List, Tuple Set dan Dictionary

Abby Rafdi C

Penyusun Modul: Ratna Aditya Apsari
Editor : Delyanda Rahmadini ,Dinar Hidayah, Hilma Arifah R, M. Hamed Bagus P, Fikri Nurachman

Bab 4.

List, tuple, set dan dictionary

Di Python, terdapat empat tipe penyimpanan atau pengoleksi data, disebut juga sebagai collection types, yaitu:

- List, yaitu koleksi yang berurutan dan dapat berubah *(mutable).* List memperbolehkan adanya elemen duplikat.
- Tuple, yaitu koleksi yang berurutan tapi tidak dapat diubah *(mutable)*. Tuple memperbolehkan adanya elemen duplikat.
- Set, yaitu koleksi yang tidak berurutan, tidak berindeks, dan tidak dapat diubah *(immutable)*. Set tidak memperbolehkan adanya elemen duplikat.
- Dictionary, yaitu koleksi yang berurutan dan dapat diubah *(mutable)*. Dictionary tidak memperbolehkan adanya elemen duplikat.

List

List adalah struktur data yang paling umum dan sederhana di Python. Tiap elemen dalam list memiliki indeks posisi yang dimulai dari nol. Menggunakan list, kita dapat melakukan operasi *indexing*, *slicing*, penambahan, perkalian, dan mengecek membership dari suatu elemen list.

List ditulis dengan braket kotak (“[ ]”) dan tiap elemennya dipisahkan dengan koma. Hal yang menarik tentang list adalah list dapat menyimpan elemen-elemen yang tipe datanya berbeda, misalnya *string*, *integer*, dan *float* di dalam list yang sama. Contoh sederhana dari list adalah sebagai berikut:

```
list1 = [2000, 2010, 2020, 2030]
list2 = [“satu”, “dua”, “tiga”, “empat”]
list3 = [“tahun”, 2022, “bulan”, 1]
```

Mengakses elemen dalam list

Menggunakan indeks *(indexing),* kita dapat mengakses elemen yang kita inginkan dalam suatu list. Contohnya adalah sebagai berikut:

```
list1 = [2000, 2010, 2020, 2030]
print("Elemen list1[0]: ", list1[0])
```

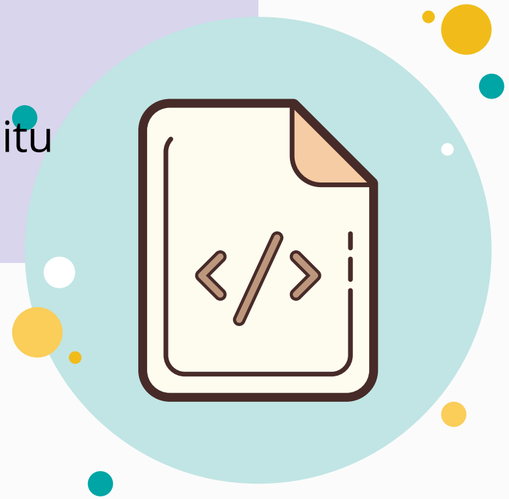
Code di atas akan menampilkan elemen ke-0 dari list1, yaitu 2000. Untuk mengakses elemen tertentu, gunakan braket kotak dan tentukan indeks dari elemen yang diinginkan.

Mengakses elemen dalam list

Kita dapat juga mengakses lebih dari satu elemen di saat yang bersamaan *(slicing),* seperti berikut:

```
list1 = [2000, 2010, 2020, 2030]
print("Elemen list1[0:3]: ", list1[0:3])
```

Dengan code di atas, kita akan menampilkan elemen ke-0 hingga 3 dari list1, yaitu `2000, 2010, dan 2020.`



Nested List

List dapat berada di dalam list lainnya atau disebut sebagai *nested list*

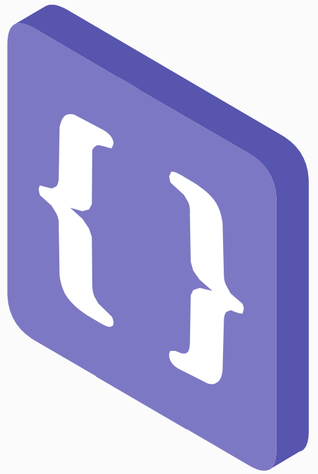
```
nested_list = [‘Anang’, ‘Depok’, [17, 7, 1994], 2022]
print(nested_list[1])
print(nested_list[2])
```

Code di atas akan memunculkan output sebagai berikut:

Depok

```
[17, 7 , 1994]
```

Mengubah elemen dari list



Elemen-elemen dalam list dapat diubah/dimodifikasi, dengan cara menambahkan atau menghilangkan elemen.

1. Penambahan elemen baru

Untuk menambahkan elemen baru pada list, kita dapat menggunakan

```
list1 = [2000, 2010, 2020, 2030]
list1[1] = 2005
```

Baris kedua akan menambahkan elemen baru, yaitu 2005, pada indeks 1 dalam list1. Jika kita lakukan perintah print pada list1, hasilnya akan menjadi seperti berikut:

```
[2000, 2005, 2010, 2020, 2030]
```

Perhatikan bahwa angka 2010 tidak hilang dari list, melainkan hanya "digeser" posisinya.

Mengubah elemen dari list

2. Penghapusan/penghilangan elemen

Untuk menghapus elemen dalam suatu list, kita dapat menggunakan perintah `del` sebagai berikut

```
list1 = [2000, 2010, 2020, 2030]
del list1[3]
```

Perintah del list1[3] akan menghapus elemen berindeks 3 pada list1, yaitu 2030. Sehingga jika kita lakukan perintah print pada list tersebut, hasilnya akan menjadi:

```
[2000, 2010, 2020]
```

Mengkonstruksi list baru menggunakan perintah `list()`

Kita dapat mengkonstruksi sebuah list menggunakan perintah list() untuk membuat list baru. Contohnya adalah sebagai berikut:

```
listbaru = list((“merah”, “biru”, “kuning”))
print(listbaru)
```

Perhatikan bahwa pembuatan list menggunakan perintah tersebut memerlukan kurung ganda. Jika kita me run code tersbut, output yang didapatkan adalah:

```
[“merah”, “biru”, “kuning”]
```

Operasi dasar list

Operasi	Output	Deskripsi
`len([1,2,3])`	`3`	Mencari "panjang" dari list tersebut (berapa banyak elemen yang ada dalam list)
`[1,2,3] + [4,5,6]`	`[1,2,3,4,5,6]`	Menggabungkan (*concatenate*) dua list yang berbeda menjadi satu list
`["A"] * 3`	`["A", "A", "A"]`	Mengulang elemen sebanyak n kali (atau pengulangan/*repetition*)
`2 in [1,2,3]`	`True`	Mengecek *membership* dari suatu elemen di dalam list tersebut
`for x in [1,2,3]:` `  print(x)`	`1 2 3`	Melakukan iterasi

Fungsi dan metode *built-in* untuk list

Beberapa fungsi *built-in* yang dapat digunakan untuk list adalah sebagai berikut:

Fungsi	**Deskripsi**
`len(list)`	Mencari panjang total dari list
`max(list)`	Mencari elemen dalam list dengan nilai maksimal
`min(list)`	Mencari elemen dalam list dengan nilai minimal
`list(seq)`	Mengubah tuple menjadi list
`sorted(seq)`	Mengurutkan list dan bisa mengembalikannya pada list yang baru (tidak mengubah list yang asli)

Fungsi dan metode *built-in* untuk list

Adapun beberapa metode built-in yang dapat digunakan untuk list adalah sebagai berikut:

Metode	Deskripsi
`list.append(obj)`	Menambahkan obyek `obj` ke dalam list
`list.count(obj)`	Menghitung berapa kali obyek `obj` berada dalam list
`list.extend(seq)`	Menambahkan konten `seq` ke dalam list (perbedaan dari append adalah, append hanya menambahkan 1 elemen obyek)
`list.index(obj)`	Memberikan indeks terkecil dalam list di mana obyek `obj` berada
`list.insert(indeks, obj)`	Menambahkan obyek `obj` ke dalam list pada indeks offset
`list.pop(obj = list[-1])`	Menghapus dan mengembalikan (*return*) obyek terakhir atau `obj` dari list
`list.sort()`	Mengurutkan list dan perubahannya langsung di*update* pada list yang asli
`list.reverse()`	Membalikkan lokasi/urutan obyek dalam list

List comprehension

List comprehension adalah cara singkat dan efisien untuk membuat suatu list dari list lain yang sudah ada. Misalnya kita memiliki list yang menyimpan tahun kelahiran dari beberapa orang. Kemudian, kita ingin membuat list baru untuk menyimpan umur dari orang-orang tersebut di tahun 2022. Menggunakan *for loop*, kita dapat membuat code seperti berikut:

```
tahun_kelahiran = [1995, 1996, 1997, 1998, 1999, 2000]
umur = []
for tahun in tahun_kelahiran:
    umur.append(2022 - tahun)
```

Output yang diperoleh adalah list dengan isi sebagai berikut:

```
print(umur)
```

```
[27, 26, 25, 24, 23, 22]
```

List comprehension

Sekarang, mari kita terapkan list comprehension untuk menyelesaikan masalah tersebut:

```
tahun_kelahiran = [1995, 1996, 1997, 1998, 1999, 2000]
umur = [2022 - tahun for tahun in tahun_kelahiran]
print(umur)
```

```
[27, 26, 25, 24, 23, 22]
```

Baris kedua adalah baris dimana kita menerapkan *list comprehension* tersebut. Kita menggunakan *for loop* di dalam list umur dan mempersingkat code awal kita menjadi satu baris. Bagian sebelum *for loop*, yaitu `2022 – tahun`, adalah item yang akan *di-append* atau ditambahkan ke list baru kita. *List comprehension* juga dapat digunakan untuk pernyataan *if* dan *looping* lainnya.

Latihan

Buatlah list yang menyimpan angka-angka dari 0 hingga 50. Kemudian, gunakan *list comprehension* untuk memisahkan bilangan-bilangan ganjil dan genap ke list masing-masing.

Solusi:

Latihan

Buatlah list yang menyimpan angka-angka dari 0 hingga 50. Kemudian, gunakan *list comprehension* untuk memisahkan bilangan-bilangan ganjil dan genap ke list masing-masing.

Solusi:

```
angka = list(range(0,51))

# list angka ganjil
angka_ganjil = [ganjil for ganjil in angka if ganjil %2 == 1]
print("angka ganjil: ", angka_ganjil)

# list angka genap
angka_genap = [genap for genap in angka if genap %2 == 0]
print("angka genap: ", angka_genap)
```

```
angka ganjil:  [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49]
angka genap:  [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50]
```

Latihan

1. Terdapat suatu list, `list1 = [30, 60, 90, 120, 150]`. Baliklah urutan dari list tersebut!

Solusi:

2. Buatlah list yang isinya angka dari 1 hingga 10. Kemudian, buat list baru yang menghitung kuadrat dari nilai-nilai tersebut.

Solusi:

Latihan

1. Terdapat suatu list, `list1 = [30, 60, 90, 120, 150]`. Baliklah urutan dari list tersebut!

Solusi:

```
list1 = [30, 60, 90, 120, 150]
list1.reverse()
print(list1)
```

```
[150, 120, 90, 60, 30]
```

```
list1 = [30, 60, 90, 120, 150]
list1.sort(reverse=True)
print(list1)
```

```
[150, 120, 90, 60, 30]
```

2. Buatlah list yang isinya angka dari 1 hingga 10. Kemudian, buat list baru yang menghitung kuadrat dari nilai-nilai tersebut.

Solusi:

```
angka = list(range(1,11))
kuadrat = [pow(i,2) for i in angka]
print(kuadrat)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
angka = list(range(1,11))
kuadrat = [i**2 for i in angka]
print(kuadrat)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Latihan

3. Buatlah program yang dapat menambahkan dua list sesuai dengan indeksnya. Pada list baru, item pertama harus merupakan item ke-0 dari list1 dan list2, item kedua harus merupakan item ke-1 dari `list1` dan `list2`, dan seterusnya (hint: menggunakan `zip()`)

```
list1 = [“Ha”, “sela”, “da”]
list2 = [“lo”, “mat”, “tang”]
```

Solusi:

TUPLE

Tuple adalah tipe data sekuensial yang berurutan dan tidak dapat diubah setelah dibuat. Berbeda dengan list, tuple menggunakan kurung biasa ( ). Contoh sederhana dari tuple adalah:

```
tuple1 = (“satu”, “dua”, “tiga”)
tuple2 = (2000, 2010, 2020, 2030)
tuple3 = “a”, “b”, “c”, “d”
```

Sama seperti list, tuple dapat menyimpan tipe data yang berbeda dalam satu variabel yang sama. Contohnya sebagai berikut:

```
tuple4 = (“xyz”, 45, True, 70, “abc”)
```

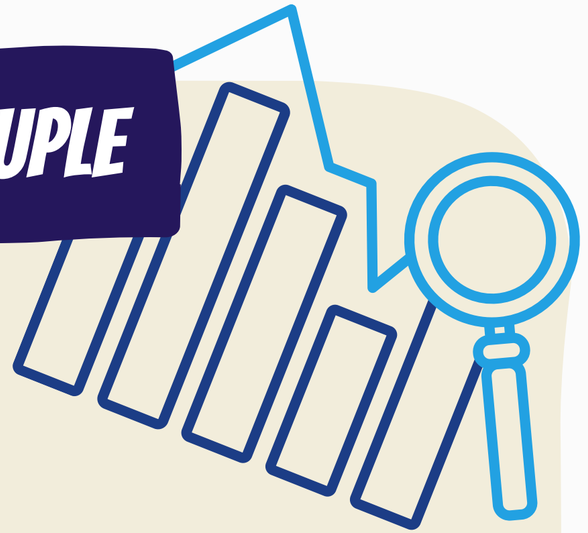
Kita dapat juga membuat tuple kosong dengan cara sebagai berikut:

```
tuple5 = ()
```

Kita dapat juga membuat tuple dengan satu nilai, penulisannya sebagai berikut:

```
tuple6 = (100)
```

MENGAKSES NILAI DALAM SUATU TUPLE



- Operasi *slicing*, *concatenation*, dan lainnya dapat dilakukan pada tuple. Pada dasarnya, nilai-nilai dalam suatu sekuens tuple memiliki indeks yang dimulai dari 0, sama dengan list. Sehingga untuk mengaksesnya kita dapat melakukan:

```
tuple1 = (“satu”, “dua”, “tiga”)
tuple2 = (2000, 2010, 2020, 2030)
print(“tuple1[0]: ”, tuple1[0])
print(“tuple2[1:3]: ”, tuple2[1:3])
```

- Dengan demikian, output yang diperoleh ketika *code* tersebut di-*run* adalah sebagai berikut:

```
tuple1[0]: satu
tuple2[1:3]: (2010, 2020)
```

MENGKONSTRUKTSI TUPLE BARU MENGGUNAKAN PERINTAH TUPLE()

Kita dapat mengkonstruksi sebuah tuple menggunakan perintah `tuple()` untuk membuat tuple baru (dapat diisi dengan *sequence* jenis apapun). Contohnya adalah sebagai berikut (pada contoh menggunakan *sequence* jenis list):

```
tuplebaru = tuple([“merah”, “biru”, “kuning”])
print(tuplebaru)
```

Perhatikan bahwa pembuatan tuple menggunakan perintah tersebut memerlukan kurung ganda. Jika kita me *run code* tersebut, output yang didapatkan adalah:

```
(‘merah’, ‘biru’, ‘kuning’)
```

MENGUBAH ELEMEN DARI TUPLE

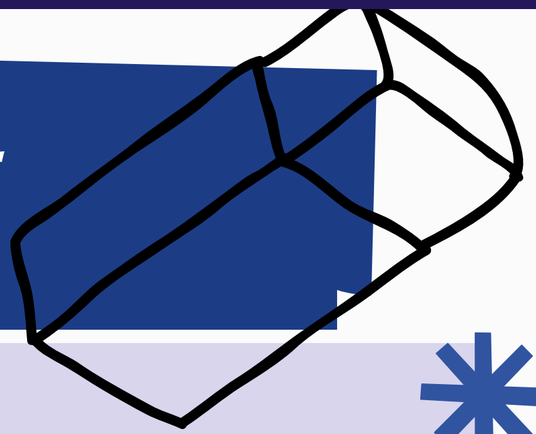
Tuple bersifat *immutable*, artinya elemen dalam tuple tidak dapat di-*update* atau diubah. Hal yang dapat dilakukan adalah mengambil sebagian dari tuple untuk membuat tuple baru yang memiliki elemen gabungan dari tuple tersebut. Misalnya:

```
tuple1 = (10, 44.77)
tuple2 = (“pqr”, “stu”)
tuple3 = tuple1 + tuple2
print(tuple3)
```

Menggunakan code di atas, output yang diperoleh akan sebagai berikut:

```
(10, 44.77, ‘pqr’, ‘stu’)
```

MENGHAPUS TUPLE



Karena tuple memiliki sifat tidak dapat diubah, maka elemen dari tuple tidak dapat dihapus setelah tuple tersebut dibuat. Hal yang bisa dilakukan hanyalah menghapus seluruh variabel tuple menggunakan fungsi `del()`. Contohnya seperti berikut:

```
tuple1 = (10, 44.77)
print(tuple1)
del(tuple1)
print(tuple1)
```

```
      2 print(tuple1)
      3 del(tuple1)
----> 4 print(tuple1)

NameError: name 'tuple1' is not defined
```

Hal ini terjadi karena variable `tuple1` sudah terhapus setelah baris `del(tuple1)` dieksekusi. Itulah mengapa error timbul ketika baris `print(tuple1)` dieksekusi, karena variabel `tuple1` sudah tidak ada.

Operasi Dasar Tuple

Operasi	Output	Deskripsi
`len((1,2,3))`	`3`	Mencari “panjang” dari list tersebut (berapa banyak elemen yang ada dalam list)
`(1,2,3) + (4,5,6)`	`(1,2,3,4,5,6)`	Menggabungkan (*concatenate*) dua list yang berbeda menjadi satu list
`(“A”,) * 3`	`(“A”, “A”, “A”)`	Mengulang elemen sebanyak n kali (pengulangan/*repetition*)
`2 in (1,2,3)`	`True`	Mengecek *membership* dari suatu elemen di dalam list tersebut
`for x in (1,2,3):` `  print x`	`1 2 3`	Melakukan iterasi

Fungsi dan metode built-in untuk tuple

Beberapa fungsi built-in yang dapat digunakan untuk tuple adalah sebagai berikut:

Fungsi	Deskripsi
`len(tuple)`	Mencari panjang total dari tuple
`max(tuple)`	Mencari elemen dalam tuple dengan nilai maksimal
`min(tuple)`	Mencari elemen dalam tuple dengan nilai minimal
`tuple(seq)`	Mengubah `seq` apapun menjadi bertipe tuple

LATIHAN!

1

Ambil setiap elemen dalam tuple berikut (*unpacking)* dan masing-masing masukkan dalam variabel baru

```
nama_sayur = (“paprika”, “kembang kol”, “wortel”, “kangkung”)
```

2

Hitunglah berapa kali nilai 8 muncul pada tuple berikut:

```
tuple1 = (0, 9, 2, 8, 45, 5, 7, 9, 8, 28, 8, 88, 8, 34, 5, 3,
32, 1, 8, 21, 18, 8, 7)
```

3

Lakukan pensortiran pada tuple sesuai dengan angka yang tersedia pada tuple didalamnya. (hint: menggunakan `sorted()` dengan key berupa fungsi `lambda`)

```
tuple1 = ((“a”, 12),(“b”, 45),(“c”, 8), (“d”,99))
Hasil yang diinginkan: ((“c”, 8), (“a”, 12), (“b”, 45), (“d”,99))
```

Solusi

```
nama_sayur = ("paprika", "kembang kol", "wortel", "kangkung")
a, b, c, d = nama_sayur
print(a)
print(b)
print(c)
print(d)
```

```
paprika
kembang kol
wortel
kangkung
```

1

```
tuple1 = (0, 9, 2, 8, 45, 5, 7, 9,
          8, 28, 8, 88, 8, 34, 5, 3,
          32, 1, 8, 21, 18, 8, 7)
print(tuple1.count(8))
```

```
6
```

2

```
tuple1 = (("a", 12),("b", 45),("c", 8), ("d", 99))
sorted_list = sorted(tuple1, key=lambda x: x[1])
sorted_tuple = tuple(sorted_list)
print(sorted_tuple)
```

```
(('c', 8), ('a', 12), ('b', 45), ('d', 99))
```

3

Catatan: untuk multidimensional tuple, proses *sorting* dapat menggunakan `sorted()` dengan `key` berupa fungsi `lambda` yang mengakses indeks elemen yang dijadikan acuan pensortiran

Set

- Set digunakan untuk menyimpan data yang tidak berurutan, tidak dapat diubah, tidak memperbolehkan adanya duplikat, dan tidak terindeks.
- Meskipun item dalam set tidak dapat diubah, tapi kita dapat menghapus dan menambahkan item baru. Set dibuat dengan kurung kurawal (“ { } ”)
- Contoh dari set adalah sebagai berikut:

```
sebuahset = {“manggis”, “mangga”, “markisa”}
print(sebuahset)
```

- Karena set memiliki sifat tidak berurutan, maka ketika kita menjalankan code di atas, urutan dari set `sebuahset` dapat berubah.
- Apa yang terjadi ketika kita memiliki nilai duplikat dalam sebuah set? Cobalah potongan code berikut!

```
sebuahset = {“manggis”, “mangga”, “markisa”, “manggis”}
print(sebuahset)
```

Mengkonstruksi set baru dengan fungsi set()

Kita dapat menggunakan fungsi `set()` untuk membuat set baru. Contohnya sebagai berikut:

```
set1 = set(("mangga", "manggis", "markisa"))
print(set1)
type(set1)
```

```
{'manggis', 'mangga', 'markisa'}

set
```

Menambahkan elemen baru pada set

Untuk menambahkan elemen baru pada sebuah set, kita dapat menggunakan metode `add()`. Contohnya adalah sebagai berikut:

```
nama = {"Agus"}
nama.add("Budi")
nama.add("Bambang")
print(nama)
```

```
{'Agus', 'Bambang', 'Budi'}
```

Bisa juga menggunakan `update()` untuk elemen yang lebih dari satu. Contohnya adalah sebagai berikut:

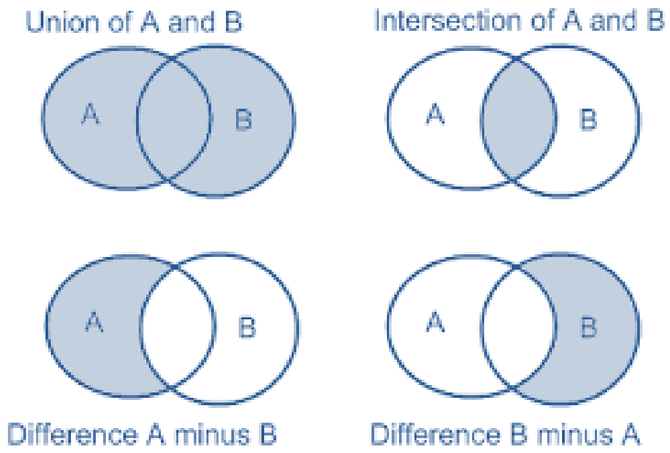
```
nama = {"Agus"}
nama_tambahan = {"Budi", "Bambang"}
nama.update(nama_tambahan)
print(nama)
```

```
{'Bambang', 'Budi', 'Agus'}
```

Metode dalam Set

Metode	Simbol	Deskripsi
`set1.intersection(set2)`	`set1 ∩ set2`	Mencari irisan dari dua buah set
`set1.union(set2)`	`set1 U set2`	Menggabungkan dua buah set
`set1.difference(set2)`	`set1 - set2`	Menghilangkan irisan antara `set1` dan `set2` pada `set1`

Catatan: metode-metode yang bisa dilakukan pada set merupakan operasi pada **himpunan** dalam matematika



Latihan

1. Lakukan proses *union* untuk kedua set di bawah:

```
set1 = {“Jakarta”, “Yogyakarta”}
set2 = {“Yogyakarta”, “Surabaya”}
```

Solusi:

```
set1 = {"Jakarta", "Yogyakarta"}
set2 = {"Yogyakarta", "Surabaya"}
set3 = set1.union(set2)
print(set3)
```

```
{'Jakarta', 'Yogyakarta', 'Surabaya'}
```

Latihan

2. Buatlah program yang dapat mengecek apakah bilangan dari 1 hingga 5 ada di dalam set berikut:

`set1 = {5, 8, 32, 54, 23, 1, 34, 4, 90}`

Solusi:

```
set1 = {5, 8, 32, 54, 23, 1, 34, 4, 90}
print("Apakah 1 ada di dalam set?", 1 in set1)
print("Apakah 2 ada di dalam set?", 2 in set1)
print("Apakah 3 ada di dalam set?", 3 in set1)
print("Apakah 4 ada di dalam set?", 4 in set1)
print("Apakah 5 ada di dalam set?", 5 in set1)
```

```
Apakah 1 ada di dalam set? True
Apakah 2 ada di dalam set? False
Apakah 3 ada di dalam set? False
Apakah 4 ada di dalam set? True
Apakah 5 ada di dalam set? True
```

Latihan

3. Buatlah program yang akan menghapus elemen *intersection* (irisan dari himpunan) dari `set2` pada `set1`

```
set1 = {4, 5, 6, 7, 8, 9}
set2 = {5, 7, 9, 11, 13}
```

Solusi:

```
set1 = {4, 5, 6, 7, 8, 9}
set2 = {5, 7, 9, 11, 13}
print("set1: ", set1.difference(set2))
```

```
set1:  {8, 4, 6}
```

DICTONARY

https://www.tutorialspoint.com/python/python_dictionary.htm
\https://www.w3schools.com/python/python_dictionaries.asp

- Dictionary digunakan untuk menyimpan data dalam pasangan *key* dan *value*. Data yang disimpan dengan tipe ini berurutan, dapat diubah, dan tidak boleh memiliki nilai duplikat.
- Dictionary dibuat dengan **kurung kurawal.** Contohnya sebagai berikut:

```
sebuah_dict = {"nama depan": "Kim",
               "nama keluarga": "Kardashian",
               "tahun kelahiran": 1980}
```

Perhatikan sintaks dalam dictionary. Dictionary dimulai dan diakhiri oleh kurung kurawal (“ { } ”), serta tiap *key* dan *value* dihubungkan dengan titik koma (“ : ”). Selain itu, pasangan *key-value* akan dipisah dengan tanda koma (,).

Mengakses nilai dalam suatu dictionary

Untuk mengakses nilai dalam dictionary, kita dapar menggunakan code sebagai berikut:

```
sebuahdict = {
    "nama depan": "Kim",
    "nama keluarga": "Kardashian",
    "tahun kelahiran": 1990 }
```

```
print ("sebuahdict ['nama depan']: ", sebuahdict['nama depan'])
print ("sebuahdict ['nama keluarga']: ", sebuahdict ['nama keluarga'])
```

```
sebuahdict ['nama depan']:  Kim
sebuahdict ['nama keluarga']:  Kardashian
```

Key dapat digunakan sebagai indeks untuk mengakses *value* pada dictionary

Key merupakan indeks dalam dictionary, bisa dalam bentuk string atau angka. Tuple dapat digunakan sebagai key jika tuple tersebut terdiri dari obyek yang bersifat immutable atau tidak dapat diubah. Jika tuple tersebut terdiri dari obyek *mutable* (dapat diubah), secara langsung ataupun tidak langsung, maka tuple tidak dapat digunakan sebagai *key*. Dengan itu, list tidak dapat digunakan sebagai *key* karena list merupakan obyek *mutable*.

Terdapat dua hal yang penting diketahui tentang *key*, yaitu:

- Nilai atau *value* dari suatu *key* tidak boleh lebih dari satu. Jika nilainya lebih dari satu, maka nilai terakhir yang akan diakui sebagai pasangan dari key tersebut.
  Contohnya:

```
dict = {'Nama': 'Tina', 'Pekerjaan': 'Penari', 'Pekerjaan': 'Programmer'}
print("dict['Pekerjaan']: ", dict['Pekerjaan'])
```

```
dict['Pekerjaan']:  Programmer
```

- *Key* harus bersifat *immutable* atau tidak dapat diubah. String, angka, atau tuple dapat digunakan sebagai *key*, tetapi obyek seperti list tidak dapat digunakan. Contohnya:

```
dict = {['Nama']: 'Tina', 'Pekerjaan': 'Penari'}
print("dict['Nama']: ", dict['Nama'])
```

```
----> 1 dict = {['Nama']: 'Tina', 'Pekerjaan': 'Penari'}
      2 print("dict['Nama']: ", dict['Nama'])

TypeError: unhashable type: 'list'
```

Meng-*update* dictionary

Dictionary dapat di-*update* dengan cara menambahkan, mengubah, atau menghapus pasangan *key-value*.

```
sebuahdict = {
    "nama depan": "Kim",
    "nama keluarga": "Kardashian",
    "tahun kelahiran": 1990 }

sebuahdict["tahun kelahiran"] = 1980
sebuahdict["umur"] = 42
```

```
print('sebuahdict["tahun kelahiran"]:', sebuahdict["tahun kelahiran"])
print('sebuahdict["umur"]:', sebuahdict["umur"])
```

```
sebuahdict["tahun kelahiran"]: 1980
sebuahdict["umur"]: 42
```

```
print(sebuahdict)
```

```
{'nama depan': 'Kim',
 'nama keluarga': 'Kardashian',
 'tahun kelahiran': 1980,
 'umur': 42}
```

Menghapus elemen dalam dictionary

Kita dapat menghapus elemen dalam dictionary secara individual atau menghapus seluruh konten variabel dictionary tersebut. Perintah yang digunakan untuk melakukan hal ini adalah `del()` atau `clear()`

1. Menghapus seluruh konten dictionary
   Untuk menghapus seluruh konten dictionary, pengguna dapat membuat code seperti berikut:

```
sebuahdict.clear()
print(sebuahdict)
```

```
{}
```

```
del(sebuahdict)
print(sebuahdict)
```

```
      1 del sebuahdict
----> 2 print(sebuahdict)

NameError: name 'sebuahdict' is not defined
```

Error ini menandakan bahwa dictionary `sebuahdict` telah terhapus.

Metode `clear()` berguna untuk menghapus semua entry dalam dictionary, sedangkan fungsi `del()` akan menghapus dictionary tersebut.

2. Menghapus sebuah elemen dictionary

Perintah `del()` juga dapat digunakan untuk menghapus elemen tertentu pada dictionary, contohnya sebagai berikut:

```
sebuahdict = {
"nama depan": "Kim",
"nama keluarga": "Kardashian",
"tahun kelahiran": 1980,
"umur": 42 }
```

```
del(sebuahdict["tahun kelahiran"])
print(sebuahdict)
```

```
{'nama depan': 'Kim', 'nama keluarga': 'Kardashian',
'umur': 42}
```

Fungsi built-in untuk dictionary

Fungsi	Deskripsi
len(dict)	Mencari panjang total dari dictionary, yaitu banyaknya item yang ada di dictionary
str(dict)	Memberikan representasi string dari dictionary
type(dict)	Mengecek tipe dari variable dict

Beberapa metode untuk dictionary

Metode	Deskripsi
`dict.clear()`	Menghapus semua elemen dalam dictionary dict
`dict.copy()`	Meng-*copy* dictionary dict
`dict.get(key)`	Mencari key yang bernama `key` dalam dictionary dict dan mengembalikan nilainya. Jika nilainya tidak ada dalam dict, maka akan mengembalikan nilai default (None).

`dict.items()`	Mengambil seluruh pasangan *key-value* yang ada di dictionary dict
`dict.keys()`	Mengambil seluruh *keys* yang ada dalam dictionary dict
`dict.update(dict2)`	Menambahkan pasangan *key-value* yang ada dalam dictionary `dict2` ke `dict`
`dict.values()`	Mengambil seluruh *value* yang ada dalam dictionary dict

Latihan

1. Print *value* dari *key* `'dasar pemrograman'` pada *nested* dictionary berikut:

```
sampleDict = {
    "kelas": "Cortana",
    "mahasiswa": {
        "nama": "Anang",
        "nilai": {
            "dasar pemrograman": 90,
            "machine learning": 80,
            "data analysis": 65
        }
    }
}
```

Solusi:

```
sampleDict["mahasiswa"]["nilai"]["dasar pemrograman"]
```

```
90
```

2. Buatlah program yang dapat menghapus *key* "nama" dan "almamater" dari dictionary berikut:

```
contoh_dict = {
            "nama": "Julie",
            "umur": 25,
            "almamater": "Brown University",
            "kewarganegaraan": "Inggris"
}
```

Solusi:

```
# key yang akan dihapus
key = ["nama", "almamater"]

new_dict = {}
for i in contoh_dict.keys()-key:
    new_dict[i] = contoh_dict[i]
print(new_dict)
```

```
{'umur': 25, 'kewarganegaraan': 'Inggris'}
```

```
# alternatif dengan comprehension
new_dict1 = {i: contoh_dict[i] for i in contoh_dict.keys()-key}
print(new_dict1)
```

```
{'umur': 25, 'kewarganegaraan': 'Inggris'}
```

`zip()` function

- Fungsi `zip()` mengambil 2 atau lebih parameter *sequence* (list, tuple, string, range)
- Kedua parameter dengan indeks yang sama akan bergabung dan menjadi objek baru bertipe zip (dengan konten berupa tuple)

```
# define a string and a list
text = 'xyzt'
a_list = [1, 2, 3, 4]

# zip the string and the list
zip_obj = zip(text, a_list)

print(zip_obj)
for i in zip_obj:
    print(i)
```

```
<zip object at 0x000001DFA8CBDB40>
('x', 1)
('y', 2)
('z', 3)
('t', 4)
```

Latihan

1. Terdapat dua list yang dapat dijadikan pasangan *key-value* dari dictionary. Buatlah dictionary dengan *key* dari variable `key` dan *value* dari variable `value` tersebut!

```
key = [‘tanggal lahir’, ‘umur’, ‘pekerjaan’]
value = [‘28-07-1996’, 25, ‘pro-gamer’]
```

Solusi:

```
key = ['tanggal lahir', 'umur', 'pekerjaan']
value = ['28-07-1996', 25, 'pro-gamer']
dict1 = dict(zip(key, value))
print(dict1)
```

```
{'tanggal lahir': '28-07-1996', 'umur': 25, 'pekerjaan': 'pro-gamer'}
```

Latihan

2. Buatlah program yang dapat menambahkan dua list sesuai dengan indeksnya. Pada list baru, item pertama harus merupakan item ke-0 dari list1 dan list2, item kedua harus merupakan item ke-1 dari `list1` dan `list2`, dan seterusnya

`list1 = [“Ha”, “sela”, “da”]`

`list2 = [“lo”, “mat”, “tang”]`

Solusi:

```
list1 = ["Ha", "sela", "da"]
list2 = ["lo", "mat", "tang"]
listbaru = [a + b for a,b in zip(list1,list2)]
print(listbaru)
```

```
['Halo', 'selamat', 'datang']
```