|  |  |
| --- | --- |
| A picture containing text, clipart  Description automatically generated | **Laporan Praktikum Algoritma dan Pemrograman**  Semester Genap 2023/2024 |

|  |  |
| --- | --- |
| **NIM** | **71230992** |
| **Nama Lengkap** | **Andriano Kurniawan Ladjeba** |
| **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.

## MATERI 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 ini mengharuskan pengguna untuk memasukkan nama, kemudian program akan menerima nama yang dimasukkan pengguna. Dalam program ini, dua fungsi digunakan yaitu **input()** dan **print()**. Keduanya merupakan fungsi bawaan Python (atau biasa disebut built-in function). Masing-masing fungsi mempunyai kegunaannya masing-masing, seperti pada contoh, fungsi input() digunakan untuk membaca input yang di masukan oleh pengguna, sedangkan fungsi print() digunakan untuk menampilkan teks di layar.

Secara umum fungsi adalah kumpulan perintah yang disatukan yang mempunyai tujuan dan kegunaan tertentu serta dapat digunakan kembali. Apa hubungan fungsi ini dengan modular programming? Jika Anda membuat program yang memerlukan banyak langkah, Anda perlu mengelompokkan beberapa kode program menjadi beberapa bagian (blok) dari program besar. Karena disebut modul, maka bila program Anda terdiri dari beberapa bagian modular yang mempunyai kegunaan khusus dan dapat digunakan kembali. Berdasarkan asal, 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/function.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:

def tambah(a,b):

    hasil = a + b

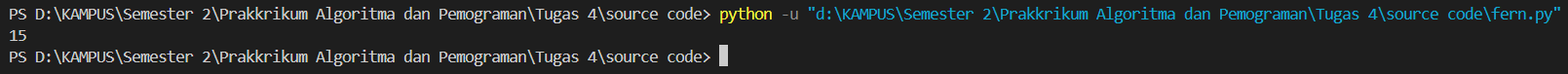
    return hasil

c = tambah(10, 5)

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 ditunjukan pada gambar dibawah.
* Jadi urutan program tersebut secara berturut-turut adalah baris 1-2-3-4-5-6-7-2-3-4-7-8.



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 dibawah

**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:

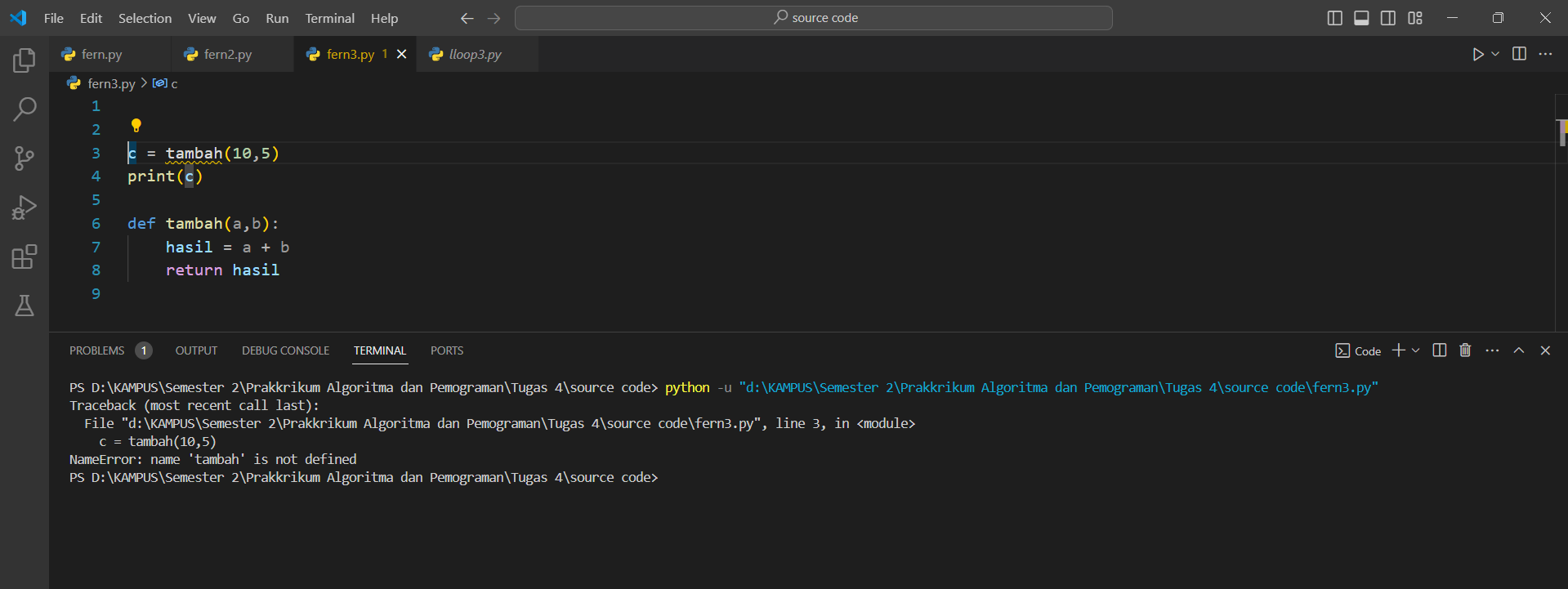
def print\_twice(message):

    print(message)

    print(message)

print\_twice("Hello World")

Fungsi **print\_twice()** memerlukan satu parameter, pesan. Kemudian fungsi **print\_twice()** akan mencetak nilai dari variabel pesan dua kali. Fungsi **print\_twice()** tidak menghasilkan nilai yang dapat digunakan pada proses selanjutnya. Tapi fungsi **print\_twice** akan mengembalikan nilai None. Contohnya sebagai berikut:

  
  
**def print\_twice(message):  
print(message)  
print(message)  
print\_twice("Hello World")**  
Output yang dihasilkan adalah:Hello World!   
Hello World!   
None

Berbeda dengan fungsi tambah() berikut ini:

def tambah(a, b, c):

    hasil = a + b + c

    return hasil

Fungsi tambah() memerlukan 3 parameter: a, b dan c. Kemudian, dalam fungsi tambah(), variabel hasil didefinisikan dan diisi oleh a + b + c. Variabel hasil kemudian dikeluarkan dari fungsi sebagai hasil dari fungsi tambah() dengan kata kunci **return**. Kata kunci return digunakan untuk:

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

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

def tambah(a, b, c):

    hasil = a + b + c

    return hasil

nilai1 = 70

nilai2 = 85

nilai3 = 55

rata\_rata = tambah(nilai1, nilai2, nilai3)/3

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

**Optional Argument dan Named Argument**

Fungsi dapat memiliki opsional parameter, yaitu parameter bersifat opsional dan memiliki nilai default yang telah ditentukan sebelumnya. Untuk mendefinisikan parameter opsional, anda harus menyetel nilai defaultnya terlebih dahulu seperti pada contoh berikut:

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

Panggilan pertama hanya menggunakan satu argumen, sedangkan panggilan kedua dan ketiga menggunakan dua argumen.  
 Setiap parameter fungsi memiliki nama. Oleh karena itu, pemanggilan fungsi juga dapat menyertakan nama parameter dan tidak perlu dalam urutan tertentu.   
Perhatikan dalam contoh berikut:

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)

Fungsi print() dipanggil untuk menyebutkan nama argumennya (named argument) Jika menggunakan metode ini, urutan argumennya tidak harus sama dengan urutan parameter pada fungsi, seperti pada contoh berikut:

def cetak(a, b, c):

    print("Nilai a: ",a)

    print("Nilai b: ",b)

    print("Nilai c: ",c)

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

**Anonymous Function (Lambda)**

Fungsi anonim adalah fungsi yang tidak memiliki nama (anonymous). Fungsi anonymous di Python adalah fitur tambahan, bukan fitur utama. Berbeda dengan bahasa pemrograman seperti Haskell, Lisp dan Erlang merupakan bahasa pemrograman fungsional. Di Python, kata kunci lambda digunakan untuk mendefinisikan fungsi anonymous.

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.

## MATERI 2

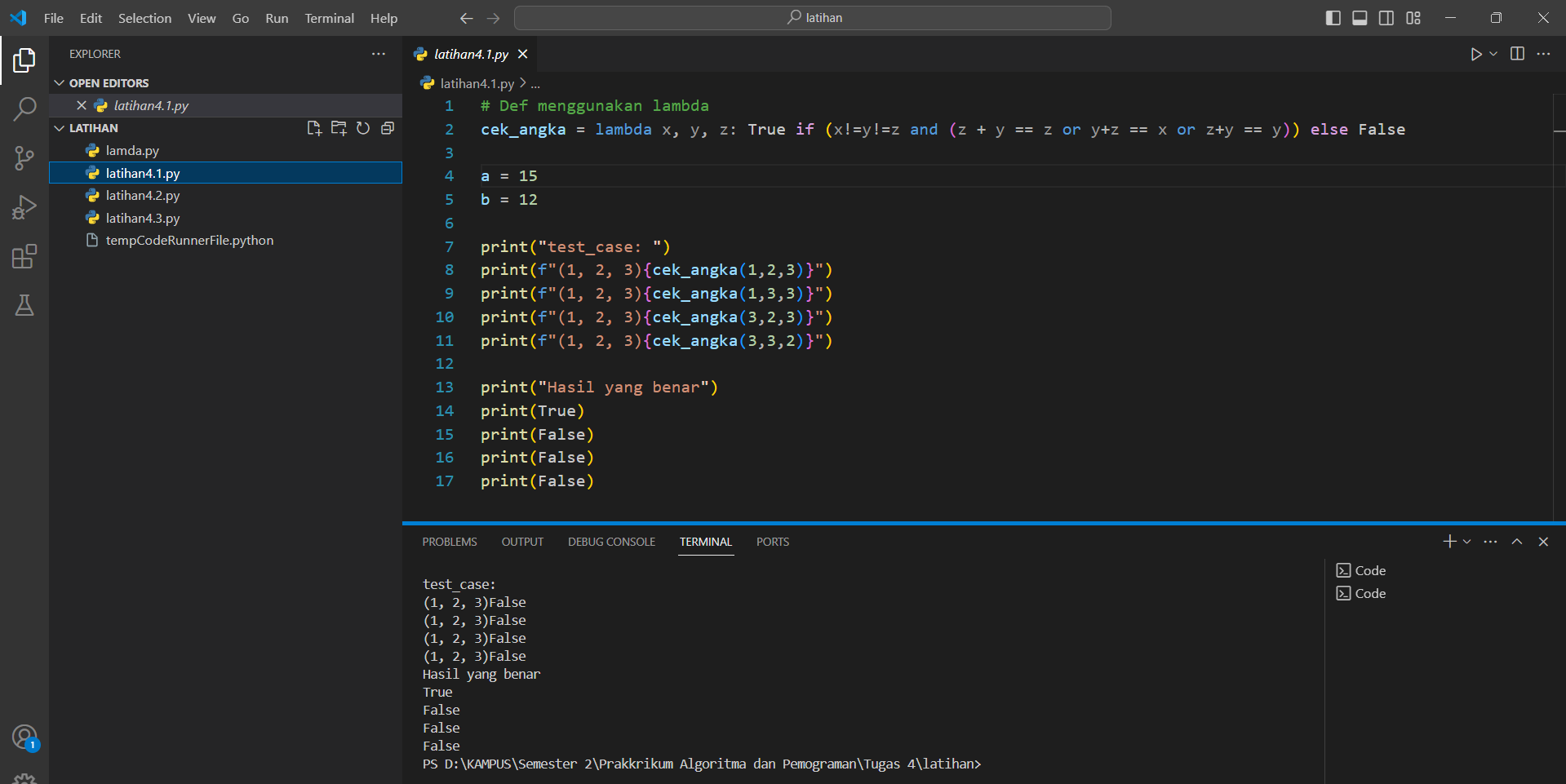
Penjelasan materi 2, dst… sesuai format ini.

# 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.

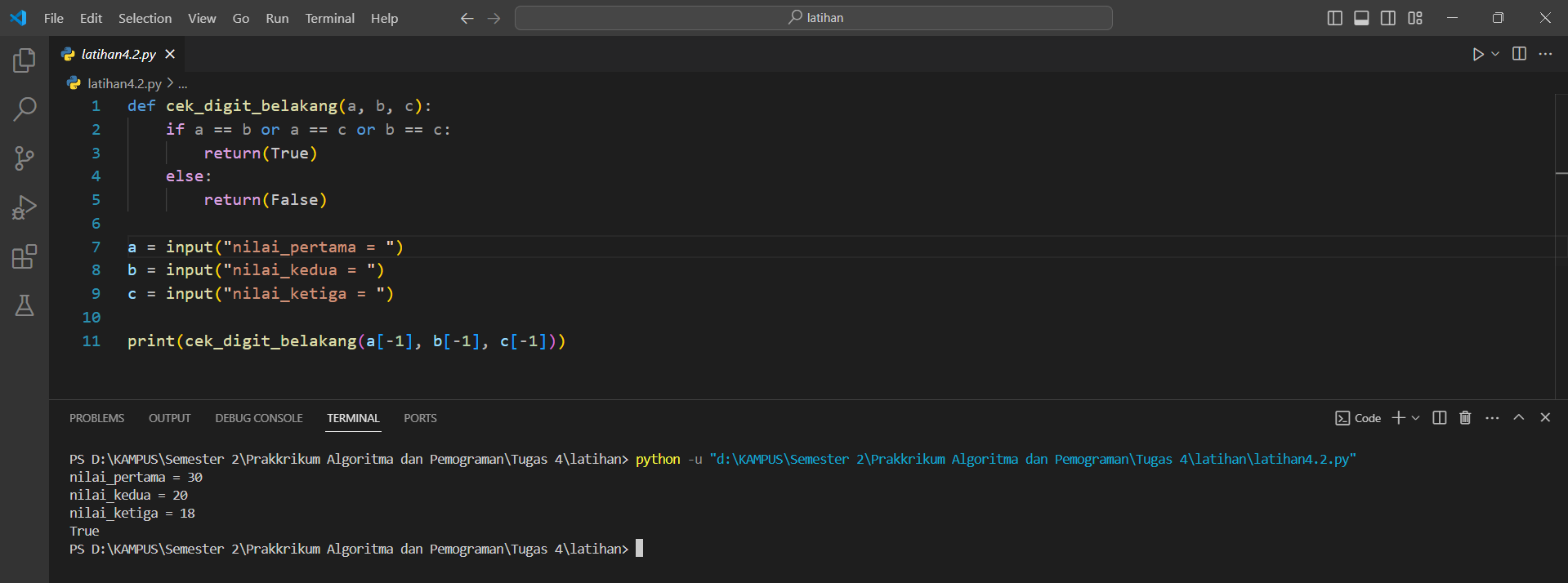
## SOAL 1

**Latihan 4.1**



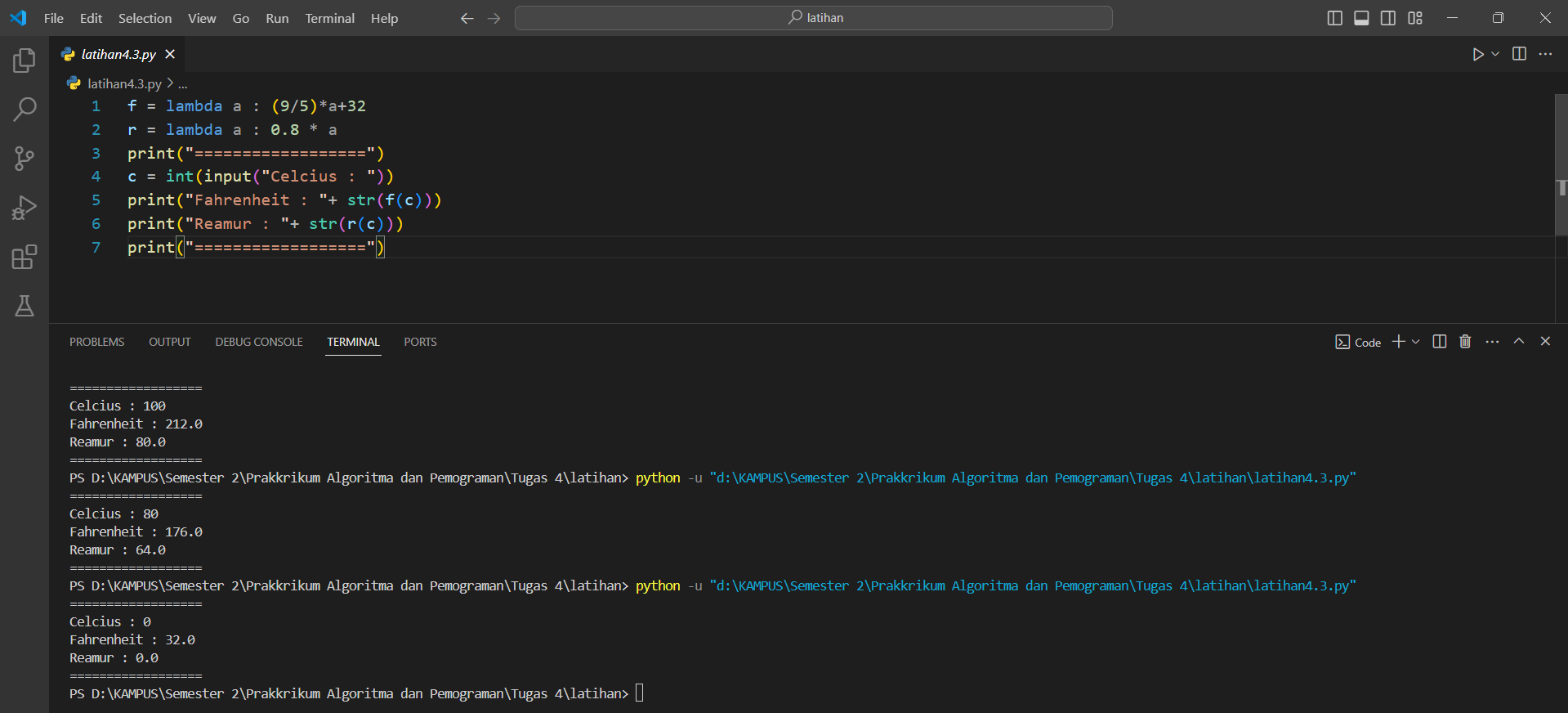
## SOAL 2

**Latihan 4.2**



## SOAL 3

**Latihan 4.3**



Link github : https://github.com/EchoGinDev/tugasAlpro4.git