BUKU PENDAMPING PRAKTIKUM
DATA WAREHOUSE PERTEMUAN 2
(SEMESTER 4)

Oleh:

Wahyu Pebrianto

PROGRAM STUDI MANAJEMEN INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI JEMBER

TAHUN 2020

PRAKTIKUM 2

PRAKTIKUM DATA WAREHOUSE PERTEMUAN 2

DATA COLLECTION [LIST, TUPLE, SET, DICTIONARY, MODUL]

Data collection pada Bahasa pemograman Python digunakan untuk mengolah sebuah data, data collection banyak digunakan oleh seorang data science dalam menganalisa sebuah data sesuai kebutuhan. Salah satunya sebagian proses dalam menghasilkan sebuah produk data. Ada empat data collection dalam bahasa pemrograman Python yaitu List, Tuple, Set dan Dictionary. Yang akan kita bahas satu persatu.

LIST [1/9]

List adalah tipe data yang berisi satu atau beberapa nilai di dalamnya. Nilai – nilai ini sering juga disebut item, elemen, atau anggota list. List dibuat dengan menempatkan semua item di dalam tanda kurung [ ], dipisahkan oleh tanda koma. Anggota list bisa berisi satu tipe data, atau campuran, contoh list sebagai berikut :

```
1 my_list = [1,2,3,4,5]
2 print(my_list)
3
4 my_list = [1, 3.5, "Hello"]
5 print(my_list)
```

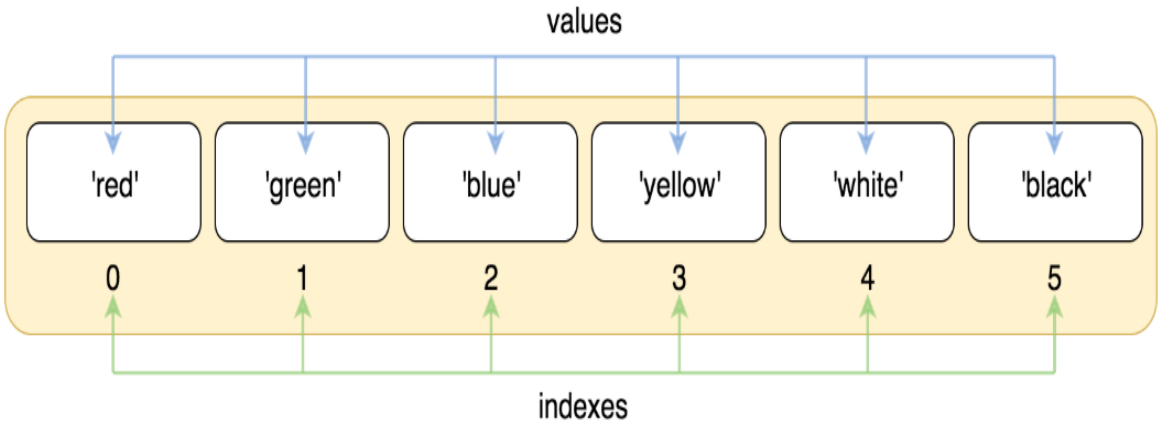
```
[1, 2, 3, 4, 5]
[1, 3.5, 'Hello']
```

(2) INDEXING [1/2]

[POSITIF INDEX]

positif index adalah bagaimana kita mengakses sebuah list dengan memulai dengan index positif, index positif di mulai dari index 0, berikut contoh gambar index dan value pada list :

Perhatikan Gambar ?



Berikut contoh program akses index list dengan Bahasa pemograman python:

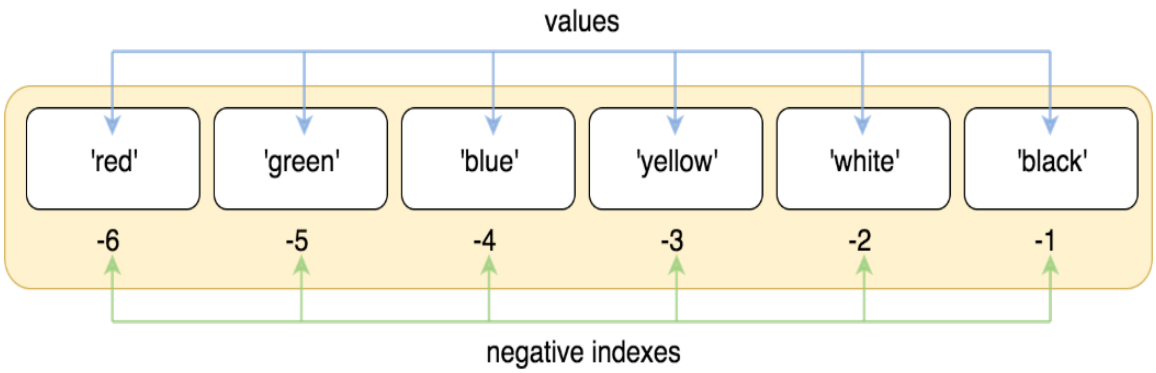
```
1 list_saya=['red', 'green', 'blue', 'yellow', 'white', 'black']
2 print(list_saya[0])
3 print(list_saya[1])
4 print(list_saya[5])
```

```
red
green
black
```

[NEGATIF INDEX]

negatif index adalah kebalikan index positif bagaimana kita mengakses sebuah list dengan dengan mengambil index paling akhir dari nilai list, index negaitf di mulai dari index -1, berikut contoh gambar index negatif dan value pada list :

Perhatikan Gambar ?



Berikut contoh program akses index negatif dengan Bahasa pemograman python **:**

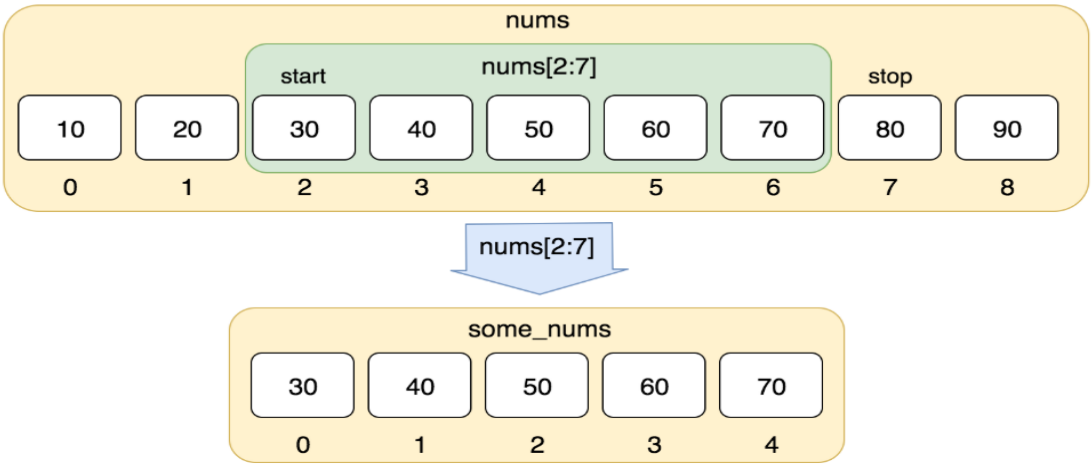
```
1 listsaya=['red', 'green', 'blue', 'yellow', 'white', 'black']
2 print(listsaya[-1])
3 print(listsaya[-2])
4 print(listsaya[-6])
```

```
black
white
red
```

(2) SLICING A PYTHON LIST [1/2]

Slicing digunakan untuk mengiris sebuah list, yang digunakan untuk menampilkan isi list berdasarkan keinginan kita, salah satunya bila kita ingin mengambil data dari Kita bisa mengakses anggota list dari range tertentu dengan menggunakan operator slicing titik dua (:).Slicing akan lebih mudah bila kita memahami indeks dengan baik. Perhatikan program berikut:

Perhatikan Gambar ?



Dari gambar di atas [2:7] berarti kita akan menampilkan indeks dimulai dari indeks 2 sampai sebelum indeks 7, berikut contoh programnya :

[POSITIVE SLICING]

```
1  nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
2
3  print(nums[0:4])
4  print(nums[:5])
5  print(nums[5:])
6  print(nums[:])
7  #print(nums)
```

```
[10, 20, 30, 40]
[10, 20, 30, 40, 50]
[60, 70, 80, 90]
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

[NEGATIVE SLICING]

Negative slicing, kebalikan dari positif slicing, bila positif slicing kita mulai dari indeks positif contoh [2:7] berarti kita akan menampilkan indeks dimulai dari indeks 2 sampai sebelum indeks 7, bila negative slicing akan melakukan slicing dimulai dari indeks negative, contoh [:-1] karena kosong berarti secara otomatis kita memesan dari indeks 0 sampai indeks sebelum -1, sebelum -1 adalah -2 maka akan di tampilkan nilai 10(indeks 0) sd nilai 80(indeks -2) bila mengacu pada gambar program di bawah, contoh programnya:

```
1 nums = [10, 20, 30, 40, 50, 60, 70, 80, 90]
2
3 print(nums[:-1])
4 print(nums[-5:])
5 print(nums[1:-2])
6 #print(nums)
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
[50, 60, 70, 80, 90]
[20, 30, 40, 50, 60, 70]
```

(3) MENGHITUNG PANJANG LIST : len()

```
1 listsaya = ["apel", "pisang", "pepaya"]
2 print(len(listsaya))
```

```
3
```

(4) MENGHITUNG KEMUNCULAN NILAI TERTENTU : count()

```
1 hitung = [1, 4, 2, 9, 7, 8, 8, 3, 1,8]
2 x = hitung.count(8)
3 print(x)
```

```
3
```

(5) MENAMBAH NILAI LIST : append() dan insert()

Fungsi append() : menambahkan nilai list dari paling belakang.

Fungsi insert() : menambahkan list berdasarkan indeks yang kita tentukan.

append()

```
1 listsaya = ["apel", "pisang", "pepaya"]
2 listsaya.append("mangga")
3 print(listsaya)
```

```
['apel', 'pisang', 'pepaya', 'mangga']
```

insert()

```
1 listsaya = ["apel", "pisang", "pepaya"]
2 listsaya.insert(0, "mangga")
3 print(listsaya)
```

```
['mangga', 'apel', 'pisang', 'pepaya']
```

(6) HAPUS LIST : remove(), pop() dan del()

Fungsi **remove ()** : menghapus list dengan elemen list yang di tentukan.

Fungsi **pop ()** : menghapus list dengan indeks list yang di tentukan.

Fungsi **del()** : menghapus list bisa berdasarkan indeks atau menghapus list beserta isinya.

```
1 list = ["apel", "pisang", "pepaya"]
2 list.remove("pepaya")
3 print(list)
4
5 #remove elemen yang ditentukan
```

```
['apel', 'pisang']
```

```
list = ["apel", "pisang", "pepaya"]
list.pop(0)
print(list)

#remove indeks yang ditentukan
```

```
['pisang', 'pepaya']
```

```
list = ["apel", "pisang", "pepaya"]
del list[2]
print(list)
```

```
['apel', 'pisang']
```

```
listsaya = ["apel", "pisang", "pepaya"]
del listsaya
#print(listsaya)
```

(7) MENYAMBUNG LIST : extends()

Fungsi extends() : dapat digunakan untuk menyambung beberapa list menjadi satu.

```
list1 = [1,2,3,4,5]
list2 = ["apel", "pisang", "pepaya"]
list1.extend(list2)
print(list1)
```

```
[1, 2, 3, 4, 5, 'apel', 'pisang', 'pepaya']
```

(8) MEMBALIK URUTAN LIST SORTING : reverse() dan sort()

Fungsi reverse() : digunakan untuk membalik isi list.

Fungsi sort() : digunakan untuk mengurutkan nilai list.

Contoh program dengan fungsi reverse() :

```
1  list1 = [2,1,5,8,6,7,4]
2  #list1 = ['anak','cantik','baik']
3  list1.reverse()
4  print(list1)
```

```
[4, 7, 6, 8, 5, 1, 2]
```

Contoh program dengan fungsi sort() :

```
1  list1 = [2,1,5,8,6,7,4]
2  #list1 = ['anak','cantik','baik']
3  list1.sort()
4  print(list1)
```

```
[1, 2, 4, 5, 6, 7, 8]
```

(9) MENCARI NILAI MAX, MIN, SUM : max(), min() dan sum()

max() : untuk mencari nilai maksimum nilai list.

min() : untuk mencari nilai minimum nilai list.

sum() : untuk menjumlah nilai list

Contoh program dengan fungsi max() :

```
1  list1 = [2,1,5,8,6,7,4]
2  print(max(list1))
```

```
8
```

Contoh program dengan fungsi min() :

```
1 list1 = [2,1,5,8,6,7,4]
2 print(min(list1))
```

```
1
```

Contoh program dengan fungsi sum() :

```
1 list1 = [2,1,5,8,6,7,4]
2 print(sum(list1))
```

```
33
```

TUPLE [1/7]

Tuple mirip dengan list. Bedanya, tuple bersifat immutable, sehingga anggotanya tidak bisa diubah. Kemudian muncul sebuah pertanyaan 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.

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 :

```
1  my_tuple = (1,2,3,4,5)
2  print(my_tuple)
3
4  my_tuple = (1, 3.5, "Hello")
5  print(my_tuple)
```

```
(1, 2, 3, 4, 5)
(1, 3.5, 'Hello')
```

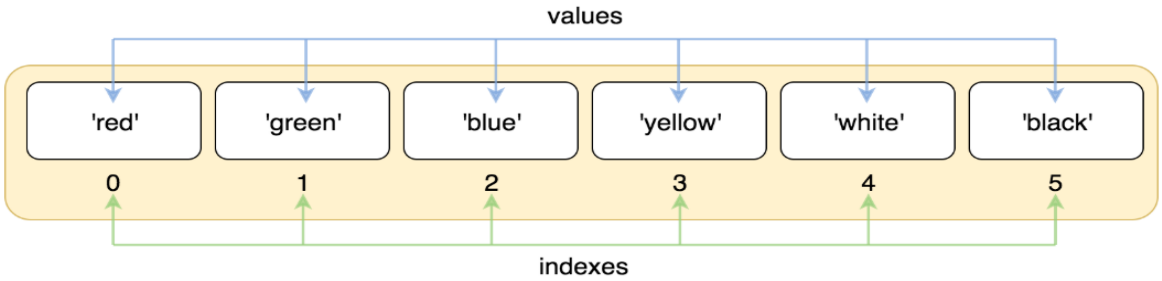
(2) INDEXING [1/2]

Sama seperti list, tuple juga dapat melakukan indexing, baik itu untuk indeks positif dan indeks negatif.

[POSITIF INDEX]

positif index adalah bagaimana kita mengakses sebuah tuple dengan memulai dengan index positif, index positif di mulai dari index 0, berikut contoh gambar index dan value pad tuple :

Perhatikan Gambar ?



Berikut contoh program akses index tuple dengan Bahasa pemograman python:

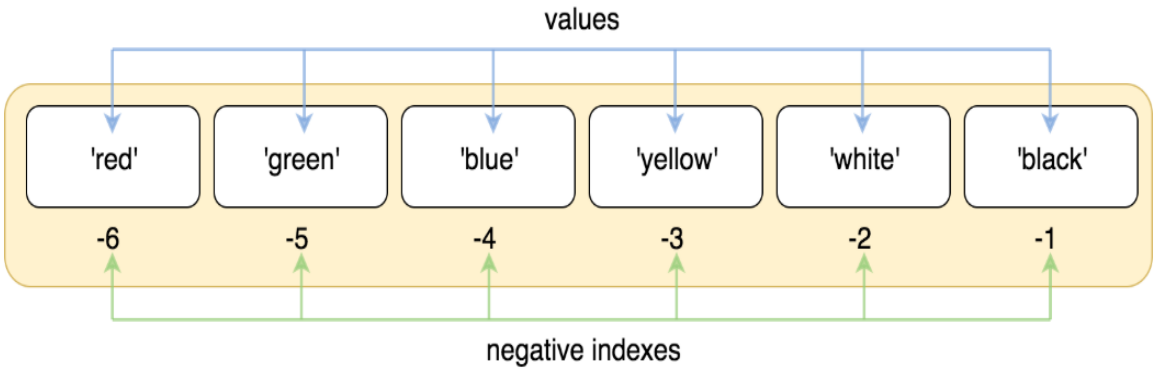
```
1 tuplesaya=('red', 'green', 'blue', 'yellow', 'white', 'black')
2 print(tuplesaya[0])
3 print(tuplesaya[1])
4 print(tuplesaya[5])
```

```
red
green
black
```

[NEGATIF INDEX]

negatif index adalah kebalikan index positif bagaimana kita mengakses sebuah tuple dengan dengan mengambil index paling akhir dari nilai tuple, index negatif di mulai dari index -1, berikut contoh gambar index negatif dan value pada tuple :

Perhatikan Gambar ?



Berikut contoh program akses tuple index negatif dengan Bahasa pemograman python**:**

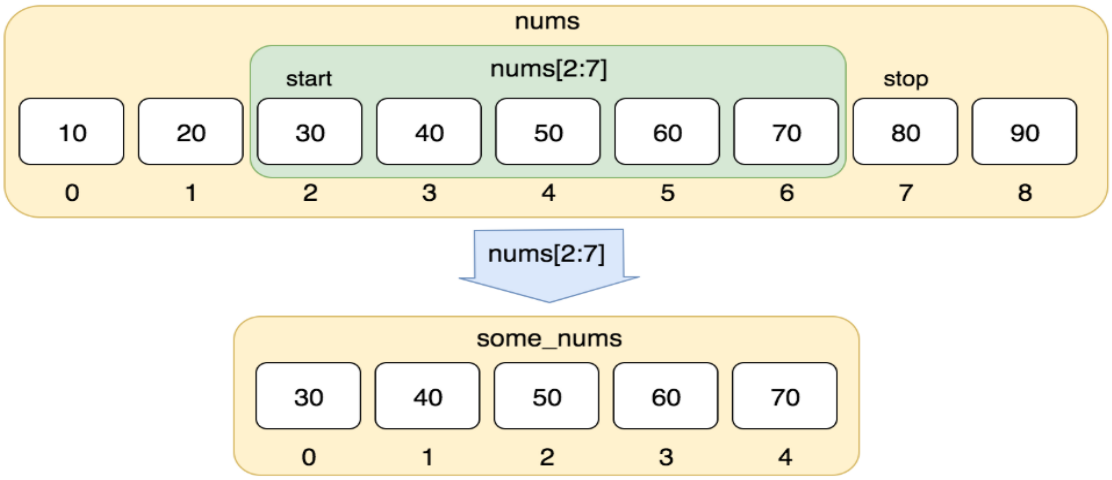
```
1 tuple_saya=('red', 'green', 'blue', 'yellow', 'white', 'black')
2 print(tuple_saya[-1])
3 print(tuple_saya[-2])
4 print(tuple_saya[-6])
```

```
black
white
red
```

(3) SLICING A PYTHON TUPLE [1/2]

Slicing digunakan untuk mengiris sebuah tuple, yang digunakan untuk menampilkan isi tuple berdasarkan keinginan kita, salah satunya bila kita ingin mengambil data, Kita bisa mengakses anggota tuple dari range tertentu dengan menggunakan operator slicing titik dua (:).Slicing akan lebih mudah bila kita memahami indeks dengan baik. Perhatikan program berikut:

Perhatikan Gambar ?



Dari gambar di atas [2:7] berarti kita akan menampilkan indeks dimulai dari indeks 2 sampai sebelum indeks 7, berikut contoh programnya :

[POSITIVE SLICING]

```
1  nums = (10, 20, 30, 40, 50, 60, 70, 80, 90)
2
3  print(nums[0:4])
4  print(nums[:5])
5  print(nums[5:])
6  print(nums[:])
7  print(nums)
```

```
(10, 20, 30, 40)
(10, 20, 30, 40, 50)
(60, 70, 80, 90)
(10, 20, 30, 40, 50, 60, 70, 80, 90)
(10, 20, 30, 40, 50, 60, 70, 80, 90)
```

[NEGATIVE SLICING]

Negative slicing, kebalikan dari positif slicing, bila positif slicing kita mulai dari indeks positif contoh [2:7] berarti kita akan menampilkan indeks dimulai dari indeks 2 sampai sebelum indeks 7, bila negative slicing akan melakukan slicing dimulai dari indeks negative, contoh [:-1] karena kosong berarti secara otomatis kita memesan dari indeks 0 sampai indeks sebelum -1, sebelum -1 adalah -2 maka akan di tampilkan nilai 10(indeks 0) sd nilai 80(indeks -2) bila mengacu pada gambar program di bawah, contoh programnya:

```
1 nums = (10, 20, 30, 40, 50, 60, 70, 80, 90)
2
3 print(nums[0:4])
4 print(nums[:-1])
5 print(nums[-5:])
6 print(nums[1:-2])
7 #print(nums)
```

```
(10, 20, 30, 40)
(10, 20, 30, 40, 50, 60, 70, 80)
(50, 60, 70, 80, 90)
(20, 30, 40, 50, 60, 70)
```

(4) MENGHITUNG PANJANG TUPLE : len()

```
1 tuple_saya = ["apel", "pisang", "pepaya"]
2 print(len(tuple_saya))
```

```
3
```

(5) MENGHITUNG KEMUNCULAN NILAI TERTENTU : count()

fungsi count() digunakan untuk menemukan berapa kali jumlah kemunculan data pada suatu tuple yang kita punya.

berikut contoh program dengan fungsi count():

```
hitung = (1, 4, 2, 9, 7, 8, 8, 3, 1,8)
x = hitung.count(1)
print(x)
```

```
2
```

(6) MENAMBAH DAN MERUBAH NILAI TUPLE

Setelah tuple dibuat, Anda tidak dapat menambahkan item atau merubah item, kecuali anda memiliki data collection lain di dalam tupel, contohnya list, maka anda dapat merubahnya :

Tuple tidak mau di rubah :

```
tuple_saya = ("apel", "pisang", "pepaya")
tuple_saya[3] = "anggur"
print(list_saya)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-2d26e018dae2> in <module>
      1 tuple_saya = ("apel", "pisang", "pepaya")
----> 2 tuple_saya[3] = "anggur"
      3 print(list_saya)

TypeError: 'tuple' object does not support item assignment
```

Program data collection lain di dalam tuple, contohnya list, anda dapat merubahnya :

```
my_tuple = (2, 3, 4,[5, 6])
my_tuple[3][0] = 7
print(my_tuple)
```

```
(2, 3, 4, [7, 6])
```

(7) MENGHAPUS NILAI TUPLE

Tupel yang tidak berubah dan tidak bisa di hapus, namun Anda dapat menghapus tupel sepenuhnya:

```
1 tuple_saya = ("apel", "pisang", "pepaya")
2 del tuple_saya
3 print(tuple_saya)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-9d3d0e5164f3> in <module>
      1 tuple_saya = ("apel", "pisang", "pepaya")
      2 del tuple_saya
----> 3 print(tuple_saya)

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

Program tersebut error karena tuple telah kita hapus sepenuhnya dengan fungsi *del()*

(8) MENGGABUNGKAN DUA TUPLE NILAI TUPLE

Kita dapat menggabungkan dua tuple dengan operator + :

```
1 tuple1 = ("a", "b" , "c")
2 tuple2 = (1, 2, 3)
3
4 tuple3 = tuple1 + tuple2
5 print(tuple3)
```

```
('a', 'b', 'c', 1, 2, 3)
```

SET [1/7]

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.

Kegunaan :

Set di tandai dengan tanda { }, set tidak memiliki sebuah indeks, set berbeda dengan list dan tuple yang memiliki indeks. Set bisa digunakan untuk melakukan operasi himpunan matematika seperti gabungan, irisan, selisih, dan lain – lain.

Contoh program jika set tidak mengandung data duplikasi :

```
1 my_set1 = {1,2,3}
2 my_set2 = {1,2,2,3,3,3}
3 print(my_set1)
4 print(my_set2)
```

```
{1, 2, 3}
{1, 2, 3}
```

(2) MENGUBAH dan menambah ITEM SET : add() dan update()

Fungsi add() : Untuk menambahkan satu item untuk satu set.

Fungsi update() : Untuk menambahkan lebih dari satu item atau segerombolan item kedalam set.

Contoh program fungsi add() :

```
1 set_saya = {"apel", "pisang", "pepaya"}
2 set_saya.add("nanas")
3 print(set_saya)
```

```
{'apel', 'pepaya', 'nanas', 'pisang'}
```

Contoh program fungsi update() :

```
1 set_saya = {"apel", "pisang", "pepaya"}
2 set_saya.update(['a', 'b', 'c', 1, 2, 3])
3 print(set_saya)
```

```
{1, 2, 3, 'a', 'c', 'pisang', 'apel', 'pepaya', 'b'}
```

(3) MENGHITUNG PANJANG SET : len()

Fungsi len() : digunakan untuk mengetahui panjang set.

Berikut contoh programnya :

```
1 set_saya = {"apel", "pisang", "pepaya"}
2 print(len(set_saya))
```

```
3
```

(4) MENGHAPUS ANGGOTA SET : discard(), remove(), pop() dan clear()

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.

Fungsi remove() :akan memunculkan eror bila nilai set yang akan di hapus tidak ada di dalam set:

Contoh program discard() :

```
1 set_saya = {1, 2, 3, 4, 5}
2 print(set_saya)
3 set_saya.discard(6)
4 print(set_saya)
```

```
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5}
```

Contoh program dengan fungsi remove():

```
1 set_saya = {1, 2, 3, 4, 5}
2 set_saya.remove(6)
3 print(set_saya)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-18-3eadbb55423a> in <module>
      1 set_saya = {1, 2, 3, 4, 5}
----> 2 set_saya.remove(6)
      3 print(set_saya)

KeyError: 6
```

Program tersebut eror karena 6 tidak ada didalam 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).

Contoh program dengan fungsi pop():

```
1 set_saya = {1, 2, 3, 4, 5}
2 set_saya.pop()
3 print(set_saya)
```

```
{2, 3, 4, 5}
```

Contoh program dengan fungsi clear():

```
1 set_saya = {1, 2, 3, 4, 5}
2 set_saya.clear()
3 print(set_saya)
```

```
set()
```

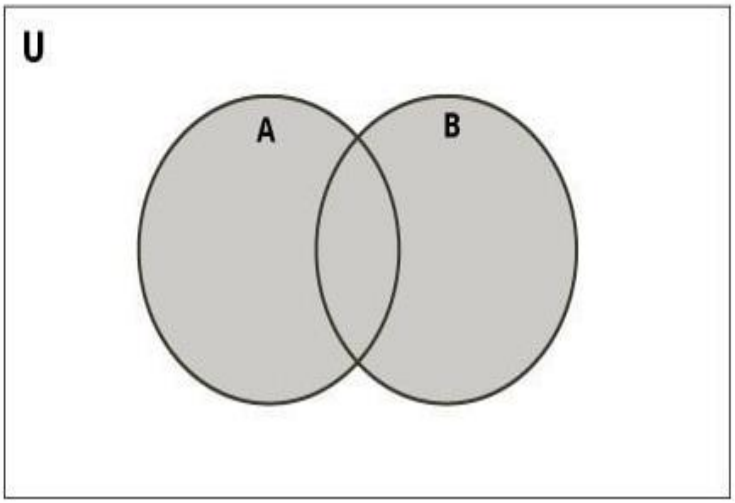
Fungsi clear() dia menghapus semua isi data di dalam set yang kita punya, dengan tetap menyisakan nilai set kosong.

(5)OPERASI SET PADA PYTHON[1/4]

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

HIMPUNAN MATEMATIKA [UNION]

Perhatikan Gambar ?



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().

Contoh program dengan operator (|) :

```
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  print(A | B)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

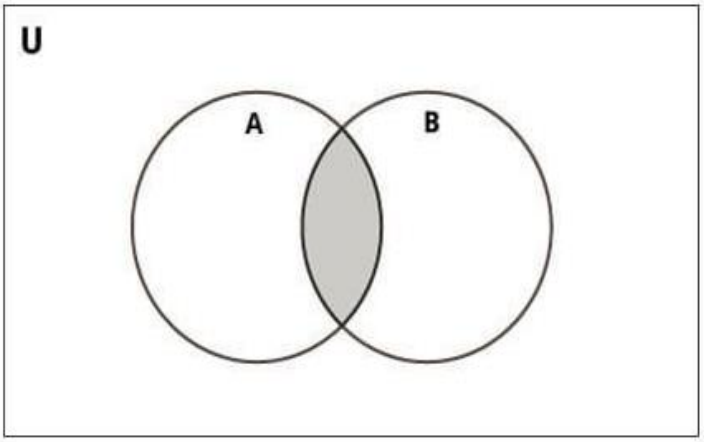
Contoh program dengan fungsi union() :

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3  # Menggunakan fungsi union()
4  print(A.union(B))
5
6  print(B.union(A))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
```

OPERASI IRISAN [INTERSECTION]

Perhatikan Gambar ?



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().

Contoh program menggunakan operator jangkar (&) :

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3
4  # Irisan menggunakan operator &
5  print(A & B)
```

```
{4, 5}
```

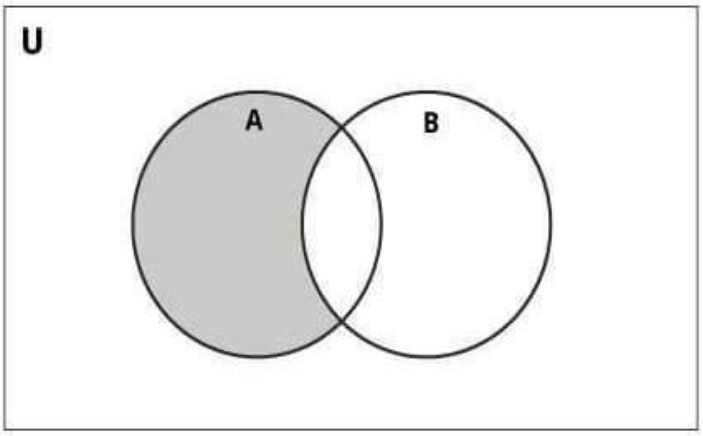
Contoh program dengan fungsi intersection():

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3  # Menggunakan fungsi intersection()
4  print(A.intersection(B))
5
6  print(B.intersection(A))
```

```
{4, 5}
{4, 5}
```

OPERASI SELISIH [DIFFERENCE]

Perhatikan Gambar ?



Selisih (intersection) 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().

Contoh program dengan operator kurung (-) :

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# Menggunakan operator - pada A
print(A - B)
print(B - A)
```

```
{1, 2, 3}
{8, 6, 7}
```

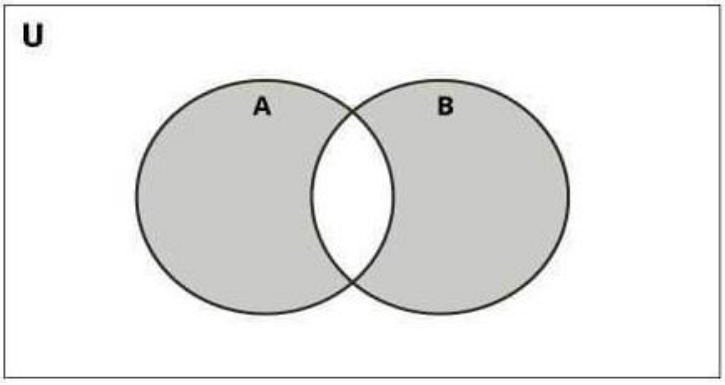
Contoh program dengan fungsi difference() :

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

A.difference(B)
B.difference(A)
```

```
{6, 7, 8}
```

OPERASI KOMPLEMEN [SYMETRIC DIFFERENCE]

Perhatikan Gambar ?



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().

Contoh program dengan operator ^:

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# Menggunakan operator ^ pada A
print(A ^ B)
print(B ^ A)
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

Contoh program dengan fungsi difference() :

```
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print(A.symmetric_difference(B))
print(B.symmetric_difference(A))
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

(6) Metode (Fungsi) Set

Sebagian fungsi sudah kita bahas di atas. Untuk lebih lengkapnya anda dapat melihat metode di bawah ini :

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

DICTIONARY

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

(1) MEMBUAT DICTIONARY

```
1  dict_saya = {
2    "nama": "Ari",
3    "Tinggal": "Jember",
4    "year": 1997
5  }
6  print(dict_saya)
7
8  #Dictionary berisi nilai List
9  dict_saya = {'warna': 'merah',
10              1: [2,3,5]}
11 print(dict_saya[1][-1])
```

```
{'nama': 'Ari', 'Tinggal': 'Jember', 'year': 1997}
5
```

Dari gambar di atas dapat kita perhatikan, nama, Tinggal, year adalah kunci atau keys dalam dictionary, Ari, Jember, 1997 adalah values dari dictionary.

Untuk dictionary kedua kita dapat melihat bahwa dictionary dapat mengandung nilai data collection lain. *print(dict_saya[1][-1])* adalah tanda bahwa kita mengakses keys 1, karena keys 1 adalah list maka kita dapat melakukan indexing, bila di mulai dari -1 maka indeks paling belakang yang akan di tampilkan.

(2) AKSES DICTIONARY : get()

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 hendak diakses tidak ada di dalam dictionary.

Contoh program dengan akses keys/kunci :

```
1  dict_saya = {'Nama': 'Ari',
2               'Usia': 25,
3               'Status': 'Mahasiswa'}
4
5  print ("dict_saya['Nama']: ", dict_saya['Nama'])
6  print ("dict_saya['Status']: ", dict_saya['S'])
```

```
dict_saya['Nama']:  Ari

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-43-6cd0fc2c3ac9> in <module>
      4
      5 print ("dict_saya['Nama']: ", dict_saya['Nama'])
----> 6 print ("dict_saya['Status']: ", dict_saya['S'])

KeyError: 'S'
```

Bila kita perhatikan pada gambar di atas, eror karena ‘S’ bukan merupakan keys/kunci dalam dictionary,

Contoh program dengan fungsi get() :

```
1  dict_saya = {'Nama': 'Ari',
2               'Usia': 25,
3               'Status': 'Mahasiswa'}
4
5  print("dict_saya['Usia']: ",dict_saya.get('S'))
```

```
dict_saya['Usia']:  None
```

Dari program di atas ‘S’ bukan merupakan keys/kunci di dalam dictionary, karena menggunakan fungsi get(), maka program tersebut tidak muncul notif eror, namun hanya muncul nilai none.

(3) MERUBAH 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. Contoh program merubah nilai/values dictionary dengan mengakses keys/kunci dalam dictionary:

```
1 dict_saya = {'Nama': 'Ari',
2               'Usia': 25,
3               'Status': 'Mahasiswa'}
4 dict_saya["Usia"] = 27
5 print(dict_saya)
```

```
{'Nama': 'Ari', 'Usia': 27, 'Status': 'Mahasiswa'}
```

(4) HAPUS DICTIONARY : pop(), popitem(), clear() dan del()

Ada beberapa fungsi untuk menghapus dictionary :

fungsi pop() : Metode menghilangkan item dengan nama kunci yang ditentukan.

fungsi popitem(): Metode menghilangkan item secara acak.

fungsi clear() : Metode mengosongkan kamus:

del(): untuk menghapus anggota tertentu atau menghapus dictionary itu sendiri.

Contoh program dengan fungsi pop():

```
1  dict_saya = {'Nama': 'Ari',
2               'Usia': 25,
3               'Status': 'Mahasiswa'}
4  dict_saya.pop("Nama")
5  print(dict_saya)
```

```
{'Usia': 25, 'Status': 'Mahasiswa'}
```

Contoh program dengan fungsi popitem() :

```
1  dict_saya = {'Nama': 'Ari',
2               'Usia': 25,
3               'Status': 'Mahasiswa'}
4  dict_saya.popitem()
5  print(dict_saya)
```

```
{'Nama': 'Ari', 'Usia': 25}
```

Contoh program dengan fungsi clear():

```
dict_saya = {'Nama': 'Ari',
             'Usia': 25,
             'Status': 'Mahasiswa'}
dict_saya.clear()
print(dict_saya)
```

```
{}
```

Contoh program dengan fungsi del():

```
dict_saya = {'Nama': 'Ari',
             'Usia': 25,
             'Status': 'Mahasiswa'}
del dict_saya
print(dict_saya)
```

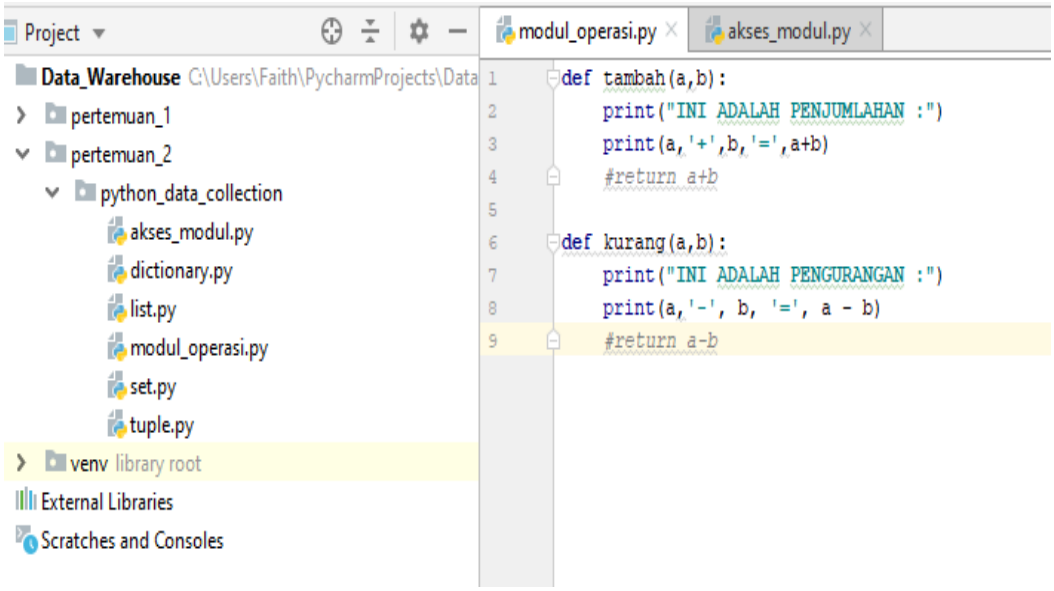
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-5107a7ed1e7c> in <module>
      3              'Status': 'Mahasiswa'}
      4 del dict_saya
----> 5 print(dict_saya)

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

Program di atas eror karena dictionary telah di hapus dengan fungsi del().

MODUL

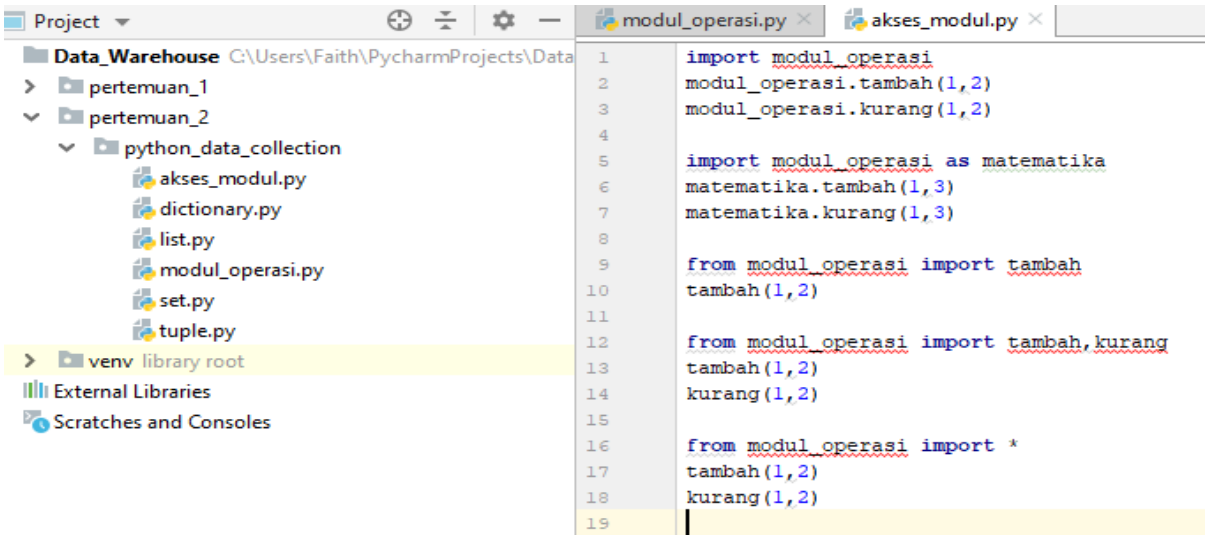
Modul juga bisa di artikan sebagai kumpulan dari beberapa fungsi. Modul digunakan untuk memecah sebuah program besar menjadi file – file yang lebih kecil agar lebih mudah di*manage* dan diorganisir. Modul membuat kode bersifat *reusable*, artinya satu modul bisa dipakai berulang dimana saja diperlukan.

Contoh membuat sebuah modul :



Dari gambar di atas kita membuat sebuah file python yang akan di jadikan sebuah modul, disana kita membuat sebuah fungsi, namun kita ingin memproses fungsi tersebut pada file python baru yang yang telah kita siapkan.

Mengakses modul pada file lain :



Dari program di atas kita dapat mengakses modul yang telah kita buat dengan nama modul_operasi, fungsi yang di tampilkan pada gambar scripts di atas adalah contoh cara memanggil sebuah modul agar dapat di akses di file python baru kita.

LATIHAN

LATIHAN 1

- buatlah sebuah TUPLE (integer) dengan indeks sebanyak 10.
- ubahlah TUPLE tersebut di bagian indeks ke 10 menjadi (22) dengan convert TUPLE to LIST.
- setelah di rubah convert LIST to TUPLE.
- lakukan looping daftar TUPLE dengan for.
- tampilkan TUPLE

LATIHAN 2

- buatlah sebuah SET yang berisi 10 teman satu kelas.
- ubahlah SET menjadi LIST dengan fungsi list()

LATIHAN 3

- buatlah sebuah function yang mengandung perulangan (for) dan fungsi append(), function tersebut bertugas mengkonfersi keys() dictionary kedalam list.
- kemudian buat dictionary di luar function dengan nilai keys() 1 sd 10 dengan value() nama teman satu kelas.
- ketika function di panggil maka secara otomatis list akan terbuat dari keys() dictionary.
- tampung nilai list pada variabel dengan nama 'a'.
- tampilkan variabel yang terisi list dengan fungsi type().