Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230970**
Nama Lengkap	**Gregorius Daniel Jodan Perminas**
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 (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

Pengertian Modular Programming

Secara sederhananya *modular programming* adalah proses untuk membagi program komputer yang panjang menjadi beberapa sub-program yang terpisah. Beberapa baris kode dengan tujuan yang serupa akan dijadikan satu dan dibuat menjadi sebuah fungsi.

Manfaat atau Kelebihan Modular Programming:

1. Kode menjadi lebih ringkas
2. Kode dapat digunakan berkali-kali tanpa harus mengetikkan ulang
3. Mempermudah perbaikan atau *debugging* program
4. Dapat menambahkan fitur pada program tanpa harus mengganggu operasi program secara keseluruhan

Fungsi, Argument, dan Parameter

Fungsi dalam Python terbagi menjadi dua macam, bawaan dan buatan *programmer*. Di bawah ini adalah contoh penggunaan fungsi bawaan Python:

```
1  nama = input("Siapa nama kamu? ")
2  print(f"Hallo! {nama}, salam kenal!")
```

Gambar 1 - Fungsi Bawaan di Python

Program di atas akan meminta pengguna untuk memasukkan nama, kemudian memberi output untuk menyapa pengguna. Fungsi bawaan *(built-in function)* yang digunakan dalam Gambar 1 di atas adalah input() yang digunakan untuk membaca hasil masukan dari pengguna, sedangkan print() digunakan untuk menampilkan tulisan pada terminal.

Pengertian dari *function* sendiri adalah sekumpulan perintah yang dijadikan satu, memiliki suatu tujuan atau kegunaan khusus dan dapat digunakan kembali. Di bawah ini merupakan contoh *function* yang dibuat oleh *programmer*:

```
1  def perkalian (a,b):
2      hasil = a * b
3      return hasil
4
5  c = perkalian(2,5)
6  print(c)
```

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scri
10
```

Gambar 2 - Fungsi Buatan Programmer di Python

Penjelasan Kode Per Baris:

1. Def digunakan untuk mendefinisikan sebuah fungsi. Nama fungsi di sini adalah ”perkalian” yang memiliki 2 parameter yaitu a dan b, atau bisa dibilang ketika pemanggilan fungsi ini akan membutuhkan 2 arguments.
2. Baris ini merupakan isi dari fungsi ”perkalian” yang telah dibuat. Hasil perkalian antara a dan b akan disimpan ke dalam variabel bernama ”hasil.”
3. Hasil dari perkalian dari baris sebelumnya masih ditampung dalam variabel ”hasil.” Keyword ”return” dapat digunakan untuk mengembalikan/mengeluarkan nilai dari suatu fungsi.
4. ---
5. Di sini variabel ”c” akan digunakan untuk menyimpan nilai yang dihasilkan oleh pemanggilan fungsi perkalian(). Fungsi tambah dipanggil dengan argument 2 dan 5 (sesuai dengan urutan). Ketika fungsi dipanggil program akan langsung pergi ke baris *function* yaitu baris 1 dengan parameter a = 2 dan b = 5. Program kemudian lanjut ke baris berikutnya, sehingga variabel hasil akan diisi nilai 2 x 5. Pada baris 3 terdapat penggunaan keyword ”return”, diikuti oleh variabel ”hasil” yang nilainya 10. Return di sini menandakan fungsi perkalian() sudah selesai dan akan mengeluarkan nilai ”hasil”. Dari baris 3, program akan kembali ke baris 5, maka baris 5 sekarang nilai c diisi oleh 10.
6. Fungsi bawaan print digunakan untuk menampilkan nilai c.

Void Functions and Functions Returning Values

Berdasarkan hasil yang dikeluarkan oleh fungsi, secara umum fungsi terbagi menjadi dua jenis yaitu fungsi yang tidak mengembalikan nilai dan fungsi yang mengembalikan nilai. Fungsi yang tidak mengembalikan nilai disebut *void function*. Dibawah ini merupakan contoh *void function*:

```
1  def printDuaNihBos (pesan):
2      print(pesan)
3      print(pesan)
4  printDuaNihBos("Oke Gas!")
```

Gambar 3 - Void Function di Python

Fungsi di atas berfungsi untuk menampilkan nilai dari variabel pesan sebanyak dua kali. Fungsi printDuaNihBos tidak menghasilkan nilai apapun yang bisa digunakan untuk proses berikutnya atau mengembalikan nilai **None**. Berikut adalah tampilan output program yang menggunakan *void function:*

```
1    def printDuaNihBos (pesan):
2        print(pesan)
3        print(pesan)
4    print(printDuaNihBos("Oke Gas!"))
```

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Oke Gas!
Oke Gas!
None
```

Gambar 4 - Nilai Yang Dikembalikan Void Function

```
1    def perkalian (a,b):
2        hasil = a * b
3
4    print(perkalian(2,5))
```

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scripts\python.exe
None
```

Gambar 5 - Hasil Program Gambar 2 Tanpa Return

Hal ini akan berbeda ketika kita menggunakan *function* yang melakukan *return values*.

```
1    def perkalian (a,b):
2        hasil = a * b
3        return hasil
4    print(perkalian(2,5))
```

Gambar 6 – Diambil Dari Gambar 2

Sama dengan Gambar 2 sebelumnya, keyword ”return” digunakan untuk mengembalikan atau mengeluarkan nilai dari fungsi, sehingga ketika terdapat perintah print, hasil perhitungan bisa ditampilkan yaitu 10.

Optional dan Named Argument

Sebuah fungsi dapat memiliki *optional parameter* yang artinya memiliki nilai bawaan (*default)* yang telah didefinisikan sebelumnya. Untuk mendefinisikan *optional parameter*, nilai bawaannya harus didefinisikan terlebih dahulu, contohnya:

```python
def penghitung_diskon (harga, diskon = 0):
    harga_akhir = harga - harga * (diskon / 100)
    return harga_akhir

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

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scripts\python.exe
100000.0
90000.0
50000.0
```

Gambar 7 - Contoh Optional Parameter di Python

Dapat dilihat bahwa pemanggilan pertama hanya menggunakan satu argument, sedangkan pemanggilan kedua dan ketiga menggunakan dua argument. Pemanggilan pertama tetap bisa berjalan dikarenakan adanya *optional parameter.* Ketika argument tidak disebutkan saat pemanggilan *function* maka nilai *default* yang telah ditentukan sebelumnya akan dimasukkan, dalam gambar 7 di atas nilai yang dimasukkan untuk diskon adalah 0, maka pembayaran tetap 100000.

Setiap parameter pada fungsi memiliki nama, maka pemanggilan fungsi juga dapat dilakukan dengan menyebutkan nama parameternya dan tidak perlu sesuai dengan urutan yang diberikan. Perhatikan contoh kasus di bawah ini:

```python
def cetak(a, b, c):
    print("Nilai a: ",a)
    print("Nilai b: ",b)
    print("Nilai c: ",c)
cetak(20, 30, 40)
```

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Nilai a:  20
Nilai b:  30
Nilai c:  40
```

Gambar 8 – Pemanggilan Fungsi Tanpa Named Argument di Python

```python
def cetak(a, b, c):
    print("Nilai a: ",a)
    print("Nilai b: ",b)
    print("Nilai c: ",c)
cetak(b=30, c=40, a=20)
```

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scripts\python.exe
Nilai a:  20
Nilai b:  30
Nilai c:  40
```

Gambar 9 - Pemanggilan Fungsi dengan Named Argument di Python

Pemanggilan fungsi cetak() pada Gambar 9 dilakukan dengan menyebutkan nama argumentnya (named argument). Jika kita menggunakan cara tersebut, maka urutan argument tidak harus sama dengan urutan parameter yang ditentukan pada fungsi. Tampak pada Gambar 8 maupun 9 output dari kedua eksekusi program sama yaitu nilai a: 20, nilai b: 30, dan nilai c: 40.

Anonymous Function (Lambda)

Anonymous function adalah fungsi yang tidak memiliki nama. Anonymous function pada Python merupakan salah satu fitur tambahan dan bukan merupakan fitur utama. Pada Python, kita dapat menggunakan keyword ”lambda” untuk mendefinisikan anonymous function. *Lambda function* bisa menampung arguments sebanyak apapun, tapi hanya bisa memiliki satu *expression.*

Syntax

lambda arguments : expression

Perhatikan contoh di bawah ini:

```
1  def tambah(a, b):
2      hasil = a + b
3      return hasil
4  print(tambah(10,20))
```

Gambar 10 - Function Pada Python

```
1  tambah = lambda a, b: a + b
2  print(tambah(10,20))
```

```
C:\Users\ASUS\PycharmProjects\pythonProject3\venv\Scripts\python.exe
30
```

Gambar 11 - Lambda Function Pada Python

Kedua *function* di atas memiliki kegunaan yang sama yaitu menjumlahkan 10 dengan 20. Bedanya hanya pada cara penulisannya saja. *Lambda Function* jauh lebih ringkas dan tidak memerlukan keyword return untuk mengeluarkan nilainya, namun hanya bisa memiliki satu *expression* saja.

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

SOAL 1

```
# Latihan Mandiri 4.1

def cek_angka(a,b,c):
    if a != b and a != c and b != c:
        if a == b + c or b == a + c or c == a + b:
            return True
        else:
            return False
    else:
        return False

angka1 = int(input("Masukkan nilai parameter 1: "))
angka2 = int(input("Masukkan nilai parameter 2: "))
angka3 = int(input("Masukkan nilai parameter 3: "))
print(cek_angka(angka1,angka2,angka3))
```

```
Masukkan nilai parameter 1:  14
Masukkan nilai parameter 2:  16
Masukkan nilai parameter 3:  30
True
```

Gambar 12 - Program untuk Latihan 4.1

```
Masukkan nilai parameter 1:  17
Masukkan nilai parameter 2:  34
Masukkan nilai parameter 3:  17
False
```

Gambar 13 - Uji Coba 1 Program 4.1

```
Masukkan nilai parameter 1:  14
Masukkan nilai parameter 2:  17
Masukkan nilai parameter 3:  39
False
```

Gambar 14 - Uji Coba 2 Program 4.1

Soal meminta program bisa menghasilkan dua jenis output antara True atau False. Ada dua syarat yang harus dipenuhi agar program menghasilkan True yaitu ketiga parameter harus memiliki nilai yang berbeda semua. Untuk memenuhi permintaan ini maka pada baris ke-2 program diberi operator perbandingan (!=) yang artinya nilai variabel yang dibandingkan harus berbeda antara kiri dan kanan, a tidak boleh sama dengan b DAN a tidak boleh sama dengan c DAN b tidak boleh sama dengan c. Operator logika AND digunakan supaya program hanya menjalankan blok berikutnya ketika ketiganya sudah terpenuhi, dengan begini maka syarat pertama sudah terpenuhi. Syarat yang kedua adalah ketika diambil dua parameter dan dijumlahkan hasilnya harus sama dengan parameter yang tersisa. Dari informasi ini maka terdapat 3 kemungkinan yaitu a = b + c ATAU b = a + c ATAU c = a + b. Operator logika yang dipakai adalah ATAU karena asal salah satu syarat terpenuhi maka sudah dianggap benar, maka dimasukkan ke dalam program pada baris ketiga ”if a == b + c or b == a + c or c == a + b: return True”. Sisa semua kemungkinan yang lain akan menghasilkan nilai False karena tidak sesuai dengan syarat.

SOAL 2

```
# Latihan Mandiri 4.2
```

```
def cek_digit_belakang(a,b,c):
    nilai_akhir1 = a % 10
    nilai_akhir2 = b % 10
    nilai_akhir3 = c % 10
    if nilai_akhir1 == nilai_akhir2 == nilai_akhir3:
        return True
    elif nilai_akhir1 == nilai_akhir2 or nilai_akhir1 == nilai_akhir3 or nilai_akhir2 == nilai_akhir3:
        return True
    else:
        return False

angka1 = int(input("Masukkan nilai parameter 1: "))
angka2 = int(input("Masukkan nilai parameter 2: "))
angka3 = int(input("Masukkan nilai parameter 3: "))
print(cek_digit_belakang(angka1, angka2, angka3))
```

```
Masukkan nilai parameter 1:  2245
Masukkan nilai parameter 2:  345
Masukkan nilai parameter 3:  2
True
```

Gambar 15 - Program Untuk Latihan 4.2

Soal meminta dibuatkan sebuah fungsi dengan nama cek_digit_belakang() yang dapat menentukan apakah minimal dua dari tiga parameter yang diberikan memiliki digit paling kanan yang sama. Fungsi tersebut akan menghasilkan nilai True jika memenuhi dan False jika tidak memenuhi. Berdasarkan permintaan soal untuk mencari digit paling akhir maka operasi aritmatika yang bisa dilakukan adalah dengan modulus 10. Hal ini dilakukan untuk mencari sisa hasil bagi yang berupa satuan dari setiap bilangan. Tampak pada program mulai dari baris 2-4 semua parameter (a,b,c) akan di modulus 10. Kemudian untuk mengecek kondisi apakah minimal 2 parameter memiliki digit belakang yang sama kita harus memetakan semua kondisi yang bisa diperoleh terlebih dahulu.

Kondisi yang bisa didapat ada 3 macam yaitu ketiganya sama, ada dua yang sama, atau tidak ada yang sama, maka pada baris ke-5 digunakan operator perbadingan (==)pada ketiga parameter, baru pada baris ke-6 yang dibandingkan hanya dua parameter saja dan disambung dengan operator logika OR supaya ketika salah satu terpenuhi program tetap akan menjalankan blok kode berikutnya yaitu return True. Sisanya yang tidak sesuai syarat akan dianggap False oleh program. Karena soal meminta agar data diinput pengguna, maka baris 12-14 dibuat variabel dengan perintah input yang kemudian dikonversi menjadi integer. Ketiga variabel ini akan dimasukkan ke perintah print yang berisi pemanggilan *function* untuk menampilkan hasil / output.

Di bawah ini adalah hasil Test Case sesuai dengan permintaan soal:

```
Masukkan nilai parameter 1:  30
Masukkan nilai parameter 2:  20
Masukkan nilai parameter 3:  18
True
```

```
Masukkan nilai parameter 1:  145
Masukkan nilai parameter 2:  5
Masukkan nilai parameter 3:  100
True
```

```
Masukkan nilai parameter 1:  71
Masukkan nilai parameter 2:  187
Masukkan nilai parameter 3:  18
False
```

```
Masukkan nilai parameter 1:  1024
Masukkan nilai parameter 2:  14
Masukkan nilai parameter 3:  94
True
```

```
Masukkan nilai parameter 1:  53
Masukkan nilai parameter 2:  8900
Masukkan nilai parameter 3:  658
False
```

SOAL 3

```
# Latihan Mandiri 4.3

celcius_to_fahrenheit = lambda c: (9/5 * c) + 32
celcius_to_reamur = lambda c: 0.8 * c

nominalCelcius = int(input("Masukkan Suhu dalam Satuan Celcius: "))
print(f"Input C = {nominalCelcius}. Output F = {celcius_to_fahrenheit(nominalCelcius)}")
print(f"Input C = {nominalCelcius}. Output R = {celcius_to_reamur(nominalCelcius)}")
```

```
Masukkan Suhu dalam Satuan Celcius:  32
Input C = 32. Output F = 89.6
Input C = 32. Output R = 25.6
```

Gambar 16 - Program Untuk Latihan 4.3

Soal meminta untuk dibuatkan fungsi yang dapat mengkonversi suhu menggunakan lambda function. Fungsi yang harus diimplementasikan ada dua yaitu:

1. Celcius to Fahrenheit. F = (9/5) ∗C +32
2. Celcius to Reamur. R = 0.8 ∗C

Syntax dari Lambda Function adalah:

lambda arguments : expression

Maka pada baris 1-2 Anonymous Function dibuat dengan syntax yang sama seperti teori di atas. Setelah itu dibuat dua variable di sebelah kirinya untuk menyimpan nilai dari Lambda Function yaitu ”celcius_to_fahrenheit” dan ”celcius_to_reamur.”

Perhitungan atau rumus ditulis sama persis dengan permintaan soal. Karena program harus bisa menerima *input* dari pengguna maka diberi variabel dan perintah input untuk menyimpan dan menerima *input.* Baru setelah itu perintah print diberikan untuk menampilkan *output* dari program.

Di bawah ini adalah hasil Test Case sesuai dengan permintaan soal:

```
Masukkan Suhu dalam Satuan Celcius:  100
Input C = 100. Output F = 212.0
Input C = 100. Output R = 80.0
```

```
Masukkan Suhu dalam Satuan Celcius:  80
Input C = 80. Output F = 176.0
Input C = 80. Output R = 64.0
```

```
Masukkan Suhu dalam Satuan Celcius:  0
Input C = 0. Output F = 32.0
Input C = 0. Output R = 0.0
```