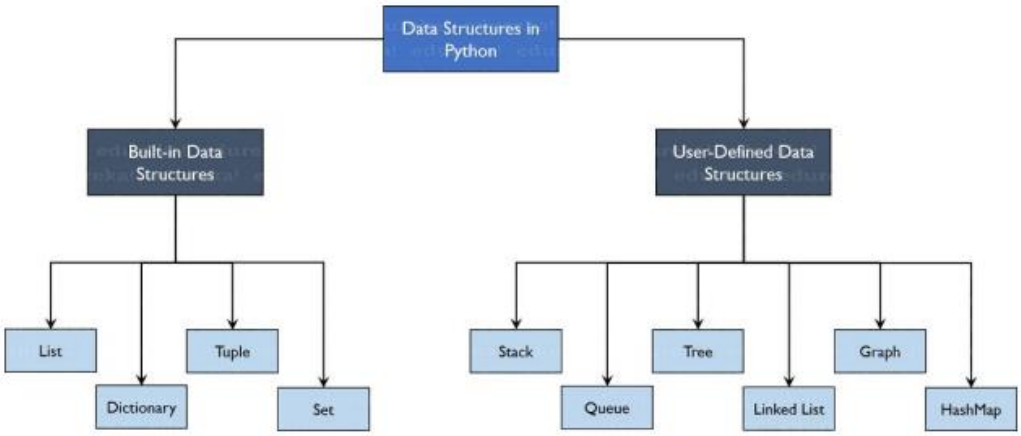
MODUL 2

(MINGGU KEUDA)

Praktikum 3

Judul : Struktur Data dalam Python



Gambar 3. 1 Strukur data dalam Python

Struktur data adalah cara yang digunakan untuk mengorganisir dan menyimpan data dalam suatu program. Python menyediakan beberapa jenis struktur data yang dapat digunakan untuk berbagai keperluan, seperti list, tuple, set, dictionary, dan lain-lain. Berikut adalah penjelasan tentang struktur data dalam Python:

1. **LIST**

   List adalah struktur data yang paling umum digunakan di Python. List adalah kumpulan nilai yang terurut dan dapat diubah-ubah. Setiap nilai dalam list dapat diakses menggunakan indeksnya, yang dimulai dari nol. List dibuat menggunakan ***tanda kurung siku***:

   Contoh penggunaan list:

```
list.py
1   thislist = ["apple", "banana", "cherry"]
2   print(thislist)
```

   Output :

```
PS D:\praktikum\data mining\python> python .\list.py
['apple', 'banana', 'cherry']
```

   Untuk menentukan berapa banyak item yang dimiliki daftar, gunakan fungsi len():

```
1   thislist = ["apple", "banana", "cherry"]
2   print(len(thislist))
```

Output :

```
PS D:\praktikum\data mining\python> python .\list.py
3
```

Item list dapat berupa tipe data apa pun, baik itu numerik, string, boolean ataupun lainnya.

```
1   list1 = ["apple", "banana", "cherry"]
2   list2 = [1, 5, 7, 9, 3]
3   list3 = [True, False, False]
4   print(list1)
5   print(list2)
6   print(list3)
```

Output

```
PS D:\praktikum\data mining\python> python .\list.py
['apple', 'banana', 'cherry']
[1, 5, 7, 9, 3]
[True, False, False]
```

Dalam satu variable list dapat berisi tipe data yang berbeda:

```
1   list1 = ["abc", 34, True, 40, "male"]
2   print(list1)
```

Output :

```
PS D:\praktikum\data mining\python> python .\list.py
['abc', 34, True, 40, 'male']
```

Item list diindeks dengan aturan item pertama memiliki index [0], item kedua memiliki indeks [1],

```
1   thislist = ["apple", "banana", "cherry"]
2   print(thislist[1])
```

Output

```
PS D:\praktikum\data mining\python> python .\list.py
banana
```

Pengindeksan negatif berarti mulai dari akhir. -1 mengacu pada item terakhir, -2 mengacu pada item terakhir kedua dll.

```
1   thislist = ["apple", "banana", "cherry"]
2   print(thislist[-1])
```

Output

```
PS D:\praktikum\data mining\python> python .\list.py
cherry
```

Anda dapat menentukan rentang indeks dengan menentukan di mana harus memulai dan di mana harus mengakhiri rentang. Saat menentukan rentang, nilai yang dikembalikan (hasilnya) akan menjadi daftar baru dengan item yang ditentukan.

```
1   thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
2   print(thislist[2:5])
```

Output :

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['cherry', 'orange', 'kiwi']
```

Untuk kasus rentang, dengan mengabaikan nilai awal, rentang akan dimulai pada item pertama:

```
1   thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
2   print(thislist[:4])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'orange']
```

Masih pada kasus rentang, dengan meninggalkan nilai akhir, rentang nilai akhir akan diset ke akhir daftar:

```
1   thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
2   print(thislist[4:])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['kiwi', 'melon', 'mango']
```

Untuk menentukan apakah item tertentu ada dalam list, gunakan **in** :

```
1   thislist = ["apple", "banana", "cherry"]
2   if "apple" in thislist:
3       print("Ya, 'apple' ada di dalam list")
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Ya, 'apple' ada di dalam list
```

Latihan 3.1

1. Buat list yang isinyanya adalah nama depan dan nama belakang masing-masing praktikan, kemudian hitung huruf vocal yang ada di nama masing-masing !

MENGUBAH NILAI LIST

Untuk mengubah nilai list, tinggal tentukan saja pada indeks yang diinginkan. Perhatikan contoh berikut:

```
1   thislist = ["apple", "banana", "cherry"]
2   thislist[1] = "blackcurrant"
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'blackcurrant', 'cherry']
```

Perhatikan, pada indeks 1, yang awalnya adalah **banana**, diganti nilainya dengan **blackcurrent**.

APPEND / MENAMBAHKAN ITEM

Untuk menambahkan item ke akhir list, kita dapat gunakan metode append() :

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'orange']
```

Untuk menyisipkan item baru di tengah-tengah / tidak di akahir, kita dapat menggunakan metode insert(). Metode insert() ini menyisipkan item pada indeks yang ditentukan:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'orange']
```

Perhatikan pada kode thislist.insert(2, “watermelon”). Kode ini artinya adalah kita akan menyisipkan item baru “watermelon” pada tepat indeks 2. Sehingga indeks 2 (cherry) yang awalnya akan mundur menjadi indeks ke-3. Ingat kembali indeks pertama dinamakan indeks ke-0.
Untuk menambahkan elemen dari list lain ke list saat ini, kita dapat gunakan metode extend().

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

Perhatikan pada list thislist telah ditambahkan konten dari list tropical.

MENGHAPUS ITEM

Untuk menghapus item di dalam list, kita dapat menggunakan metode remove() terhadap item dengan nilai yang sama sepertiyang diinginkan.

```
1   thislist = ["apple", "banana", "cherry"]
2   thislist.remove("banana")
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'cherry']
```

Untuk menghapus item berdasarkan nomor indeks, kita dapat menggunakan metode pop().

```
1   thislist = ["apple", "banana", "cherry"]
2   thislist.pop(1)
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'cherry']
```

Perhatikan pada kode thislist.pop(1). Kode ini artinya adalah kita akan menghapus item indeks ke-1. Lihat pada hasilnya, indeks ke-1 yaitu banana telah terhapus.

Sedang untuk menghapus isi dari list atau mengosongkannya dapat menggunakan metode clear()

```
1   thislist = ["apple", "banana", "cherry"]
2   thislist.clear()
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
[]
```

Perhatikan, dengan metode clear(), maka list akan menjadi kosong total, namun varibel list tersebut tetap ada.
Untuk menghapus variabel list, dari memory, kita dapat menggunakan metode del().

FOR LOOP PADA LIST

Anda dapat membuat perintah berulang untuk setiap item pada list. Untuk agar lebih memahami perhatikan cotoh berikut:

```
1   thislist = ["apple", "banana", "cherry"]
2   for x in thislist:
3       print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
apple
banana
cherry
```

Perhatikan pada kode for x in thislist:, pada kode ini memberitahu mesin bahwa kita ingin melakukan perulangan perintah untuk setiap item yang ada di list thislist dan disimbolkan dengan x. Kemudian pada bagian menjorok kedepan ada kode print(x), kode ini artinya kita memberitahu mesin untuk memprint-out ke layar yaitu varible x. Dimana variable x adalah item pada list thislist.

Perlu dipahami, jika di bahasa pemrograman lain untuk menunjukkan isi dari kumpulan statement adalah dengan kurung kurawa { ….. }, jika di phyton adalah dengan indent (kode yang agak menjorok ke depan dari parent-nya)

SORTING / PENGURUTAN LIST

Untuk mengurutkan berdasarkan alfanumerik list, dapat menggunakan metode sort():

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Perhatikan contoh lagi dibawah ini untuk mengurutkan angka.

```
1   thislist = [100, 50, 65, 82, 23]
2   thislist.sort()
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
[23, 50, 65, 82, 100]
```

Secara default, metode **sort**() akan mengurukan secara naik yaitu dari yang terkecil ke terbesar. Untuk mengurutkan sebaliknya / menurun kita dapat menggunakan metode **sort**(**reverse=True**).

```
1   thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
2   thislist.sort(reverse = True)
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

Phyton juga mempunyai fungsi reverse() untuk membalikkan urutan dari list. Artinya yang paling belakang akan menjadi yang paling depan dan sebaliknya. Perhatikan contoh berikut:

```
1   thislist = ["banana", "Orange", "Kiwi", "cherry"]
2   thislist.reverse()
3   print(thislist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['cherry', 'Kiwi', 'Orange', 'banana']
```

MENYALIN LIST (COPY)

Untuk mengopy variable list ke variable lain, di phyton tidak bisa sesederhana dengan mengetik *list2 = list1*, karena: *list2* hanya akan menjadi referensi ke *list1*, dan perubahan yang dibuat *list1* secara otomatis juga akan dibuat di *list2*. Mungkin ini sedikit membingungkan, tapi nanti anda akan terbiasa, ini seperti pointer jika di bahasa pemrograman lain.

Ada cara untuk membuat salinan, salah satu caranya adalah dengan menggunakan metode bawaan copy().

```
1   thislist = ["apple", "banana", "cherry"]
2   mylist = thislist.copy()
3   print(mylist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry']
```

Cara lain untuk membuat salinan adalah dengan menggunakan metode bawaan list().

```
1   thislist = ["apple", "banana", "cherry"]
2   mylist = list(thislist)
3   print(mylist)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['apple', 'banana', 'cherry']
```

Latihan 3.2

1. Buat sebuah list yang isinya adalah nama masing-maisng praktikan, kemudian balik nama tersebut menggunakan fungsi yang sudah didefiniskan sendiri.
   a. Contoh : “Susi Susanti” menjadi “Susanti Susi”

MENGGABUNGKAN BEBERAPA LIST (JOIN)

Ada beberapa cara untuk menggabungkan dua atau lebih daftar dengan Python. Salah satu cara termudah adalah dengan menggunakan operator +.

```
1   list1 = ["a", "b", "c"]
2   list2 = [1, 2, 3]
3   list3 = list1 + list2
4   print(list3)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['a', 'b', 'c', 1, 2, 3]
```

Kita juga dapat menggunakan metode **extend**(), yang tujuannya adalah untuk menambahkan elemen dari satu daftar ke daftar lain:

```
1   list1 = ["a", "b" , "c"]
2   list2 = [1, 2, 3]
3   list1.extend(list2)
4   print(list1)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
['a', 'b', 'c', 1, 2, 3]
```

2. **Tuple**

Tuple digunakan untuk menyimpan beberapa item dalam satu variabel. Berbeda dengan list yang bisa kita ubah-ubah, Tuple adalah kumpulan yang dipesan dan *tidak dapat diubah.* Ingat kembali bab sebelumnya**,**

jika list ditulis dengan kurung siku, tuple ditulis dengan kurung bulat.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
('apple', 'banana', 'cherry')
```

Sama seperti list, Item tuple diindeks, item pertama memiliki index [0], item kedua memiliki indeks [1], dst

Tuple tidak dapat diubah, artinya kita tidak dapat mengubah, menambah, atau menghapus item setelah tuple dibuat. Urutannya pun tidak dapat diubah, diurutkan, dan lainnya.
Untuk menentukan berapa banyak item yang dimiliki sebuah tuple, gunakan fungsi len():

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
3
```

Untuk membuat Tuple dengan hanya satu item, Anda harus menambahkan koma setelah item, jika tidak Python tidak akan mengenalinya sebagai Tuple.

```
thistuple = ("apple",)
print(type(thistuple))
#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
<class 'tuple'>
<class 'str'>
```

Item tuple dapat berupa tipe data apa pun baik itu strig, numerik, boolean, dan lain sebagainya. Dalam satu tuple juga bisa mengizinkan tipe data yang berbeda-beda.

Kita dapat mengakses/memanggil item Tuple dengan mengacu pada nomor indeks, di dalam tanda kurung siku, sama seperti list:

```
1   thistuple = ("apple", "banana", "cherry")
2   print(thistuple[1])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
banana
```

Setelah tuple dibuat, Anda tidak dapat mengubah nilainya. Tuple tidak dapat diubah , atau tidak dapat diubah seperti yang juga disebut. Tapi ada solusi. Anda dapat mengonversi Tuple menjadi list, mengubah list, dan mengonversi list kembali menjadi Tuple.

```
1   x = ("apple", "banana", "cherry")
2   y = list(x)
3   y[1] = "kiwi"
4   x = tuple(y)
5   print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
('apple', 'kiwi', 'cherry')
```

Anda diperbolehkan untuk menambahkan tupel ke tupel, jadi jika Anda ingin menambahkan satu item, (atau banyak), buat tupel baru dengan item tersebut, dan tambahkan ke tupel yang ada:

Sama seperti list, anda dapat mengulang item Tuple dengan menggunakan forloop **in.**

Latihan 3.3

1. Buat tuple dengan isi adalah nim masing-masing praktikan, kemudian tambahkan tanggal lahir masing-masing dalam Tuple tersebut !

3. Set

Set digunakan untuk menyimpan beberapa item dalam satu variabel. Berbeda dengan list dan tuple, **set tidak diindeks yang artinya tidak bisa diurutkan, tidak bisa diubah**. Kita hanya bisa menghapus dan menambahkan saja. **Jika list dituliskan dengan kurung siku, tuple kurung bulat, set dituliskan dengan kurung kurawa.**

Perhatikan contoh berikut

```
1   thisset = {"apple", "banana", "cherry"}
2   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple', 'cherry', 'banana'}
```

Set tidak boleh memiliki dua item dengan nilai yang sama. Perhatikan contoh berikut:

```
1   thisset = {"apple", "banana", "cherry", "apple"}
2   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple', 'cherry', 'banana'}
```

Perhatikan contoh diatas, oleh karena pada item terakhir (apple) sama dengan item yang pertama, maka secara otomatis python akan menghapus item yang lebih awal.

Untuk menentukan berapa banyak item yang dimiliki satu set, gunakan fungsi len(). Item pada set dapat berupa tipe data apa pun, sama seperti list ataupun tuple.

Dan satu set dapat berisi tipe data yang berbeda.

```
1   set1 = {"abc", 34, True, 40, "male"}
2   print(set1)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{True, 34, 40, 'male', 'abc'}
```

Jangan kaget jika saat dirun, urutan yang keluar berbeda dari saat awal kita deklarasikan. Karena memang set tidak menyimpan urutan item.

Kita tidak dapat mengakses item dalam set dengan mengacu pada indeks atau kunci. Tetapi Kita dapat mengulang item yang ditetapkan menggunakan for loop.

```
1   thisset = {"apple", "banana", "cherry"}
2   for x in thisset:
3       print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
apple
cherry
banana
```

Kita juga bisa menanyakan apakah nilai yang ditentukan ada dalam satu set, dengan menggunakan kata kunci **in**

```
1   thisset = {"apple", "banana", "cherry"}
2   print("banana" in thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
True
```

Untuk menambahkan satu item ke set gunakan fungsi add().

```
1   thisset = {"apple", "banana", "cherry"}
2   thisset.add("orange")
3   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple', 'cherry', 'orange', 'banana'}
```

MENGHAPUS ITEM

Untuk menghapus item dalam set dapat menggunakan remove(), discard(), atau clear(). Kita akan bahasa satu per satu.
Pertama, remove(), coba perhatikan contoh berikut:

```
1   thisset = {"apple", "banana", "cherry"}
2   thisset.remove("banana")
3   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'cherry', 'apple'}
```

Fungsi remove() akan menimbulkan error jika kata kunci yang mau kita hapus tidak ditemukan di dalam set yang terkait, perhatikan cotoh berikut:

```
1   thisset = {"apple", "banana", "cherry"}
2   thisset.remove("kiwi")
3   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Traceback (most recent call last):
  File "tes.py", line 2, in <module>
    thisset.remove("kiwi")
KeyError: 'kiwi'
```

Berbeda dengan remove(), dengan fungsi discard(), saat kata kunci yang kita tentukan untuk dihapus tidak ditemukan di set maka python tidak akan error.

Untuk menghapus keseluruhan isi dari set kita dapat menggunakan fungsi clear().

INTERAKSI DUA SET

Untuk menambahkan item dari set lain ke set saat ini, gunakan fungsi update().

```
1   thisset = {"apple", "banana", "cherry"}
2   tropical = {"pineapple", "mango", "papaya"}
3   thisset.update(tropical)
4   print(thisset)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'pineapple', 'mango', 'banana', 'papaya', 'cherry', 'apple'}
```

Untuk mendapatan nilai irisan, kita dapat menggunaka, Metode **intersection_update**().
Dengan fungsi ini, kita hanya akan menyimpan item yang ada di kedua set (irisan).

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y)
print(x)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'apple'}
```

Untuk mendapatkan nilai yang tidak ada di kedua set (berkebalikan dengan irisan), maka kita dapat menggunakan metode **symmetric_difference_update()**. Dengan fungsi ini hanya akan menyimpan elemen yang tidak ada di kedua set.

4. Dictionary
Dictionary (kamus) digunakan untuk menyimpan nilai data dalam pasangan **key:value** atau dalam bahasa Indonesia **kunci:nilai**.

Dictionary adalah kumpulan yang dipesan*, dapat diubah dan tidak memungkinkan duplikat.

Dictionary ditulis dengan tanda kurung kurawal, dan memiliki kunci dan nilai,

perhatikan contoh dibawah ini

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964 }
print(thisdict)
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Item dictionary disajikan dalam pasangan **kunci:nilai,** dan dapat dirujuk dengan menggunakan nama kuncinya, lihat contoh dibawah ini:

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
print(thisdict["brand"])
```

Output

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Ford
```

Dictionary tidak boleh memiliki dua item dengan kunci yang sama:

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964,
"year": 2020
}
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Perhatikan contoh diatas, oleh karena ada dua kata kunci yang sama yaitu year, maka yang akan dieksekusi oleh python adalah yag didefinisikan yang paling terakhir.
Untuk menentukan berapa banyak item yang dimiliki kamus, gunakan fungsi len():

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964,
"year": 2020
}
print(len(thisdict))
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
3
```

Nilai dalam item kamus dapat berupa tipe data apa pun, bahkan berupa list, tuple, atau set.

```
thisdict = {
"brand": "Ford",
"electric": False,
"year": 1964,
"colors": ["red", "white", "blue"]
}
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']}
```

Kita dapat mengakses item dictionary dengan mengacu pada nama kuncinya, di dalam tanda kurung siku

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict["model"]
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Mustang
```

Ada juga metode yang disebut get() yang akan memberikan hasil yang sama:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.get("model")
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Mustang
```

Metode keys(), akan mengembalikan daftar semua kunci dalam dictionary.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.keys()
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
dict_keys(['brand', 'model', 'year'])
```

Metode values() akan mengembalikan daftar semua nilai dalam dictionary.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.values()
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
dict_values(['Ford', 'Mustang', 1964])
```

Untuk mengembalikan pasangan kunci dan nilai untuk setiap item dari dictionary dapat menggunakan metode items(). Fungsi ini akan mengembalikan setiap item dalam dictionary, sebagai tupel dalam dictionary.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = thisdict.items()
print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

Untuk menentukan apakah kunci tertentu ada dalam dictionary, gunakan kata kunci **in:**

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

if "model" in thisdict:
    print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Yes, 'model' is one of the keys in the thisdict dictionary
```

MENGUBAH NILAI DICTIONARY

Untuk mengubah nilai suatu kata kunci pada dictionary, kita tinggal panggil saja kata kuncinya dengan kurung siku.

Perhatikan contoh berikut:

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict["year"] = 2018
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

Di contoh diatas kita mengubah kata kunci “year” yang awalnya 1964 menjadi 2018.

Untuk mengubah nilai dictionary, selain menggunakan kurung siku, kita juga bisa menggunakan fungsi update()

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
thisdict.update({"year": 2023, "model": "SUV"})
print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'SUV', 'year': 2023}
```

MENAMBAHKAN ITEM

Menambahkan item ke dictionary dapat dilakukan dengan menggunakan kunci dan indeks baru dan menetapkan nilai untuk itu. Perhatikan contoh berikut.

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964
5   }
6   thisdict["color"] = "red"
7   print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Pada contoh diatas, perhatikan kode thisdict[“color”]=“red”. Oleh karena kunci “color” belum ada di dictionary, maka python akan mengindikasikan ini sebagai penambahan item baru yaitu dengan kunci “color” dan nilai “red”. Pada kasus yang lain, apabila kata kunci yang dipanngil sudah ada di dictionary maka nilainya akan diupdate.

MENGHAPUS ITEM

Ada beberapa metode untuk menghapus item dari dictionary. Yang pertama adalah menggunakan fungsi pop().

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964
5   }
6   thisdict.pop("model")
7   print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Pada contoh diatas, kode thisdict.pop("model"), artinya adalah kita akan menghapus item dengan kata kunci "model". Cara yang lain untuk menghapus item di dictionary adalah dengan fungsi del. Perhatikan contoh berikut, kita akan coba menghapus item dengan kata kunci "model" namun dengan cara **del.**

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964
5   }
6   del thisdict["model"]
7   print(thisdict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'year': 1964}
```

Sedangkan untuk menghapus total keseulurhan isi dari dictionary kita dapat menggunakan fungsi clear().

PERULANGAN FOR LOOP PADA DICTIONARY

Sama seperti list, set, dan tuple, kita juga bisa melakukan pengulangan perintah untuk setiap item di dictionary menggunakan for loop in. Perhatikan contoh berikut.

```
1   thisdict = {
2   "brand": "Ford",
3   "model": "Mustang",
4   "year": 1964
5   }
6   for x in thisdict:
7       print(x)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
brand
model
year
```

Perlu ingat kembali seperti yang sudah kita bahas di bab sebelumnya, jangan lupa perintah yang ingin kita jalankan di dalam for loop, harus menjorok ke depan (indent).

Di contoh diatas, kita dapat lihat bahwa ketika kita gunakan kode x in thisdict, maka x akan menunjukkan kata kunci. Untuk menampilkan nilainya, alih-alih kata kuncinya, kita bisa menggunakan contoh berikut

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x in thisdict:
    print(thisdict[x])
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
Ford
Mustang
1964
```

Atau jika kita ingin menampilkan pasangan kunci dan nilai, dengan sedikit strategi, kita bisa menggunakan fungsi items() yang sudah kita bahas sebelumnya. Perhatikan contoh berikut:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
for x, y in thisdict.items():
    print(x, y)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
brand Ford
model Mustang
year 1964
```

MENYALIN DICTIONARY

Kita tidak dapat menyalin dictionary hanya dengan mengetik dict2 = dict1, karena: dict2 hanya akan menjadi referensi ke dict1, dan perubahan yang dibuat dict1 secara otomatis juga akan dibuat di dict2.
Ada cara untuk membuat salinan, salah satu caranya adalah dengan menggunakan metode **copy()**. Perhatikan contoh berikut.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
mydict = thisdict.copy()

print(mydict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Cara lain untuk membuat salinan adalah dengan menggunakan fungsi bawaan dict(). Perhatikan contoh berikut.

```python
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

NESTED DICTIONARY

Di dalam dictionary, juga dapat berisi dictionary, kita sebut ini nested dictionary. Perhatikan contoh berikut.

```python
myfamily = {
"child1" : {
"name" : "Emil",
"year" : 2004
},
"child2" : {
"name" : "Tobias",
"year" : 2007
},
"child3" : {
"name" : "Linus",
"year" : 2011
}
}
print(myfamily)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name':
 'Linus', 'year': 2011}}
```

Pada contoh diatas kita membuat dictionary myfamily yang isinya adalah 3 item dengan kunci child1, child2, dan child3

Atau, jika kita ingin menambahkan tiga kamus ke dalam kamus baru juga bisa, perhatikan contoh berikut.

```python
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}

print(myfamily)
```

```
PS C:\Users\dwiaj\OneDrive\Documents\1. python> python tes.py
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name':
 'Linus', 'year': 2011}}
```

Pada contoh diatas hasilnya sama juga dengan contoh sebelumya.

Latihan 3.3

1. Buatlah sebuah dictionary yang berisi biodata masing-masing praktikan
   - Nama
   - Nim
   - Kelas
   - Jurusan

   Kemudian tambahkan alamat pada dictionry tersebut

Praktikum 4

Judul : preprocessing data

Deskripsi :

Preprocessing data adalah tahap awal dalam proses data mining yang sangat penting. Tujuannya adalah untuk membersihkan, mengubah, dan mempersiapkan data mentah menjadi format yang lebih mudah dipahami dan diproses oleh algoritma data mining.

Python adalah bahasa pemrograman yang sangat populer untuk melakukan preprocessing data dalam data mining. Ada beberapa teknik preprocessing data yang dapat dilakukan menggunakan Python, di antaranya:

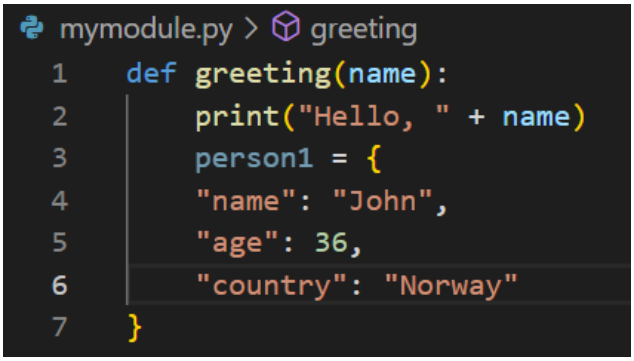
1. Data Cleaning: Teknik untuk membersihkan data mentah dari noise dan missing values. Beberapa library Python seperti pandas dan numpy menyediakan fungsi untuk mengatasi masalah ini.
2. Data Transformation: Teknik untuk mengubah format data menjadi format yang lebih mudah diproses oleh algoritma data mining. Beberapa contoh transformasi data adalah normalisasi data, encoding kategori data, dan pengurangan dimensi data.
3. Data Integration: Teknik untuk menggabungkan data dari berbagai sumber menjadi satu. Library pandas menyediakan fitur untuk menggabungkan data dari berbagai sumber.
4. Data Reduction: Teknik untuk mengurangi ukuran data agar lebih mudah diproses oleh algoritma data mining. Contoh teknik ini adalah feature selection dan feature extraction.
5. Data Discretization: Teknik untuk mengubah data kontinu menjadi data diskrit. Contoh teknik ini adalah binning, clustering, dan decision tree.

Setelah melakukan preprocessing data, data siap untuk diproses menggunakan algoritma data mining yang sesuai. Dalam Python, banyak library seperti Scikit-learn dan TensorFlow yang menyediakan berbagai algoritma data mining yang dapat digunakan.

1. Modul dalam python
   Modul adalah file yang berisi sekumpulan fungsi dan kode yang ingin Anda simpan dalam format .py dan ingin anda sertakan dalam aplikasi Anda.

   MEMBUAT MODUL

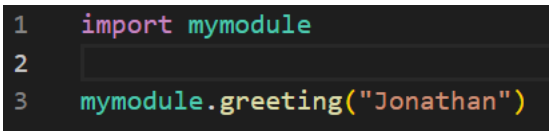
   Untuk membuat modul cukup simpan kode yang Anda inginkan dalam file dengan ekstensi file .py: Contoh, simpan kode dibawah ini sebagai **mymodule.py**.



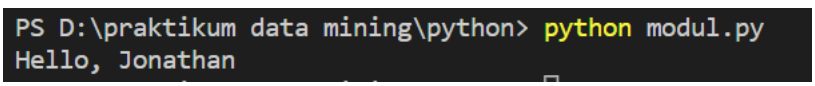
   Modul mymodule.py ini kita buat memiliki satu buah fungsi yaitu greeting yang memiliki satu argumen. Dan juga satu buah variabel global yakni sebuah dictiionary yang bernama person1 yang memiliki 3 item dengan kunci secara berurutan name, age, dan country.

   MENGIMPORT DAN MENGGUNAKAN MODUL

   Sekarang kita dapat menggunakan modul yang baru saja kita buat (mymodule.py), dengan menggunakan pernyataan import. Perhatikan contoh berikut.



   Output



   Saat menggunakan fungsi dari modul, gunakan sintaks: module_name.function_name.

Pada contoh diatas kode *import mymodule*, artinya adalah kita mengimport modul mymodule.py. Ingat, kita tidak perlu menuliskan ".py"-nya. Lalu kode *mymodule.greeting("jonathan")*, artinya adalah kita mencoba mengeksekusi fungsi greeting yang ada di dalam modul mymodule. Hasilnya kita bisa lihat di console.

Modul dapat berisi fungsi, seperti yang telah dijelaskan, tetapi juga variabel dari semua jenis (array, kamus, objek, dll). Untuk mengaksesnya adalah sama persis seperti mengakses fungsi. Berikut contoh untuk mengakses varibel person1 dari module mymodule.

```
PS D:\praktikum data mining\python> python modul.py
36
```

Anda juga dapat menamai ulang module yang diimport dengan kata kunci as. Biasanya digunakan nama yang lebih singkat.

```
1   import mymodule as mx
2
3   a = mx.person1["age"]
4   print(a)
```

Ada beberapa modul bawaan dalam Python, yang dapat Anda impor kapan pun Anda mau seperti platform, dan lain-lain.
Untuk melihat list dari semua nama fungsi dan variabel dari suatu module kita dapat menggunakan fungsi dir().

From nama_module import nama_fungsi

Perhatikan contoh berikut. Kita mau mencoba mengimport variable person1 dari module mymodule.

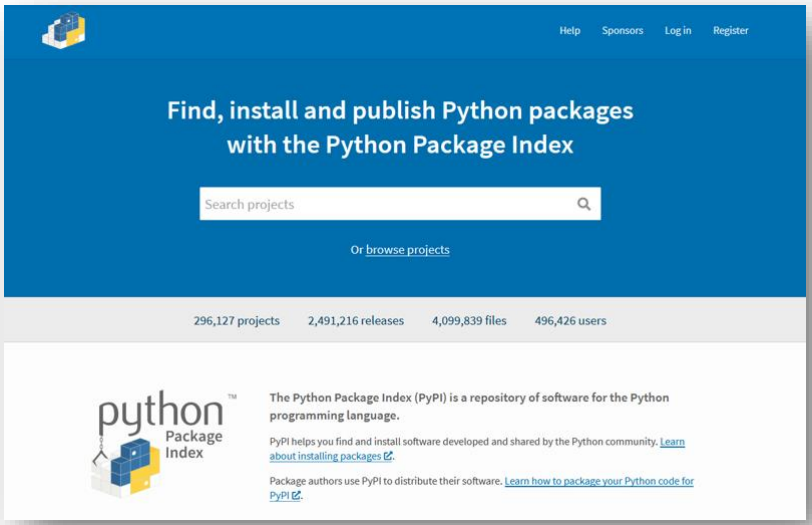
```
1   from mymodule import person1
2
3   print(person1["age"])
```

Perlu dicatat, saat mengimport hanya satu fungsi atau variabel dari suatu modul dengan syntaks from, untuk memanggil fungsi atau variabel tersebut langsung panggil saja tanpa perlu merujuk ke nama modulenya.

Latihan 3.4

a. Buatlah module yang berisi fungsi untuk menampilkan nama, nim dan kelas masing-masing
b.

2. PIP



PIP adalah singkatan dari **Package Installer for Python**. PIP adalah sebuah perangkat lunak untuk mengelola paket atau library Python, dan memudahkan pengguna untuk menginstal, menghapus, dan memperbarui paket Python.

Dalam Python, banyak library atau modul yang telah dikembangkan oleh pengembang lain yang dapat digunakan untuk memperluas fungsionalitas Python. Library tersebut dapat diinstal melalui PIP, sehingga kita tidak perlu men-download dan menginstal secara manual satu per satu.

Berikut adalah beberapa perintah dasar PIP yang perlu diketahui oleh praktikan Python:

1. Instalasi library menggunakan PIP
   Untuk menginstal sebuah library, dapat menggunakan perintah "pip install nama_library". Contoh: untuk menginstal library Numpy, gunakan perintah "pip install numpy".
2. Menghapus library menggunakan PIP
   Untuk menghapus sebuah library yang sudah terinstal, dapat menggunakan perintah "pip uninstall nama_library". Contoh: untuk menghapus library Numpy, gunakan perintah "pip uninstall numpy".
3. Melihat daftar library yang sudah terinstal menggunakan PIP
   Untuk melihat daftar library yang sudah terinstal, dapat menggunakan perintah "pip list".
4. Mencari informasi tentang sebuah library menggunakan PIP

Untuk mencari informasi tentang sebuah library, dapat menggunakan perintah "pip search nama_library". Contoh: untuk mencari informasi tentang library Numpy, gunakan perintah "pip search numpy".

5. Memperbarui library menggunakan PIP
   Untuk memperbarui sebuah library yang sudah terinstal, dapat menggunakan perintah "pip install --upgrade nama_library". Contoh: untuk memperbarui library Numpy, gunakan perintah "pip install --upgrade numpy".

PIP adalah alat yang sangat berguna bagi praktikan Python untuk menginstal dan mengelola library Python yang diperlukan untuk proyek atau tugas mereka. Sebaiknya selalu menginstal library melalui PIP untuk memastikan library yang terinstal up-to-date dan terintegrasi dengan baik dalam lingkungan Python.

3. File Handling

File handling adalah teknik mengoperasikan file baik itu membaca, menulis, menghapus, dan mengedit. Ini sangat penting dan wajib untuk dipelajari.

MEMBACA FILE

Asumsikan kita memiliki file berikut, terletak di folder yang sama dengan file Python yang mau anda run.

Nama file : *demofile.txt*
Isi : “python itu mudah”

Untuk membuka file tersebut (mengimport), di python kita bisa menggunakan fungsi open() dan lalu dengan fungsi read(). Ini adalah fungsi bawaannya python jadi kita tidak perlu mengimport modul apapun.

```
1   f = open("demofile.txt", "r")
2
3   print(f.read())
4   f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
python itu mudah
```

Jika file berada di folder atau directory yang berbeda maka, kita harus tuliskan full path nya.

Secara default fungsi read() mengembalikan seluruh teks, tetapi Anda juga dapat menentukan berapa banyak karakter yang ingin Anda kembalikan.

Sebagai catatan, setelah mengoperasikan file, kita juga harus menutupnya dengan fungsi close()

```
1   f = open("demofile.txt", "r")
2   print(f.read(5))
3   f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
pytho
```

MEMBACA FILE PER BARIS

Kita dapat mengembalikan satu baris dengan menggunakan fungsi**readline()**.

Buat file Pancasila.txt denga nisi adalah 5 sila Pancasila

pancasila.txt

```
1. Ketuhanan yang Maha Esa
2. Kemanusiaan yang adil dan beradab
3. Persatuan Indonesia
4. Kerakyatan yang dipimpin oleh hikmat kebijaksanaan dalam permusyawaratan/perwakilan, serta
5. Keadilan sosial bagi seluruh rakyat Indonesia
```

```
1   f = open("pancasila.txt", "r")
2   print(f.readline())
3   f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
1. Ketuhanan yang Maha Esa
```

Dengan memanggil readline() dua kali, kita dapat membaca dua baris pertama dari file yang didefinisikan.

```
1   f = open("pancasila.txt", "r")
2   print(f.readline())
3   print(f.readline())
4   f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
1. Ketuhanan yang Maha Esa

2. Kemanusiaan yang adil dan beradab
```

Selain menggunakan fungsi readline(), untu membaca file baris per baris, kita juga bisa menggunakan looping for in. Perhatikan contoh berikut.

```
f = open("pancasila.txt", "r")
for x in f:
    print(x)
f.close()
```

```
PS D:\praktikum data mining\python> python modul.py
1. Ketuhanan yang Maha Esa

2. Kemanusiaan yang adil dan beradab

3. Persatuan Indonesia

4. Kerakyatan yang dipimpin oleh hikmat kebijaksanaan dalam permusyawaratan/perwakilan, serta

5. Keadilan sosial bagi seluruh rakyat Indonesia
```

MENAMBAHKAN KONTEN KE FILE YANG ADA

Untuk menulis ke file yang ada, maka sintaks

```
f=open("namafile.txt", "a")
f.write("kontan yang baru")
```

Perhatikan pada contoh sebeumnya, di fungsi open, kita menggunakan parameter "r", yang artinya adalah read-only.

Latihan 3.5

1. Buatlah sebuah file yang berisi biodata masing-masing yang berisi
   - Nama
   - Nim
   - Kelas
   - Jurusan
   - Alamat
   - Hoby

   Kemudian tampilkan dengan python