



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230971
Nama Lengkap	James Marvin Santoso
Minggu ke / Materi	04 / Modular Programming

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI

Fungsi, Argument dan Parameter :

- Fungsi :

Fungsi dalam Python adalah blok kode yang dapat dipanggil untuk melakukan tugas tertentu. Fungsi dapat menerima argumen sebagai input, melakukan operasi tertentu, dan mengembalikan nilai. Fungsi digunakan untuk mengorganisir kode menjadi unit yang lebih kecil dan dapat digunakan ulang. Sebuah fungsi dalam Python didefinisikan menggunakan kata kunci `def`, diikuti oleh nama fungsi dan parameter dalam tanda kurung.

Contoh sederhana fungsi yang menambahkan dua angka, antara lain :

```
contoh.py > ...
1  def tambah(a, b):
2      return a + b
3
4  hasil = tambah(3, 5)
5  print("Hasil penjumlahan:", hasil)]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Hasil penjumlahan: 8
PS C:\Tugas>
```

Di sini, fungsi `tambah` menerima dua parameter `a` dan `b`, kemudian mengembalikan hasil penjumlahan keduanya. Saat fungsi dipanggil dengan `tambah(3, 5)`, ia mengembalikan 8, yang kemudian disimpan dalam variabel `hasil` dan dicetak.

- Argument :

Argument dalam Python merujuk pada nilai yang diberikan kepada sebuah fungsi saat fungsi tersebut dipanggil. Argumen digunakan untuk memberikan input kepada fungsi sehingga fungsi dapat bekerja dengan nilai yang diberikan.

Ada beberapa jenis argumen dalam Python beserta contohnya, antara lain :

1. Argumen Posisional: Argumen yang diberikan berdasarkan posisinya sesuai dengan urutan parameter dalam definisi fungsi.

```
contoh.py > ...
1  def tambah(a, b):
2      return a + b
3
4  hasil = tambah(3, 5)]
```

Di sini, 3 dan 5 adalah argumen posisi yang diberikan kepada fungsi `tambah`.

2. Argumen Kata Kunci: Argumen yang diberikan dengan menyebutkan nama parameter saat memanggil fungsi.

```
contoh.py > hello
1 def hello(nama, pesan):
2     print(f"Halo, {nama}! {pesan}")
3
4     hello(nama="James", pesan="Pie kabare?")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Halo, James! Pie kabare?
PS C:\Tugas>
```

Di sini, nama="James" dan pesan="Pie kabare?" adalah argumen kata kunci yang diberikan kepada fungsi hello.

3. Argumen Default: Parameter dalam definisi fungsi dapat memiliki nilai default. Jika argumen tidak diberikan saat pemanggilan fungsi, nilai default akan digunakan.

```
contoh.py > ...
1 def pangkat(angka, pangkat=2):
2     return angka ** pangkat
3
4     hasil = pangkat(3)
```

Di sini, pangkat=2 adalah argumen default untuk parameter pangkat dalam fungsi

- Parameter :

Parameter dalam Python adalah variabel yang digunakan dalam definisi fungsi untuk menerima nilai yang akan dioperasikan oleh fungsi tersebut. Parameter didefinisikan di dalam tanda kurung setelah nama fungsi dan digunakan untuk menentukan jenis dan jumlah nilai yang diterima oleh fungsi.

Contoh Parameter, antara lain:

```
contoh.py > tambah
1 def tambah(a, b):
2     return a + b
```

Dalam contoh di atas, a dan b adalah parameter fungsi tambah. Ketika fungsi tambah dipanggil, nilai yang diberikan untuk a dan b akan digunakan dalam operasi penjumlahan yang dilakukan oleh fungsi. Parameter memungkinkan kita untuk membuat fungsi yang dapat beroperasi dengan nilai yang berbeda setiap kali fungsi dipanggil.

Return Value

Return value dalam Python adalah nilai yang dikembalikan oleh sebuah fungsi setelah proses komputasi atau manipulasi data selesai dilakukan. Ketika sebuah fungsi dipanggil, kita dapat menggunakan pernyataan return untuk mengembalikan nilai tertentu ke pemanggil fungsi. Hal ini memungkinkan fungsi untuk memberikan hasil yang dihasilkan kembali kepada bagian kode yang memanggilmnya, sehingga hasil tersebut dapat digunakan atau diproses lebih lanjut.

Berikut contoh penggunaan Return Value, antara lain :

```
contoh.py > ...
1  def tambah(a, b):
2      return a + b
3
4  hasil = tambah(3, 5)
5  print("Hasil penjumlahan:", hasil)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Hasil penjumlahan: 8

Di sini, fungsi tambah mengembalikan hasil penjumlahan dari dua argumen yang diberikan. Pernyataan return a + b mengembalikan nilai a + b ke pemanggil fungsi, yang kemudian disimpan dalam variabel hasil dan dicetak. Jika sebuah fungsi tidak memiliki pernyataan return, maka secara implisit akan mengembalikan None.

Optional Argument dan Named Argument

- Optional Argument :

Optional argument dalam Python adalah parameter dalam definisi fungsi yang memiliki nilai default. Ini berarti saat fungsi dipanggil, argumen untuk parameter tersebut dapat diabaikan. Jika argumen tidak diberikan, nilai default parameter akan digunakan. Kelebihan optional argument adalah meningkatkan fleksibilitas fungsi. Dengan menggunakan optional argument, dapat memiliki fungsi yang dapat berfungsi dengan berbagai cara, tergantung pada argumen yang diberikan saat pemanggilan. Ini memungkinkan untuk memiliki perilaku default yang bisa diubah tanpa mengubah pemanggilan fungsi yang sudah ada.

Berikut adalah contoh penggunaan Optional Argument, antara lain :

```
contoh.py > ...
1  def sapa(nama="Kamu"):
2      print("Halo,", nama, "!")
3
4  sapa()
5
6  sapa("James")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

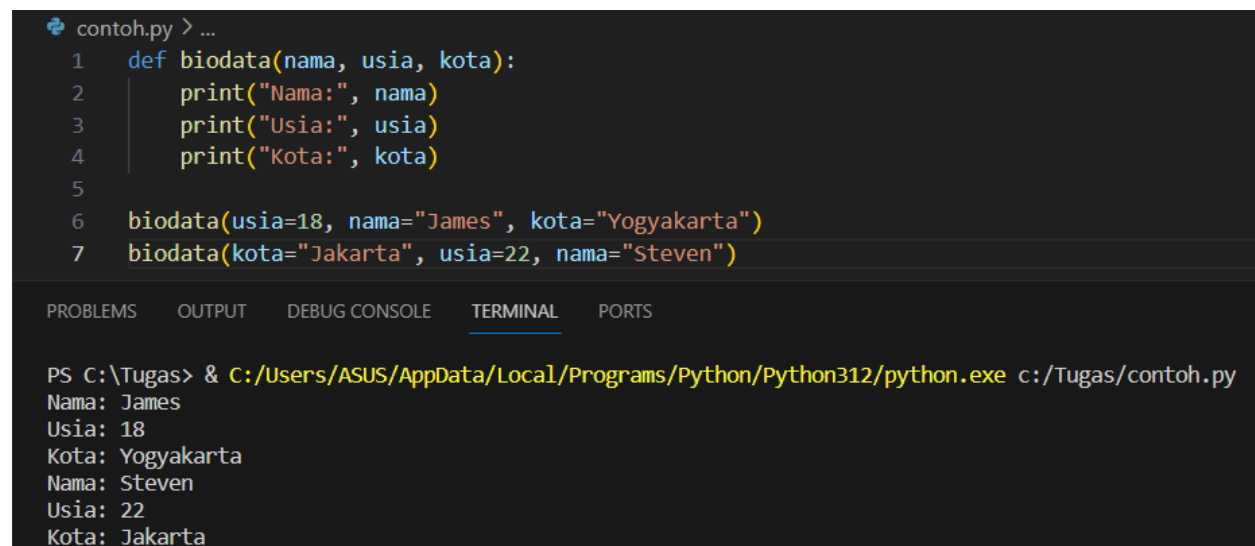
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Halo, Kamu !
Halo, James !

Dalam contoh ini, nama adalah optional argument dalam fungsi sapa dengan nilai default "Kamu". Ketika fungsi dipanggil tanpa argumen, nilai default "Kamu" akan digunakan. Namun, jika argumen "James" diberikan saat pemanggilan, nilai argumen tersebut akan menggantikan nilai default dan outputnya menjadi "Halo, James!".

- Named Argument :

Named argument dalam Python adalah cara untuk memberikan nilai ke parameter fungsi dengan menyebutkan nama parameter tersebut saat pemanggilan fungsi. Hal ini memungkinkan untuk menentukan nilai argumen berdasarkan nama parameter, sehingga tidak perlu mengikuti urutan parameter dalam definisi fungsi. Keuntungan menggunakan named argument adalah fleksibilitas dalam memberikan nilai argumen. Dapat memberikan nilai untuk beberapa parameter tertentu tanpa harus mengisi semua parameter dalam urutan yang telah ditentukan dalam definisi fungsi. Ini membuat kode lebih mudah dibaca dan lebih mudah dipahami, terutama untuk fungsi yang memiliki banyak parameter.

Berikut adalah contoh penggunaan Named Argument, antara lain :



```
contoh.py > ...
1 def biodata(nama, usia, kota):
2     print("Nama:", nama)
3     print("Usia:", usia)
4     print("Kota:", kota)
5
6 biodata(usia=18, nama="James", kota="Yogyakarta")
7 biodata(kota="Jakarta", usia=22, nama="Steven")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

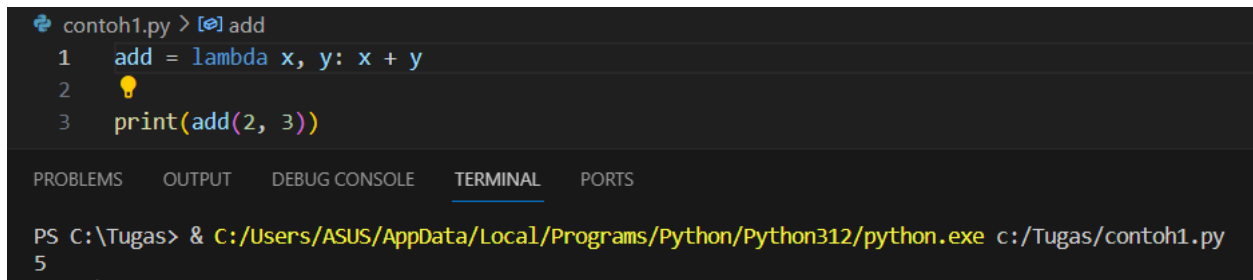
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Nama: James
Usia: 18
Kota: Yogyakarta
Nama: Steven
Usia: 22
Kota: Jakarta
```

Dalam contoh ini, kita menggunakan named argument untuk memanggil fungsi biodata. Kita dapat memberikan nilai untuk usia, nama, dan kota dalam urutan yang berbeda dan hanya memberikan nilai untuk argumen yang kita inginkan.

Anonymous Function (Lambda)

Anonymous functions, juga dikenal sebagai lambda functions, adalah fungsi tanpa nama atau fungsi sederhana yang didefinisikan tanpa menggunakan kata kunci `def` secara langsung. Mereka sering digunakan ketika Anda hanya perlu membuat fungsi sederhana yang digunakan sekali atau dalam konteks tertentu.

Contoh penggunaan lambda function dalam Python, antara lain :

A screenshot of a Python IDE interface. The top part shows a code editor with three lines of Python code: `1 add = lambda x, y: x + y`, `2` (empty line), and `3 print(add(2, 3))`. Below the code editor is a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the command `PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh1.py` and the output `5`.

```
contoh1.py > [E] add
1  add = lambda x, y: x + y
2
3  print(add(2, 3))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh1.py
5
```

Dalam contoh ini, `lambda x, y: x + y` adalah lambda function yang mengambil dua argumen `x` dan `y`, dan mengembalikan hasil penjumlahan `x + y`. Sesuai dengan sifatnya yang sederhana dan tidak memiliki nama, lambda functions sering digunakan dalam fungsi-fungsi seperti `map()`, `filter()`, dan `sort()` di Python.

BAGIAN 2: LATIHAN MANDIRI

SOAL 1.1

- Source Code :

```
Latihan 4,1.py > ...
1  def cek_angka(x, y, z):
2      if x != y and y != z and x != z:
3          if x + y == z or x + z == y or y + z == x:
4              return True
5      return False
6
7  print(cek_angka(1, 2, 3))
8  print(cek_angka(1, 2, 4))
9  print(cek_angka(1, 2, 2))
10 print(cek_angka(1, 1, 2))
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 4,1.py"
True
False
False
False
PS C:\Tugas>
```

- Penjelasan :

Fungsi `cek_angka` menerima tiga argumen `x`, `y`, dan `z`, dan mengembalikan `True` jika semua angka berbeda satu sama lain, dan salah satu dari jumlah dua angka sama dengan angka ketiga. Jika kondisi tersebut tidak terpenuhi, maka fungsi mengembalikan `False`.

SOAL 1.2

- Source Code :

```
Latihan 4,2.py > cek_digit_belakang
1 def cek_digit_belakang(x, y, z):
2     digit_x = x % 10
3     digit_y = y % 10
4     digit_z = z % 10
5
6     if digit_x == digit_y or digit_x == digit_z or digit_y == digit_z:
7         return True
8     return False
9
10 input_x = int(input("Masukkan bilangan pertama: "))
11 input_y = int(input("Masukkan bilangan kedua: "))
12 input_z = int(input("Masukkan bilangan ketiga: "))
13
14 hasil = cek_digit_belakang(input_x, input_y, input_z)
15
16 print("Output yang diharapkan =", hasil)
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 4,2.py"
Masukkan bilangan pertama: 1
Masukkan bilangan kedua: 1
Masukkan bilangan ketiga: 1
Output yang diharapkan = True
```

- Penjelasan :

Fungsi `cek_digit_belakang` memeriksa apakah setidaknya dua dari tiga bilangan yang diberikan memiliki digit terakhir yang sama. Jika ya, fungsi mengembalikan `True`, jika tidak, fungsi mengembalikan `False`. Kemudian, program meminta tiga input bilangan dari pengguna, memanggil fungsi `cek_digit_belakang` dengan input tersebut, dan mencetak hasilnya.

SOAL 1.3

- Source Code :

```
Latihan 4,3.py > ...
1  celcius_to_fahrenheit = lambda C: (9/5) * C + 32
2
3  celcius_to_reamur = lambda C: 0.8 * C
4
5  C = 100
6  F = celcius_to_fahrenheit(C)
7  print(f"Input C = {C}. Output F = {F}.")
8
9  C = 80
10 R = celcius_to_reamur(C)
11 print(f"Input C = {C}. Output R = {R}.")
12
13 C = 0
14 F = celcius_to_fahrenheit(C)
15 print(f"Input C = {C}. Output F = {F}.")
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 4,3.py"
Input C = 100. Output F = 212.0.
Input C = 80. Output R = 64.0.
Input C = 0. Output F = 32.0.
```

- Penjelasan :

Kode ini mendefinisikan dua fungsi lambda, `celcius_to_fahrenheit` untuk mengonversi suhu dari Celcius ke Fahrenheit, dan `celcius_to_reamur` untuk mengonversi suhu dari Celcius ke Reamur. Kemudian, kode ini mengonversi suhu 100°C ke Fahrenheit, suhu 80°C ke Reamur, dan suhu 0°C ke Fahrenheit, dan mencetak hasilnya.

- Link GitHub :

<https://github.com/Jamesmarvins/Tugas-Alpro-4.git>