4. Modular Programming

4.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat memahami anatomi dan struktur dari suatu fungsi.
2. Dapat memahami penggunaan parameter dan hasil dari suatu fungsi.
3. Dapat mendefinisikan fungsi dan parameter yang dibutuhkan dalam memecahkan suatu masalah.

4.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

4.3 Materi

4.3.1 Fungsi, Argument dan Parameter

Anda tentunya sudah mengerti tentang apa yang dilakukan oleh kode program berikut ini:

```
nama = input("Masukkan nama: ")
print("Hallo ", nama, " selamat pagi!")
```

Program tersebut meminta pengguna untuk memasukkan nama, kemudian program akan menyapa nama yang diinputkan oleh pengguna. Pada program tersebut ada dua fungsi yang digunakan, yaitu **input()** dan **print()**. Keduanya merupakan fungsi bawaan dari Python (atau biasa disebut sebagai *built-in function*). Setiap fungsi memiliki kegunaan masing-masing, seperti pada contoh tersebut fungsi input() digunakan untuk membaca input yang diberikan oleh pengguna, sedangkan fungsi print() digunakan untuk menampilkan tulisan di layar.

Secara umum fungsi adalah kumpulan perintah-perintah yang dijadikan satu, memiliki suatu tujuan dan kegunaan khusus serta dapat digunakan ulang. Apa kaitan fungsi dengan modular programming? Jika anda membuat program yang membutuhkan langkah yang banyak, anda akan membutuhkan untuk mengelompokkan beberapa kode program menjadi bagian-bagian (block) dari suatu program yang besar. Karena itu disebut sebagai modular, di mana program anda terdiri dari beberapa bagian modular yang memiliki kegunaan khusus dan dapat digunakan ulang. Berdasarkan asalnya, fungsi dibagi menjadi dua jenis yaitu:

- Fungsi bawaan (built-in function). Daftar fungsi bawaan Python 3 dapat dilihat di `https://docs.python.org/3/library/functions.html`
- Fungsi yang dibuat sendiri oleh programmer.

Sebagai contoh, perhatikan fungsi tambah() berikut ini yang dapat digunakan untuk menghitung jumlah dari dua bilangan yang diberikan:

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

Fungsi tambah tersebut terdiri dari beberapa hal yang perlu diperhatikan:

- Keyword **def** digunakan untuk mendefinisikan sebuah fungsi.
- Nama fungsi yang dibuat adalah **tambah()**.
- Isi dari fungsi harus anda tuliskan menjorok ke dalam 1 tab. Perhatikan bagian tambah(a,b)**:** sebagai penanda blok.
- Fungsi tambah() membutuhkan dua argument, yang nantinya akan dikenali sebagai parameter **a** dan **b**.
- Fungsi tersebut akan menghasilkan hasil penjumlahan yang dapat ditampung di sebuah variabel. Keyword **return** digunakan untuk mengembalikan/mengeluarkan nilai dari suatu fungsi.

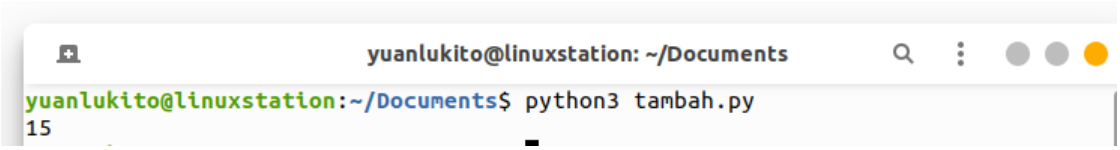
Penggunaan fungsi tersebut dapat dilihat pada kode program berikut ini:

```
1 # definisikan fungsi tambah terlebih dahulu
2 def tambah(a, b):
3     hasil = a + b
4     return hasil
5
6 # panggil fungsi tambah dengan dua arguments berupa nilai: 10 dan 5
7 c = tambah(10, 5)
8 print(c)
```

Jalannya program tersebut adalah sebagai berikut:

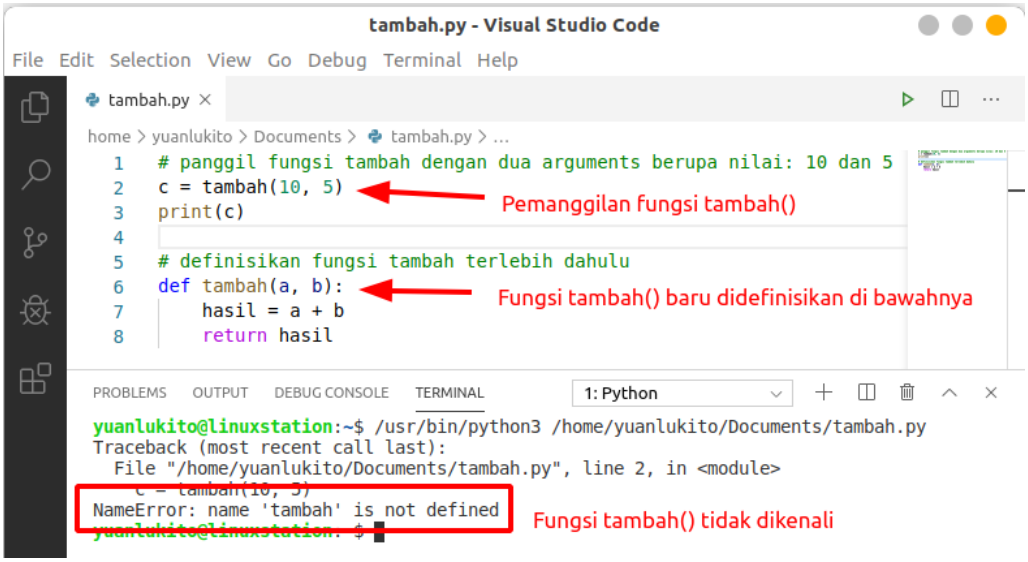
- Baris 1 adalah komentar, akan diabaikan oleh interpreter Python.
- Baris 2-4 adalah mendefinisikan fungsi tambah(). Baris 2-4 tidak dijalankan sampai suatu saat fungsi tambah() dipanggil.
- Baris 5-6 adalah baris kosong dan baris komentar akan diabaikan oleh Python.

- Baris 7 variabel c diisi oleh nilai yang dihasilkan oleh pemanggilan fungsi tambah(). Fungsi tambah dipanggil dengan argument 10 dan 5 (sesuai dengan urutan).
- Program akan lompat ke baris 2 karena fungsi tambah() dipanggil. Pada baris 2 terdapat parameter a dan b. Karena fungsi tambah() dipanggil dengan arguments 10 dan 5, maka parameter a = 10 dan b = 5 (sesuai urutan).
- Program lanjut ke baris 3. Pada baris ini variabel hasil akan berisi 10 + 5 = 15.
- Program lanjut ke baris 4. Ada penggunaan keyword return, diikuti oleh variabel hasil yang nilainya 15. Return di sini menandakan bahwa fungsi tambah() sudah berakhir dan menghasilkan nilai 15 (sesuai dengan variabel hasil).
- Dari baris 4, program akan kembali ke baris 7. Ingat, jika fungsi sudah selesai, program akan kembali ke baris di mana fungsi tersebut dipanggil. Pada baris 7 sekarang nilai c diisi oleh 15 (hasil dari menjalankan fungsi tambah().
- Pada baris 8 tampilkan nilai dari variabel c ke layar. Output yang dihasilkan adalah 15 seperti yang ditunjukkan pada Gambar 4.1.
- Jadi urutan jalannya program tersebut secara berturut-turut adalah baris 1-2-3-4-5-6-7-2-3-4-7-8.



Gambar 4.1: Hasil dari pemanggilan fungsi tambah().

Fungsi dapat dipanggil jika sudah didefinisikan sebelumnya. Jika anda memanggil fungsi yang belum didefinisikan, atau jika fungsi tersebut didefinisikan di bawah, program anda akan mengalami kesalahan. Contoh pemanggilan fungsi sebelum fungsi didefinisikan dapat dilihat pada Gambar 4.2.



Gambar 4.2: Fungsi tambah() belum dikenali karena didefinisikan di bawahnya.

4.3.2 Return Value

Berdasarkan hasil yang dikeluarkan oleh fungsi, secara umum ada dua jenis yaitu: (1) fungsi yang tidak mengembalikan nilai dan (2) fungsi yang mengembalikan nilai. Fungsi yang tidak

mengembalikan nilai sering disebut sebagai **void function**. Sebagai contoh, fungsi print_twice() berikut ini adalah fungsi yang tidak mengembalikan nilai:

```
def print_twice(message):
    print(message)
    print(message)

print_twice("Hello World!")
```

Fungsi print_twice() membutuhkan 1 parameter yaitu message. Kemudian fungsi print_twice() akan menampilkan nilai dari variabel message sebanyak 2 kali. Fungsi print_twice() tidak menghasilkan suatu nilai yang dapat digunakan untuk proses berikutnya. Fungsi print_twice() sebenarnya mengembalikan nilai **None** jika anda coba panggil dengan cara berikut:

```
def print_twice(message):
    print(message)
    print(message)

print(print_twice("Hello World!"))
```

Output yang dihasilkan adalah:

```
Hello World!
Hello World!
None
```

Berbeda halnya dengan fungsi tambah() berikut ini:

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

Fungsi tambah() membutuhkan 3 parameter yaitu a, b dan c. Kemudian di dalam fungsi tambah() didefinisikan suatu variabel hasil yang diisi oleh a+b+c. Kemudian variabel hasil tersebut dikeluarkan dari fungsi sebagai hasil dari fungsi tambah() dengan keyword **return**. Keyword return digunakan untuk:

- Mengeluarkan nilai yang merupakan hasil dari fungsi.
- Mengakhiri fungsi.

Fungsi tambah() tersebut bisa digunakan untuk mencari hasil penjumlahan tiga bilangan. Hasil dari penjumlahan tiga bilangan tersebut dapat digunakan untuk proses/langkah berikutnya, seperti contoh program untuk mencari rata-rata dari tiga bilangan berikut ini:

```
1  def tambah(a, b, c):
2      hasil = a + b + c
3      return hasil
4
5  nilai1 = 70
6  nilai2 = 85
7  nilai3 = 55
8
9  rata_rata = tambah(nilai1, nilai2, nilai3)/3
10 print(rata_rata)
```

Urutan jalannya program tersebut adalah sebagai berikut:

- Baris 1, 2, 3 adalah definisi fungsi tambah. Belum ada yang dijalankan.
- Baris 5, 6, 7 mendefinisikan tiga variabel dan langsung diisi nilainya.
- Baris 9 memanggil fungsi tambah, dengan mengirimkan tiga arguments yaitu nilai1 (70), nilai2 (85) dan nilai 3 (55). Program akan berjalan ke baris 1.
- Baris 1 ada 3 parameter yang diisi nilainya oleh tiga arguments yang dikirimkan, sehingga a = 70, b = 85 dan c = 55.
- Baris 2 mendefinisikan variabel hasil yang diisi oleh nilai a+b+c, sehingga hasil = 70 + 85 + 55 = 210.
- Baris 3 return nilai dari hasil, berarti fungsi tambah() mengeluarkan hasil dengan nilai 210. Dengan adanya return, artinya fungsi tambah() sudah berakhir dan program akan berjalan ke baris 9 (baris di mana fungsi tambah() dipanggil sebelumnya).
- Baris 9 sudah didapatkan hasil dari fungsi tambah, sehingga menjadi rata_rata = 210/3 = 70. Variabel rata_rata bernilai 70.
- Baris 10 menampilkan nilai dari variabel rata_rata, yaitu 70.

4.3.3 Optional Argument dan Named Argument

Fungsi dapat memiliki optional parameter, yaitu parameter yang bersifat opsional dan memiliki nilai bawaan (default) yang sudah didefinisikan sebelumnya. Untuk mendefinisikan optional parameter, anda harus mendefinisikan nilai bawaannya terlebih dahulu seperti pada contoh berikut ini:

```
def hitung_belanja(belanja, diskon=0):
    bayar = belanja - (belanja * diskon)/100
    return bayar
```

Fungsi hitung_belanja() memiliki dua parameter yaitu **belanja** dan **diskon**. Parameter diskon secara default memiliki nilai 0 (yang artinya 0%). Untuk memanggil fungsi hitung_belanja() tersebut, anda dapat melakukan dengan beberapa cara seperti berikut ini:

```
def hitung_belanja(belanja, diskon=0):
    bayar = belanja - (belanja * diskon)/100
    return bayar

print(hitung_belanja(100000))
print(hitung_belanja(100000, 10))
print(hitung_belanja(100000, 50))
```

Output yang dihasilkan adalah sebagai berikut:

```
100000.0
90000.0
50000.0
```

Pemanggilan pertama hanya menggunakan satu argument, sedangkan pemanggilan kedua dan ketiga menggunakan dua argument.

Setiap parameter pada fungsi memiliki nama. Karena itu pada pemanggilan fungsi juga dapat disertakan nama parameternya dan tidak perlu sesuai dengan urutan yang diberikan. Perhatikan contoh berikut ini:

```
def cetak(a, b, c):
    print("Nilai a: ",a)
    print("Nilai b: ",b)
    print("Nilai c: ",c)

cetak(20, 30, 40)
```

Output yang dihasilkan adalah:

```
Nilai a: 20
Nilai b: 30
Nilai c: 40
```

Anda juga dapat memanggil fungsi cetak() tersebut dengan cara berikut ini:

```
def cetak(a, b, c):
    print("Nilai a: ",a)
    print("Nilai b: ",b)
    print("Nilai c: ",c)

cetak(a=20, b=30, c=40)
```

Pemanggilan fungsi cetak() dilakukan dengan menyebutkan nama argumentnya (named argument). Jika anda menggunakan cara tersebut, maka urutan argument tidak harus sama dengan urutan parameter pada fungsi, seperti pada contoh berikut ini:

```
def cetak(a, b, c):
    print("Nilai a: ",a)
    print("Nilai b: ",b)
    print("Nilai c: ",c)

cetak(b=30, c=40, a=20)
```

4.3.4 Anonymous Function (Lambda)

Sesuai dengan namanya, anonymous function adalah fungsi tanpa nama (anonymous). Anonymous function pada Python adalah fitur tambahan, bukan merupakan fitur utama. Berbeda dengan bahasa-bahasa pemrograman seperti Haskell, Lisp dan Erlang yang merupakan bahasa pemrograman fungsional. Pada Python, digunakan keyword lambda untuk mendefinisikan anonymous function. Sebagai contoh, perhatikan fungsi tambah() berikut ini:

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

print(tambah(10,20))
```

Untuk mendefinisikan fungsi tambah menggunakan lambda adalah sebagai berikut:

```
tambah = lambda a, b: a + b

print(tambah(10,20))
```

Setiap anonymous function pada Python terdiri dari beberapa bagian berikut ini:

- Keyword: lambda
- Bound variable: argument pada lambda function
- Body: bagian utama lambda, berisi ekspresi atau statement yang menghasilkan suatu nilai.

4.4 Kegiatan Praktikum

Kegiatan praktikum yang akan dilakukan adalah sebagai berikut:

1. Mendefinisikan fungsi sesuai dengan spesifikasi yang dibutuhkan.
2. Penggunaan optional argument dan named argument.
3. Mendefinisikan dan menggunakan lambda function

4.4.1 Mendefinisikan Fungsi

■ Contoh 4.1 Fungsi Menghitung Tagihan Listrik

Buatlah sebuah fungsi yang dapat menghitung tagihan listrik seseorang (pasca bayar) dengan beberapa informasi berikut ini:

- Jumlah pemakaian (dalam kwh)
- Golongan tarif (1 - 4). Diasumsikan input golongan tarif selalu valid.
-
  - Golongan 1: Rp. 1500/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 2000/kwh.
  - Golongan 2: Rp. 2500/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 3000/kwh.
  - Golongan 3: Rp. 4000/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 5000/kwh.
  - Golongan 4: Rp. 5000/kwh untuk 100 kwh pertama, selanjutnya dikenakan Rp. 7000/kwh.

Jika tidak ada informasi golongan tarif, maka diasumsikan menggunakan tarif golongan 3. Definisikan fungsi tersebut! ■

Ada beberapa hal yang perlu diperhatikan sebelum anda mendefinisikan fungsi untuk menghitung tagihan listrik tersebut:

- Fungsi membutuhkan dua parameter, yaitu jumlah pemakaian dan golongan tarif.
- Jumlah pemakaian menggunakan tarif yang berbeda untuk 100 kwh pertama, dan setelah 100 kwh.
- Golongan tarif memiliki nilai default = 3, sehingga harus didefinisikan sebagai optional argument.

Fungsi untuk menghitung tagihan listrik dan contoh penggunaannya dapat dilihat pada kode program berikut ini:

```
1  def tagihan_listrik(pemakaian, golongan=3):
2      bayar = 0
3      pemakaian_100 = 100 if pemakaian > 100 else pemakaian
4      pemakaian_100_lebih = pemakaian - pemakaian_100
5      if golongan == 1:
6          bayar = pemakaian_100 * 1500 + pemakaian_100_lebih * 2000
7      elif golongan == 2:
8          bayar = pemakaian_100 * 2500 + pemakaian_100_lebih * 3000
9      elif golongan == 3:
10         bayar = pemakaian_100 * 4000 + pemakaian_100_lebih * 5000
```

```
    elif golongan == 4:
        bayar = pemakaian_100 * 5000 + pemakaian_100_lebih * 7000
    return bayar

print(tagihan_listrik(130))
print(tagihan_listrik(80, 4))
print(tagihan_listrik(golongan=1, pemakaian=175))
```

Pada kode program tersebut ada beberapa konsep yang diterapkan: optional argument, named argument dan ternary operator.

4.4.2 Argument dan Parameter

■ Contoh 4.2 Pengiriman Argument pada Fungsi

Perhatikan definisi fungsi abc() dan penggunaannya berikut ini:

```
def abc(a, b, c):
    a = b + c
    b = c + a
    c = a + b

nilai1 = 20
nilai2 = 30
nilai3 = 40
abc(nilai1, nilai2, nilai3)
print(nilai1)
print(nilai2)
print(nilai3)
```

Apa output yang dihasilkan dari kode program tersebut? Jelaskan! ■

Pengiriman parameter pada Python menggunakan prinsip yang disebut sebagai **Call-by-Object**. Perlakuan terhadap parameter tergantung dari apakah argument yang diberikan immutable atau tidak. Jika argument yang dikirimkan bersifat immutable artinya argument tersebut tidak bisa diubah oleh fungsi. Tipe data seperti integer, string dan tuple bersifat immutable.

Pada fungsi abc() dikirimkan 3 argument integer, yaitu nilai1, nilai2 dan nilai3. Karena itu dapat dikatakan argument yang dikirimkan akan bersifat immutable, sehingga tidak akan terpengaruh nilainya oleh perubahan nilai parameter dalam fungsi abc(). Output yang dihasilkan saat program tersebut dijalankan adalah sebagai berikut:

```
20
30
40
```

4.4.3 Anonymous/Lambda Function

■ Contoh 4.3 Konversi ke Lambda Function

Perhatikan fungsi yang dapat menentukan apakah suatu bilangan merupakan kelipatan 9 atau tidak berikut ini:

```
def kelipatan_sembilan(angka):
    if angka % 9 == 0:
        return True
    else
        return False
```

Ubahlah fungsi tersebut menjadi bentuk lambda function!

■

Setiap lambda function pada Python terdiri dari beberapa bagian. Dalam kasus ini, bagian-bagian tersebut adalah:

- Keyword: lambda
- Bound variable: angka
- Body: pengecekan apakah habis dibagi 9 atau tidak (kelipatan 9)

Sehingga bentuk lambda function-nya adalah sebagai berikut:

```
kelipatan_sembilan = lambda angka: angka % 9 == 0
```

Contoh penggunaan lambda function tersebut adalah sebagai berikut:

```
kelipatan_sembilan = lambda angka: angka % 9 == 0

print(kelipatan_sembilan(81))
print(kelipatan_sembilan(2000))
```

4.5 Latihan Mandiri

Latihan 4.1 Buatlah sebuah fungsi yang dapat menentukan apakah ketiga parameter memenuhi semua ketentuan berikut ini:

- Ketiga parameter tersebut nilainya berbeda semua.
- Ada kemungkinan jika diambil dua parameter dan dijumlahkan hasilnya sama dengan parameter lainnya (yang tersisa).

Fungsi tersebut akan menghasilkan nilai True jika semua ketentuan tersebut dipenuhi. Jika tidak terpenuhi maka fungsi akan menghasilkan nilai False. Fungsi anda harus diberi nama cek_angka(). ■

Latihan 4.2 Buatlah sebuah fungsi yang dapat menentukan apakah **minimal dua** dari tiga parameter yang diberikan memiliki digit paling kanan yang sama. Fungsi tersebut menghasilkan nilai True jika memenuhi dan False jika tidak memenuhi. Gunakan fungsi tersebut untuk mengecek beberapa test-case berikut ini:

- Input = 30, 20, 18. Output yang diharapkan = True
- Input = 145, 5, 100. Output yang diharapkan = True
- Input = 71, 187, 18. Output yang diharapkan = False
- Input = 1024, 14, 94. Output yang diharapkan = True
- Input = 53, 8900, 658. Output yang diharapkan = False

Ketiga bilangan tersebut diinputkan oleh pengguna, sehingga anda perlu membaca input dari pengguna. Fungsi anda harus diberi nama cek_digit_belakang(). ■

Latihan 4.3 Buatlah fungsi-fungsi konversi suhu menggunakan lambda function. Fungsi-fungsi yang harus anda implementasikan:

- Celcius to Fahrenheit. $F = (9/5) * C + 32$
- Celcius to Reamur. $R = 0.8 * C$

Berikan contoh penggunaannya untuk test-case berikut ini:

- Input C = 100. Output F = 212.
- Input C = 80. Output R = 64.
- Input = 0. Output F = 32.

■