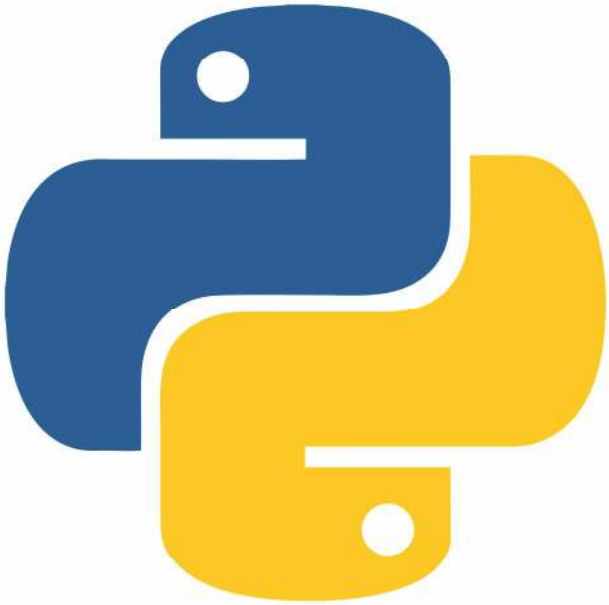
PYTHON

Pemrograman IV



Tuple

Tuple mirip dengan list. Bedanya, tuple bersifat immutable, sehingga anggotanya tidak bisa diubah. Kalau mirip, mengapa harus menggunakan tuple?
Kita menggunakan tuple tergantung kebutuhan. Untuk beberapa hal, tuple memiliki kelebihan sebagai berikut :

- Karena tuple adalah immutable, maka iterasi pada tuple lebih cepat dibandingkan list.
- Tuple bisa berisi anggota yang immutable yang dapat digunakan sebagai key untuk dictionary. List tidak bisa dipakai untuk itu.
- Kalau kita memerlukan data yang memang tidak untuk diubah, maka menggunakan tuple bisa menjamin bahwa data tersebut akan write-protected.

6.1 Membuat Tuple

Tuple dibuat dengan meletakkan semua anggota di dalam **tanda kurung ( )**, masing-masing dipisahkan oleh **tanda koma**. Menggunakan tanda kurung sebenarnya hanya opsional, tapi kita sebaiknya tetap menggunakannya untuk kemudahan pembacaan kode.

tuple_bersarang.py ▸ ...

```
1   # membuat tuple kosong
2   # Output: ( )
3   tuple1 = ()
4   print(tuple1)
5
6   # tuple dengan 1 elemen
7   # Output: (1,)
8   tuple1 = (1,)
9   print (tuple1)
10
11  # tuple berisi integer
12  # output = (1, 2, 3)
13  tuple1 = (1, 2, 3)
14  print(tuple1)
15
16  # tuple bersarang
17  # Output: ("hello", [1, 2, 3], (4, 5, 6))
18  tuple1 = ("hello", [1, 2, 3], (4, 5, 6))
19  print(tuple1)
20
21  # Tuple bisa tidak menggunakan tanda ()
22  # Output (1, 2, 3)
23  tuple1 = 1, 2, 3
24  print(tuple1)
25
26  # memasukkan anggota tuple ke variabel yang bersesuaian
27  # a akan berisi 1, b berisi 2, dan c berisi 3
28  # output 1 2 3
29  a, b, c = tuple1
30  print(a, b, c)
```

Gambar 51 Membuat Tuple

Kevin Perdana, ST
https://medium.com/@kevinperdana

6.2 Mengakses Anggota Tuple

Seperti halnya list, kita bisa mengakses anggota tuple lewat indeksnya menggunakan format namatuple[indeks]. Indeks dimulai dari 0 untuk anggota pertama. Selain itu, indeks negatif juga bisa dipakai mulai dari -1 untuk anggota terakhir tuple.

tuple_akses.py ...

```
1   tuple1 = ('p','y','t','h','o','n')
2   # Output: 'p'
3   print(tuple1[0])
4
5   # Output: 'y'
6   print(tuple1[1])
7
8   # Output: 'n'
9   print(tuple1[-1])
10
11  # Output: 'o'
12  print(tuple1[-2])
13
14  # IndexError
15  #print(tuple1[6])
```

Gambar 52 Mengakses Tuple

Sama seperti list, kita bisa mengakses satu range anggota tuple dengan menggunakan operator **titik dua** (:).

tuple_akses_range.py ...

```
1   tuple1 = ('p','r','o','g','r','a','m','m','i','n','g')
2   # akses dari indeks 0 s/d 2
3
4   # output: ('p','r','o')
5   print(tuple1[:3])
6
7   # Akses dari indeks 2 s/d 5
8   # output: ('o', 'g', 'r', 'a')
9   print(tuple1[2:6])
10
11  # Akses dari indeks 3 sampai akhir
12  # output: ('g', 'r', 'a', 'm', 'm', 'i', 'n', 'g')
13  print(tuple1[3:])
```

Gambar 53 Mengakses Tuple dengan Range

6.3 Mengubah Anggota Tuple

Setelah tuple dibuat, maka anggota tuple tidak bisa lagi diubah atau dihapus. Akan tetapi, bila anggota tuple-nya adalah **tuple bersarang** dengan anggota seperti list, maka item pada list tersebut dapat diubah.

tuple_ubah_anggota.py ▸ ...

```python
tuple1 = (2, 3, 4, [5, 6])
# kita tidak bisa mengubah anggota tuple
# bila kita hilangkan tanda komentar # pada baris ke 6
# akan muncul error: # TypeError: 'tuple' object does not support item assignment

# tuple1[1] = 8

# tapi list di dalam tuple bisa diubah
# output: (2, 3, 4, [7, 6])
tuple1[3][0] = 7
print(tuple1)

# tuple bisa diganti secara keseluruhan dengan penugasan kembali
# output: ('p','y','t','h','o','n')
tuple1 = ('p','y','t','h','o','n')
print(tuple1)

# anggota tuple juga tidak bisa dihapus menggunakan del
# perintah berikut akan menghasilkan error TypeError
# kalau Anda menghilangkan tanda komentar #

#del tuple1[0]
# kita bisa menghapus tuple keseluruhan
del tuple1
```

Gambar 54 Mengubah Anggota Tuple

6.4 Menguji Keanggotaan Tuple

Seperti halnya string dan list, kita bisa menguji apakah sebuah objek adalah anggota dari tuple atau tidak, yaitu dengan menggunakan operator **in** atau **not in** untuk kebalikannya.

tuple_uji_anggota.py ▸ ...

```python
tuple1 = (1, 2, 3, 'a', 'b', 'c')

# menggunakan in
# output: True
print(3 in tuple1)

# output: False
print('2' in tuple1)

# output: False
print('e' in tuple1)

# menggunakan not in
# output True
print('k' not in tuple1)
```

Gambar 55 Menguji Anggota Tuple

6.5 Iterasi pada Tuple

Kita bisa menggunakan **for** untuk melakukan iterasi pada tiap anggota dalam tuple.

tuple_iterasi.py ▸ ...

```
1   # output:
2   # Hai Sistem
3   # Hai Informasi
4   nama = ('Sistem', 'Informasi')
5   for a in nama:
6       print('Hai', a)
```

Gambar 56 Iterasi pada Tuple

6.6 Metode dan Fungsi Bawaan Tuple

Tuple hanya memiliki dua buah metode yaitu count() dan index().

- Metode count(x) berfungsi mengembalikan jumlah item yang sesuai dengan x pada tuple.
- Metode index(x) berfungsi mengembalikan indeks dari item pertama yang sama dengan x.

tuple_fungsi.py ▸ ...

```
1   tuple1 = ('p','y','t','o','n','s','a','y','a')
2   # count
3   # output: 2
4   print(tuple1.count('a'))
5
6   # index
7   # Output 4
8   print(tuple1.index('n'))
```

Gambar 57 Fungsi Bawaan Tuple

Walaupun hanya memiliki dua metode, banyak fungsi bawaan python yang berfungsi untuk melakukan operasi pada tuple. Berikut adalah daftarnya :

Fungsi	Deskripsi
all()	Mengembalikan True jika semua anggota tuple adalah benar (tidak ada yang kosong)
any()	Mengembalikan True jika salah satu atau semua bernilai benar. Jika tuple kosong, maka akan mengambalikan False.
enumerate()	Mengembalikan objek enumerasi. Objek enumerasi adalah objek yang terdiri dari pasangan indeks dan nilai.
len()	Mengembalikan panjang (jumlah anggota) tuple
max()	Mengembalikan anggota terbesar di tuple
min()	Mengembalikan anggota terkecil di tuple
sorted()	Mengambil anggota tuple dan mengembalikan list baru yang sudah diurutkan
sum()	Mengembalikan jumlah dari semua anggota tuple
tuple()	Mengubah sequence (list, string, set, dictionary) menjadi tuple

Gambar 58 Fungsi Bawaan Python

Set

Set adalah salah satu tipe data di Python yang tidak berurut (unordered). Set memiliki anggota yang unik (**tidak ada duplikasi**). Jadi misalnya kalau kita meletakkan dua anggota yang sama di dalam set, maka otomatis set akan menghilangkan yang salah satunya.

Set bisa digunakan untuk melakukan operasi himpunan matematika seperti gabungan, irisan, selisih, dan lain - lain.

7.1 Membuat Set

Set dibuat dengan meletakkan anggota - anggotanya di dalam tanda **kurung kurawal {}**, dipisahkan menggunakan **tanda koma**. Kita juga bisa membuat set dari list dengan memasukkan list ke dalam fungsi set()
Set bisa berisi data campuran, baik **integer**, **float**, **string**, dan lain sebagainya. Akan tetapi set tidak bisa berisi list, set, dan dictionary.

set1.py ▸ ...

```
1   # set integer
2   set_saya = {1,2,3}
3   print(set_saya)
4
5   # set dengan menggunakan fungsi set()
6   set_saya = set([1,2,3])
7   print(set_saya)
8
9   # set data campuran
10  set_saya = {1, 2.0, "Python", (3,4,5)}
11  print(set_saya)
12
13  # bila kita mengisi duplikasi, set akan menghilangkan salah satu
14  # output: {1,2,3}
15  set_saya = {1,2,2,3,3,3}
16  print(set_saya)
17
18  # set tidak bisa berisi anggota list
19  # contoh berikut akan muncul error TypeError
20  set_saya = {1,2,[3,4,5]}
```

Gambar 59 Membuat Set

Untuk membuat set kosong, kita tidak bisa menggunakan **{ }**, karena itu akan dianggap sebagai **dictionary**. Kita harus menggunakan fungsi **set()** tanpa argumen untuk membuat set kosong.

set_kosong.py ▸ ...

```
1   # membuat variabel a dengan {}
2   a = {}
3   print(type(a))
4   # output <class 'dict'>
5
6   # harus menggunakan fungsi set()
7   a = set()
8   print(type(a))
9   #output <class 'set'>
```

Gambar 60 Set Kosong

7.2 Mengubah Anggota Set

Set bersifat mutable. Tapi, karena set adalah tipe data tidak berurut (unordered), maka kita tidak bisa menggunakan indeks. Set tidak mendukung **indeks** ataupun **slicing**.

Untuk menambah satu anggota ke dalam set, kita bisa menggunakan fungsi **add()**, dan untuk menambahkan beberapa anggota sekaligus kita bisa menggunakan fungsi **update()**. List, tuple, maupun string bisa digunakan sebagai masukan dari fungsi update().

set_ubah_anggota.py ▸ ...

```
1   # membuat set baru
2   set_saya = {1,2,3}
3   print(set_saya)
4
5   # bila kita hilangkan tanda # dari baris 9 akan muncul error TypeError
6   #set_saya[0]
7
8   # menambah satu anggota
9   # output: {1,2,3,4}
10  set_saya.add(4)
11  print(set_saya)
12
13  # menambah beberapa anggota
14  # set akan menghilangkan duplikasi
15  # output: {1,2,3,4,5,6}
16  set_saya.update([3,4,5,6])
17  print(set_saya)
```

Gambar 61 Mengubah Anggota Set

7.3 Menghapus Anggota Set

Kita bisa menghapus anggota set dengan menggunakan fungsi **discard()** dan **remove()**. Perbedaannya, fungsi discard() tidak akan memunculkan error bila anggota yang ingin dihapus ternyata tidak ada di dalam set, sedangkan remove() sebaliknya.

set_hapus_anggota.py ▸ ...

```python
# membuat set baru
set_saya = {1, 2, 3, 4, 5}
print(set_saya)

# menghapus 4 dengan discard
# output: {1, 2, 3, 5}
set_saya.discard(4)
print(set_saya)

# menghapus 5 dengan remove
# output : {1, 2, 3}
set_saya.remove(5)
print(set_saya)

# anggota yang mau dihapus tidak ada dalam set
# discard tidak akan memunculkan error
# output: {1, 2, 3}
set_saya.discard(6)
```

Gambar 62 Menghapus Anggota Set

Selain **discard()** dan **remove()**, kita bisa menghapus anggota set dengan menggunakan fungsi **pop()**. Dengan menggunakan fungsi **pop()**, kita menghapus salah satu anggota **secara acak (random)**.

Untuk mengosongkan atau menghapus seluruh anggota set, kita bisa menggunakan fungsi **clear()**.

set_hapus_anggota2.py ▸ ...

```python
# membuat set baru
# output: set berisi anggota yang unik
set_saya = set("HelloPython")
print(set_saya)

# pop anggota
# output: anggota acak
print(set_saya.pop())
print(set_saya)

# pop anggota lainnya
# output: anggota acak
print(set_saya.pop())
print(set_saya)

# mengosongkan set
# output: set()
set_saya.clear()
print(set_saya)
```

Gambar 63 Menghapus Anggota Set Secara Random dengan pop()

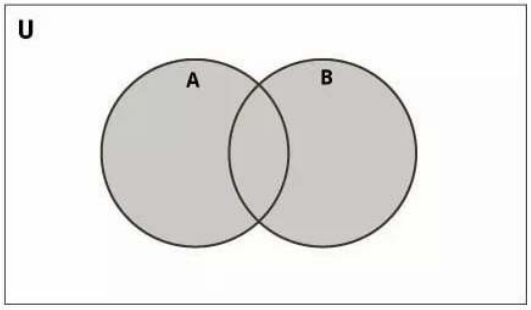
Kevin Perdana, ST
https://medium.com/@kevinperdana

7.4 Operasi Set di Python

Set dapat digunakan untuk melakukan operasi himpunan matematika seperti gabungan, irisan, selisih, dan komplemen.

7.4.1 Operasi Gabungan (Union)

Mari kita ambil dua contoh set berikut :
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}



Gambar 64 Operasi Gabungan (Union)

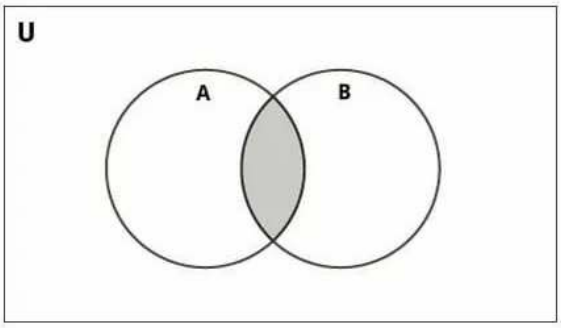
Gabungan (union) dari A dan B adalah himpunan atau set anggota yang ada di A dan B. Gabungan dapat dibuat dengan menggunakan operator **palang (|)**. Selain itu juga bisa dilakukan dengan menggunakan fungsi **union()**.

set_union.py ▸ ...

```
1   # Membuat set A and B
2   A = {1, 2, 3, 4, 5}
3   B = {4, 5, 6, 7, 8}
4
5   # Gabungan menggunakan operator |
6   # output: {1, 2, 3, 4, 5, 6, 7, 8}
7   print(A | B)
8
9   # Menggunakan fungsi union()
10  # output: {1, 2, 3, 4, 5, 6, 7, 8}
11  A.union(B)
12
13  # output: {1, 2, 3, 4, 5, 6, 7, 8}
14  B.union(A)
```

Gambar 65 Operasi Gabungan (Union) dengan Set

7.4.2 Operasi Irisan (Intersection)



Gambar 66 Operasi Irisan (Intersection)

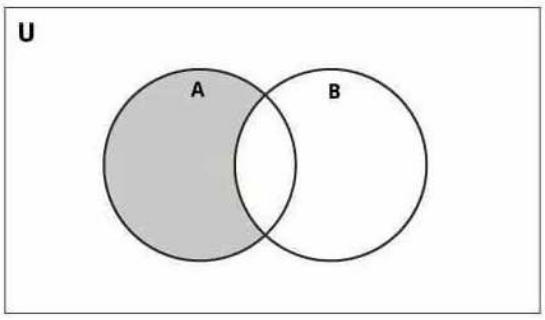
Irisan (intersection) dari A dan B adalah himpunan atau set anggota yang sama di A dan B.
Irisan dilakukan dengan menggunakan operator **jangkar (&)**. Irisan juga bisa dilakukan dengan menggunakan fungsi **intersection()**.

set_intersection.py ▸ ...

```
1   # Membuat set A and B
2   A = {1, 2, 3, 4, 5}
3   B = {4, 5, 6, 7, 8}
4
5   # Irisan menggunakan operator &
6   # output: {4,5}
7   print(A & B)
8   # Menggunakan fungsi intersection()
9   # output: {4,5}
10  A.intersection(B)
11
12  # output: {4,5}
13  B.intersection(A)
```

Gambar 67 Operasi Irisan (Intersection) dengan Set

7.4.3 Operasi Selisih (Difference)



Gambar 68 Operasi Selisih (Difference)

Kevin Perdana, ST
https://medium.com/@kevinperdana

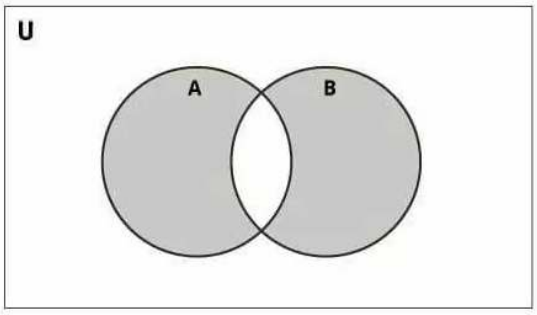
Selisih (difference) dari A dan B adalah himpunan atau set anggota yang hanya ada di A dan tidak ada di B. Begitu juga sebaliknya, ada di B tapi tidak ada di A.
Selisih dilakukan dengan menggunakan operator **kurang (-)**. Bisa juga dengan menggunakan fungsi **difference()**.

set_difference.py ▸ ...

```
1   # membuat A and B
2   A = {1, 2, 3, 4, 5}
3   B = {4, 5, 6, 7, 8}
4
5   # Menggunakan operator - pada A
6   # Output: {1, 2, 3}
7   print(A - B)
8
9   # Output: {1, 2, 3}
10  A.difference(B)
11
12  # Menggunakan operator - pada B
13  # Output: {8, 6, 7}
14  print(B - A)
15
16  # Output: {8, 6, 7}
17  B.difference(A)
```

Gambar 69 Operasi Selisih (Difference) dengan Set

7.4.4 Operasi Komplemen (Symmetric Difference)



Gambar 70 Operasi Komplemen (Symmetric Difference)

Operasi komplemen (symmetric difference) dari A dan B adalah himpunan atau set anggota yang ada di A dan di B, tapi tidak di keduanya.
Komplemen dilakukan dengan menggunakan **operator ^**. Bisa juga dengan menggunakan fungsi **symmetric_difference()**.

set_symmetric_difference.py ▸ ...

```
1   # membuat A and B
2   A = {1, 2, 3, 4, 5}
3   B = {4, 5, 6, 7, 8}
4
5   # Menggunakan operator ^ pada A
6   # Output: {1, 2, 3, 6, 7, 8}
7   print(A ^ B)
8
9   # Output: {1, 2, 3, 6, 7, 8}
10  A.symmetric_difference(B)
11
12  # Menggunakan operator ^ pada B
13  # Output: {1, 2, 3, 6, 7, 8}
14  print(B ^ A)
15
16  # Output: {1, 2, 3, 6, 7, 8}
17  B.symmetric_difference(A)
```

Gambar 71 Operasi Komplemen (Symmetric Difference) dengan Set

7.5 Metode (Fungsi) Set

Set memiliki banyak metode atau fungsi. Beberapa di antaranya adalah yang sudah kita gunakan di atas. Tabel berikut berisi daftar metode atau fungsi set yang disediakan oleh python.

Metode	Deskripsi
add()	Menambahkan satu anggota ke set
clear()	Menghapus semua anggota set
copy()	Mengembalikan *shallow copy* dari set
difference()	Mengembalikan set baru berisi selisih dua atau lebih set
difference_update()	Menghapus semua anggota set lain yang ada di set ini
discard()	Menghapus satu anggota dari set
intersection()	Mengembalikan set baru berisi irisan antara dua atau lebih set
intersection_update()	Mengupdate set dengan irisan set bersangkutan dan set lainnya
isdisjoint()	Mengembalikan True jika dua set tidak memiliki irisan
issubset()	Mengembalikan True jika set lain berisi set ini
issuperset()	Mengembalikan True jika set ini berisi set lain
pop()	Menghapus dan mengembalikan anggota acak dari sebuah set
remove()	Menghapus satu anggota dari set
symmetric_difference()	Mengembalikan set baru berisi komplemen dari dua set
symmetric_difference_update()	Mengupdate set dengan komplemen dari set ini dan set lainnya
union()	Mengembalikan set baru berisi gabungan dua atau lebih set
update()	Mengupdate set dengan gabungan set ini dan set lainnya

Gambar 72 Metode/ Fungsi Set

Kevin Perdana, ST
https://medium.com/@kevinperdana

Dictionary

Dictionary adalah tipe data yang anggotanya terdiri dari pasangan **kunci:nilai (key:value)**. Dictionary bersifat tidak berurut (unordered) sehingga anggotanya tidak memiliki indeks.

8.1 Membuat Dictionary

Dictionary dibuat dengan menempatkan anggotanya di dalam tanda kurung **kurawal {}**, dipisahkan oleh tanda koma.

Anggota dictionary terdiri dari pasangan **kunci:nilai**. Kunci harus bersifat **unik**, tidak boleh ada dua kunci yang sama dalam dictionary.

dictionary1.py ▸ ...

```
1   # Membuat dictionary kosong
2   dict1 = {}
3   print(dict1)
4
5   # dictionary dengan kunci integer
6   dict1 = {1: 'sepatu', 2: 'tas'}
7   print(dict1)
8
9   # dictionary dengan kunci campuran
10  dict1 = {'warna': 'merah', 1: [2,3,5]}
11  print(dict1)
12
13  # membuat dictionary menggunakan fungsi dict()
14  dict1 = dict([('1','sepatu'), ('2','bola')])
15  print(dict1)
16
17  dict1 = dict(m=8, n=9, o=10)
18  print(dict1)
19
```

Gambar 73 Membuat Dictionary

8.2 Mengakses Anggota Dictionary

Dictionary tidak menggunakan indeks. Anggota dictionary diakses dengan menggunakan kuncinya. Selain itu, bisa juga diakses dengan menggunakan fungsi **get()**.

Dengan menggunakan fungsi **get()**, bila kunci tidak ada di dalam dictionary, maka akan dikembalikan **None**. Bila tidak menggunakan fungsi **get()**, maka akan terjadi error **KeyError** bila kunci yang ingin diakses tidak ada di dalam dictionary.

dictionary_akses_anggota.py ▸ ...

```python
dict_saya = {'nama':'Budi', 'usia':27}

# output: Budi
print(dict_saya['nama'])

# output 27
print(dict_saya.get('usia'))

# output None
print(dict_saya.get('alamat'))

# Mengakses kunci yang tidak tersedia menyebabkan KeyError
#dict_saya['alamat']
```

Gambar 74 Mengakses Anggota Dictionary

8.3 Mengubah Anggota Dictionary

Dictionary bersifat mutable. Kita bisa menambahkan atau mengubah nilai dari anggotanya menggunakan operator penugasan. Bila kunci sudah ada, maka nilainya yang akan diupdate. Bila kunci belum ada, maka akan ditambahkan sebagai kunci baru.

dictionary_ubah_anggota.py ▸ ...

```python
dict_saya = {'nama':'Ikhsan', 'usia':35}

# mengupdate nilai
dict_saya['usia'] = 36
# Output: {'nama': 'Ikhsan', 'usia':36}
print(dict_saya)

# menambah anggota
dict_saya['alamat'] = 'Tanjungpinang'
# output: {'alamat': 'Tanjungpinang', 'nama':'Ikhsan', 'usia':36}
print(dict_saya)
```

Gambar 75 Mengubah Anggota Dictionary

8.4 Menghapus Anggota Dictionary

Kita bisa menghapus anggota tertentu pada dictionary dengan menggunakan fungsi **pop()**. Fungsi ini menghapus anggota dengan mengembalikan kunci dari anggota tersebut.

Fungsi lain, **popitem()** digunakan untuk menghapus anggota acak dari dictionary. Untuk menghapus semua anggota dictionary, bisa menggunakan fungsi **clear()**.

Selain itu kita juga bisa menggunakan kata kunci del untuk menghapus anggota tertentu atau menghapus dictionary itu sendiri.

dictionary_hapus_anggota.py ▸ ...

```python
# membuat dictionary baru
dict_saya = {1:1, 2:4, 3:9, 4:16, 5:25}

# menghapus anggota tertentu
# output: 9
print(dict_saya.pop(3))

# menghapus anggota secara acak
# output: (5, 25)
print(dict_saya.popitem())

# yang tersisa adalah {1:1, 2:4, 4:16}
print(dict_saya)

# delete 5
del dict_saya[2]

# output: {2:4, 4:16}
print(dict_saya)

# menghapus semua anggota
dict_saya.clear()

# menghapus dictionary dict_saya
del dict_saya

# Error karena dict_saya sudah dihapus
#print(dict_saya)
```

Gambar 76 Menghapus Anggota Dictionary

8.5 Metode (Fungsi) Dictionary

Dictionary memiliki beberapa metode/fungsi untuk melakukan berbagai operasi. Beberapa di antaranya sudah digunakan di atas. Selengkapnya bisa dilihat pada tabel berikut :

Metode	Deskripsi
clear()	Menghapus semua anggota dictionary
copy()	Mengembalikan *shallow copy* dari dictionary
fromkeys(seq[, v])	Mengembalikan dictionary baru dengan kunci-kuncinya dari `seq`, dan nilainya sama dengan `v` (defaultnya `None`)
get(key[,d])	Mengembalikan nilai dari `key`. Bila `key` tidak tersedia, kembalikan d (defaultnya `None`)
items()	Mengembalikan view baru (berisi semua pasangan key,value dari dictionary
keys()	Mengembalikan view baru (berisi semua kunci pada dictionary)
pop(key[,d])	Menghapus anggota yang memiliki kunci `key` dan mengembalikan nilai `d` jika kunci tidak ada dalam dictionary. Bila `d` tidak dibuat, dan `key` tidak ditemukan, akan menghasilkan KeyError
popitem()	Menghapus anggota secara acak. Menghasilkan KeyError jika dictionary kosong
setdefault(key[,d])	Bila `key` ada dalam dictionary, kembalikan nilainya. Bila tidak, sisipkan `key` dengan nilai `d` dan kembalikan `d` (defaulnya `None`)
update([other])	Mengupdate dictionary dengan menambahkan anggota dari dictionary lain `other`, timpa (*overwrite*) bila ada kunci yang sama
values()	Mengembalikan view baru (berisi semua *value* pada dictionary)

Gambar 77 Metode/ Fungsi Dictionary

Kevin Perdana, ST
https://medium.com/@kevinperdana