LAPORAN RESMI

MODUL IV

FUNGSI (FUNCTION)

ALGORITMA PEMROGRAMAN

NAMA	**:**	**KEVIN MALIK FAJAR**
N.R.P	**:**	**200441100014**
DOSEN	**:**	**IMAMAH, S.Kom., M.Kom.**
ASISTEN	**:**	**NOVI LIANA**
TGL PRAKTIKUM	**:**	**2 November 2020**

Disetujui : 8 Desember 2020

Asisten

NOVI LIANA
190441100087

LABORATORIUM BISNIS INTELIJEN SISTEM

JURUSAN SISTEM INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Fungsi digunakan untuk mengumpulkan beberapa perintah yang sering dipakai dalam sebuah program. Fungsi (Function) adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai sub-program (modul program) yang merupakan sebuah program kecil untuk memproses sebagian dari pekerjaan program utama. Dengan memakai fungsi, program yang kita buat menjadi lebih terstruktur. Lebih mudah diikuti oleh orang lain yang membaca program kita. Dan yang paling penting adalah mempersingkat waktu yang kita perlukan untuk mengembangkan suatu perangkat lunak. Karena perangkat lunak yang kita buat, bisa jadi memakai komponen-komponen yang sama.

Pada laporan ini, saya akan membahas mengenai Fungsi (function) pada Bahasa Pemrograman Python. Fungsi digunakan untuk mengumpulkan beberapa perintah yang sering dipakai dalam sebuah program. Fungsi dalam Python didefinisikan menggunakan kata kunci def. Setelah def ada nama pengenal fungsi diikut dengan parameter yang diapit oleh tanda kurung dan diakhir dingan tanda titik dua.

Seperti layaknya sebuah bahasa pemrograman, Python juga memberikan fasilitas pembuatan fungsi yang sangat bagus. Konsep fungsi dalam Python sama dengan bahasa pemrograman C/C++. Python menganggap fungsi dan prosedur adalah sesuatu yang sama, dalam artian cara mendeklarasikan fungsi dan prosedur adalah sama. Hanya bedanya, kalau fungsi mengembalikan suatu nilai setelah proses sedangkan prosedur tidak.

1.2 Tujuan

- Dapat mengetahui dan memahami fungsi dalam Python
- Dapat mengetahui bentuk umum dari fungsi
- Dapat menggunakan dan mendeklarasikan fungsi
- Dapat membuat program sederhana untuk fungsi

BAB II
DASAR TEORI

2. Dasar Teori

2.1 Pengertian Fungsi

Fungsi (Function) adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai sub-program (modul program) yang merupakan sebuah program kecil untuk memproses sebagian dari pekerjaan program utama. Fungsi digunakan untuk mengumpulkan beberapa perintah yang sering dipakai dalam sebuah program.

Fungsi juga bisa diartikan sebagai bagian dari program yang dapat digunakan kembali. Hal ini bisa dicapai dengan memberi nama pada blok statemen, kemudian nama ini dapat dipanggil di manapun dalam program. Kita telah menggunakan beberapa fungsi builtin seperti range.

Fungsi dalam Python didefinisikan menggunakan kata kunci def. Setelah def ada nama pengenal fungsi diikut dengan parameter yang diapit oleh tanda kurung dan diakhir dingan tanda titik dua :. Baris berikutnya berupa blok fungsi yang akan dijalankan jika fungsi dipanggil.

```
def halo_dunia():
      print 'Halo Dunia!'

halo_dunia()   # memanggil fungsi halo_dunia
halo_dunia()   # fungsi halo_dunia dipanggil lagi
```

Keuntungan Menggunakan Fungsi

- Program besar dapat di pisah-pisah menjadi program-program kecil melalui function.
- Kemudahan dalam mencari kesalahan-kesalahan karena alur logika jelas dan kesalahan dapat dilokalisasi dalam suatu modul tertentu.
- Memperbaiki atau memodifikasi program dapat dilakukan pada suatu modul tertentu saja tanpa menggangu keseluruhan program.
- Dapat digunakan kembali (Reusability) oleh program atau fungsi lain.
- Meminimalkan penulisan perintah yang sama.

Kategori Fungsi

1. Standard Library Function
   fungsi-fungsi yang telah disediakan oleh Interpreter Python dalam file-file atau librarinya.

   Misalnya: `raw_input()`, `input()`, `print()`, `open()`, `len()`, `max()`, `min()`, `abs()` dll.
2. Programme-Defined Function

   function yang dibuat oleh programmer sendiri. Function ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library buatan programmer itu sendiri.

2.2 Mendeklarasikan dan Memakai Fungsi

Dalam python terdapat dua perintah yang dapat digunakan untuk membuat sebuah fungsi, yaitu

a) Statemen *Def*

Statemen *def* adalah perintah standar dalam python untuk mendefinisikan sebuah fungsi. *def* dalam python merupakan perintah yang executable, artinya function tidak akan aktif sampai python me*running* perintah *def* tersebut

Statemen *def* digunakan untuk mendeklarasikan fungsi. Sedangkan statemen *return* digunakan untuk mengembalikan suatu nilai kepada bagian program yang memanggil fungsi. Bentuk umum untuk mendeklarasikan fungsi adalah sebagai berikut :

```
def <nama_fungsi>(arg1, arg2, arg3, …,argN) :
     <statemen-statemen>
```

Sebuah fungsi diawali dengan statemen *def* kemudian diikuti oleh sebuah *nama_fungsi* nya. Sebuah fungsi dapat memiliki daftar argumen (parameter) ataupun tidak. Tanda titik dua (:) menandakan awal pendefinisian tubuh dari fungsi yang terdiri dari statemen-statemen.

Tubuh fungsi yang memiliki statemen *return* :

```
def <nama_fungsi>(arg1, arg2, arg3, …,argN)

     <statemen-statemen>
     ...
     return <value>
```

Statemen *return* dapat diletakkan di bagian mana saja dalam tubuh fungsi. Statemen *return* menandakan akhir dar pemanggilan fungsi dan akan mengirimkan suatu nilai balik kepada program yang memanggil fungsi tersebut. Statemen *return* bersifat opsional, artinya jika sebuah fungsi tidak memiliki statemen *return*, maka sebuah fungsi tidak akan mengembalikan suatu nilai apapun.

b) **Statemen *Lambda***

Selain statemen *def*, Python juga menyediakan suatu bentuk ekspresi yang menghasilkan objek fungsi. Karena kesamaannya dengan tools dalam bahasa Lisp, ini disebut *lambda* .Seperti *def*, ekspresi ini menciptakan sebuah fungsi yang akan dipanggil nanti, tapi mengembalikan fungsi dan bukan untuk menetapkan nama.

lambda dalam python lebih dikenal dengan nama Anonymous Function (Fungsi yang tidak disebutkan bukanlah sebuah perintah (statemen) namun lebih namanya). *Lambda* kepada ekspresi (expression). Dalam prakteknya, mereka sering digunakan sebagai cara untuk inline definisi fungsi, atau untuk menunda pelaksanaan sepotong kode.

Bentuk umum lambda adalah kata kunci *lambda*, diikuti oleh satu atau lebih argument (persis seperti daftar argumen dalam tanda kurung di *def* header), diikuti oleh ekspresi setelah tandatitik dua:

```
lambda argument1, argument2,... argumentN
:expression using arguments
```

lambda memiliki perbedaan dengan *def* antara lain :

1. *Lambda* adalah sebuah ekspresi, bukan pernyataan. Karena ini, sebuah *lambda* dapat muncul di tempat-tempat *def* tidak diperbolehkan oleh sintaks Python-di dalam daftar harfiah atau pemanggilan fungsi argumen, misalnya. Sebagai ekspresi, *lambda* mengembalikan nilai (fungsi baru) yang opsional dapat diberi nama. Sebaliknya, pernyataan *def* selalu memberikan fungsi baru ke nama di header, bukannya kembali sebagai hasilnya.
2. Tubuh *lambda* adalah ekspresi tunggal, bukan satu blok statemen. Tubuh lambda sama dengan apa yang akan dimasukkan ke dalam statemen *return* dalam tubuh *def*.

Fungsi Rekursif

Fungsi Rekursif merupakan suatu fungsi yang memanggil dirinya sendiri. Artinya, fungsi tersebut dipanggil di dalam tubuh fungsi itu sendiri. Tujuan di lakukan rekursif adalah untuk menyederhanakan penulisan program dan menggantikan bentuk iterasi. Dengan rekursi, program akan lebih mudah dilihat.

Mencari nilai faktorial dari suatu bilangan bulat positif adalah salah satu pokok bahasan yang memudahkan pemahaman mengenai fungsi rekursif.

Scope Variabel

Scope variabel atau cakupan variabel merupakan suatu keadaan dimana pendeklarasian sebuah variabel di tentukan. Dalam scope variabel dikenal dua istilah yaitu local dan global .

- Variabel local **:** ketika variabel tersebut didefinisikan didalam sebuah fungsi (*def*). Artinya, variabel tersebut hanya dapat di gunakan dalam cakupan fungsi tersebut saja
- Variabel global **:** didefinisikan diluar fungsi . Artinya, variabel tersebut dapat digunakan oleh fungsi lain atau pun program utamanya.

BAB III

TUGAS PENDAHULUAN

BAB III
TUGAS PENDAHULUAN

3.1 Pertanyaan

1. Sebutkan dan Jelaskan macam-macam Perintah fungsi?
2. Buatlah contoh soal dan Program Sederhana dengan menggunakan fungsi ?
3. Jelaskan Perbedaan antara Def dan Lamda? Dengan menggunakan Tabel
4. Buatlah Program dengan menggunakan fungsi Len(), max(), min(), abs()!

3.2 Jawaban

1. - Perintah def adalah Perintah standar dalam Python untuk mendefinisikan Sebuah fungsi. def dalam Python merupakan Perintah Yang executable, artinya function tidak akan aktif sampai Python merunning Perintah def tersebut.
   - Perintah lambda menciptakan Sebuah fungsi Yang akan dipanggil nanti, tapi mengembalikan fungsi dan bukan untuk menetapkan nama.
2. Buatlah Program Ucapan Selamat Pagi menggunakan fungsi!

```
def Salam():
    Print('Halo, Selamat Pagi')
Salam()
```

3.

No	Def	Lambda
1.	Def merupakan sebuah Perintah standar	Lambda merupakan sebuah ekspresi, bukan Pernyataan
2.	Pernyataan def selalu memberikan fungsi baru ke nama di header	Lamda mengembalikan nilai (fungsi baru) Yang opsional dapat diberi nama

4. - Fungsi Len():

Buatlah Program Penghitung banyak huruf dalam kata "INDONESIAKU"!

```
ind = "INDONESIAKU"
Print(ind, 'Panjangnya =', len(ind))
```

Output = INDONESIAKU Panjangnya = 11

- Fungsi max():

  Buatlah Program Pencari Nilai tertinggi dari list b : 1, 2, 3, 4, 5

  b = 1, 2, 3, 4, 5

  Print ('Nilai maksimum dari b adalah = ', max(b))

  OutPut : Nilai maksimum dari b adalah = 5

- Fungsi min():

  Buatlah Program Pencari Nilai terendah dari list b : 1, 2, 3, 4, 5

  b = 1, 2, 3, 4, 5

  Print ('Nilai minimum dari b adalah =', min(b))

  OutPut : Nilai minimum dari b adalah = 1

- Fungsi abs():

  Buatlah Program Penghitung nilai absolut dari -100!

  x = -100

  Print ('Nilai absolut dari', x, 'adalah = ', abs(x))

  OutPut : Nilai absolut dari -100 adalah = 100

BAB IV

IMPLEMENTASI

4.1. Implementasi

1. Buatlah program kalkulator dengan menggunakan fungsi.

Source Code:

coding > praktikum > praktikumm1.py > ...

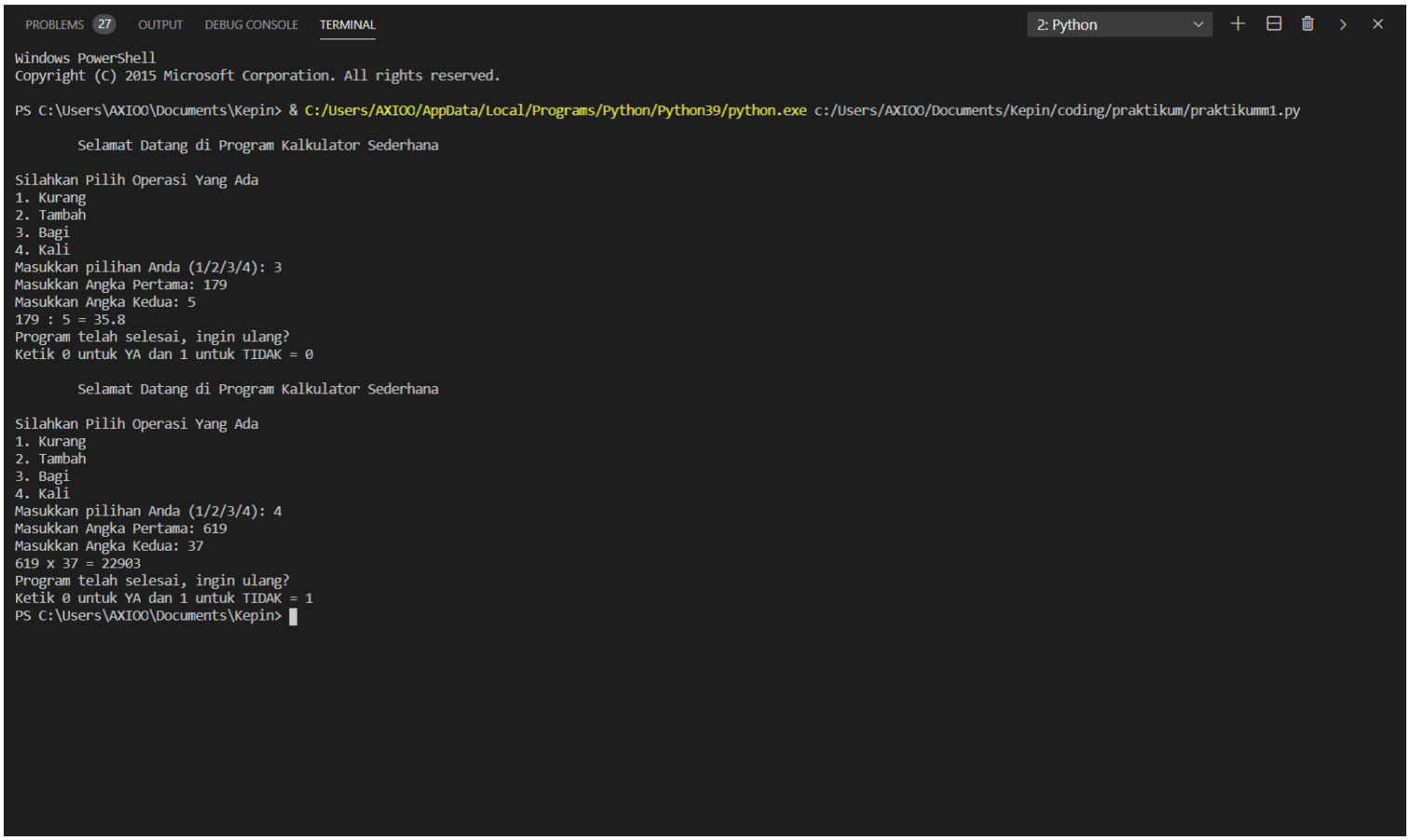
```
def Kalkulator():
    i = 0
    while i == 0:
        print('\n\tSelamat Datang di Program Kalkulator Sederhana')
        print("\nSilahkan Pilih Operasi Yang Ada", "\n1. Kurang", "\n2. Tambah", "\n3. Bagi", "\n4. Kali")

        pilihan2 = int(input('Masukkan pilihan Anda (1/2/3/4): '))
        if pilihan2 == 1:
            angka1 = int(input('Masukkan Angka Pertama: '))
            angka2 = int(input('Masukkan Angka kedua: '))
            print(angka1, "-", angka2, "=", angka1 - angka2)
        elif pilihan2 == 2:
            angka1 = int(input('Masukkan Angka Pertama: '))
            angka2 = int(input('Masukkan Angka Kedua: '))
            print(angka1, "+", angka2, "=", angka1 + angka2)
        elif pilihan2 == 3:
            angka1 = int(input('Masukkan Angka Pertama: '))
            angka2 = int(input('Masukkan Angka Kedua: '))
            print(angka1, ":", angka2, "=", angka1 / angka2)
        elif pilihan2 == 4:
            angka1 = int(input('Masukkan Angka Pertama: '))
            angka2 = int(input('Masukkan Angka Kedua: '))
            print(angka1, "x", angka2, "=", angka1 * angka2)
        else:
            print('Pilihan error, silahkan coba lagi')
            continue

        print('Program telah selesai, ingin ulang?')
        y = 0
        while y == 0:
            ulang = int(input('Ketik 0 untuk YA dan 1 untuk TIDAK = '))
            if ulang == 0:
                y += 1
                break
            elif ulang == 1:
                y += 2
                break
            else:
                print('\nPerintah yang anda masukan error, silahkan coba lagi')
                continue
        if y == 1:
            continue
        else:
            break

Kalkulator()
```

Output :



2. Buatlah program factorial dengan menggunakan fungsi

Source Code :

coding > praktikum > praktikumm2.py > ...

```python
def faktorial():
    i  = 0
    while i == 0:
        angka2 = []
        print('\n\tSelamat Datang di Program Menghitung Bilangan Faktorial')
        angka = int(input('\nSilahkan Masukan angka yang ingin dihitung = '))
        if angka <= 0:
            print('Angka yang dimasukan harus bilangan asli, silahkan coba lagi')
            continue
        else:
            hasil_faktorial = 1
            for p in range(1, angka + 1):
                angka2.append(p)
                hasil_faktorial = hasil_faktorial * p
            print('Faktorial dari', angka, 'adalah', hasil_faktorial, '| Susunan Bilangan Faktorial', angka2)

        print('Program telah selesai, ingin ulang?')
        y = 0
        while y == 0:
            ulang = int(input('Ketik 0 untuk YA dan 1 untuk TIDAK = '))
            if ulang == 0:
                y += 1
                break
            elif ulang == 1:
                y += 2
                break
            else:
                print('\nPerintah yang anda masukan error, silahkan coba lagi')
                continue
        if y == 1:
            continue
        else:
            break

faktorial()
```

Output :

```
PROBLEMS 27  OUTPUT  DEBUG CONSOLE  TERMINAL                                  2: Python

Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\AXIOO\Documents\Kepin> & C:/Users/AXIOO/AppData/Local/Programs/Python/Python39/python.exe c:/Users/AXIOO/Documents/Kepin/coding/praktikum/praktikumm2.py

        Selamat Datang di Program Menghitung Bilangan Faktorial

Silahkan Masukan angka yang ingin dihitung = 79
Faktorial dari 79 adalah 894618213078297528685144171539831652069808216779571907213868063227837990693501860533361810841010176000000000000000000 | Susunan Bilangan Faktorial [1,
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79]
Program telah selesai, ingin ulang?
Ketik 0 untuk YA dan 1 untuk TIDAK = 0

        Selamat Datang di Program Menghitung Bilangan Faktorial

Silahkan Masukan angka yang ingin dihitung = 10
Faktorial dari 10 adalah 3628800 | Susunan Bilangan Faktorial [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Program telah selesai, ingin ulang?
Ketik 0 untuk YA dan 1 untuk TIDAK = 0

        Selamat Datang di Program Menghitung Bilangan Faktorial

Silahkan Masukan angka yang ingin dihitung = 3
Faktorial dari 3 adalah 6 | Susunan Bilangan Faktorial [1, 2, 3]
Program telah selesai, ingin ulang?
Ketik 0 untuk YA dan 1 untuk TIDAK = 0

        Selamat Datang di Program Menghitung Bilangan Faktorial

Silahkan Masukan angka yang ingin dihitung = 7
Faktorial dari 7 adalah 5040 | Susunan Bilangan Faktorial [1, 2, 3, 4, 5, 6, 7]
Program telah selesai, ingin ulang?
Ketik 0 untuk YA dan 1 untuk TIDAK = 1
PS C:\Users\AXIOO\Documents\Kepin>
```

BAB V
PENUTUP

5.1 Analisa

Dalam python terdapat dua perintah yang dapat digunakan untuk membuat sebuah fungsi , yaitu statemen def dan statemen lambda . Statemen def untuk mendeklarasikan fungsi . Statemen def dalam python merupakan perintah yang executable , artinya fungsi tidak akan aktif sampai python merunning perintah def tersebut . sebuah fungsi diawali dengan statemen def kemudian diikuti oleh sebuah nama_fungsi nya. Sebuah fungsindapat memiliki daftar argument(parameter) ataupun tidak Tanda titik dua(:) menandakan awal pendefisian tubuh dari fungsi yang terdiri dari statemen- statemen.

Sedangkan statemen lambda digunakan untuk menghasilkan objek fungsi . seperti def , statemen lambda menciptakan sebuah fungsi yang akan dipamggil nanti , tetapi mengembalikan fungsi dan bukan untuk menetapkan nama sehingga lambda.

Selain itu terdapat fungsi Rekursif yang merupakan suatu fungsi yang memanggil dirinya sendiri . Artinya , fungsi tersebut dipanggil didalam tubuh fungsi itu sendiri . Tujuan dilakukan rekursif adalah untuk menyederhanakan penulisan program dan menjadikan bentuk iterasi . Dengan rekursi , program akan lebih mudah dilihat . Contoh dari fungsi rekursif yaitu mencari nilai factorial.

5.2 Kesimpulan

Dalam belajar pemrograman terlebih dahulu harus mengerti tentang macam - macam dan fungsinya setiap perintah. Dalam belajar pemrograman operator harus memahami, mengingat, meneliti symbol - symbol yang dijadikan sebagai operator masing - masing

Dalam Fungsi (function) sangat berguna untuk meminimalkan penulisan perintah yang sama terutama pada program dengan perintah yang sering dipanggil ulang sehingga penulisan perintah menjadi lebih efisien .