Lambda

Fungsi Lambda adalah fungsi *one line* yang didefinisikan tanpa nama. Fungsi lambda dapat mengambil sejumlah *arguments*, tetapi hanya dapat memiliki suatu *expression*. Sementara fungsi normal didefinisikan menggunakan kata kunci *def* , dalam python fungsi anonim didefinisikan menggunakan kata kunci lambda.

- Fungsi Lambda digunakan ketika fungsi sederhana hanya digunakan sekali atau untuk waktu yang singkat dalam kode kita. Penggunaannya yang paling umum adalah sebagai argumen untuk fungsi tingkat tinggi(fungsi yang menggunaka fungsi lain sebagai argumen)
- Lambda expression di Python adalah sebuah ekspressi untuk membuat fungsi
- Lambda sedenriri berasal dari teori kalkulus, yakni Lambda Calculus yang dikenalkan oleh Alonzo Church di tahun 1930.
- Berkat Lambda, kita bisa membuat fungsi tanpa nama atau dikenal juga dengan anonymous function.

Lambda bisa memiliki lebih dari satu argumen atau parameter, tapi hanya bida memiliki satu ekspressi atau isi.

```
greeting = lambda name : print(f"hello, {name}")
```

Karena fungsi Lambda tidak punya mana, jadi kita butuh variabel untuk menyimpannya.

Nanti saat mau kita panggil, kita tinggal tuliskan saja nama varibelnya seperti ini :

```
greeting("Dian")
greeting("Ayu")
```

```
    hello, Dian
    hello, Ayu
```

Beberapa contoh penggunakan fungsi Lambda

```
# Fungsi Lambda untuk menambahkan angka 10 kepada argument input
f = lambda x : x+10
val1 = f(5)
val2 = f(100)
print(val1, val2)
```

```
15 110
```

```
# Fungsi Lambda untuk mengalihkkan dua argumen input dan menampilkan hasilnnya
f = lambda x,y : x*y
val3 = f(2,10)
val4 = f(7,5)
print(val3, val4)
```

```
20 35
```

Anonymous Function

Karena Lambda adalah annoymous function, ia akan bebas menggunakan nama apa saja. denagan kata lain, fungsi lambda bisa disimpan di variabel mana pun. Ini tentunya tidak bisa dilakukan oleh fungsi yang dibuat dengan def. Coba perhatikan ini :

```
def sat_hello(nama):
  print(f"hello"{nama}, apa kabar?)

ucapin = say_hello(nama)
```

```
  File "<ipython-input-5-80dad1e00897>", line 2
    print(f"hello"{nama}, apa kabar?)
          ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

TELUSURI STACK OVERFLOW

Saya mencoba membuat fungsi say_hello() dengan def, kemudian menyimpannya ke dalam variabel ucapin. Tapi malah error ini karena fungsi tersebut bukan fungsi anounymous yang boleh diubah-ubah namannya. Lalu coba perhatikan di baris terakhir, fungsi lambda greeting() yang sudah disimpan lagi ke variabel yang berbeda, yakni sapa

```
greeting = lambda name : print(f"hello,(name)")
sapa = greeting
greeting("Andi")
sapa("Neli")
```

```
hello,(name)
hello,(name)
```

Hasilnya tidak error, karena ini adalah fungsi anonymous atau lambda

Eksekusi Lambda Secara Langsung

Eksekusi lambda secara langsung kelebihan fungsi dibandingkan def adalah bisa kita eksekusi langsung.

Contohnya :

```
(lambda x,y : x**2 +y**2)(4,6)
```

```
52
```

Tanda kurung yang mengapit fungsi lamda artinya akan langsung megeksekusi fungsi tersebut.

Lalu kurnag berikutnya akan berisi paramerte.

Angka 4 dan 6 adalah parameter x dan y yang akan diberikan kepada fungsi lambda

Maka kita akan langsung mendapatkan hasilnnya, yakni 42 + 62 =52

Hasil dari eksekusi ini bida juga kita simpan ke dalam variabel

```
hasil = (lambda x,y : x**2 + y**2)(4,6)
```

Mengapa Harus Pakai Lambda

Kita memang tidak harus selalu menggunakan Lambda, tapi dalam kasusu tertentu lambda lebih baik dibandigkan fungsi biasa.

Lambda biasannya dibutuhkan saat kita ingin membuat fungsi dalam satu baris.

Biasannya saat menggunakan fungsi-fungsi seperti *filter()*, *map()*, dan *reduce()* kita akan membutuhkan lambda.

Mengapa?

Karena di fungsi-fungsi tersebut membutuhkan parameter fungsi.

Contoh :

```
bilangan = [10,2,8,7,5,4,3,11,0,1]
filtered_result = map (lambda x : x*x, bilangan)
print(list(filtered_result))
```

```
[100, 4, 64, 49, 25, 16, 9, 121, 0, 1]
```

Pada fungsi map() kita memberikan parameter dengan fungsi lambda.

mengapa bisa begitu?

Karena Lambda bisa disimpan ke dalam variabel, otomatis dia akan bisa juga jadi parameter

satu lagi dengan fungsi filter()

```
# Menentukan bilangan genap
genap = lambda x : x%2 == 0
list(filter(genap, range(11)))
```

```
[0, 2, 4, 6, 8, 10]
```