KONSEP PEMROGRAMAN

[Python 3.6.2]

Abstract

Immutable dan Mutable Object, Data Structure (Tuple dan Set)

Puji Winar Cahyo

github.com/pwcahyo

A. Tujuan Pembelajaran.

Mahasiswa diharapkan dapat :

1) Memahami konsep mutable dan immutable object
2) Memahami konsep data structure lebih lanjut (tuple dan set)
3) Dapat melakukan implementasi data structure (tuple dan set) pada python

B. Pengantar

Bahasa pemrograman python mengakomodasi object pada pendefinisian data structure maupun data type, Object tersebut dibagi menjadi dua jenis yaitu immutable object dan mutable object. Object – object tersebut memiliki sifat tersendiri dalam penggolongan immutable atau mutable object. Penjelasan mengenai pengelompokan object tersebut dapat dilihat sebagai berikut.

1. Immutable object

Immutable object merupakan object yang tidak bisa dilakukan alter secara langsung. Alter adalah aksi perubahan pada object, apabila ada aksi perubahan (contohnya adalah perubahan nilai), maka perubahan tersebut akan melakukan penghapusan object beserta memori penyimpanan terhadap object yang dilalkukan aksi, kemudian akan menciptakan object baru hasil dari aksi perubahan yang dilakukan. Ada beberapa object yang bersifat immutable, diantaranya adalah :

int	bool	frozenset
float	string	bytes
decimal	tuple	
complex	range	

berikut ini contoh pembuktian immutable object pada tuple :

```
>>> tuple = (1,2,3,4,5)
>>> tuple[1] = 6
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

dapat dilihat dari pembuktian diatas, nilai tuple pada index ke 1 yaitu 2 tidak dapat diganti secara menjadi 6 karena tuple adalah immutable object. Apabila ingin memasukan nilai 6 kedalam tuple dengan paksa dapat menggunakan cara

concatination, dari proses concatination tersebut, maka python akan melakukan penghapusan object dan memori pada object yang dikenakan aksi. Kemudian akan menciptakan object baru dengan memori baru. Hal tersebut dapat kita ketahui dengan menggunakan fungsi id(object).

```
>>> tuple = (1,2,3,4,5)
>>> id(tuple)
4323397481
>>> tuple += (6,)
>>> id(tuple)
4323397482
```

Table 1.1 (tuple) awal sebelum dilakukan perubahan

Address	Value	Address	Value
4323397481	(1,2,3,4,5)		

Table 1.2 (tuple) setelah dilakukan perubahan

Address	Value	Address	Value
		4323397482	(1,2,3,4,5,6)

Perbandingan dari Table 1.1 dengan Table 1.2 dapat diketahui, Table 1.1 nilai tuple (1,2,3,4,5) mempunyai alamat memori 4323397481. Karena tuple tersebut tidak bisa dilakukan aksi penambahan nilai secara langsung maka dilakukan concatination dengan menambahkan nilai 6 pada tuple (1,2,3,4,5) sehingga nilai tuple menjadi (1,2,3,4,5,6). Perubahan tersebut menyebabkan nilai dan memori pada object sebelumnya dihapus kemudian menciptakan memori pada object yang baru yaitu 4323397482

immutable tidak sepenuhnya immutable

tidak sepenuhnya object menjadi immutable, apabila didalam immutable object terdapat mutable object, contohnya pada tuple yang didalamnya ada list. data pada list (mutable object) dapat diganti nilainya, seperti contoh berikut.

```
>>> tuple = (1,[6,7,8],3,4,5)
>>> tuple[1][0] = 7
>>> tuple
(1, [7, 7, 8], 3, 4, 5)
```

contoh diatas memperlihatkan bahwa nilai 6 dapat diganti menjadi 7, dikarenakan [6,7,8] merupakan list (mutable object) yang berada pada tuple (immutable object).

2. Mutable object

Berbeda dengan immutable object, mutable object adalah kebalikan dari immutable object. mutable object adalah object yang dapat dilakukan alter atau perubahan. Sehingga tidak membutuhkan proses penghapusan object dan memori untuk melakukan proses perubahan. Ada beberapa object yang tergolong kedalam mutable object diantaranya adalah :

list	set
dict	bytearray

berikut contoh pembuktian mutable object pada set :

```
>>> set = {1,2,3,4,5}
>>> id(tuple)
4324901608
>>> tuple.add(6)
>>> tuple
{1,2,3,4,5,6}
>>> id(tuple)
4324901608
```

Dari pembuktian diatas dapat diketahui bahwa set adalah mutable object dikarenakan untuk melakukan penambahan nilai dapat dilakukan secara langsung,

dengan menggunakan tuple.add(). Object semula dengan alamat memori 4324901608 kemudian dilakukan penambahan nilai pada object tersebut maka alamat memori object tetap sama yaitu pada alamat 4324901608.

resume

Penggunakan immutable dan mutable object tergantung kebutuhan si pemrogram, apabila dilihat dari segi efisiensi mutable object lebih efisien daripada immutable object. Dikarenakan proses dari mutable object tidak membutuhkan memori yang cukup tinggi dibanding dengan immutable object, dikarenakan pada mutable object tidak ada proses penghapusan memori kemudian penempatan pada memori baru. Penggunakan mutable object sering dipilih oleh kebanyakan programmer untuk processing data yang besar, salah satunya programmer pada bidang data scientist.

C. Implementasi

1. tuple

Tuple merupakan immutable object, dengan artian nilai didalam tuple tidak bisa dilakukan perubahan. Akan tetapi ada beberapa method yang dapat digunakan untuk operasional tuple, diantaranya adalah :

a) mencetak data tuple

data tuple dapat dicetak menggunakan pemanggilan index tuple, seperti contoh berikut mencetak data 4, yang berada pada index tuple ke 3.

```
>>> tuple = (1,2,3,4,5)
>>> print(tuple[3])
```

menggunakan negative data di ambil dari minus index dimulai dari kanan dengan contoh mengambil data 3, berarti -3.

```
>>> tuple = (1,2,3,4,5)
>>> print(tuple[-3])
```

pengambilan nilai tuple dimulai dari index yang ditentukan, dengan contoh mengambil data dimulai dari index 2, menggunakan format [2:]

```
>>> tuple = (1,2,3,4,5)
>>> print(tuple[2:])
```

mencetak nilai tuple menggunakan perulangan

```
>>> tuple = (1,2,3,4,5)
>>> for x in tuple:
>>>     print(x)
```

a. tuple.count(element)

menghitung banyaknya elemen sejenis yang telah didefinisikan, dengan contoh mencari banyaknya data dengan nilai 5.

```
>>> tuple.count(5)
>>> 2
```

b. tuple.index(element)

mencari index element

```
>>> tuple.index(5)
>>> 4
```

c. len(tuple)

mengetahui jumlah panjang atau banyaknya isi tuple

```
>>> len(tuple)
>>> 5
```

d. membership

tuple dapat dicari nilai kebenaran keanggotaannya, seperti contoh apakah 3 termasuk dalam anggota tuple.

```
>>> tuple = (1,2,3,4,5)
>>> member = 3 in tuple
>>> print(member)
```

e. max & min

mencari nilai maksimal didalam tuple.

```
>>> tuple = (1,2,3,4,5)
>>> print(max(tuple))
```

mencari nilai minimal didalam tuple.

```
>>> tuple = (1,2,3,4,5)
>>> print(min(tuple))
```

f. concatination tuple

penambahan data tuple, dengan contoh menambahkan data 6 pada tuple (1,2,3,4,5).

```
>>> tuple = (1,2,3,4,5)
>>> tuple+=(6,)
```

penggabungan dua tuple

```
>>> tuple1 = (1,2,3,4,5)
>>> tuple2 = (5,6,7,8,9)
>>> tuple3 = tuple1 + tuple2
>>> print(tuple3)
```

g. convert data tuple kedalam variable

data tuple dapat dimasukan kedalam variable python, dengan contoh memasukan data 2 kedalam variable y.

```
>>> x,y,z = (1,2,3)
>>> print(y)
```

h. tuple key/value pairs convert kedalam dictionary

data tuple dapat diconvert kedalam dictionary, dengan contoh sebagai berikut.

```
>>> tuple = dict([(“jan”, 1), (“feb”, 2), (“march”, 3)])
>>> print(tuple[“jan”])
```

i. mutable tuple

tuple tidak selamanya immutable, isi tuple yang berbentuk data structure list dapat diubah isi datanya, karena list adalah mutable. Sebagai contoh mengubah nilai data 3 didalam list menjadi 9. Data tersebut ada didalam index tuple 2 dan index list 0.

```
>>> tuple = (1, 2, [3, 10])
>>> tuple[2][0] = 9
>>> print(tuple)
```

j. repetition

nilai didalam tuple dapat dilakukan operasi repetition (duplikasi berulang). Seperti contoh string “Hae PWCahyo” diulang menjadi 5 kali

```
>>> tuple = ("Hae PWCahyo")
>>> repetition = tuple*5
>>> print(repetition)
```

k. join string tuple

nilai string tuple dapat digabungkan menggunakan fungsi join. Dapat dilihat seperti contoh berikut.

```
>>> tuple = ("saya","makan","nasi")
>>> join_tuple = “ ”.join(tuple)
>>> print(join_tuple)
```

l. convert list ke tuple

data structure berbentuk list dapat diconvert kedalam tuple. Seperti contoh berikut. List [1,2,3,4,5] dilakukan convert kedalam tuple

```
>>> list = [1,2,3,4,5]
>>> tuple = tuple(list)
>>> print(tuple)
```

m. operator perbandingan

tuple dapat dilakukan perbandingan (>) (<) (<=) (>=) (!=) (= =) sehingga akan menghasilkan nilai boolean. Sebagai contoh berikut melakukan perbandingan sama dengan.

```
>>> tuple1 = [1,2,3,4,5]
>>> tuple2 = [1,2,3,4,5]
>>> print(tuple1 == tuple2)
```

n. delete tuple

nilai didalam tuple dapat diremove dengan menggunakan fungsi del. Seperti contoh berikut, yaitu delete nilai tuple kemudian mengisikan kembali dengan nilai data (4,5,6,7).

```
>>> tuple = (1,2,3,4,5)
>>> del tuple
>>> print(tuple)
>>> tuple = (4,5,6,7)
>>> print(tuple)
```