**Materi Lab 6**

**Tuple**

Tuple adalah suatu struktur data *built-in* di bahasa pemrograman Python. Tuple berisi koleksi data yang terurut dan **tidak dapat diubah** (***immutable***), berbeda dengan struktur data list yang dapat diubah-ubah (*mutable*).

**Membuat tuple**

Tuple dapat dibuat dengan beberapa cara:

* Menggunakan sepasang tanda kurung untuk menunjukkan tuple kosong: ()
* Menggunakan tanda koma untuk singleton tuple: a, atau (a,)
* Memisahkan item/elemen dengan koma: a, b, c atau (a, b, c)
* Menggunakan tuple() built-in: tuple() or tuple(*iterable*)

|  |
| --- |
| **>>>** empty = ()  **>>>** singleton = 'hello', *# <-- note trailing comma*  **>>>** t = ('apple', 'banana', 'avocado')  **>>>** tt = 'hello', 'world'  **>>>** ttt = tuple([1, 2, 3])  **>>>** singleton  ('hello',)  **>>>** tt  ('hello', 'world')  **>>>** ttt  (1, 2, 3)  **>>>** u = t, (1, 2, 3, 4, 5) *# Tuples may be nested*  **>>>** u  (('apple', 'banana', 'avocado'), (1, 2, 3, 4, 5))  **>>>** len(empty)  0  **>>>** len(t)  3 |

***Packing* dan *unpacking***

|  |
| --- |
| **>>>** t = 'apple', 'banana', 'avocado' *# Packing*  **>>>** x, y, z = t *# Unpacking*  **>>>** x  'apple'  **>>>** z, y  ('avocado', 'banana') |

**Mengakses data pada tuple**

|  |
| --- |
| **>>>** t = ([1, 2, 3], [98, 99, 100], 'ciluk ba', 5)  **>>>** t[1:3]  ([98, 99, 100], 'ciluk ba')  **>>>** t[-1]  5  **>>>** t[1][2]  100  **>>>** t[0]  [1, 2, 3] |

Data pada tuple tidak dapat diubah

|  |
| --- |
| **>>>** t = ([1, 2, 3], [98, 99, 100], 'ciluk ba', 5)  **>>>** t[0] = 2020 *# Tuples are immutable!*  Traceback (most recent call last):  File "<stdin>", line 1, in <module>  TypeError: 'tuple' object does not support item assignment |

**Set**

Suatu set (himpunan) adalah koleksi yang tidak berurutan dan tidak mengandung elemen duplikat. Biasanya set digunakan untuk menghilangkan entri duplikat dan *membership testing*. Objek set juga mendukung operasi matematika seperti gabungan (*union*), irisan (*intersection*), perbedaan (*difference*), dan perbedaan simetris (*symmetric difference*).

**Membuat set**

Untuk membuat sebuah set kosong, gunakan set(), bukan {}. Jika menggunakan {}, maka akan terbuat dictionary kosong, bukan set kosong. Kalian bisa membuat set dengan set(*iterable*).

|  |
| --- |
| **>>>** s = set()  **>>>** kulkas = {'keju', 'telur', 'es', 'daging', 'es'}  **>>>** kulkas  {'keju', 'telur', 'daging', 'es'} *# data duplikat terhapus*  **>>>** npm = set([123, 456, 789])  **>>>** npm  {456, 123, 789}  **>>>** c = set('kalamanawahaya')  {'w', 'k', 'h', 'm', 'a', 'y', 'n', 'l'} |

**Mengakses data pada set**

Kalian tidak bisa mengakses data pada sebuah set dengan menunjuk ke suatu index atau elemen. Namun, kalian bisa mengakses data dari sebuah set dengan menggunakan for loop atau melakukan *membership testing* dengan keyword in.

|  |
| --- |
| **>>>** npm = {123, 456}  **>>>** for x in npm:  ... print(x)  ...  456  123  **>>>** kulkas = {'keju', 'telur', 'es', 'daging', 'es'}  **>>>** 'daging' in kulkas  True  **>>>** 'susu' in kulkas  False |

Note: Setelah membuat set, kalian tidak bisa mengganti elemen tertentu di dalam set tersebut, namun kalian bisa menambah dan menghapus elemen. Kalian juga dapat meng-update set (bukan meng-update elemen). Meng-update set dapat kalian lakukan jika kalian ingin menambahkan banyak elemen ke dalam suatu set.

**Menambahkan elemen ke suatu set dan meng-update set**

Untuk menambahkan suatu data ke dalam set, kalian dapat menggunakan suatu\_set**.add(data)**. Untuk meng-update set (menambahkan banyak data ke dalam set), kalian dapat menggunakan suatu\_set**.update(*iterable*)**.

|  |
| --- |
| **>>>** kulkas = {'keju', 'telur', 'es', 'daging', 'es'}  **>>>** kulkas.add('es krim')  **>>>** kulkas  {'es', 'keju', 'daging', 'telur', 'es krim'}  **>>>** kulkas.update(['bayam', 'wortel'])  **>>>** kulkas  {'es', 'bayam', 'keju', 'daging', 'telur', 'es krim', 'wortel'} |

**Menghapus elemen pada suatu set**

Untuk menghapus suatu elemen dalam suatu set, kalian dapat menggunakan suatu\_set**.remove(elemen)** atau suatu\_set**.discard(elemen)**. Jika elemen tidak ada dalam list, method remove() akan me-raise error, sedangkan method discard() tidak akan me-raise error.

|  |
| --- |
| **>>>** kulkas = {'keju', 'telur', 'es', 'daging', 'es krim'}  **>>>** kulkas.remove('es')  **>>>** kulkas  {'keju', 'daging', 'telur', 'es krim'}  **>>>** kulkas.remove('es')  Traceback (most recent call last):  File "<stdin>", line 1, in <module>  KeyError: 'es'  **>>>** kulkas.discard('es')  **>>>** kulkas  {'keju', 'daging', 'telur', 'es krim'}  **>>>** kulkas.clear() *# method clear() mengosongkan set*  **>>>** kulkas  set() |

**Menggabungkan sets**

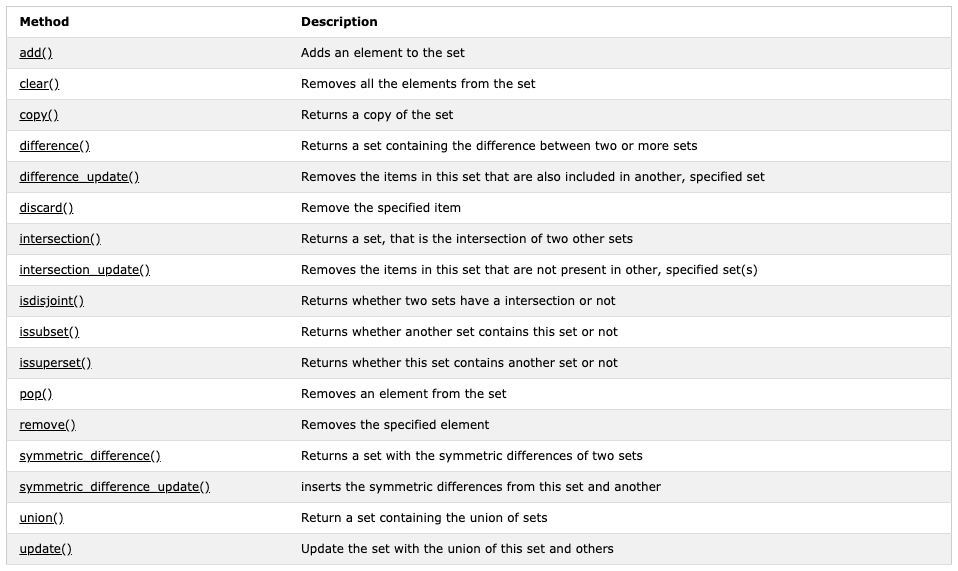
Untuk menggabungkan dua atau lebih sets, kalian dapat menggunakan set1**.union(set2)** atau set1**.update(set2)**.

|  |
| --- |
| **>>>** set1 = {'a', 'b', 'c'}  **>>>** set2 = {1, 2, 3}  **>>>** set1.union(set2)  **>>>** set1  {1, 2, 3, 'c', 'a', 'b'}  **>>>** set3 = {'d', 'e', 'f'}  **>>>** set3.update(set2)  **>>>** set3  {'d', 'e', 1, 2, 3, 'f'} |

**Operasi matematika (*union, intersection, difference, symmetrical difference*) pada sets**

|  |
| --- |
| **>>>** a = set('kalamanawahaya')  **>>>** b = set('kalasenja')  **>>>** a  {'w', 'k', 'h', 'm', 'a', 'y', 'n', 'l'}  **>>>** b  {'e', 'k', 's', 'a', 'n', 'l', 'j'}  **>>>** *# difference: elemen-elemen yang ada di a tetapi tidak ada di b*  **...** a - b  {'y', 'w', 'h', 'm'}  **>>>** a.difference(b)  {'y', 'w', 'h', 'm'}  **>>>** *# union: elemen-elemen yang ada di a atau di b atau keduanya*  **...** a | b  {'n', 'l', 'a', 's', 'j', 'w', 'e', 'h', 'm', 'y', 'k'}  **>>>** a.union(b)  {'n', 'l', 'a', 's', 'j', 'w', 'e', 'h', 'm', 'y', 'k'}  **>>>** *# intersection: elemen-elemen yang ada di a dan di b*  **...** a & b  {'n', 'k', 'l', 'a'}  **>>>** a.intersection(b)  {'n', 'k', 'l', 'a'}  **>>>** *# symmetrical difference: elemen-elemen yang ada di a atau di b, tetapi tidak di keduanya*  **...** a ^ b  {'s', 'j', 'e', 'h', 'w', 'y', 'm'}  **>>>** a.symmetric\_difference(b)  {'s', 'j', 'e', 'h', 'w', 'y', 'm'} |

Python Set Methods lainnya



sumber: https://www.w3schools.com/python/python\_ref\_set.asp

**Dictionary**

Dictionary merupakan struktur data berisi kumpulan data yang tidak terurut. **DIctionary bersifat *mutable***. Setiap elemen dari dictionary merupakan pasangan key dan value yang setiap elemennya dipisahkan dengan koma. Setiap key bersifat unik, yang artinya key dari satu elemen tidak boleh sama dengan elemen lainnya.

Key dan value setiap elemen dapat berupa tipe data apapun, asalkan **tipe data untuk key bersifat *immutable*** (seperti integer, string, float, ataupun boolean). Sehingga, key tidak dapat berupa list atau dictionary karena keduanya bersifat *mutable.*

Sedangkan untuk value, dapat berbentuk tipe data apapun, termasuk dictionary dan list, atau bahkan objek yang kalian buat sendiri.

**Membuat dictionary**

Untuk membuat sebuah objek dictionary, dapat dilakukan dengan berbagai cara seperti berikut.

* Menggunakan sepasang tanda kurung untuk menunjukkan dictionary kosong: {}. Atau dengan constructor dict().
* Mendefinisikan dictionary beserta isinya.

|  |
| --- |
| **>>>** empty\_dict = {}  **>>>** empty\_dict2 = dict()  **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** orion\_stars  {'alpha': 'Betelgeuse', 'beta': 'Rigel', 'gamma': 'Bellatrix', 'delta': 'Mintaka'} |

**Mengakses dictionary**

Mengakses suatu elemen dalam dictionary yaitu dengan cara memanggil key elemen. Dapat menggunakan kurung siku atau method get(key).

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** orion\_stars  {'alpha': 'Betelgeuse', 'beta': 'Rigel', 'gamma': 'Bellatrix', 'delta': 'Mintaka'}  **>>>** orion\_stars[“beta”]  ‘Rigel’  **>>>** orion\_stars.get(“alpha”)  ‘Betelgeuse’ |

**Menambah elemen pada dictionary**

Untuk menambahkan elemen ke dalam dictionary, dapat digunakan kurung siku. Namun, jika menggunakan key yang sudah ada pada dictionary, dictionary tidak akan menambahkan elemen tersebut, melainkan meng-update value dari elemen dengan key yang sama. Untuk mengubah value dari sebuah elemen, dapat digunakan cara yang sama.

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** orion\_stars[“epsilon”] = ”Alnilam”  **>>>** orion\_stars  {'alpha': 'Betelgeuse', 'beta': 'Rigel', 'gamma': 'Bellatrix', 'delta': 'Mintaka', ‘epsilon’:’Mintaka’}  **>>>** orion\_stars[“epsilon”] = ”Alnitak”  **>>>** orion\_stars  {'alpha': 'Betelgeuse', 'beta': 'Rigel', 'gamma': 'Bellatrix', 'delta': 'Mintaka', ‘epsilon’:’Alnitak’} |

**Menghapus elemen pada dictionary**

* Menggunakan method .pop(suatu\_key) yang menghapus elemen dengan key suatu\_key dan mengembalikan value dari elemen yang dihapus.
* Menggunakan method .popitem() yang menghapus elemen yang terakhir kali ditambahkan dan mengembalikan value dari elemen yang dihapus.
* Menggunakan keyword del.

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** orion\_stars[“epsilon”] = ”Alnilam”  **>>>** orion\_stars.pop(“alpha”)  ‘Betelgeuse’  **>>>** orion\_stars.popitem()  ‘Alnilam’  **>>>** orion\_stars  {'gamma': 'Bellatrix', 'delta': 'Mintaka'}  **>>>** del orion\_stars[“gamma”]  **>>>** orion\_stars  {'delta': 'Mintaka'} |

**Menghapus dictionary**

Untuk menghapus sebuah dictionary, dapat pula digunakan keyword del untuk menghapus objeknya, atau dengan method .clear() untuk menghapus seluruh elemennya.

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** constellations = {“Orion”:“Betelgeuse”, “Aquarius”:“Sadalmelik”, “Cassiopeia”:“Schedar”, “Canis Major”:”Sirius”}  **>>>** del orion\_stars  **>>>** orion\_stars  Traceback (most recent call last):  File "<stdin>", line 1, in <module>  NameError: name “orion\_stars” is not defined  **>>>** constellations.clear()  **>>>** constellations  {} |

**Mengecek elemen di dalam dictionary**

Kita dapat mengecek apakah suatu key telah ada di dalam sebuah dictionary dengan keyword in.

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”} |

|  |
| --- |
| **>>>** “beta” in orion\_stars  True  **>>>** “zeta” in orion\_stars  False |

**Panjang Dictionary**

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** len(orion\_stars)  4 |

**Looping sebuah dictionary**

Ada berbagai cara untuk mengiterasi sebuah dictionary, kalian dapat memilih caranya sesuai kebutuhan.

|  |
| --- |
| **>>>** orion\_stars = {“alpha”:“Betelgeuse”, “beta”:“Rigel”, “gamma”:“Bellatrix”, “delta”:”Mintaka”}  **>>>** for key in orion\_stars:  ... print(key) *# mencetak semua key dari*  *# dictionary, satu persatu*  ‘alpha’  ‘beta’  ‘gamma’  ‘delta’  **>>>** for key in orion\_stars:  ... print(orion\_stars[key]) *# mencetak semua value dari*  *# dictionary, satu persatu*  ‘Betelgeuse’  ‘Rigel’  ‘Bellatrix’  ‘Mintaka’ |

|  |
| --- |
| **>>>** for value in orion\_stars.values():  ... print(value) *# mencetak semua value dari*  *# dictionary, satu persatu*  ‘Betelgeuse’  ‘Rigel’  ‘Bellatrix’  ‘Mintaka’  **>>>** for key, value in orion\_stars.items():  ... print(key, value) *# mencetak semua key dan value dari*  *# dictionary, satu persatu*  ‘alpha Betelgeuse’  ‘beta Rigel’  ‘gamma Bellatrix’  ‘delta Mintaka’ |

Referensi:

<https://docs.python.org/3/tutorial/datastructures.html>

<https://docs.python.org/3/library/stdtypes.html>

<https://www.w3schools.com/python/python_sets.asp>

<https://www.w3schools.com/python/python_tuples.asp>

<https://www.w3schools.com/python/python_dictionaries.asp>