



UNS
UNIVERSITAS
SEBELAS MARET



MODUL PRAKTIKUM PROGRAMA KOMPUTER

**PROGRAM STUDI TEKNIK INDUSTRI
UNIVERSITAS SEBELAS MARET**

**TIM ASISTEN LABORATORIUM
PERANCANGAN DAN OPTIMASI
SISTEM INDUSTRI 2020**

MODUL VI

FUNGSI

A. Tujuan

Berikut merupakan tujuan Praktikum Programa Komputer Modul VI.

1. Mampu memahami konsep dari fungsi dalam pemrograman
2. Mampu menerapkan penggunaan dari fungsi dalam pemrograman
3. Mampu memahami serta menerapkan fungsi *built-in* pada bahasa pemrograman Python

B. Pengertian Fungsi

Fungsi adalah merupakan suatu blok yang berisi kode program yang dirancang untuk melaksanakan tugas khusus. Fungsi dibutuhkan untuk memudahkan proses pemrograman terutama untuk menyelesaikan keputusan yang spesifik. Dalam pemrograman, fungsi sering disebut sebagai subprogram. Keuntungan dalam penggunaan fungsi adalah :

1. Kode program yang kita buat lebih mudah untuk dipahami.
Memecah program menjadi sub-sub fungsi program dengan tugas spesifik tertentu, membuat pemrogram mengetahui dengan cepat alur proses pengolahan data. Selain itu jika terjadi error, dapat diketahui lebih cepat sumber errornya.
2. Fungsi mudah untuk digunakan kembali.

Suatu fungsi dapat digunakan kembali dengan hanya cukup memanggil fungsi tersebut sesuai nama dan parameternya, maka fungsi tersebut akan menjalankan perintah.

3. Menghemat baris perintah.

Programmer tidak perlu lagi membuat deretan perintah secara berulang untuk menyelesaikan suatu masalah yang sama. Cukup dengan memanggil fungsi yang sudah ada.

C. Jenis-jenis Fungsi

Ada dua jenis fungsi dalam pemrograman Python:

1. ***Standard library functions* atau *built-in function*** - Ini adalah fungsi bawaan dalam Python yang tersedia untuk digunakan. Contoh: **print()**, **sqrt()**, **pow()**, dll.
2. ***User-defined functions*** – User dapat membuat fungsi sendiri berdasarkan kebutuhan.

D. Membuat dan Memanggil Fungsi

Pada Python, untuk membuat fungsi adalah dengan menggunakan kata kunci **def()**

Secara keseluruhan, seperti inilah penjelasan dari setiap bagian fungsi:

```
def function_name(argumen):  
    # badan fungsi  
  
    return
```

- **def** – kata kunci untuk melakukan deklarasi fungsi
- **function_name** – nama yang diberikan pada fungsi
- **argumen** – nilai yang akan dimasukan pada fungsi
- **return** (optional) – mengembalikan nilai dari fungsi

Agar lebih mudah dipahami, perhatikan contoh berikut ini:

```
def salam():  
    print("Halo Praktikan!")
```

Di atas, telah dibuat sebuah fungsi bernama **salam()** dengan perintah mencetak teks **Halo Praktikan!**. Fungsi ini tidak memiliki argumen apa pun dan tidak mengembalikan nilai apa pun.

Untuk menggunakannya, fungsi tersebut harus dipanggil.

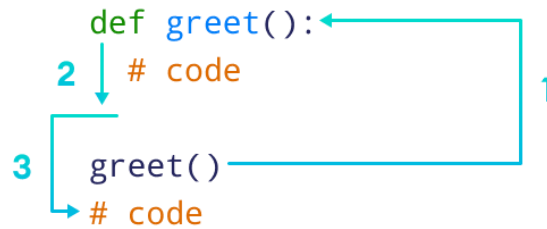
```
def salam():  
    print("Halo Praktikan!")  
  
# memanggil fungsi  
salam()
```

```
Halo Praktikan!
```

Selain itu, cara kerja fungsi juga harus diperhatikan, seperti pada contoh di bawah:

```
def greet():  
    print('Hello World!')  
  
# memanggil fungsi  
greet()  
  
print('Ada di luar fungsi')
```

```
Hello World!  
Ada di luar fungsi
```



- 1) Saat fungsi dipanggil, kontrol program beralih ke definisi fungsi.
- 2) Semua kode di dalam fungsi dijalankan.
- 3) Kontrol program melompat ke pernyataan berikutnya setelah pemanggilan fungsi.

E. Fungsi dengan Argumen

Seperti disebutkan sebelumnya, suatu fungsi juga dapat memiliki argumen. Argumen adalah nilai yang diteruskan di dalam tanda kurung fungsi. Suatu fungsi dapat memiliki sejumlah argumen yang dipisahkan oleh koma.

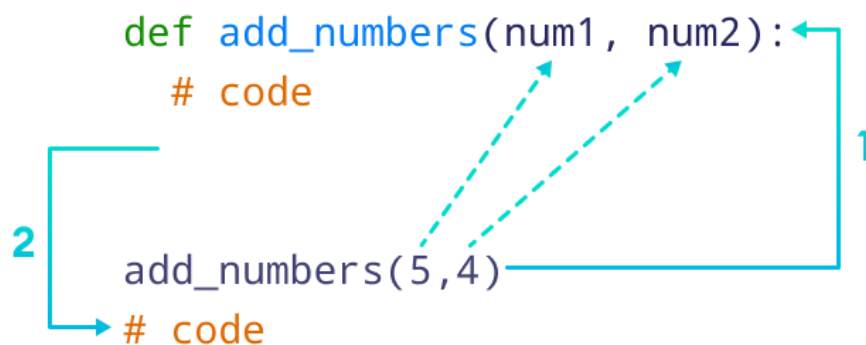
Secara keseluruhan, berikut contoh dari penggunaan fungsi argumen:

```
# Contoh keseluruhan
def add_numbers(num1, num2):
    sum = num1 + num2
    print("Sum: ",sum)

add_numbers(5, 4)
```

Sum: 9

Pada contoh di atas, kita telah membuat fungsi bernama `add_numbers()` dengan argumen: `num1` dan `num2`.



Python mendukung berbagai jenis argumen yang dapat diteruskan pada saat pemanggilan fungsi. Di Python, kami memiliki 4 jenis argumen fungsi berikut.

- **Default argument**

Argumen default adalah argumen yang mengasumsikan nilai default jika nilai tidak diberikan dalam pemanggilan fungsi untuk argument tersebut. Perhatikan Contoh 1 - 4 untuk membantu memahami *default argument*.

```
# Contoh 1
def say_hello(nama = "Ganteng"):
    '''fungsi dengan default argument'''
    print(f"Hallo {nama}")

say_hello("Cakep")
say_hello()
```

✓ 0.0s

Hallo Cakep
Hallo Ganteng

```
# Contoh 2
def sapa_dia(nama, pesan = "Apa kabar?"):
    '''fungsi dengan satu input biasa,
    dan satu default argument'''
    print(f"Hai {nama}, {pesan}")

sapa_dia("Ucup", "kamu Ganteeeng")
sapa_dia("Kamu")
```

✓ 0.0s

Hai Ucup, kamu Ganteeeng
Hai Kamu, Apa kabar?

```
# Contoh 3
def hitung_pangkat(angka, pangkat=2):
    hasil = angka**pangkat
    return hasil

print(hitung_pangkat(2,4))

hasil = hitung_pangkat(pangkat=3, angka=5)
print(hasil)
```

✓ 0.0s

16
125

```
# Contoh 4
def fungsi(input1=1,input2=2,input3=3,input4=4):
    hasil = input1 + input2 + input3 + input4
    return hasil

print(fungsi())
print(fungsi(input3=10))

✓ 0.0s
10
17
```

- **Keyword arguments (named arguments)**

Dengan menggunakan named parameter, kita tidak harus bergantung kepada urutan parameter pada saat menjalankan sebuah fungsi. Urutan argumen bisa ditulis acak selama nama argumen sama dengan nama parameter.

```
def student(firstname, lastname):
    print(firstname, lastname)

# Keyword arguments
student(firstname='POSI', lastname='2020')
student(lastname='2020', firstname='POSI')
```

```
POSI 2020
POSI 2020
```

- **Positional arguments**

Argumen posisi adalah argumen yang diteruskan ke fungsi dalam urutan tertentu. Urutan akan penting karena fungsi akan menggunakan argumen dalam urutan yang diterima.

```
def identitas(nama, umur):
    print("Hi, Saya", nama)
    print("Umur saya adalah", umur)

# Hasilnya adalah benar karena urutannya benar
print("Case-1:")
identitas("Andeca", 27)

# Bandingkan dengan yang ke dua ini
print("\nCase-2:")
identitas(27, "Andeca")
```

```
Case-1:
Hi, Saya Andeca
Umur saya adalah 27

Case-2:
Hi, Saya 27
Umur saya adalah Andeca
```

- **Arbitrary arguments (*args dan **kwargs)**

- ***args**

Parameter `*args` adalah sintaks yang digunakan saat mendefinisikan parameter panjang dari sebuah variabel. Parameter ini dapat menampung banyak argumen sesuai dengan yang diinginkan di dalam program. Simbol `*` artinya akan membuat parameter menjadi tuple.

```
def tambah(*data):
    # data tipenya adalah tuple
    output = 0
    for angka in data:
        output += angka
    return output
hasil = tambah(10,5,15,20)
print(f"hasil = {hasil}")
✓ 0.0s
hasil = 50
```

- ****kwargs**

Nama `kwargs` sendiri merupakan singkatan dari “*keyword arguments*” dimana parameter `*kwargs` adalah sintaks yang digunakan dalam mengubah semua keyword argument (`key=val`) kedalam *dictionary*. Cara penggunaan sintaks ini seperti mengakses *dictionary*. Simbol `**` artinya akan membuat parameter menjadi *dictionary*.

```
#membuat fungsi cetak dengan argumen *kwargs
def cetak(**mahasiswa):
    print(mahasiswa)

#memanggil fungsi cetak
cetak(nama="Aji", alamat="Cirebon")
cetak(nama="Hani", alamat="Bogor")
✓ 0.0s
{'nama': 'Aji', 'alamat': 'Cirebon'}
{'nama': 'Hani', 'alamat': 'Bogor'}
```

F. Fungsi dengan Pengembalian (*return value*)

Fungsi dengan pengembalian sudah muncul di beberapa contoh di atas.

Dalam Fungsi Python, jika kita ingin fungsi kita mengembalikan beberapa nilai ke pemanggilan fungsi, kita menggunakan pernyataan **return**. Misalnya,

```
def find_square(num):
    result = num * num
    return result

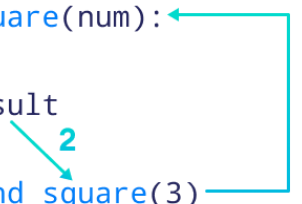
square = find_square(3)

print('Square:', square)

# Output = Square: 9
```

```
def find_square(num):
    # code
    return result

Square = find_square(3)
# code
```



G. List pada Argumen Fungsi

Tipe data apapun (string, integer, list, dictionary, dll.) dapat dikirim argumen ke suatu fungsi, dan itu akan diperlakukan sebagai tipe data yang sama di dalam fungsi.

Misalnya, jika ingin mengirim List sebagai argumen, itu akan tetap menjadi List saat mencapai fungsi:

```
def my_function(food):
    for x in food:
        print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

apple
banana
cherry

H. Fungsi Rekursi

Fungsi rekursi adalah fungsi yang dapat memanggil dirinya sendiri secara berulang-ulang hingga suatu kondisi yang di definisikan terpenuhi atau bernilai benar (**True**).

Fungsi rekursi juga bisa disebut perulangan rekursi (karena fungsinya memang sama seperti *looping*) perbedaannya adalah jika pada perulangan iteratif menggunakan instruksi perulangan seperti **for** dan **while**, sementara pada fungsi rekursi perulangan dilakukan pada fungsi itu sendiri.


```
def faktorial(a):
    if a == 1:
        return (a)
    else:
        return (a*faktorial(a-1))

bil = int(input("Masukan Bilangan : "))

print("%d! = %d" % (bil, faktorial(bil)))
```

3! = 6

Fungsi **faktorial()** merupakan fungsi rekursi karena di dalam fungsi tersebut terdapat pemanggilan fungsi-nya sendiri. Ketika pengguna memasukkan bilangan bil maka nilai tersebut akan di kirim ke fungsi **faktorial()** lewat argumen a. Selama nilai a tidak sama dengan 1, fungsi **faktorial()** akan terus melakukan pemanggilan dirinya sendiri. Perulangan akan berhenti ketika nilai adalah 1.

I. Fungsi *Built-in* pada Python

Fungsi *Built-in* adalah fungsi yang sudah tersedia atau sudah disediakan oleh *python* dan dapat digunakan tanpa harus mengimpor terlebih dahulu fungsi tersebut. Maka dari itu, dapat dikatakan bahwa fungsi *Built-in* adalah fungsi bawaan python. Dalam penggunaannya, python menyediakan banyak fungsi bawaan. Fungsi bawaan tersebut dapat dikategorikan menjadi 3, seperti sebagai berikut:

1. *Built-in* konversi
2. *Built-in* tipe
3. *Built-in* selain 2 jenis di atas.

Ketiga kategori *built-in* tersebut akan dijelaskan lebih lanjut sebagai berikut.

1. Kategori *Built-in* konversi

Kategori *built-in* ini berfokus pada fungsi mengkonversi suatu tipe data menjadi tipe data yang lain. Beberapa fungsi sudah dipelajari pada modul sebelumnya seperti fungsi **bin()** untuk melakukan konversi dari bilangan integer ke biner. Sama halnya dengan fungsi **hex()** mengubah bilangan integer menjadi string hexadesimal dan fungsi **oct()** yang mengubah bilangan integer menjadi string oktal. Berikut merupakan beberapa fungsi *built-in* kategori konversi yang akan dibahas pada modul ini.

- Fungsi **ascii()**

ASCII merupakan singkatan dari (American Standard Code for Information Interchange). Fungsi **ascii()** akan melakukan eksekusi ke objek yang terdaftar dalam ASCII. Jika terdapat karakter non-ASCII dalam objek tersebut, maka fungsi **ascii()** akan mengembalikan nilai tersebut menjadi sebuah kode unik yang dapat disimpan dan diolah di komputer. Berikut merupakan contoh penggunaan fungsi **ascii()**.

```
teks = 'Aku mau makan nih'
print(ascii(teks))

teks_alay = 'Âku maÛ maîn yöyö'
print(ascii(teks_alay))
```

✓ 0.0s

```
'Aku mau makan nih'
'\u0202ku ma\u0216 ma\u020an y\u04e7y\u04e7'
```

- Fungsi **chr()** dan **ord()**

Fungsi **chr()** berguna untuk mengubah data integer menjadi sebuah string yang berupa kode unik. Jadi kode unik tersebut akan menyimpan sebuah data integer didalamnya dan dapat diakses kembali menggunakan fungsi **ord()**. Dalam ASCII, terdapat 256 non-printable karakter yang dimulai dari 0 hingga 255. Fungsi **chr()** mengambil karakter yang terdaftar pada ASCII untuk menyimpan nilai integer. Contoh penggunaannya adalah sebagai berikut.

```
integer = 78
print(integer)
print(type (integer))

ubah = chr(integer)
print(ubah)
print(type (ubah))

kembali = ord(ubah)
print(kembali)
print(type (kembali))
```

✓ 0.0s

```
78
<class 'int'>
N
<class 'str'>
78
<class 'int'>
```

2. Kategori *Built-in* Tipe

Kategori *built-in* ini berfokus pada fungsi menentukan tipe sebuah variabel ataupun mengubahnya. Beberapa fungsi sudah dipelajari pada modul sebelumnya seperti fungsi **str()** untuk membuat variabel string, fungsi **bool()** untuk menghasilkan output boolean (True atau False), **int()** untuk membuat variabel integer, dan lain-lain seperti **float()**, **list()**, **tuple()**, **range()**, dan masih banyak lagi.

Berikut merupakan beberapa fungsi *built-in* kategori tipe yang akan dibahas pada modul ini.

- Fungsi **bytes()**

Fungsi **bytes()** berfungsi untuk membuat objek berupa byte yang tidak dapat diubah-ubah. Hampir sama dengan string, bytes menggunakan urutan integer 8 bit dalam range $0 < x < 256$ seperti dalam ASCII. Contoh penggunaannya adalah seperti berikut.

```

y = bytes()
print(y)

x = bytes(5)
print(x)
print(type(x))

list = bytes([2,3,4,6])
print(list)

angka_besar = bytes([3,2,1,300])
print(angka_besar)

```

⊗ 0.0s

```

b''
b'\x00\x00\x00\x00\x00'
<class 'bytes'>
b'\x02\x03\x04\x06'

```

```

ValueError                                Traceback (most recent call last)
Cell In[18], line 11
      8 list = bytes([2,3,4,6])
      9 print(list)
--> 11 angka_besar = bytes([3,2,1,300])
     12 print(angka_besar)

ValueError: bytes must be in range(0, 256)

```

3. Kategori *Built-in* lainnya

Kategori fungsi bawaan ini tidak terkategoriikan secara spesifik berdasarkan kegunaannya karena cakupannya yang sangat luas tetapi sama pentingnya dengan dua kategori diatas, contoh fungsi dari kategori ini adalah **min()**, **max()**, **abs()**, **len()**, **type()**, **map()**, dll. Berikut merupakan beberapa contoh penulisannya:

- Fungsi **min()** dan **max()**

Fungsi ini memiliki kegunaan untuk mengembalikan item terkecil (min) atau terbesar (max) dalam tipe data iterable

```
1 list = [3, 4, 7, 11, 18, 29, 42]
2 print(f"Nilai maksimal adalah: {max(list)}")
3 print(f"Nilai minimal adalah: {min(list)}")
✓ 0.1s

Nilai maksimal adalah: 42
Nilai minimal adalah: 3
```

- Fungsi **type()**

Fungsi ini memiliki kegunaan untuk mengecek tipe data pada objek atau variabel

```
1 angka = [3, 4, 7, 11, 18, 29, 42]
2 print(f"Tipe data dari variabel 'angka' adalah: {type(angka)}")
✓ 0.1s

Tipe data dari list adalah: <class 'list'>
```

- Fungsi **len()**

Fungsi ini memiliki kegunaan untuk mengembalikan panjang objek

```
1 angka = [3, 4, 7, 11, 18, 29, 42]
2 print(f"panjang list adalah: {len(angka)}")
✓ 0.8s

panjang list adalah: 7
```

- Fungsi **map()**

Fungsi ini memiliki kegunaan untuk mengembalikan iterator yang ditentukan dengan fungsi yang ditentukan diterapkan ke setiap item

```
1 angka_list = []
2 angka = input("Masukkan angka: ")
3 angka_list = angka.split(",")
4
5 angka_list = list(map(int, angka_list))
6 print(angka_list)
✓ 7.9s

[10, 13, 15]
```

Pada contoh diatas, fungsi map digunakan untuk mengubah input user menjadi list

Referensi

https://www.w3schools.com/python/python_functions.asp

<https://core-electronics.com.au/guides/functions-of-python/>

https://www.w3schools.com/python/python_ref_functions.asp

<https://www.programiz.com/python-programming/function>

<https://www.geeksforgeeks.org/python-functions/>

<https://www.duniailkom.com/tutorial-belajar-bahasa-pemrograman-python-untuk-pemula/>

<https://youtube.com/playlist?list=PLZS-MHyEIRo7cgStrKAMhgnOT66z2qKz1>