalah dengan menggunakan format `namafungsi.__doc__`

```
>>> print(sapa.__doc__)
```

"""Fungsi ini untuk menyapa seseorang sesuai
nama yang dimasukkan sebagai
parameter"""

Pernyataan Return

Pernyataan return digunakan untuk keluar dari fungsi dan kembali ke baris selanjutnya dimana fungsi dipanggil.

Adapun sintaks dari return adalah:

```
return [expression_list]
```

return bisa berisi satu atau beberapa ekspresi atau nilai yang dievaluasi dan nilai tersebut akan dikembalikan. Bila tidak ada pernyataan return yang dibuat atau ekspresi dikosongkan, maka fungsi akan mengembalikan objek None. Perhatikan bila hasil keluaran dari fungsi sapa kita simpan dalam variabel.

```
>>> keluaran = sapa('Gani')  
>>> print(keluaran) None
```

Argumen Fungsi

Kita bisa memanggil fungsi dengan menggunakan salah satu dari empat jenis argumen berikut:

- Argumen wajib (required argument)

Argumen wajib adalah argumen yang dilewatkan ke dalam fungsi dengan urutan posisi yang benar. Di sini, jumlah argumen pada saat pemanggilan fungsi harus sama persis dengan jumlah argumen pada pendefinisian fungsi. Pada contoh fungsi sapa() di atas, kita perlu melewatkan satu argumen ke dalam fungsi sapa(). Bila tidak, maka akan muncul error.

```
>>> sapa('Umar')
```

```
Hi Umar. Apa kabar?
```

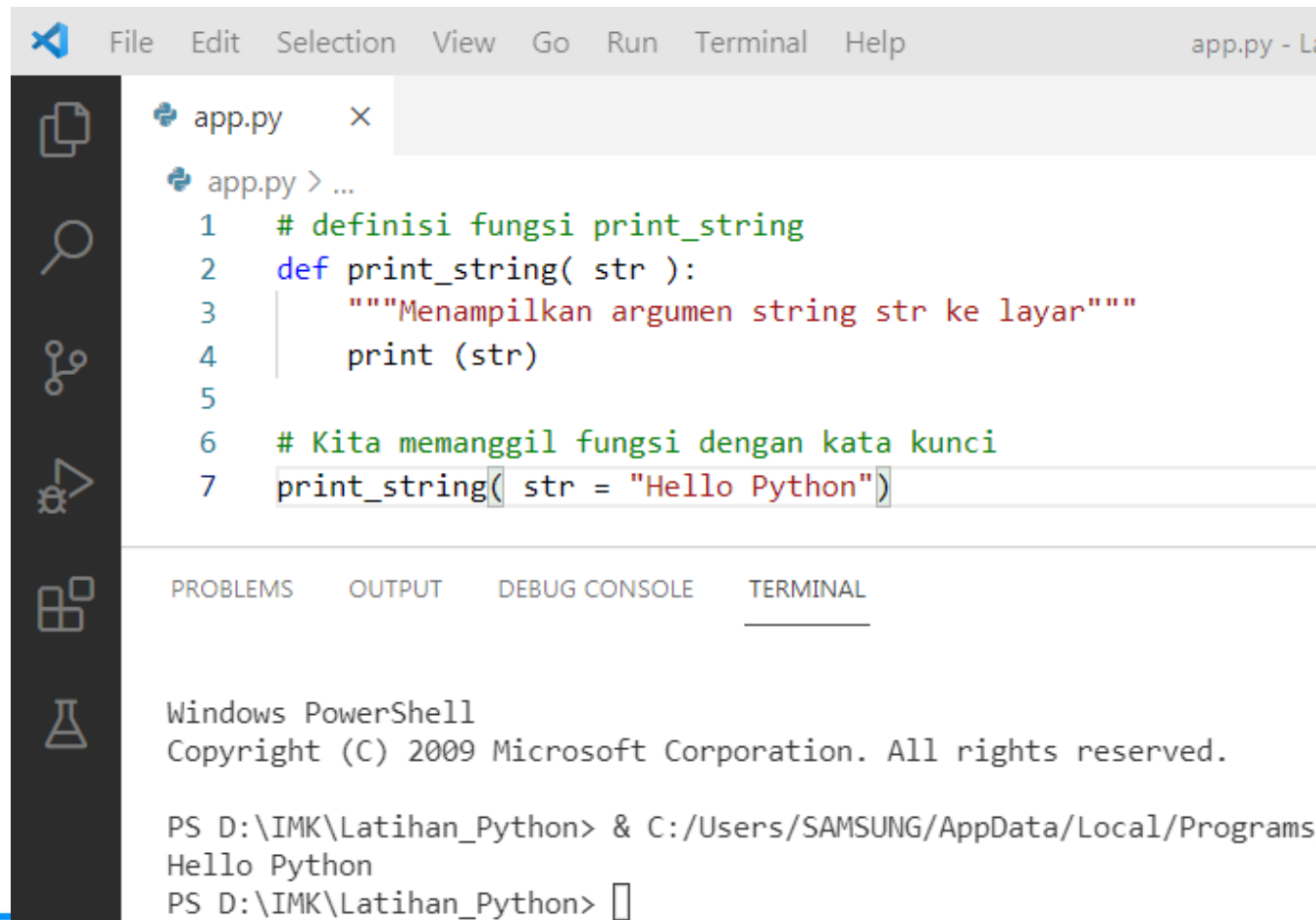
```
>>> # akan muncul error
```

```
>>> sapa()
```

```
Traceback (most recent call last):  
File "<pyshell#5>", line 1, in  
<module>  
    sapa()  
TypeError: sapa() missing 1 required  
positional argument: 'nama'
```


- Argumen kata kunci (keyword argument)

Argumen dengan kata kunci berkaitan dengan cara pemanggilan fungsi. Ketika menggunakan argumen dengan kata kunci, fungsi pemanggil menentukan argumen dari nama parameter. Hal ini membuat kita bisa mengabaikan argumen atau menempatkannya dengan sembarang urutan. Python dapat menggunakan kata kunci yang disediakan untuk mencocokkan nilai sesuai dengan parameter. Jelasnya ada pada contoh berikut:

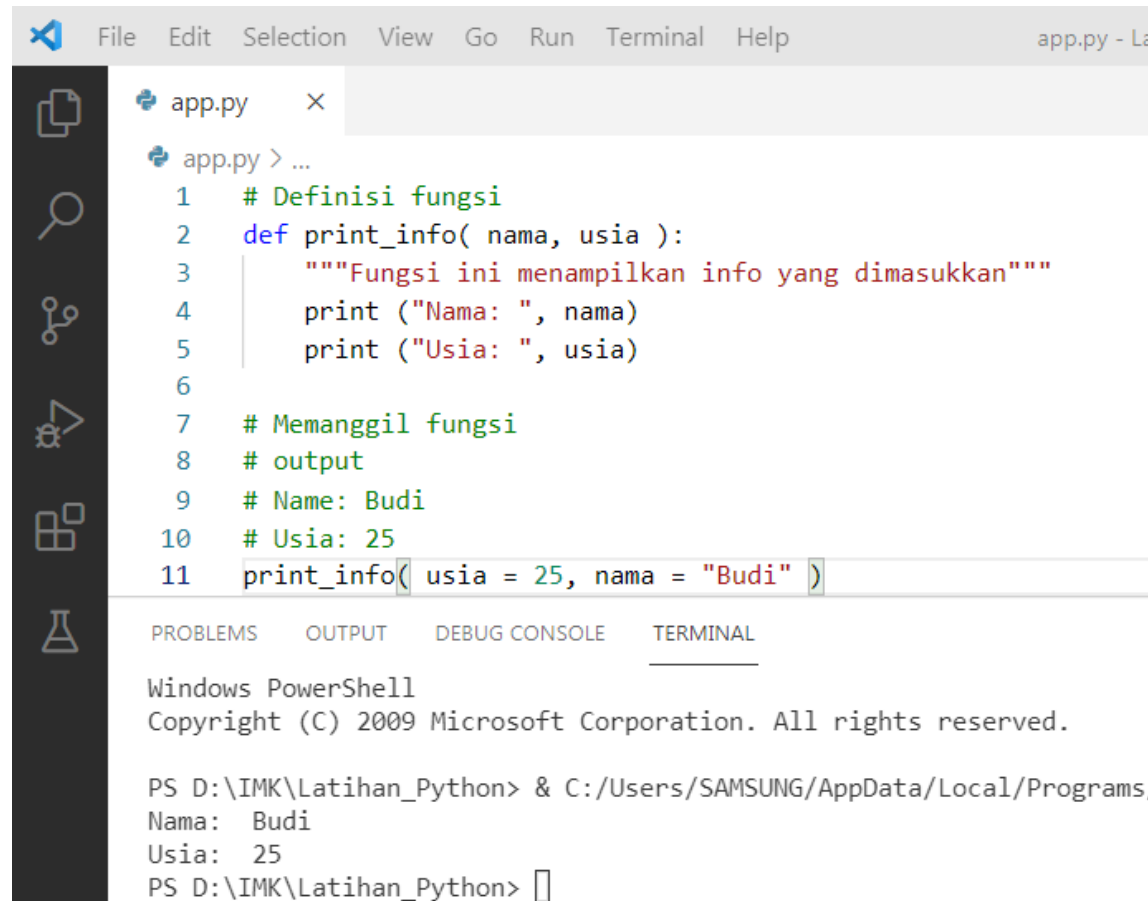


```
File Edit Selection View Go Run Terminal Help app.py - La
app.py x
app.py > ...
1 # definisi fungsi print_string
2 def print_string( str ):
3     """Menampilkan argumen string str ke layar"""
4     print (str)
5
6 # Kita memanggil fungsi dengan kata kunci
7 print_string( str = "Hello Python")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs
Hello Python
PS D:\IMK\Latihan_Python> 
```


Urutan parameter tidak menjadi masalah. Perhatikan contoh berikut:



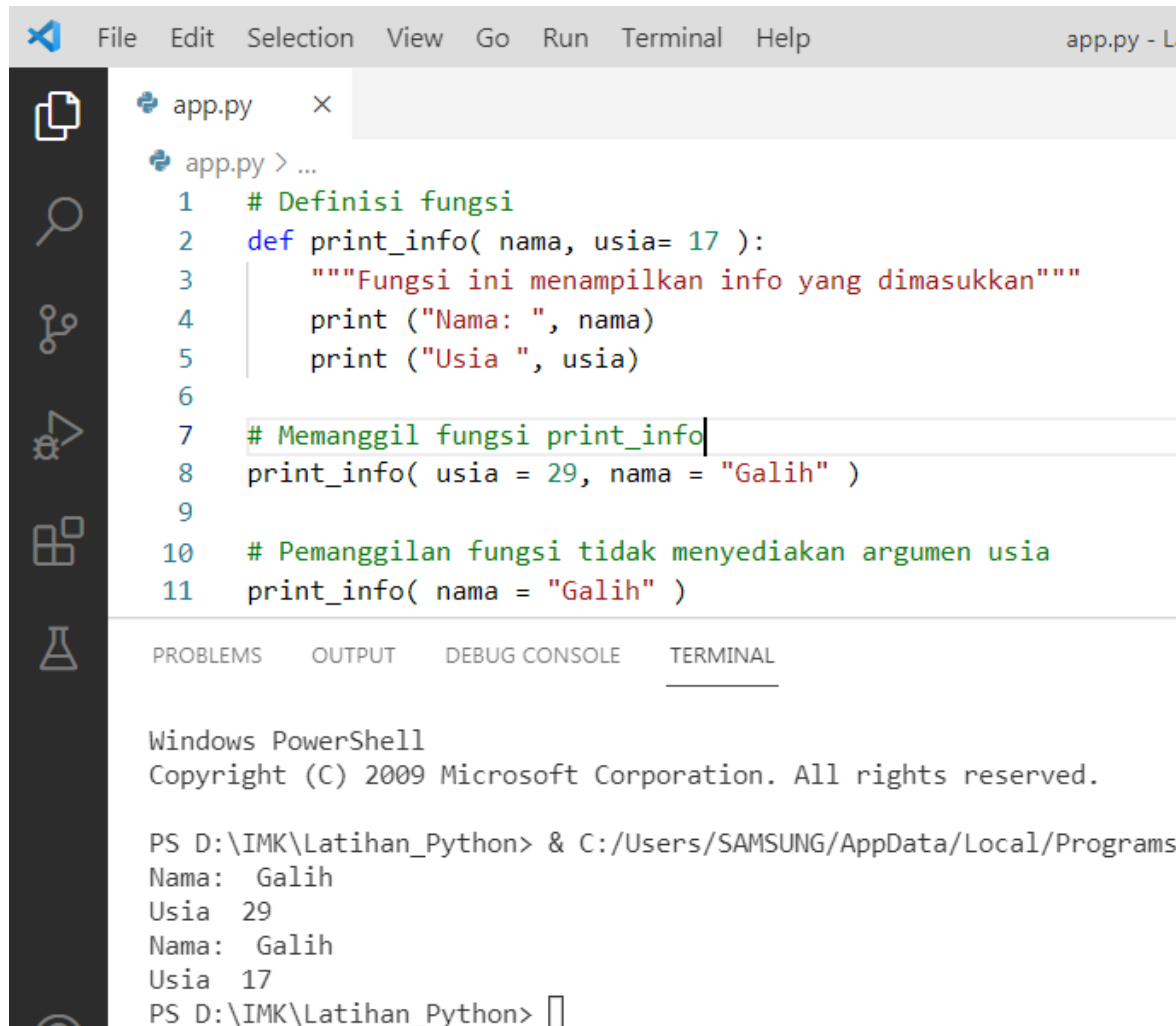
```
File Edit Selection View Go Run Terminal Help app.py - La
app.py x
app.py > ...
1 # Definisi fungsi
2 def print_info( nama, usia ):
3     """Fungsi ini menampilkan info yang dimasukkan"""
4     print ("Nama: ", nama)
5     print ("Usia: ", usia)
6
7 # Memanggil fungsi
8 # output
9 # Name: Budi
10 # Usia: 25
11 print_info( usia = 25, nama = "Budi" )

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs,
Nama: Budi
Usia: 25
PS D:\IMK\Latihan_Python> 
```

- Argumen default

Fungsi dengan argumen default menggunakan nilai default untuk argumen yang tidak diberikan nilainya pada saat pemanggilan fungsi. Pada contoh berikut, fungsi akan menampilkan usia default bila argumen usia tidak diberikan:



```

File Edit Selection View Go Run Terminal Help
app.py - L
app.py x
app.py > ...
1  # Definisi fungsi
2  def print_info( nama, usia= 17 ):
3      """Fungsi ini menampilkan info yang dimasukkan"""
4      print ("Nama: ", nama)
5      print ("Usia ", usia)
6
7  # Memanggil fungsi print_info
8  print_info( usia = 29, nama = "Galih" )
9
10 # Pemanggilan fungsi tidak menyediakan argumen usia
11 print_info( nama = "Galih" )

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs
Nama: Galih
Usia 29
Nama: Galih
Usia 17
PS D:\IMK\Latihan_Python>

```

Pada contoh di atas, pemanggilan fungsi kedua tidak menyediakan nilai untuk parameter usia, sehingga yang digunakan adalah nilai default yaitu 17.

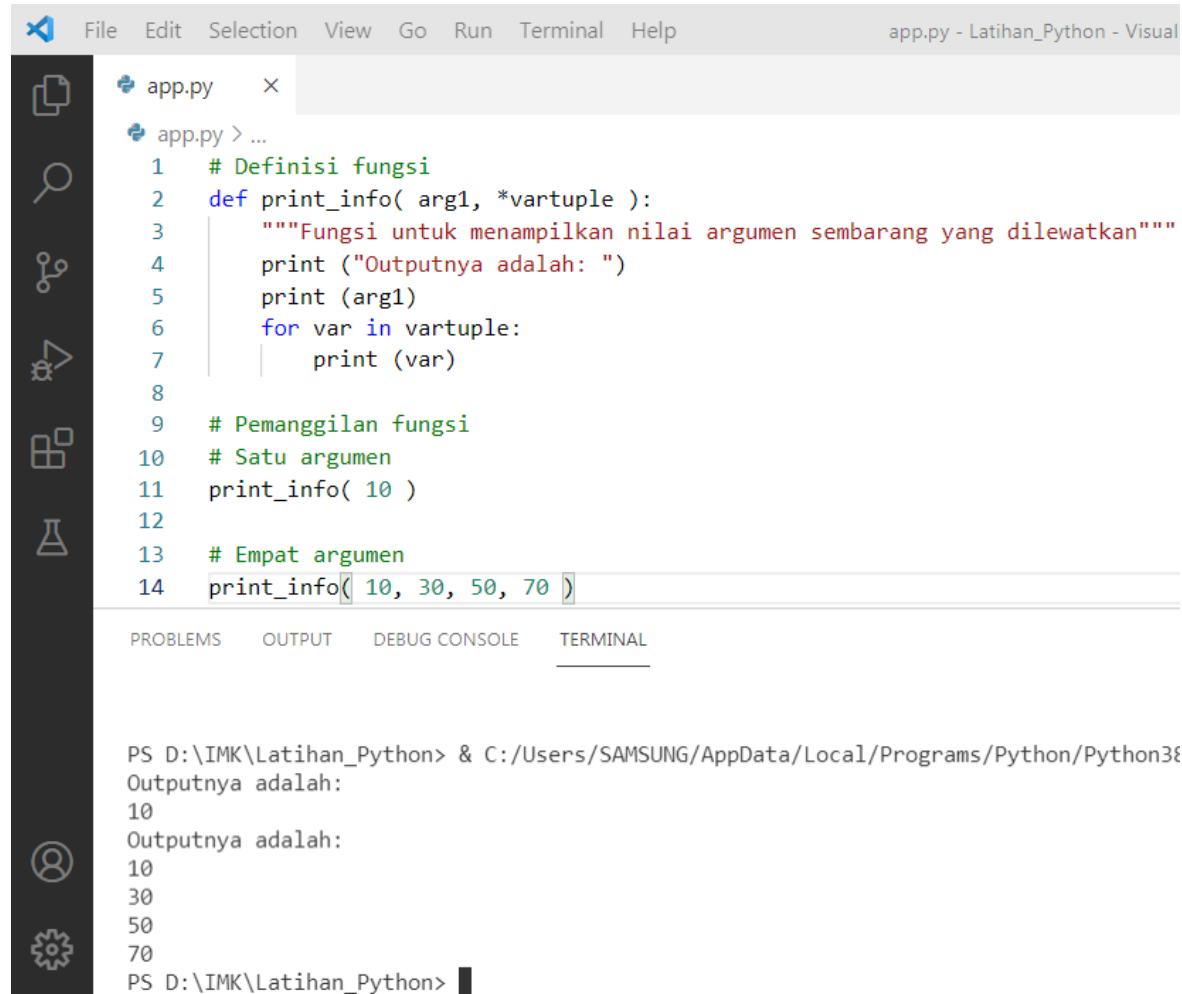
- Argumen dengan panjang sembarang

Terkadang kita butuh untuk memproses fungsi yang memiliki banyak argumen. Nama – nama argumennya tidak disebutkan saat pendefinisian fungsi, beda halnya dengan fungsi dengan argumen wajib dan argumen default. Sintaksnya fungsi dengan argumen panjang sembarang adalah seperti berikut:

```
def function_name([formal_args,]  
*var_args_tuple):  
    """function_docstring"""  
    statement(s)
```

```
        .  
        return [expression]
```

Tanda asterisk (*) ditempatkan sebelum nama variabel yang menyimpan nilai dari semua argumen yang tidak didefinisikan. Tuple ini akan kosong bila tidak ada argumen tambahan pada saat pemanggilan fungsi. Berikut adalah contohnya:



```

app.py
app.py > ...
1  # Definisi fungsi
2  def print_info( arg1, *vartuple ):
3      """Fungsi untuk menampilkan nilai argumen sembarang yang dilewatkan"""
4      print ("Outputnya adalah: ")
5      print (arg1)
6      for var in vartuple:
7          print (var)
8
9  # Pemanggilan fungsi
10 # Satu argumen
11 print_info( 10 )
12
13 # Empat argumen
14 print_info( 10, 30, 50, 70 )

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs/Python/Python38
Outputnya adalah:
10
Outputnya adalah:
10
30
50
70
PS D:\IMK\Latihan_Python>

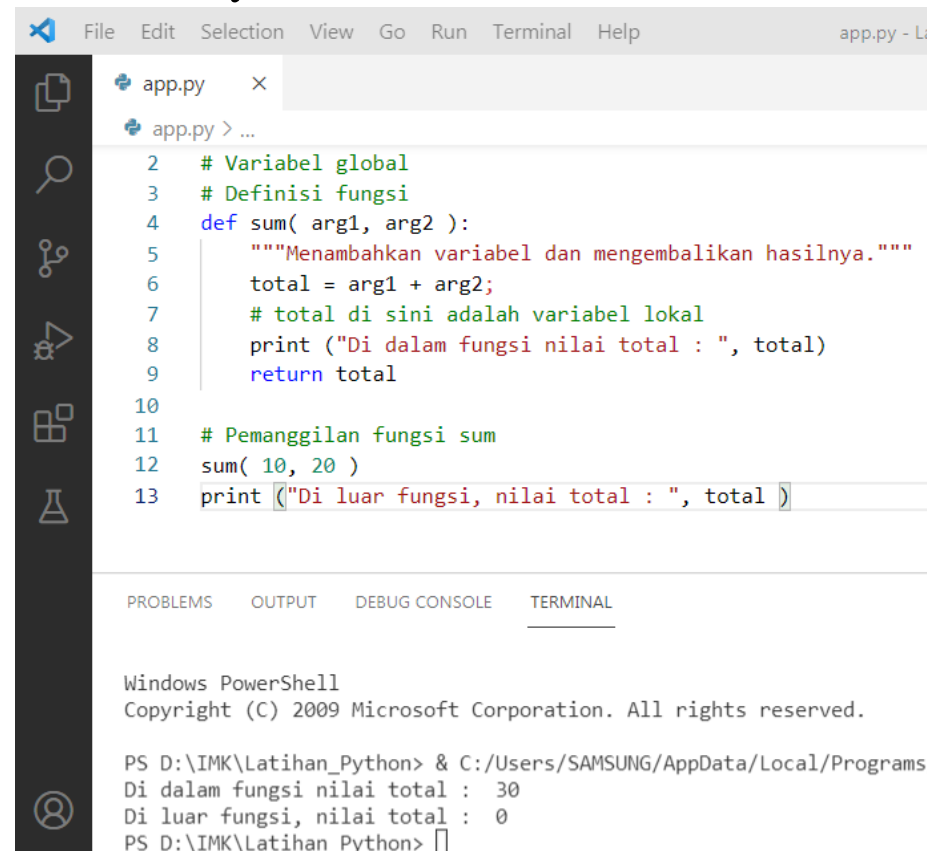
```

Ruang Lingkup (Scope) Variabel

Di Python, tidak semua variabel bisa diakses dari semua tempat. Ini tergantung dari tempat dimana kita mendefinisikan variabel. Ruang lingkup variabel ada dua, yaitu:

- Global
- Local

Variabel yang didefinisikan di dalam fungsi memiliki scope lokal, sedangkan variabel yang didefinisikan di luar fungsi memiliki scope global. Ini berarti, variabel lokal hanya bisa diakses dari dalam fungsi di mana ia di definisikan, sedangkan variabel global bisa diakses dari seluruh tempat dimanapun di dalam program. Berikut adalah contohnya:



```

File Edit Selection View Go Run Terminal Help
app.py - L
app.py x
app.py > ...
2 # Variabel global
3 # Definisi fungsi
4 def sum( arg1, arg2 ):
5     """Menambahkan variabel dan mengembalikan hasilnya."""
6     total = arg1 + arg2;
7     # total di sini adalah variabel lokal
8     print ("Di dalam fungsi nilai total : ", total)
9     return total
10
11 # Pemanggilan fungsi sum
12 sum( 10, 20 )
13 print ("Di luar fungsi, nilai total : ", total )

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs
Di dalam fungsi nilai total : 30
Di luar fungsi, nilai total : 0
PS D:\IMK\Latihan_Python>

```

Perhatikan bagaimana variabel total di dalam dan di luar fungsi adalah dua variabel yang berbeda.