



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	<71230989>
Nama Lengkap	<Yohanes Nevan Adventus Wibawa>
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.

Fungsi, Argument dan Parameter

```
nama = input("Masukan nama anda = ")  
  
print("Hallo aku", nama, "selamat pagi")
```

Program tersebut meminta pengguna untuk memasukkan nama, lalu menyapa nama yang diinputkan. Dalam program ini, terdapat dua fungsi bawaan Python, yaitu `input()` untuk membaca input dari pengguna dan `print()` untuk menampilkan teks di layar. Fungsi-fungsi ini adalah kumpulan perintah dengan tujuan dan kegunaan tertentu. Secara umum, fungsi-fungsi ini terkait dengan pemrograman modular. Modular programming melibatkan pengelompokan kode program ke dalam bagian-bagian yang memiliki tujuan khusus dan dapat digunakan kembali. Ketika Anda mengembangkan program yang melibatkan banyak langkah, penting untuk memisahkan dan mengorganisir kode ke dalam modul-modul agar lebih mudah dikelola 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 memiliki beberapa poin yang perlu diperhatikan: Gunakan kata kunci `def` untuk mendefinisikan fungsi. Beri nama fungsi dengan `tambah()`. Isi fungsi harus diindent satu tab. Perhatikan `tambah(a, b)`: sebagai penanda blok. Fungsi `tambah()` memerlukan dua argumen, yang disebut sebagai parameter `a` dan `b`. Hasil penjumlahan dari fungsi dapat disimpan dalam variabel. Kata kunci `return` digunakan untuk mengembalikan nilai dari fungsi. Implementasi fungsi ini dapat dilihat dalam kode program berikut:

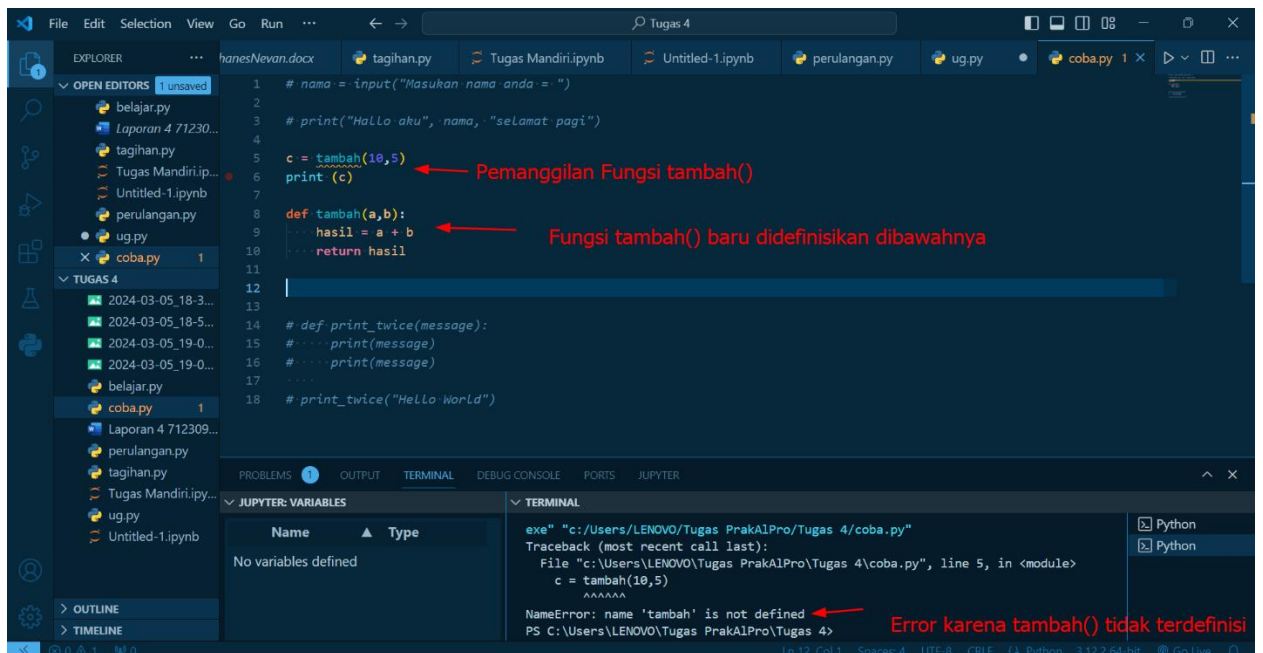
```
def tambah(a,b):  
    ... hasil = a + b  
    ... return hasil  
  
c = tambah(10,5)  
print (c)
```

Return Value

Dari output fungsi, dapat dibedakan secara umum menjadi dua jenis, yaitu: (1) fungsi tanpa nilai kembalian dan (2) fungsi dengan nilai kembalian. Fungsi tanpa nilai kembalian sering disebut sebagai void function. Sebagai contoh, fungsi `print_twice()` berikut merupakan contoh fungsi tanpa mengembalikan nilai:

```
def print_twice(message):  
    print(message)  
    print(message)  
    ....  
print_twice("Hello World")
```

Fungsi `print_twice()` memerlukan satu parameter, yaitu `message`. Selanjutnya, fungsi ini akan menampilkan nilai dari variabel `message` dua kali. Meskipun fungsi `print_twice()` tidak menghasilkan nilai yang dapat digunakan dalam proses selanjutnya. Sebenarnya, jika Anda mencoba memanggilnya, fungsi `print_twice()` akan mengembalikan nilai `None`, seperti contoh berikut:



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a terminal at the bottom. The main editor displays a Python script with the following code:

```
1 # nama = input("Masukan nama anda = ")  
2  
3 # print("Hallo aku", nama, "selamat pagi")  
4  
5 c = tambah(10,5)  
6 print(c)  
7  
8 def tambah(a,b):  
9     hasil = a + b  
10    return hasil  
11  
12  
13  
14 # def print_twice(message):  
15 #     print(message)  
16 #     print(message)  
17 #  
18 # print_twice("Hello World")
```

Annotations in the image point to specific lines:

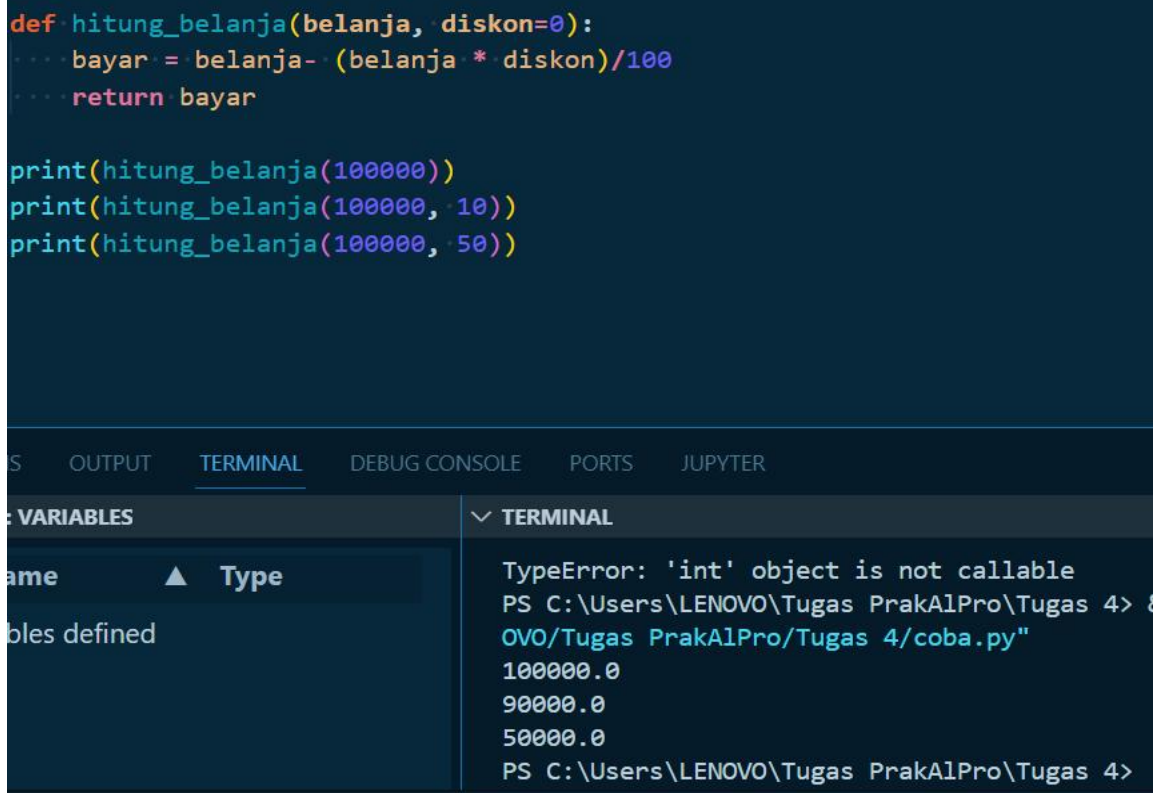
- Red arrow pointing to line 5: **Pemanggilan Fungsi tambah()**
- Red arrow pointing to line 8: **Fungsi tambah() baru didefinisikan dibawahnya**

The terminal at the bottom shows the execution of the script, resulting in a `NameError: name 'tambah' is not defined`. The error message is highlighted with a red arrow and the text **Error karena tambah() tidak terdefinisi**.

Optional Argument and Named Argument

Fungsi memiliki kemampuan untuk memiliki parameter opsional, yaitu parameter yang bersifat tidak wajib dan memiliki nilai bawaan (default) yang telah ditentukan sebelumnya. Untuk menentukan parameter opsional, Anda perlu mendefinisikan nilai default terlebih dahulu, seperti yang ditunjukkan dalam contoh berikut:

```
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))
```



The screenshot shows a Jupyter Notebook interface with a terminal window. The terminal output displays the results of the function calls: 100000.0, 90000.0, and 50000.0. The terminal also shows a TypeError message: 'int' object is not callable. The terminal output is as follows:

```
TypeError: 'int' object is not callable  
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 4> 8  
OVO/Tugas PrakAlPro/Tugas 4/coba.py"  
100000.0  
90000.0  
50000.0  
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 4>
```

Anonymous Function (Lambda)

Seiring dengan namanya, anonymous function adalah fungsi tanpa nama yang pada Python dikenal sebagai fitur tambahan, bukan fitur utama. Berbeda dengan bahasa pemrograman fungsional seperti Haskell, Lisp, dan Erlang, di mana konsep ini lebih mendalam. Dalam Python, kita menggunakan kata kunci lambda untuk mendefinisikan anonymous function. Sebagai ilustrasi, lihatlah contoh fungsi tambah() berikut:

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

Fungsi biasa

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

Lambda

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.

Kegiatan Praktikum Mendefinisikan Fungsi

The screenshot shows a Jupyter Notebook interface with a code editor and a terminal output.

Code Editor:

```

1  def tagihan_listrik(jmlh_pemakaian,golongan=3):
2      ... pemaiaikan_awal = 100
3      ... pemakaian_selanjutnya = jmlh_pemakaian - pemaiaikan_awal
4      ... if golongan == 1:
5      ...     bayar = pemaiaikan_awal * 1500 + pemakaian_selanjutnya * 2000
6      ... elif golongan == 2:
7      ...     bayar = pemaiaikan_awal * 2500 + pemakaian_selanjutnya * 3000
8      ... elif golongan == 3:
9      ...     bayar = pemaiaikan_awal * 4000 + pemakaian_selanjutnya * 5000
10     ... elif golongan == 4:
11     ...     bayar = pemaiaikan_awal * 5000 + pemakaian_selanjutnya * 7000
12     ... return bayar
13
14     print(tagihan_listrik(130))
15     print(tagihan_listrik(80,4))
16     print(tagihan_listrik(golongan=1,jmlh_pemakaian=175))
17
18
19
20
21

```

Terminal Output:

```

PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 4> & "C:/Program Files/Python39/Python.exe" "C:/Users/LENOVO/Tugas PrakAlPro/Tugas 4/tagihan listrik.py"
550000
360000
300000
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 4>

```

Kode ini adalah implementasi dari suatu fungsi untuk menghitung tagihan listrik berdasarkan jumlah pemakaian dan golongan tertentu. Berikut adalah penjelasan singkatnya: - Fungsi `tagihan_listrik` memiliki dua parameter, yaitu `jmlh_pemakaian` (jumlah pemakaian listrik) dan `golongan` (golongan pelanggan) yang memiliki nilai default 3. - Variabel `pemaiaikan_awal` diinisialisasi dengan nilai 100. - Variabel `pemakaian_selanjutnya` menghitung jumlah pemakaian selanjutnya setelah mencapai batas awal. - Melalui serangkaian kondisi `if-elif`, fungsi menghitung nilai `bayar` berdasarkan golongan pelanggan dan tarif yang sesuai. - Fungsi kemudian mengembalikan nilai `bayar`. Contoh pemanggilan fungsi: 1. `print(tagihan_listrik(130))` akan menghitung tagihan untuk 130 kWh pada golongan default 3. 2. `print(tagihan_listrik(80,4))` akan menghitung tagihan untuk 80 kWh pada golongan 4. 3. `print(tagihan_listrik(golongan=1,jmlh_pemakaian=175))` akan menghitung tagihan untuk 175 kWh pada golongan 1.

Argument dan Parameter

```
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)
```

VARIABLES		TERMINAL
Name	Type	PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 4> & "C:/P OVO/Tugas PrakAlPro/Tugas 4/coba.py"
ables defined		20 30 40 PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 4>

Pengiriman parameter dalam Python mengikuti prinsip yang disebut Call-by-Object. Perlakuan terhadap parameter bergantung pada apakah argumen yang diberikan bersifat immutable atau tidak. Argumen yang bersifat immutable tidak dapat diubah oleh fungsi. Contoh tipe data yang bersifat immutable meliputi integer, string, dan tuple. Dalam fungsi `abc()`, tiga argumen integer dikirimkan, yaitu `nilai1`, `nilai2`, dan `nilai3`. Oleh karena itu, dapat disimpulkan bahwa argumen yang dikirimkan bersifat immutable. Sebagai hasilnya, nilai-nilai ini tidak akan terpengaruh oleh perubahan nilai parameter dalam fungsi `abc()`.

Anonymus/Lamda Function

```
def kelipatan Sembilan(angka):  
    if angka % 9 == 0:  
        return True  
    else:  
        return False  
  
print(kelipatan Sembilan(81))  
print(kelipatan Sembilan(2000))  
  
kelipatan Sembilan = lambda angka: angka % 9 == 0  
  
print(kelipatan Sembilan(81))  
print(kelipatan Sembilan(2000))
```

Function biasa

Lamda

Kedua kode tersebut memiliki tujuan yang sama, yaitu untuk menentukan apakah suatu angka merupakan kelipatan sembilan atau tidak. Perbedaan utama terletak pada cara implementasinya: 1. Fungsi dengan Pendekatan Tradisional:- Fungsi `kelipatan Sembilan` ditulis menggunakan pendekatan tradisional dengan menggunakan pernyataan `if-else`. - Fungsi ini mengembalikan `True` jika angka tersebut kelipatan sembilan, dan `False` jika tidak. 2. Fungsi dengan Lambda Expression:- Fungsi `kelipatan Sembilan` diimplementasikan sebagai lambda expression. - Lambda expression secara langsung mengembalikan hasil evaluasi dari ekspresi `angka % 9 == 0`. - Kode ini lebih singkat dan langsung menyatakan kondisi kelipatan sembilan dalam satu baris. Keduanya memberikan output yang sama, tetapi pendekatan lambda expression lebih ringkas dan dapat digunakan untuk fungsi sederhana tanpa perlu mendefinisikan blok fungsi secara terpisah.

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

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

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 4.1

A. Source Code & Output

```
def cek_angka(a, b, c):  
    """  
    if (a != b and b != c and a != c) and (a + b == c or a + c == b or b + c == a):  
    """  
    return True  
    else:  
    return False  
  
hasil = cek_angka(1, 2, 3)  
print(hasil)  
  
hasil = cek_angka(1, 2, 2)  
print(hasil)
```

True
False

B. Penjelasan

Fungsi `cek_angka(a, b, c)` mengevaluasi tiga parameter untuk memastikan bahwa semua parameter memiliki nilai yang berbeda. Ada kemungkinan jika dua parameter dijumlahkan, hasilnya sama dengan parameter ketiga yang tersisa. Jika kedua kondisi tersebut terpenuhi, fungsi mengembalikan nilai `True`, jika tidak, mengembalikan nilai `False`. Fungsi ini menggunakan operator logika dan perbandingan untuk melakukan pengecekan tersebut.

SOAL 4.2

A. Source Code & Output

```
def cek_digit_belakang(a,b,c):  
    digit_a= a % 10  
    digit_b= b % 10  
    digit_c= c % 10  
    if digit_a == digit_b or digit_a == digit_c or digit_b == digit_c:  
        return True  
    else:  
        return False  
  
print(cek_digit_belakang(30,20,1))
```

True

B. Penjelasan

Fungsi `cek_digit_belakang(a, b, c)` menentukan apakah minimal dua dari tiga parameter memiliki digit paling kanan yang sama. Fungsi ini menghasilkan nilai `True` jika kondisi tersebut terpenuhi, dan `False` jika tidak. Pertama, fungsi menghitung digit paling kanan dari setiap parameter menggunakan operasi modulo (%). Selanjutnya, menggunakan perbandingan, fungsi memeriksa apakah terdapat kesamaan digit paling kanan di antara dua atau lebih parameter. Jika ada kesamaan, fungsi mengembalikan nilai `True`, dan jika tidak, mengembalikan nilai `False`.

Soal 4.3

A. Source Code & Output

```
celsius_to_fahrenheit = lambda c : (9/5) * c + 32  
celsius_to_reamur = lambda c : 0.80 * c  
  
print(celsius_to_fahrenheit(100))  
print(celsius_to_reamur(80))  
print(celsius_to_fahrenheit(0))
```

✓ 0.0s

212.0

64.0

32.0

B. Penjelasan

Dalam implementasi ini, digunakan lambda functions untuk membuat dua fungsi konversi suhu: satu untuk mengubah suhu dari Celsius ke Fahrenheit, dan yang lainnya untuk mengubah suhu dari Celsius ke Reamur. Lambda functions memungkinkan definisi fungsi yang ringkas dan langsung dievaluasi, tanpa perlu mendeklarasikan fungsi secara terpisah.

Fungsi pertama, ``celsius_to_fahrenheit``, menggunakan rumus Fahrenheit $(9/5) * c + 32$. sedangkan fungsi kedua, ``celsius_to_reamur``, menggunakan rumus Reamur $c : 0.80 * c$

Penggunaan kedua fungsi tersebut untuk beberapa test-case memastikan bahwa konversi suhu berjalan sesuai harapan. Sebagai contoh, ketika suhu dalam Celsius adalah 100, ``celsius_to_fahrenheit`` menghasilkan 212, sementara ``celsius_to_reamur`` menghasilkan 64. Pendekatan ini mempermudah konversi suhu tanpa perlu menentukan fungsi secara konvensional.

Link GitHub

Link = <https://github.com/YohanesNevan/Tugas-Laporan-AIPro-4.git>