8. Tipe Data Koleksi

Pada pertemuan ini, kita akan membahas tentang **tipe data koleksi** (*collection*) pada python.

Apa maksud dari tipe data koleksi?

Ia adalah suatu jenis atau **tipe data yang digunakan untuk menghimpun kumpulan data**, atau data yang berjumlah lebih dari satu.

Secara umum, terdapat 4 tipe data koleksi pada python, yaitu:

- List
- Tuple
- Set
- Dictionary

Masing-masing dari 4 tipe data di atas memiliki sifat dan kegunaan sendiri-sendiri. Agar kita tahu kapan kita membutuhkan tipe data a dan kapan kita membutuhkan tipe data b, maka kita harus mempelajari semuanya dengan baik.

Tipe Data List

Tipe data list adalah tipe data koleksi yang bersifat *ordered* (terurut) dan juga bersifat *changeable* (bisa diubah). Tipe data ini bisa kita definisikan dengan tanda kurung siku `[]` di dalam Python.

Bagaimana cara membuat list?

Langsung saja, buka teks editor, lalu coba tuliskan kode program di bawah ini.

```
# list kosong
list_kosong = []

# list yang berisi kumpulan string
list_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']

# list yang berisi kumpulan integer
list_nilai = [80, 70, 90, 60]

# list campuran berbagai tipe data
list_jawaban = [150, 33.33, 'Presiden Sukarno', False]
```

Kode program 1

Pada **Kode program 1**, kita lihat bahwa **sebuah list didefinisikan menggunakan tanda kurung siku** (`[]`).

Kita juga saksikan bahwa list pada python, bisa berisi berbagai macam tipe data. Bisa terdiri dari tipe data yang sejenis mau pun dari tipe data yang berbeda-beda.

Menampilkan List

Kita bisa menggunakan perintah `print()` untuk melihat isi dari sebuah list, baik secara menyeluruh maupun sebagian.

Pada **Kode program 1** di atas, tambahkan kode program berikut ini untuk menampilkan semua isi dari list:

```
print('list_kosong:', list_kosong)
print('list_buah:', list_buah)
print('list_nilai:', list_nilai)
print('list_jawaban:', list_jawaban)
```

Jika dijalankan, kita akan mendapatkan output sebagai berikut:

```
list_kosong: []
list_buah: ['Pisang', 'Nanas', 'Melon', 'Durian']
list_nilai: [80, 70, 90, 60]
list_jawaban: [150, 33.33, 'Presiden Sukarno', False]
```

Kita juga bisa **menampilkan isi tertentu dari list dengan menggunakan indeks**. Setiap data pada list memiliki indeks sebagai alamat. Dan indeks adalah sebuah nilai integer dimulai dari 0 yang menjadi acuan di mana sebuah data disimpan di dalam list.

Tambahkan kode program berikut:

```
print(list_buah[0])
print(list_buah[2])
print(list_buah[1])
print(list_buah[3])
```

Output:

```
Pisang
Melon
Nanas
Durian
```

Kita juga bisa menggunakan indeks negatif untuk menampilkan data dari belakang. Perhatikan contoh berikut:

```
print(list_buah[-1])
print(list_buah[-2])
print(list_buah[-3])
print(list_buah[-4])
```

Output:

```
Durian
Melon
Nanas
Pisang
```

NB: yang perlu diperhatikan adalah: bahwa indeks negatif tidak dimulai dari 0, akan tetapi dimulai dari angka 1.

Slicing List

Slicing list adalah **teknik untuk memotong nilai pada list**. Maksudnya adalah: kita mengambil beberapa nilai dari anggota list dengan mendefinisikan indeks kiri dan indeks kanan.

Perhatikan contoh kode program berikut:

```
list_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']

print(list_buah[0:1])
print(list_buah[0:2])
print(list_buah[1:3])
print(list_buah[0:-1])
print(list_buah[-1:-3])
print(list_buah[-1:3])
print(list_buah[-3:-1])
```

Jika kita eksekusi, program di atas akan menghasilkan output:

```
['Pisang']
['Pisang', 'Nanas']
['Nanas', 'Melon']
['Pisang', 'Nanas', 'Melon']
[]
[]
['Nanas', 'Melon']
```

Keterangan

- parameter indeks sebelah kiri mendefinisikan awal indeks dari nilai yang akan ditampilkan.
- parameter indeks sebelah kanan mendefinisikan batas yang harus ditampilkan.

Slicing tanpa batas

Kita juga bisa melakukan slicing data tanpa mendefinisikan indeks batas. Coba perhatikan contoh berikut:

```
list_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']

print(list_buah[0:])
print(list_buah[1:])
print(list_buah[2:])
print(list_buah[3:])
print(list_buah[:0])
print(list_buah[:1])
print(list_buah[:2])
print(list_buah[:3])
print(list_buah[:4])
```

Kode program di atas akan menampilkan output sebagai berikut:

```
['Pisang', 'Nanas', 'Melon', 'Durian']
['Nanas', 'Melon', 'Durian']
['Melon', 'Durian']
['Durian']
[]
['Pisang']
['Pisang', 'Nanas']
['Pisang', 'Nanas', 'Melon']
['Pisang', 'Nanas', 'Melon', 'Durian']
```

Mengubah data di dalam list

Pada awal pembahasan, dikatakan bahwa list adalah **tipe data yang bersifat *changable* alias bisa diubah**.

Caranya seperti mengubah nilai variabel pada umumnya. Perhatikan contoh berikut:

```
list_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']

print(list_buah)

# ubah data pertama
list_buah[0] = 'Jeruk'

print(list_buah)

# ubah data terakhir
list_buah[-1] = 'Mangga'

print(list_buah)
```

Output:

```
['Pisang', 'Nanas', 'Melon', 'Durian']
['Jeruk', 'Nanas', 'Melon', 'Durian']
['Jeruk', 'Nanas', 'Melon', 'Mangga']
```

Kita juga bisa mengubah data dalam range

Di dalam python, kita juga bisa mengubah data dalam *range* tertentu secara sekaligus. Caranya tidak jauh berbeda dengan apa yang telah kita pelajari pada poin *slicing data list*.

Pada kode program di atas, tambahkan lagi kode program berikut:

```
# ubah data dalam range
list_buah[1:3] = ['Naga', 'Pepaya']

print(list_buah)
```

Maka kita akan mendapati bahwa nilai `Nanas` dan `Melon` akan berubah menjadi `Naga` dan `Pepaya`.

Output:

```
['Jeruk', 'Naga', 'Pepaya', 'Mangga']
```

Menambah item ke dalam list

Setelah kita mengubah data pada list, sekarang kita akan mencoba untuk menambahkan sebuah data baru ke dalam list.

Menambah data di belakang

Yang pertama, kita bisa menggunakan fungsi `append()`. Fungsi ini menerima satu parameter, yang mana parameter tersebut akan dimasukkan sebagai nilai baru pada list, dan nilai baru tersebut berada pada akhir item.

```
list_buah = ['Jeruk', 'Naga', 'Pepaya', 'Mangga']
print(list_buah)

# tambah data di belakang list
list_buah.append('Sirsak')
print(list_buah)
```

Menambah data di depan

Selain fungsi `append()`, kita juga bisa menambahkan item ke dalam list dengan menggunakan fungsi `insert()`. Fungsi insert ini menerima dua buah parameter:

1. Parameter pertama untuk mendefinisikan posisi indeks dari data yang akan dimasukkan
2. Parameter kedua untuk mendefinisikan nilai yang akan dimasukkan ke dalam list

Berikut ini contoh untuk memasukkan nilai `Jambu` ke dalam `list_buah` pada indeks `0`.

```
# tambah data di awal list
list_buah.insert(0, 'Jambu')
print(list_buah)
```

Menambah data di mana pun

Tidak hanya terbatas indeks `0`, kita juga bisa memasukkan nilai pada indeks berapa pun pada list.

```
# tambah data di index mana pun dalam list
list_buah.insert(2, 'Manggis')
print(list_buah)
```

Jika 3 potongan kode program di atas dijalankan, maka kita akan mendapatkan output seperti berikut:

```
['Jeruk', 'Naga', 'Pepaya', 'Mangga']
['Jeruk', 'Naga', 'Pepaya', 'Mangga', 'Sirsak']
['Jambu', 'Jeruk', 'Naga', 'Pepaya', 'Mangga', 'Sirsak']
['Jambu', 'Jeruk', 'Manggis', 'Naga', 'Pepaya', 'Mangga', 'Sirsak']
```

Menghapus item dari list

Untuk menghapus item dari list, kita bisa menggunakan dua buah fungsi; fungsi `pop()` dan fungsi `remove()`, kita juga bisa menggunakan *statement* `del`.

Menghapus item dengan fungsi pop()

Fungsi `pop()` akan mengambil item terakhir dari sebuah list, lalu menghapusnya. Karena ia juga "mengambil", maka kita bisa menyimpan hasil kembalian dari fungsi `pop()` ke dalam sebuah variabel.

Perhatikan contoh berikut:

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

# hapus satu angka di belakang
angka_yang_terhapus = list_angka.pop()
print('angka yang terhapus:', angka_yang_terhapus)
print(list_angka)
```

Output:

```
[1, 2, 3, 4, 5]
angka yang terhapus: 5
[1, 2, 3, 4]
```

Menghapus dengan fungsi remove()

Selanjutnya adalah fungsi `remove()`. Fungsi ini akan menghapus data yang memiliki nilai yang sama dengan parameter yang dimasukkan. Perhatikan contoh berikut:

```
list_buah = ['Mangga', 'Jambu', 'Jeruk', 'Jambu']
print(list_buah)

# hapus item pertama dengan nilai 'Jambu'
list_buah.remove('Jambu')

print(list_buah)
```

Output:

```
['Mangga', 'Jambu', 'Jeruk', 'Jambu']
['Mangga', 'Jeruk', 'Jambu']
```

Menghapus dengan statement del

Selanjutnya, kita juga bisa menghapus item pada list dengan menggunakan statement `del`. Dengan statement ini, kita bisa menghapus indeks berapa pun dari item list.

Perhatikan contoh berikut:

```
print('\n' * 2)

list_buah = ['Mangga', 'Jambu', 'Jeruk', 'Jambu']
print(list_buah)

del list_buah[1]
print(list_buah)

del list_buah[0:2]
print(list_buah)
```

Output:

```
['Mangga', 'Jambu', 'Jeruk', 'Jambu']
['Mangga', 'Jeruk', 'Jambu']
['Jambu']
```

Menggabungkan dua buah list atau lebih

Berikutnya hal umum yang biasa kita lakukan dengan list adalah: menggabungkan dua buah list (atau lebih) menjadi satu kesatuan.

Misal kita memiliki 3 list berikut:

```
a = [1, 2, 3]
b = ['a']
c = [True, 'b', False]
```

Kita bisa dengan mudah menggabungkan ketiganya menggunakan operator `+`.

```
listBaru = a + b + c

print(listBaru)
```

Program di atas akan menghasilkan output:

```
[1, 2, 3, 'a', True, 'b', False]
```

Mengurutkan data

Terakhir tapi bukan yang paling akhir, kita bisa mengurutkan data list pada python dengan memanggil fungsi `<list>.sort()`.

Perhatikan contoh berikut:

```
list_buah = ['Mangga', 'Jeruk', 'Zaitun', 'Apel', 'Durian']
print(list_buah)

# urutkan secara ascending
list_buah.sort()
print(list_buah)

# membalikkan posisi item list (tidak harus berurut)
list_buah.reverse()
print(list_buah)
```

Output:

```
['Mangga', 'Jeruk', 'Zaitun', 'Apel', 'Durian']
['Apel', 'Durian', 'Jeruk', 'Mangga', 'Zaitun']
['Zaitun', 'Mangga', 'Jeruk', 'Durian', 'Apel']
```

Fungsi-fungsi bawaan list

Masih ada banyak sekali fungsi-fungsi list pada python yang belum kita bahas.

Secara umum, berikut ini di antara fungsi-fungsi list yang bisa kita manfaatkan untuk menyelesaikan berbagai macam permasalahan.

Nama	Keterangan
`append()`	Menambahkan elemen baru pada list
`clear()`	Menghapus semua item pada list
`copy()`	Mengembalikan hasil duplikat dari list
`count()`	Mengembalikan jumlah item pada list sesuai yang didefinisikan
`index()`	Mengembalikan indeks pertama dari item yang sudah didefinisikan
`insert()`	Menambahakn item baru pada list pada posisi tertentu
`pop()`	Menghapus item terakhir pada list, atau juga bisa menghapus item pada posisi yang didefinisikan
`remove()`	Hapus item pada list sesuai dengan nilai yang didefinisikan
`reverse()`	Membalikkan posisi tiap item pada list
`sort()`	Mengurutkan list

Tuple

Tuple adalah 1 dari 4 tipe data kolektif pada python yang berguna untuk menyimpan lebih dari satu nilai dalam satu variabel secara sekaligus.

Tuple bersifat *ordered* (terurut) dan juga bersifat ***unchangable*** (tidak bisa diubah). *Ordered* berarti datanya bisa kita akses menggunakan indeks, dan **unchangeable** berarti datanya tidak akan pernah bisa diubah setelah pertama kali definisikan.

Dalam python, tipe data tuple didefinisikan dengan tanda kurung `()`.

Apa bedanya Tuple dengan List?

Tuple sama saja dengan list. Dia sama-sama digunakan untuk menyimpan data himpunan. Sama-sama bisa menampung berbagai macam tipe data dalam satu himpunan. Hanya saja **setelah diberi nilai, tuple tidak bisa diubah lagi**. Hal ini berbeda dengan list.

Dari segi penulisan, list menggunakan kurung siku `[]` sedangkan tuple menggunakan kurung biasa `()`.

Bagaimana cara membuat tuple?

Ada 3 cara untuk membuat tuple. Perhatikan contoh berikut:

```
# cara standar
tuple_jenis_kelamin = ('laki-laki', 'perempuan')

# tanpa kurung
tuple_status_perkawinan = 'menikah', 'lajang'

# menggunakan fungsi tuple()
tuple_lulus = tuple(['lulus', 'tidak lulus'])
```

Keterangan

- Cara yang pertama adalah cara standar dan paling dasar
- Cara yang kedua tanpa tanda kurung. Ini mungkin kelihatan agak aneh, tapi yang seperti ini normal di python
- Cara yang ketiga adalah dengan menggunakan fungsi `tuple()` dan melemparkan `list` sebagai parameternya.

Tuple kosong

Untuk membuat tuple kosong, kita cukup dengan menuliskan dua tanda kurung seperti berikut:

```
tuple_kosong = ()
```

Tuple yang hanya berisi satu item

Untuk mendefinisikan tuple yang hanya berisi satu item, **kita tetap diharuskan menulis tanda koma**.

```
tuple_tunggal = (10,)
```

Kalau tidak, maka python akan menggap tanda kurungnya tidak ada, seperti contoh berikut:

```
print(type((10))) # yang ini dianggap integer biasa
print(type((10,))) # yang ini dianggap tuple
```

Output:

```
<class 'int'>
<class 'tuple'>
```

Cara mengakses nilai tuple

Mengakses data pada tuple tidak jauh berbeda dengan cara mengakses data pada list, bahkan bisa kita bilang sama persis dalam keumumannya.

Kita bisa mengakses nilai pada tuple dengan langsung mendefinisikan indeks-nya seperti berikut:

```
# cara standar
tuple_jenis_kelamin = ('laki-laki', 'perempuan')

print(tuple_jenis_kelamin[1]) # indeks satu
print(tuple_jenis_kelamin[0]) # indeks nol
```

Output:

```
perempuan
laki-laki
```

Kita juga bisa mengakses nilai pada tuple dengan **negatif indeks**:

```
print(tuple_jenis_kelamin[-2])
print(tuple_jenis_kelamin[-1])
laki-laki
perempuan
```

Slicing tuple

Slicing adalah teknik memotong nilai dari sebuah tuple. Sintaksnya sama saja dengan teknis *slicing* di list. Tidak berbeda.

Untuk melakukan slicing, kita perlu mendefinisikan range indeks dengan pemisah tanda titik dua (`:`).

Perhatikan kode berikut:

```
tuple_buah = ('Pisang', 'Nanas', 'Melon', 'Durian')

print(tuple_buah[0:1])
print(tuple_buah[0:2])
print(tuple_buah[1:3])
print(tuple_buah[0:-1])
print(tuple_buah[-1:-3])
print(tuple_buah[-1:3])
print(tuple_buah[-3:-1])
```

Output:

```
('Pisang',)
('Pisang', 'Nanas')
('Nanas', 'Melon')
('Pisang', 'Nanas', 'Melon')
()
()
('Nanas', 'Melon')
```

Mengubah data pada tuple

Kita telah mempelajari **bahwa kita bisa mengubah data pada list**. Akan tetapi, berbeda dengan Tuple. **Kita tidak bisa mengubah data pada tuple**, kita hanya akan mendapatkan error.

Jika benar-benar ingin mengubah data yang ada pada tuple maka gunakanlah list!

Tuple memang difungsikan untuk kasus-kasus tertentu di mana nilai data bersifat tetap, tidak akan berubah selama runtime. Seperti misalnya data jenis kelamin.

Sequence Unpacking

Fitur selanjutnya dari Tuple adalah: ***sequence unpacking***. Fitur ini berfungsi untuk **mengekstrak isi dari tuple** ke dalam variabel-variabel tunggal secara berurutan. Kita hanya perlu menggunakan operator *assignment* standar (simbol sama dengan =) dan mendefinisikan nama variabel dengan koma.

Contohnya adalah seperti ini:

```
siswa = ('Nurul Huda', 'Bangkalan', 24)

# ekstrak data atau juga dinamakan sequence unpacking
nama, asal, usia = siswa

# setiap variabel di atas akan memiliki nilai dari tiap isi tuple
# secara berurutan
print('Nama:', nama)
print('Asal:', asal)
print('Usia:', usia)
```

Jika kita jalankan, kita akan mendapatkan output seperti berikut:

```
Nama: Nurul Huda
Asal: Bangkalan
Usia: 24
```

Penjelasan

Jika kita perhatikan kode program dan output di atas, kita bisa simpulkan bahwa:

1. Variabel `nama` memiliki nilai dari `siswa[0]`
2. Variabel `asal` memiliki nilai dari `siswa[1]`
3. Variabel `usia` memiliki nilai dari `siswa[2]`

Dan pengekstrakan nilai tersebut terjadi sesuai dengan urutan penulisan variabel, coba bolak-balikkan urutan variabelnya lalu lihat hasilnya seperti apa.

Menggabungkan dua buah tuple atau lebih

Hal lain yang bisa kita lakukan dengan tuple adalah: menggabungkan beberapa tuple menjadi satu tuple baru.

Kita bisa melakukan hal tersebut menggunakan operator penjumlahan `+` seperti contoh berikut:

```
>>> a = (1, 2, 3)
>>> b = (50, 60, 70)
>>>
>>> c = a + b
>>> c
(1, 2, 3, 50, 60, 70)
>>>
```

Fungsi-fungsi bawaan tuple

Sama seperti list, tuple juga memiliki fungsi-fungsi bawaan yang bisa kita gunakan seperti berikut:

Nama	Keterangan
`len()`	Menghitung jumlah item pada tuple
`max()`	Mencari nilai paling besar dari sebuah tuple
`min()`	Mencari nilai paling kecil dari sebuah tuple

Contoh penggunaan:

```
>>> nilai_semester_1 = (80, 90, 100, 88, 60)
>>> max(nilai_semester_1)
100
>>> min(nilai_semester_1)
60
>>> len(nilai_semester_1)
5
```

Set

Set dalam bahasa pemrograman python adalah tipe data kolektif yang digunakan untuk **menyimpan banyak nilai** dalam satu variabel dengan ketentuan:

- nilai anggota yang disimpan harus unik (tidak duplikat)
- nilai anggota yang sudah dimasukkan tidak bisa diubah lagi
- set bersifat unordered alias tidak berurut –yang artinya tidak bisa diakses dengan index.

Untuk lebih memahami 3 poin di atas, kita akan langsung melakukan praktik.

Cara membuat set

Secara umum kita bisa membuat set dengan 2 cara: dengan kurung kurawal `{}`, atau dengan sebuah `list` yang kita passing ke dalam fungsi `set()`.

Perhatikan kode program berikut:

```
# menggunakan kurung kurawal
himpunan_siswa = {'Huda', 'Lendis', 'Wahid', 'Basith'}
print(himpunan_siswa)

# mengkonversi list ke dalam set
himpunan_buah = set(['mangga', 'apel'])
print(himpunan_buah)

# set dengan tipe data yang berbeda-beda
set_campuran = {'manusia', 'hewan', 5, True, ('A', 'B')}
print(set_campuran)
```

Jika kita jalankan kode di atas, kita akan mendapatkan output sebagai berikut:

```
{'Wahid', 'Lendis', 'Basith', 'Huda'}
{'apel', 'mangga'}
{True, 5, ('A', 'B'), 'hewan', 'manusia'}
```

Set bersifat unordered

Tipe data set bersifat *unordered* alias tidak berurut. Itu artinya, kita tidak bisa menggunakan index untuk mengakses nilai pada set, kita hanya akan mendapatkan error:

```
set_ku = {'a'}
print(set_ku[0])
```

Pesan error:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object is not subscriptable
```

Kita juga bisa perhatikan kode program yang telah kita buat sebelumnya:

```
himpunan_siswa = {'Huda', 'Lendis', 'Wahid', 'Basith'}
print(himpunan_siswa)
```

Di mana kita mendefinisikan 4 anggota set dengan urutan: `Huda`, `Lendis`, `Wahid`, dan `Basith`.

Akan tetapi setelah kita print, kita malah mendapatkan urutan yang berbeda:

```
{'Wahid', 'Lendis', 'Basith', 'Huda'}
```

Set bersifat unchangable

Set bersifat `unchangable`, yang berarti bahwa nilai yang sudah kita masukkan ke dalam set, tidak bisa kita ubah lagi.

Akan tetapi, kita tetap bisa menambah dan menghapus anggota pada set.

Dan, karena set bersifat *unchangable*, set juga hanya bisa menerima anggota dari tipe data yang juga bersifat *immutable*.

Perhatikan contoh berikut:

```
# anggota set harus dari tipe data yang immutable
set_buah = { 'mangga', 'lemon', 'alpukat', True, 1, 2, 3 }

# kita bisa menjadikan tuple sebagai anggota
# karena ia bersifat immutable
papan_ketik = {
  (1, 2, 3),
  (4, 5, 6),
  (7, 8, 9),
  (0)
}
```

Tapi kita **tidak bisa memasukkan list** sebagai anggota karena list bersifat mutable, Perhatikan contoh berikut:

```
x = { 35, 100, ['a', 'b'] }
```

Pesan error:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

Tidak bisa menerima nilai duplikat

Selain itu `set` pada python juga tidak bisa menerima nilai duplikat. Jika kita memasukkan nilai yang sudah ada pada suatu set, maka nilai tersebut hanya akan muncul atau dimasukkan 1 kali saja.

Perhatikan contoh berikut ketika membuat list dengan kata **"pagi"** yang muncul **sebanyak 2 kali**:

```
list_kata = [
  'pagi', 'ini', 'adalah', 'pagi', 'yang',
  'sangat', 'cerah'
]

print(list_kata)
```

Output:

```
['pagi', 'ini', 'adalah', 'pagi', 'yang', 'sangat', 'cerah']
```

Sedangkan jika masukkan string **"pagi"** sebanyak dua kali dalam set seperti ini:

```
kata_unik = {
  'pagi', 'ini', 'adalah', 'pagi', 'yang',
  'sangat', 'cerah'
}

# atau bisa langsung konversi list_kata menjadi set
# supaya lebih ringkas:
# kata_unik = set(list_kata)

print(kata_unik)
```

Maka kata **"pagi"** hanya akan dimasukkan satu kali saja:

```
{'yang', 'sangat', 'ini', 'pagi', 'cerah', 'adalah'}
```

Menambah Anggota Baru

Seperti yang kita singgung di atas bahwa meskipun nilai set tidak bisa diubah, tapi tetap bisa ditambah dan dihapus.

Kita bisa menambah anggota baru ke dalam set dengan fungsi `add()` dan fungsi `update()`.

Perhatikan contoh berikut:

```
himpunan_abjad = {'a', 'b', 'c'}
print(himpunan_abjad)

# menambah satu-satu
himpunan_abjad.add('d')
himpunan_abjad.add('e')

# menambah lebih dari satu anggota sekaligus
himpunan_abjad.update({ 'f', 'g' })
# bisa juga pakai list
himpunan_abjad.update(['h', 'i'])

print(himpunan_abjad)
```

Jika kita jalankan, kita akan mendapatkan output sebagai berikut:

```
{'a', 'c', 'b'}
{'d', 'i', 'e', 'a', 'b', 'g', 'h', 'f', 'c'}
```

Menghapus Anggota

Untuk menghapus anggota pada set, terdapat 4 fungsi yang bisa kita gunakan:

- `remove(nilai)` Untuk menghapus nilai yang dicari. Jika nilai yang dicari tidak ada, maka akan error.
- `discard(nilai)` Untuk menghapus nilai yang dicari. Jika nilai yang dicari tidak ada, tidak akan error.
- `pop()` Mengambil dan menghapus nilai yang ada di sebelah kiri.
- `clear()` Menghapus semua anggota.

Mari kita coba satu persatu.

```
himpunan = {'maya', 'budi', 100, ('a', 'b'), False, True}
print(himpunan)

# akan error jika nilai 100 tidak ada di dalam set
himpunan.remove(100)
print(himpunan)

# tidak akan error jika ('a', 'b') tidak ada di dalam set
himpunan.discard(('a', 'b'))
print(himpunan)

# remove nilai yang ada di sebelah kiri
nilaiYangDihapus = himpunan.pop()
print('nilaiYangDihapus =', nilaiYangDihapus)
print(himpunan)
```

```
# hapus semua nilai
himpunan.clear()
print(himpunan)
```

Jika dijalankan, berikut keseluruhan output yang akan kita dapat:

```
{False, True, 100, 'maya', 'budi', ('a', 'b')}
{False, True, 'maya', 'budi', ('a', 'b')}
{False, True, 'maya', 'budi'}
nilaiYangDihapus = False
{True, 'maya', 'budi'}
set()
```

Fungsi Keanggotaan Pada Set

Di antara keunggulan tipe data set adalah: keunikan anggotanya. Sehingga, dengan keunikan tersebut, python menyediakan kepada kita berbagai fungsi keanggotaan yang berguna untuk pengolahan data. Di antaranya:

- fungsi union (gabungan)
- fungsi intersection (irisan)
- fungsi difference (selisih)
- fungsi symmetric difference (komplement)
- dan lain sebagainya

Agar lebih jelas, mari kita coba beberapa di antaranya.

Anggap saja kita memliki dua grup WA: yaitu grup WA SMA, dan grup WA SMP. Dan kita memiliki teman bernama **Ratna** dan **Andi** yang merupakan teman SMP dan juga teman SMA sekaligus.

Untuk merepresentasikan hal tersebut, kita akan buat dua buah set seperti berikut:

```
grup_smp = {
  'andi', 'budi', 'ratna', 'sari'
}
```

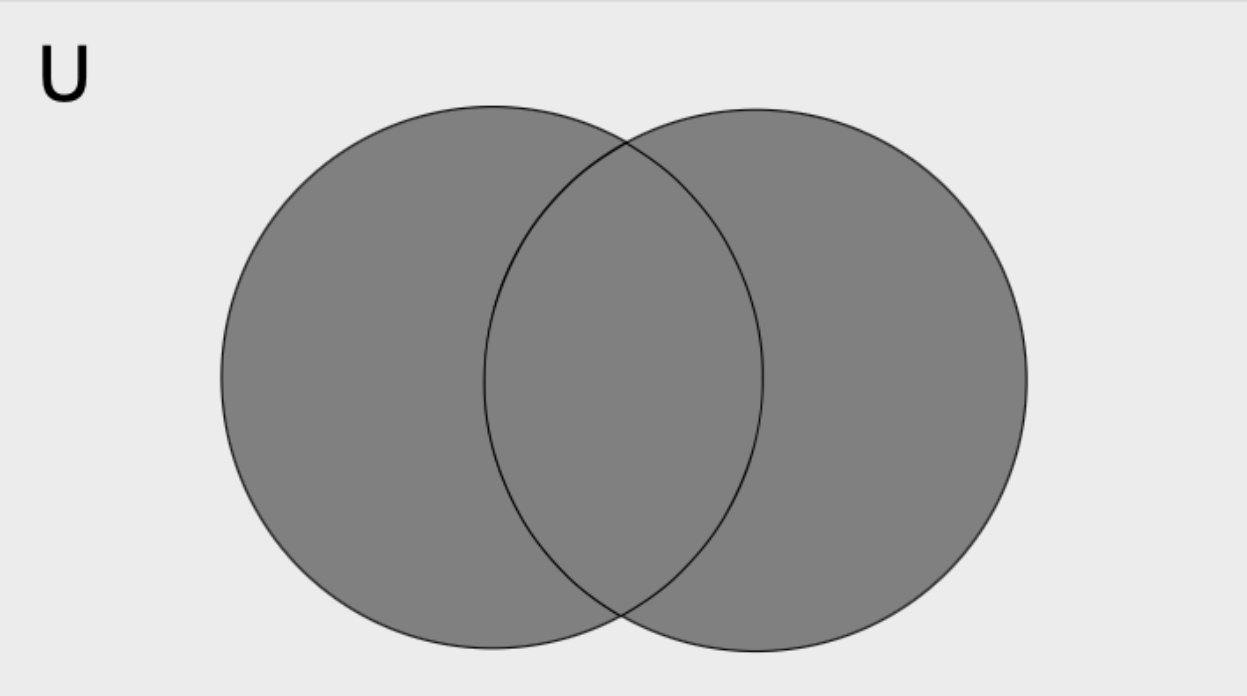
```
grup_sma = {
  'putri', 'ratna', 'andi', 'agus'
}
```

Kita sekarang punya dua buah set: satu untuk grup smp, dan satu untuk grup sma.

Union (Gabungan)

Kita bisa melakukan operasi union, alias menggabungkan kedua anggota dari grup smp dan grup sma.

Ilustrasi operasi union:



Dalam python, kita bisa melakukan operasi union dengan simbol `pipe` (`|`) atau bisa memanggil fungsi `set.union()` seperti berikut:

```
# cara 1
print(grup_smp | grup_sma)
# cara 2
print(grup_smp.union(grup_sma)
```

Jika dijalankan, kita akan mendapatkan output:

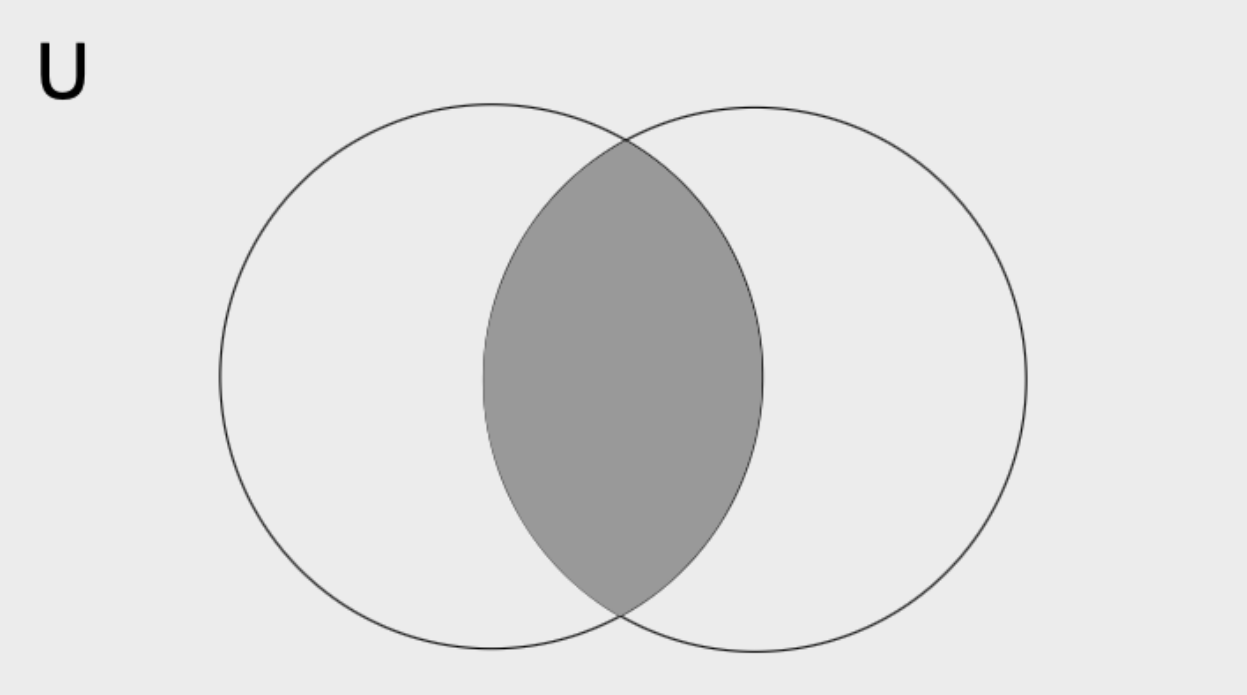
```
{'agus', 'andi', 'budi', 'ratna', 'sari', 'putri'}
{'agus', 'andi', 'budi', 'ratna', 'sari', 'putri'}
```

Perhatikan output di atas: meskipun **ratna** dan **andi** adalah anggota dari kedua grup tersebut, tapi nama mereka **hanya muncul satu kali**.

Intersection (Irisan)

Berikutnya adalah *intersection* atau irisan: yaitu kita akan mengambil siapa saja siswa yang menjadi anggota grup sma **yang juga adalah** anggota grup smp.

Ilustrasi operasi intersection:



Kita bisa melakukannya dengan dua cara: yakni menggunakan simbol `&`, atau menggunakan fungsi `set.intersection()` seperi berikut:

```
print(grup_smp & grup_sma) # cara 1
print(grup_smp.intersection(grup_sma)) # cara 2
```

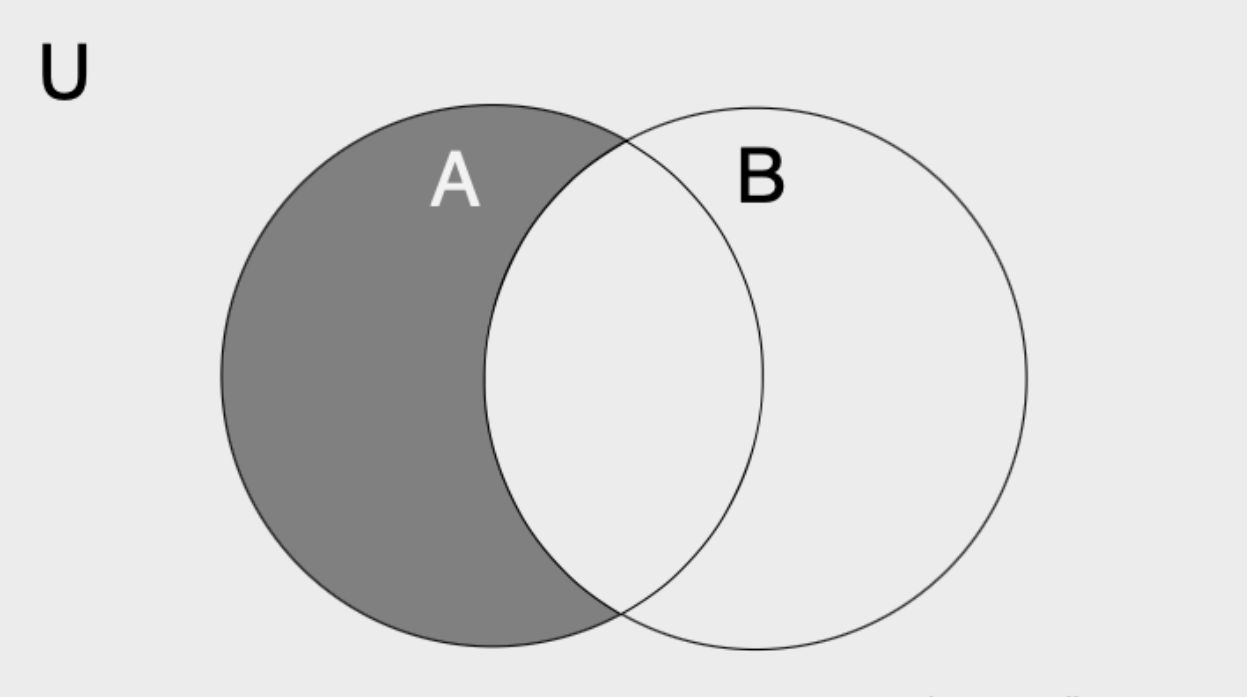
Jika dijalankan, kita akan mendapatkan output:

```
{'ratna', 'andi'}
{'ratna', 'andi'}
```

Difference (Selisih)

Difference atau selisih adalah proses mengekstrak anggota grup pertama, yang bukan anggota grup kedua.

Perhatikan ilustrasi berikut:



Untuk melakukannya, kita bisa menggunakan simbol `-` atau dengan memanggil fungsi `set.difference()` seperti ini:

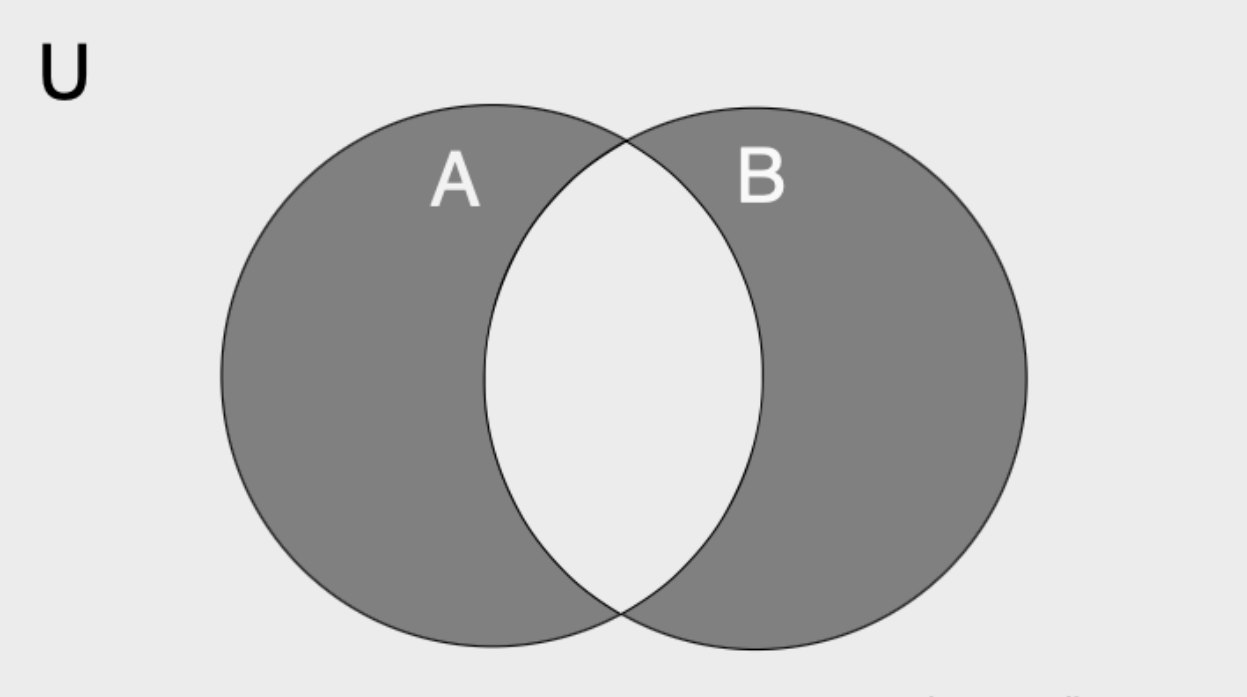
```
# difference
print('\nanggota grup smp yang bukan anggota grup sma')
print(grup_smp - grup_sma)
print(grup_smp.difference(grup_sma))

print('\ndibalik, anggota grup sma yang bukan anggota grup smp:')
print(grup_sma - grup_smp)
print(grup_sma.difference(grup_smp))
```

Outpunya akan kita lihat setelah kita mencoba fungsi keanggotaan `symmetric_difference`.

Symmetric Difference (Yang hanya menjadi anggota satu grup saja)

Bedanya dengan difference, symmetric difference akan menghasilkan anggota-anggota dari kedua grup, yang mana tiap anggota tersebut hanya menjadi anggota dari satu grup saja.



Tambahkan kode program berikut:

```
# symmetric_difference
print('\nanggota yang hanya ikut satu grup saja:')
print(grup_sma.symmetric_difference(grup_smp))
```

Jika kita jalankan, gabungan kode program dari fungsi `difference` dan juga `symmetric_difference` akan menghasilkan output sebagai berikut:

```
anggota grup smp yang bukan anggota grup sma
{'sari', 'budi'}
{'sari', 'budi'}

dibalik, anggota grup sma yang bukan anggota grup smp:
{'agus', 'putri'}
{'agus', 'putri'}

anggota yang hanya ikut satu grup saja:
{'budi', 'sari', 'agus', 'putri'}
```

Menampilkan Anggota Set Dengan Perulangan

Berikutnya kita juga bisa menampilkan tiap anggota dari sebuah set dengan melakukan perulangan for.

Perhatikan contoh berikut:

```
himpunan_buah = {
  'pepaya', 'apel', 'jagung', 'rambutan'
}

for buah in himpunan_buah:
  print(buah)
```

Output:

```
rambutan
pepaya
jagung
apel
```

Fungsi-Fungsi Bawaan Set

Berikut ini adalah kesimpulan fungsi-fungsi bawaan tipee data set yang bisa kita gunakan:

Fungsi	Kegunaan
`add()`	Menambahkan satu anggota ke dalam set
`clear()`	Menghapus keseluruhan anggota set
`copy()`	Membuat salinan satu set menjadi set baru
`difference()`	Melakukan operasi selisih antar dua set
`difference_update()`	Menghapus anggota set lain yang ada di set sekarang
`discard()`	Menghapus satu anggota dari set
`intersection()`	Melakukan operasi irisan antar dua set
`intersection_update()`	Mengupdate anggota yang menjadi irisan dari dua set
`isdisjoint()`	Mengembalikan nilai `True` jika dua set tidak memiliki irisan
`issusbset()`	Mengembalikan nilai `True` jika set lain memiliki anggota dari set sekarang
`issuperset()`	Mengembalikan nilai `True` jika set sekarang memiliki anggota dari set lain
`pop()`	Menghapus dan mengembalikan nilai yang dihapus dari sebuah set
`remove()`	Menghapus suatu nilai dari set
`symmetric_difference()`	Melakukan operasi komplemen antar dua set
`symmetric_difference_update()`	Mengupdate set dari hasil komplemen
`union()`	Melakukan operasi gabungan dari dua atau lebih set

Fungsi	Kegunaan
`update()`	Mengupdate set dengan menambahkan suatu nilai tertentu

Agar lebih memahami semua fungsi di atas, bisa dicoba satu-persatu.

Dictionary

Apa itu dictionary? Sebelum kita membahas pengertiannya, silakan perhatikan kode program berikut ini:

```
pertemuan_hari_ini = {
  "judul": "Belajar Dictionary Pada Python 3",
  "tanggal": "01 Februari 2021",
  "kategori": [
    "Python", "Python Dasar"
  ],
  "page_views": 10,
  "published": True,
  "share_count": {
    "facebook": 0,
    "twitter": 2
  }
}
```

Secara garis besar apa tugas dan isi dari variabel `pertemuan_hari_ini` bisa dipahami secara langsung.

Variabel tersebut menyimpan beberapa informasi sekaligus seperti `judul`, `tanggal`, `kategori` (yang berupa list), `page_views`, dan sebagainya.

Pengertian Dictionary

Lalu, apa sebenarnya dictionary itu?

Dictionary adalah tipe data pada python yang berfungsi untuk menyimpan kumpulan data/nilai dengan pendekatan “key-value”.

Perhatikan contoh di atas, variabel `pertemuan_hari_ini` adalah dictionary.

Ia memiliki 6 buah **“key”** atau **“atribut”**, yaitu:

1. `judul`
2. `tanggal`
3. `kategori`

4. `page_views`
5. `published`
6. dan `share_count` (bahkan value dari `share_count` sendiri adalah sebuah dictionary)

Yang mana masing-masing atribut di atas memiliki informasi/nilai sendiri-sendiri sesuai dengan namanya.

Dictionary sendiri **memiliki dua buah komponen** inti:

1. Yang pertama adalah **key**, ia merupakan nama atribut suatu item pada dictionary.
2. Yang kedua adalah **value**, ia adalah nilai yang disimpan pada suatu atribut.

Sifat Dictionary Items

Dictionary items memiliki 3 sifat, yaitu:

1. `Unordered` - tidak berurutan
2. `Changeable` - bisa diubah
3. `Unique` - alias tidak bisa menerima dua keys yang sama

Unordered artinya ia tidak berurutan, sehingga key/atribut yang pertama kali kita definisikan, tidak berarti dia akan benar-benar menjadi yang "pertama" dibandingkan dengan key yang lainnya. Juga, unordered berarti **tidak bisa diakses menggunakan index** (integer) sebagaimana halnya list.

Sedangkan **changeable** artinya kita **bisa kita mengubah value** yang telah kita masukkan ke dalam sebuah dictionary. Hal ini berbeda dengan tipe data set mau pun tuple yang mana keduanya bersifat *immutable* alias tidak bisa diubah.

Dan yang terakhir, dictionary **tidak bisa memiliki lebih dari satu key yang sama**karena ia bersifat **unique**. Sehingga jika ada dua buah key yang sama, key yang didefinisikan terakhir akan menimpa nilai dari key yang didefinisikan lebih awal.

Perhatikan contoh berikut:

```
artikel = {
  "judul": "Menu Masakan Enak",
  "judul": "Menu Masakan Enak Tradisional"
}

print(artikel.get("judul"))
```

Kode program di atas akan menghasilkan output:

```
Menu Masakan Enak Tradisional
```

Kita bisa perhatikan bahwa pada dictionary `artikel` di atas, terdapat dua buah item dengan nama "judul", akan tetapi ketika kita tampilkan hasilnya hanya nilai yang terakhir kita masukkan.

Tipe Data pada Items Dictionary

Selain 3 sifat yang telah disebutkan, item pada dictionary juga bisa menerima berbagai macam tipe data, mulai dari tipe data asli mau pun tipe data terusan seperti objek.

Yang artinya, dictionary juga bisa memiliki value berupa dictionary juga.

Dan ini telah kita lihat langsung contohnya pada variabel `pertemuan_hari_ini` di kode program yang paling atas.

Cara Membuat Dictionary

Berikutnya adalah bagaimana cara membuat dictionary pada python.

Untuk membuatnya, terdapat 2 cara:

1. Yang pertama dengan tanda kurung kurawal `{}`.
2. Dan yang kedua bisa menggunakan fungsi atau konstruktor `dict()`.

Perhatikan contoh berikut:

```
# cara pertama
buku = {
  "judul": "Daun Yang Jatuh Tidak Pernah Membenci Angin",
  "penulis": "Tere Liye"
}

# cara kedua
buku = dict(
  judul="Daun Yang Jatuh Tidak Pernah Membenci Angin",
  penulis="Tere Liye"
)
```

Cara Mengakses Item Pada Dictionary

Kita bisa mengakses item pada dictionary dengan dua cara:

- dengan menggunakan kurung siku ([])
- atau dengan menggunakan fungsi `get()`

Perhatikan contoh berikut:

```
pertemuan_hari_ini = {
  "judul": "Belajar Dictionary Pada Python 3",
  "tanggal": "01 Februari 2021",
  "kategori": [
    "Python", "Python Dasar"
  ],
  "page_views": 10,
  "published": True,
  "share_count": {
    "facebook": 0,
    "twitter": 2
  }
}
```

Kita bisa mengaksesnya dengan:

```
print('Judul:', pertemuan_hari_ini.get('judul'))
# atau
print('Tanggal:', pertemuan_hari_ini['tanggal'])

# bisa menggunakan fungsi berantai untuk dictionary bertingkat
print('Facebook share:',
pertemuan_hari_ini.get('share_count').get('facebook'))
# bisa juga dengan kurung siku dua-duanya
print('Twitter share:', pertemuan_hari_ini['share_count']['twitter'])
```

Output program:

```
Judul: Belajar Dictionary Pada Python 3
Tanggal: 01 Februari 2021
Facebook share: 0
Twitter share: 2
```

Hanya saja, di antara keunggulan menggunakan fungsi `get()` adalah:

Kita bisa mengatur nilai default jika key pada dictionary yang kita panggil tidak ditemukan. Hal itu akan mencegah sistem untuk menampilkan error.

Perhatikan contoh berikut, perintah yang pertama akan menimbulkan error\ sedangkan perintah yang kedua tidak:

```
share_count = pertemuan_hari_ini.get('share_count')

# ini error
print('Instagram share:', share_count['instagram'])
```

```
# sedangkan ini tidak error
print('Instagram share:', share_count.get('instagram', 0))
```

Coba jalankan lagi dengan menghapus perintah yang pertama.

Perulangan Untuk Dictionary

Kita juga bisa menampilkan semua isi dari sebuah dictionary dengan memanfaatkan perulangan.

Perhatikan contoh berikut:

```
buku = {
  'judul': 'Hafalan Sholat Delisa',
  'penulis': 'Tere Liye'
}

for key in buku:
  print(key, '->', buku[key])
```

Jika dijalankan, kita akan mendapatkan output:

```
judul -> Hafalan Sholat Delisa
penulis -> Tere Liye
```

Atau kita juga bisa memanggil fungsi `dictionary.items()` untuk perulangan yang lebih simpel:

```
for nama_atribut, nilai in buku.items():
  print(nama_atribut, '->', nilai)
```

Kode di atas akan menghasilkan output yang sama.

Mengubah Nilai Item

Seperti yang telah kita singgung di atas bahwasanya item pada dictionary bersifat *changeable* alias bisa diubah.

Untuk mengubah nilai item pada suatu dictionary, caranya simpel seperti mengubah variabel pada umumnya.

Contoh:

```
mahasiswa = {
  'nama': 'Lendis Fabri',
  'asal': 'Indonesia'
}
```

```
# mengubah data
print('Nama awal:', mahasiswa.get('nama'))
mahasiswa['nama'] = 'Andi Mukhlish'
print('Setelah diubah:', mahasiswa.get('nama'))
```

Jika dieksekusi, sistem akan memberikan kita output seperti ini:

```
Nama awal: Lendis Fabri
Setelah diubah: Andi Mukhlish
```

Menambahkan Item

Untuk menambahkan key dan item baru, caranya seperti mengedit item.

Jadi:

- Kalau key yang kita definisikan **ternyata sudah ada**, sistem akan **mengganti item** yang lama dengan yang baru.
- Tapi jika key yang kita definisikan **ternyata tidak ada**, maka sistem akan **menambahkannya** sebagai item baru.

Contoh:

```
mahasiswa = {
  'nama': 'Lendis Fabri',
  'asal': 'Indonesia',
}

# output None
print('Hobi:', mahasiswa.get('hobi'))
# tambah data
mahasiswa['hobi'] = 'Memancing'
# print ulang
print('Hobi dari {} adalah {}'.format(
  mahasiswa.get('nama'),
  mahasiswa.get('hobi')
))
```

Kita akan mendapatkan output:

```
Hobi: None
Hobi dari Lendis Fabri adalah Memancing
```

Menghapus Item

Untuk menghapus item, terdapat dua cara:

1. Menggunakan statement `del <dict[key]>`.
2. Atau menggunakan fungsi `dictionary.pop()`

Perhatikan contoh berikut:

```
mahasiswa = {
  'nama': 'Wahid Abdullah',
  'usia': 18,
  'asal': 'Indonesia'
}

del mahasiswa['nama']
mahasiswa.pop('usia')
mahasiswa.pop('asal')
```

Apa bedanya memakai cara pertama dan cara kedua?

Bedanya, kalau menggunakan fungsi `pop()`, kita bisa mendapatkan nilai kembalian dari data yang telah dihapus.

Contoh:

```
pesan_singkat = {
  "isi": "ISI PESAN INI HANYA BISA DIBACA SEKALI SAJA!! 😱"
}

isi_pesan = pesan_singkat.pop('isi')

# akses langsung dari dictionary
# ouput: None
print('isi pesan:', pesan_singkat.get('isi'))

# akses dari hasil kembalian yang telah disimpan
print('isi pesan:', isi_pesan)
```

Output dari program di atas:

```
isi pesan: None
isi pesan: ISI PESAN INI HANYA BISA DIBACA SEKALI SAJA!! 😱
```

Operator Keanggotaan

Kita bisa memanfaatkan operator keanggotaan untuk tipe data dictionary pada python.

Perhatikan contoh berikut:

```
siswa = {
  'nama': 'Renza Ilhami'
}
```

```
print('Apakah variabel siswa memiliki key nama?')
print('nama' in siswa)

print('\nApakah variabel siswa TIDAK memiliki key usia?')
print('usia' not in siswa)
```

Output:

```
Apakah variabel siswa memiliki key nama?
True

Apakah variabel siswa TIDAK memiliki key usia?
True
```

Panjang atau Banyak Key Pada Dictionary

Terakhir tapi bukan yang paling akhir, kita bisa juga menghitung jumlah key yang terdapat pada sebuah dictionary dengan fungsi `len()`.

Contoh:

```
sekolah = {
  'nama': 'Sekolah Dasar Negeri Surabaya 1',
  'jenjang': 'Sekolah Dasar',
  'akreditasi': 'A'
}

print(
  "Jumlah atribut variabel sekolah adalah:",
  len(sekolah)
)
```

Output:

```
Jumlah atribut variabel sekolah adalah: 3
```

Latihan

1. Buatlah suatu list bernama years_list, dimulai dengan tahun kelahiran anda, dan seterusnya sampai tahun saat anda berumur 5 tahun. Sebagai contoh, jika anda lahir pada 1980. List years_list = [1980, 1981, 1982, 1983, 1984, 1985].

   - Pada taun berapakah dalam years_list anda berumur 3 tahun?
   - Pada tahun berapakah dalam years_list anda paling tua?

2. Buatlah suatu list bernama things dengan 3 string ini sebagai elemennya: "mozzarella", "cinderella", "salmonella".

   - Huruf besarkan elemen dalam things yang merujuk ke seseorang dan kemudian cetak list tersebut. Apakah itu mengubah elemen di dalam list?
   - Buatlah elemen “cheesy” dari things menjadi huruf besar dan kemudian cetak list.
   - Hapus elemen “disease” dari things dan cetak list.

3. Buatlah suatu list bernama surprise dengan elemen: "Groucho", "Chico", and "Harpo".

   - Huruf-kecilkan elemen terakhir dari list surprise, balik hurufnya, dan kemudian besarkan.

4. Buatlah suatu kamus English-to-Indonesia bernama e2i dan cetaklah. Kata-kata awal yang harus ada: dog = anjing, cat = kucing, dan tiger = macan.

   - Menggunakan kamus tiga kata e2i, cetak kata Indonesia dari tiger.
   - Buatlah kamus Indonesia-to-English bernama i2e dari e2i.
   - Gunakan i2e, cetak kata English yang Indonesianya dalah kucing.
   - Buat dan cetak sehimpunan kata English dari kunci dalam e2i.

5. Buatlah suatu kamus multilevel bernama life. Gunakan string ini untuk kunci level tertinggi: 'animals', 'plants’, dan 'other’. Buatlah kunci 'animals’ merujuk ke kamus lain dengan kunci 'cats', 'octopi’, dan 'emus’. Buatlah kunci 'cats’ merujuk ke suatu list string dengan nilai 'Henri', 'Grumpy’, dan 'Lucy’. Buatlah semua kunci lain merujuk ke kamus kosong.

   - Cetak kunci top-level dari life.
   - Cetak kunci untuk life['animals’].
   - Cetak nilai untuk life['animals']['cats'].

Referensi

[1] https://www.w3schools.com/python/python_tuples.asp - diakses tanggal 15 Februari 2021

[2] https://www.pythoncontent.com/unpacking-sequence-in-python - diakses tanggal 15 Mei 2021