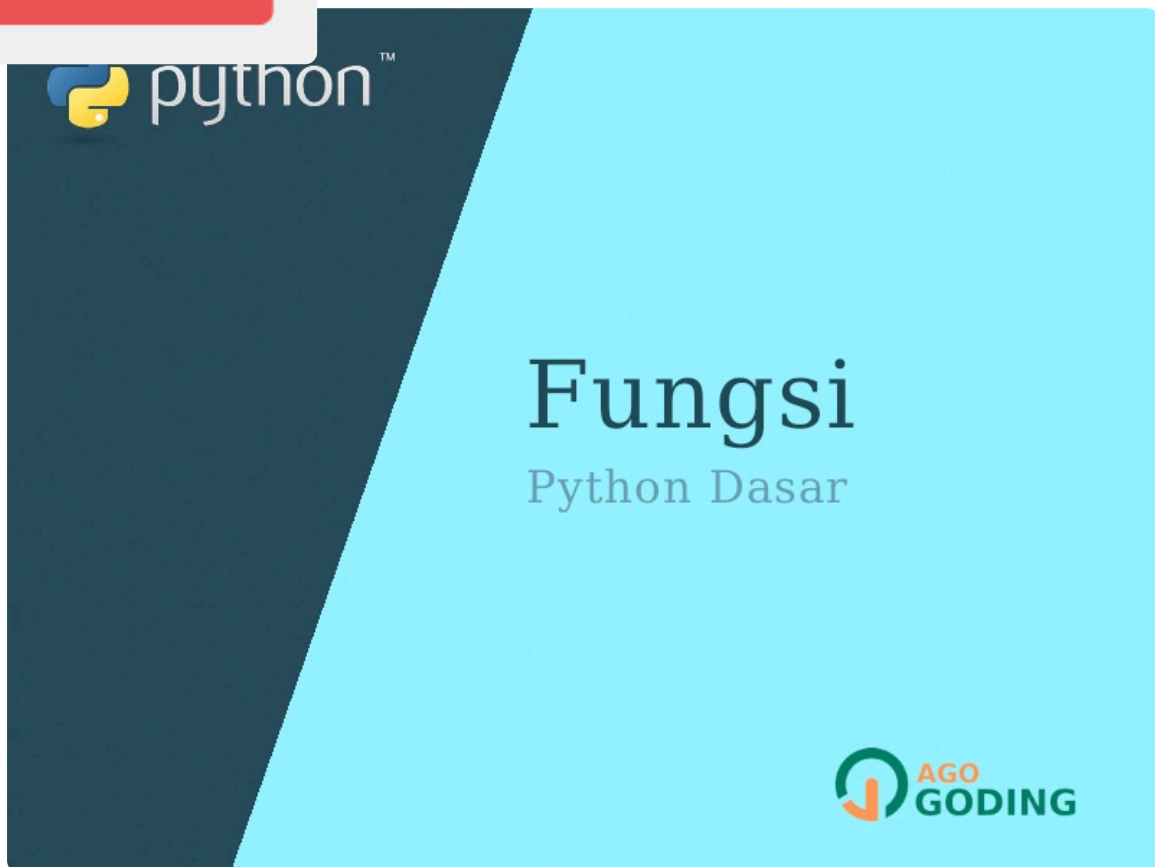




Fungsi (def) 🐍

[Python](#)[Python Dasar](#)

Daftar Isi

[Pengertian Fungsi \(def\) Pada Python](#)[Sintaks Fungsi](#)[Memanggil Fungsi](#)[Fungsi dengan Argumen atau Parameter](#)[Parameter Wajib](#)[Parameter Opsional \(atau Default\)](#)[Fungsi Dengan Parameter Tidak Berurut](#)[Fungsi yang Mengembalikan Nilai](#)

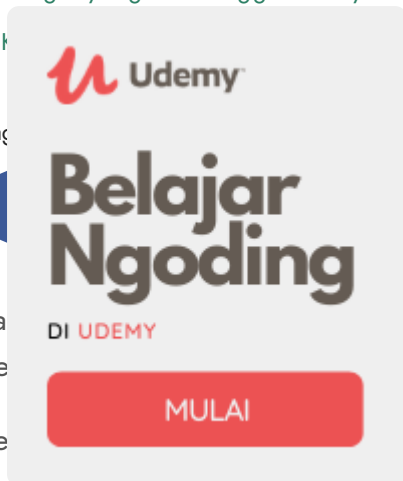
Fungsi yang Memanggil Dirinya Sendiri

Bag

Pa

pe

Se



membahas tentang fungsi pada python, jenis-jenisnya, dan juga contoh cara

as pengertiannya terlebih dahulu.

Pengertian Fungsi (def) Pada Python

Apa itu fungsi (def) pada python?

Fungsi pada python adalah kumpulan perintah atau baris kode yang dikelompokkan menjadi satu kesatuan untuk kemudian bisa dipanggil atau digunakan berkali-kali.

Sebuah fungsi bisa menerima parameter, bisa mengembalikan suatu nilai, dan bisa dipanggil berkali-kali secara independen.

Dengan fungsi kita bisa memecah program besar yang kita tulis, menjadi bagian-bagian kecil dengan tugasnya masing-masing.

Juga, fungsi akan membuat kode program kita menjadi lebih *“reusable”* dan lebih terstruktur.

Sintaks Fungsi

Di dalam python, sintaks pembuatan fungsi terlihat seperti berikut:

```
def <nama_fungsi>(parameters):  
    statements
```

Sintaks di atas secara umum terbagi menjadi 4 bagian:

1. Kata kunci `def` yang menjadi pertanda bahwa blok kode program adalah sebuah fungsi
2. Nama fungsi yang kita buat
3. Parameters yang akan diterima oleh fungsi yang kita buat (tidak wajib)
4. Dan blok kode fungsi yang di sana akan kita tulis perintah-perintah yang harus dilakukan oleh sebuah fungsi

Oiya: jangan lupa bahwa **blok kode program di dalam python didefinisikan dengan indentasi**. Silakan baca [aturan sintaks python](#) untuk lebih lengkapnya.

```
def halo_dunia():  
    print('Halo python! Halo dunia')
```

Fungsi ini akan mengeksekusi perintah `print()` yang ada di dalamnya.

Mengapa kita menggunakan `print()` yang telah kita definisikan?

Sebelumnya, kita sudah pernah menambahkan dengan tanda kurung `()` seperti berikut:

```
halo_dunia()
```

Output:

```
Halo python! Halo dunia
```

Bahkan kita bisa memanggil fungsi `halo_dunia()` berkali-kali:

```
halo_dunia()  
halo_dunia()  
halo_dunia()
```

Output:

```
Halo python! Halo dunia  
Halo python! Halo dunia  
Halo python! Halo dunia
```

Fungsi dengan Argumen atau Parameter

Sebuah fungsi juga bisa menerima parameter atau pun argumen. Ia merupakan suatu nilai/variabel yang dilemparkan ke dalam fungsi untuk diproses lebih lanjut.

Sebagai contoh, perhatikan **output** berikut:

```
Halo Nurul, selamat datang!  
Halo Lendis, selamat datang!
```

La... produksi output seperti itu dengan python?

Ac... , perulangan, dan lain sebagainya.

Ak... yang terbesit dalam benak kita adalah dengan melakukan 4x `print()` seperti

ini



```
ng!')
ang!')
ng!')
!')
```

Itu adalah cara yang sangat simpel, dan juga tidak salah.

Akan tetapi, dari pada kita melakukan 4x print seperti di atas, kita bisa memanfaatkan **fungsi dan parameter** pada python.

Sehingga kode programnya akan terlihat seperti ini:

```
def selamat_datang (nama):
    print(f'Halo {nama}, selamat datang!')

selamat_datang('Nurul')
selamat_datang('Lendis')
selamat_datang('Fabri')
selamat_datang('isa')
```

Dan kita tetap akan mendapatkan output yang sama. Lebih elegan bukan 😎

Parameter Wajib

Parameter di dalam python bisa lebih dari satu, bisa wajib semua (harus diisi), dan bisa juga bersifat opsional.

Perhatikan contoh fungsi berikut:

```
def perkenalan (nama, asal):
    print(f"Perkenalkan saya {nama} dari {asal}")
```

Jika dipanggil:

```
perkenalan("Renza Ilhami", "Jawa Timur")
```

Perkenalkan saya Renza Ilhami dari Jawa Timur

Tapi jika kita memanggil fungsi dengan parameter tidak lengkap, justru kita akan mendapatkan error:

Error: `TypeError: positional argument: 'asal'`



Kenapa? Karena kita hanya memasukkan satu parameter saja padahal parameter yang diminta ada 2.

Parameter Opsional (atau Default)

Tidak semua parameter fungsi pada python itu bersifat wajib. Ada yang opsional.

Parameter opsional adalah parameter yang seandainya tidak diisi, dia sudah memiliki nilai default.

Perhatikan contoh berikut:

```
def suhu_udara (daerah, derajat, satuan = 'celcius'):
    print(f"Suhu di {daerah} adalah {derajat} {satuan}")
```

Pada fungsi `suhu_udara()` di atas, kita mendefinisikan 3 buah parameter:

- `daerah`
- `derajat`
- `suhu = 'celcius'`

Dua parameter pertama adalah bersifat wajib dan harus diisi, sedangkan parameter ketiga tidak wajib. Jika tidak kita isi, maka nilai default-nya adalah "celcius".

Sekarang, kita coba panggil fungsi tersebut dengan 2 cara:

```
suhu_udara("Surabaya", 30)
suhu_udara("Surabaya", 86, 'Fahrenheit')
```

Jika dijalankan, outputnya akan terlihat seperti ini:

```
Suhu di Surabaya adalah 30 celcius
Suhu di Surabaya adalah 86 Fahrenheit
```

Jika kita perhatikan lagi fungsi `suhu_udara()`, kita akan dapati kalau parameter yang bersifat opsional hanya ada 1.

Tapi kalau parameter opsionalnya ada lebih dari 1?

Contoh:



**Belajar
Ngoding**

DI UDEMY

MULAI

```
= 30, satuan = 'celcius':  
    print(f"Suhu di {daerah} adalah {suhu} {satuan}")
```

Padahal kita ingin mengatur nilai default untuk parameter `derajat`. Sehingga sekarang kita memiliki dua buah parameter.

Kita coba panggil dengan 2 parameter seperti ini:

```
suhu_udara('Jakarta', 'fahrenheit')
```

Apa outputnya?

```
Suhu di Jakarta adalah fahrenheit celcius
```

Hmmm. Kok gitu?

Padahal kita inginnya hanya mengisi 2 parameter saja:

1. Satu untuk parameter `daerah`
2. Dan yang kedua untuk parameter `satuan`

Tapi di sini malah terisi adalah parameter `derajat`.

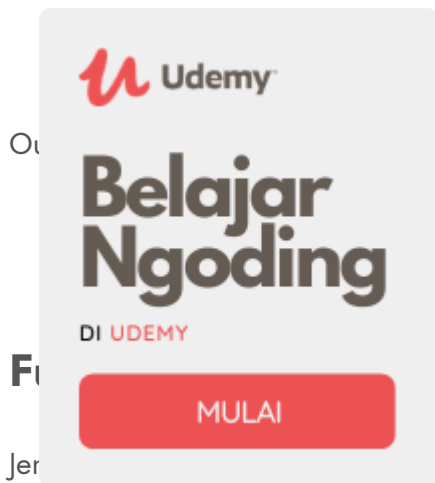
Untuk mengatasi hal ini, kita bisa mendefinisikan nama argumen/parameter yang akan kita isi.

Perhatikan contoh ini:

```
suhu_udara('Jakarta', 'fahrenheit')  
suhu_udara('Jakarta', satuan = 'fahrenheit')
```

Pemanggilan fungsi di atas akan menghasilkan output:

```
Suhu di Jakarta adalah fahrenheit celcius  
Suhu di Jakarta adalah 30 fahrenheit
```



```
rah='Makasar', derajat=100)
```

Output

in

Fun Mengembalikan Nilai

Jer berkaitan dengan nilai kembalian.

Ditinjau dari segi pengembalian nilai, fungsi **terbagi menjadi 2**:

1. Fungsi yang tidak mengembalikan nilai
2. Fungsi yang mengembalikan nilai

Pada contoh-contoh di atas, kita telah membuat dan memanggil fungsi-fungsi yang tidak memiliki nilai.

Sekarang, kita akan coba membuat fungsi yang mempunyai atau mengembalikan sebuah nilai.

```
def luas_persegi (sisi):  
    return sisi * sisi
```

Penjelasan

- Kata kunci `return` berfungsi untuk mengembalikan nilai.
- Nilai yang dikembalikan suatu fungsi, bisa kita olah kembali untuk berbagai kebutuhan.

Contoh:

```
# tidak menghasilkan output apa pun  
luas_persegi(10)  
  
# menghasilkan output  
print('Luas persegi dengan sisi 4 adalah:', luas_persegi(4))  
  
# kita juga bisa simpan di dalam variabel  
persegi_besar = luas_persegi(100)  
persegi_kecil = luas_persegi(50)  
  
print('Toal luas persegi besar dan kecil adalah:', persegi_besar + persegi_kecil)
```

Jika dijalankan, kita akan mendapatkan output:

Toal luas persegi besar dan kecil adalah: **12500**

Jika kita ingin mengembalikan nilai adalah sebuah fungsi yang jika kita panggil, dia akan mengembalikan nilai yang bisa kita olah lebih lanjut, seperti misalkan kita simpan dalam variabel untuk operasi tertentu.

Le

Jika kita mengeksekusi pada sebuah fungsi, maka semua proses yang ada di dalam blok kode fungsi akan dijalankan.

Seandainya kita memiliki lebih dari 1 buah return, maka hanya ada satu return saja yang dieksekusi. Dan ketika sebuah return telah dieksekusi, semua perintah yang ada di bawahnya akan di-skip — ini mirip dengan perintah `break` pada perulangan for mau pun while.

Perhatikan contoh berikut:

```
def persentase (total, jumlah):
    if (total >= 0 and total <= jumlah):
        return total / jumlah * 100

    return False

# output 50
print(persentase(30, 60))

# output False
print(persentase(100, 60))
```

Output:

```
50.0
False
```

Ruang Lingkup (dan Siklus Hidup) Variabel Pada Fungsi

Variabel memiliki ruang lingkup dan siklus hidup.

Secara umum, terdapat dua ruang lingkup variabel pada python:

1. Variabel global
2. Dan variabel lokal

Variable global adalah variabel yang bisa dipanggil dari manapun dari satu file python.

Perhatikan contoh berikut:



```
, kota)  
end=' ')
```

Output:

```
[print secara langsung] Lamongan  
[panggil fungsi halo] Lamongan
```

Pada kode di atas, variabel `kota` yang ada di dalam fungsi, adalah variabel `kota` yang sama dengan yang ada di luar fungsi.

Tapi, coba kita ubah kode programnya:

```
kota, provinsi = 'Lamongan', 'Jawa Timur'  
  
def hello ():  
    provinsi = 'Jawa Barat'  
    print(kota, provinsi)  
  
print('[PANGGIL FUNGSI hello()])')  
hello()  
  
print('\n[SECARA LANGSUNG]')  
print(kota, provinsi)
```

Output:

```
[PANGGIL FUNGSI hello()]  
Lamongan Jawa Barat  
  
[SECARA LANGSUNG]  
Lamongan Jawa Timur
```

Pencerahan

1. Kita membuat 2 buah variabel dengan nama `kota` dan `provinsi`

4. Tapi, ketika kita tampilkan lagi (secara langsung) variabel `provinsi`, nilainya kembali ke nilai semula.

Ke...
Ka...
va...
tet...
D...
Ya...
Ke...
Ka...
va...
tet...
D...
Ya...
Ke...
Ka...
va...
tet...
D...
Ya...

bel `provinsi` pada fungsi `hello()`, itu sebenarnya kita tidak merubah
ar, melainkan kita membuat variabel baru dengan nama yang sama, akan
yang hanya bisa diakses pada fungsi `hello()` saja.

Ya...
Ke...
Ka...
va...
tet...
D...
Ya...
Ke...
Ka...
va...
tet...
D...
Ya...
Ke...
Ka...
va...
tet...
D...
Ya...

in sebuah deskripsi terhadap fungsi yang kita buat.

Deskrip...
Ke...
Ka...
va...
tet...
D...
Ya...
Ke...
Ka...
va...
tet...
D...
Ya...
Ke...
Ka...
va...
tet...
D...
Ya...

Deskrip... akan menampilkan oleh Text Editor mau pun IDE sebagai bantuan tentang apa yang
sebenarnya dilakukan oleh sebuah fungsi.

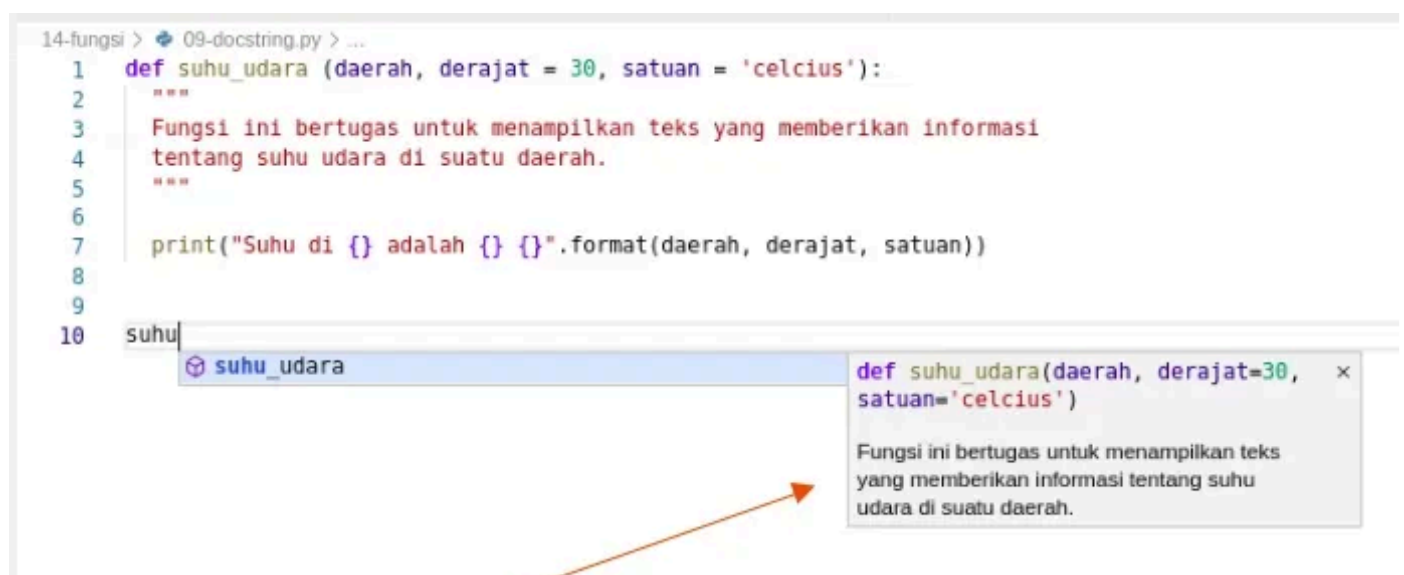
Caranya gampang.

Untuk mendefinisikan deskripsi program, kita hanya perlu menuliskan komentar multi baris tepat setelah
mendefinsikan nama fungsi.

Perhatikan contoh berikut:

```
def suhu_udara (daerah, derajat = 30, satuan = 'celcius'):  
    """  
    Fungsi ini bertugas untuk menampilkan teks yang memberikan informasi  
    tentang suhu udara di suatu daerah.  
    """  
  
    print("Suhu di {} adalah {} {}".format(daerah, derajat, satuan))
```

Berikut ini contoh tampilannya jika kita menggunakan Visual Studio Code.



```
14-fungsi > 09-docstring.py > ...  
1 def suhu_udara (daerah, derajat = 30, satuan = 'celcius'):  
2     """  
3     Fungsi ini bertugas untuk menampilkan teks yang memberikan informasi  
4     tentang suhu udara di suatu daerah.  
5     """  
6  
7     print("Suhu di {} adalah {} {}".format(daerah, derajat, satuan))  
8  
9  
10 suhu
```

def suhu_udara(daerah, derajat=30, satuan='celcius')
Fungsi ini bertugas untuk menampilkan teks yang memberikan informasi tentang suhu udara di suatu daerah.



Fungsi yang Panggil Dirinya Sendiri

Dalam fungsi yang kita buat, fungsi pada python bisa dipanggil dari berbagai tempat.

Bisa dari tempat lain yang lain. Bisa juga dari dirinya sendiri.

Fungsi yang memanggil dirinya sendiri, akan menciptakan sebuah perulangan. Dan perulangan ini biasa disebut sebagai perulangan rekursif.

Inshaallah, pada pertemuan selanjutnya kita akan membahas tentang fungsi rekursif pada python.

Kode Program Lengkap

Untuk kalian yang ingin mengakses kode program lengkap dari pertemuan ini. Langsung saja kunjungi [link ini](#).

Terima kasih banyak!

Mengangkangi Python: Level 1

Ikuti Kursus Cara Paling Cepat Menguasai Bahasa Python.

Baru 4.5 ★★★★★ (261 Peserta)

[Ambil Kelas](#)

Bagikan:



Nurul Huda

Dukung Jago Ngoding 🍵

Selanjutnya →

Python Dasar: Fungsi Rekursif (4 Contoh Program) 🐍



Belajar Ngoding

DI UDEMY

MULAI

Python Dictionary 🐍

Python Data Set 🐍

Python Dasar: Perulangan Bersarang / Bertingkat 🐍

Python Dasar: Mempelajari Perulangan While 🐍

Python Dasar: Mempelajari Perulangan For 🐍

© 2022 Jago Ngoding

Icons made by Freepik from www.flaticon.com