DASPRO

ALPRO -20

MODUL 4

Tipe Data: List, Dictionary, Tuple, dan Set

Tim Penyusun

NIM	NAMA	KODE ASISTEN
1202180127	**ABDURRAHMAN AZIZ**	ABZ
1202184189	**AGRIVA DETTA GINTING**	GIP
1202171049	**AHMAD SHOHIBUS SULTHONI**	TON
1202184143	**ARDDHANA ZHAFRAN AMANULLAH**	ZAF
1202184085	**ARDY RIKARDO**	RDY
1202184347	**DANIEL ALEXANDER POLII**	DNI
1202184135	**DEWA MADE SURYA PERMANA MASTRA**	DEM
1202184072	**EKKY CHANDRA WIBOWO**	EKY
1202183302	**FAUZI ARIFIN ALGHIFARI**	OZI
1202184126	**GHUFRON FIKRIANTO**	GHF
1202180229	**IFEN FARIDIAN RAHMADAN**	FEN
1202184159	**ILMA NUR HIDAYATI**	ILM
1202180092	**JODY MARDIKA**	JDM
1202184138	**M. FAIZ TRIPUTRA**	XAX
1202184310	**MUHAMMAD DIFAGAMA IVANKA**	PAA
1202184077	**NANDA ARFAN HAKIM**	NAN
1202174288	**NUR AZIZAH HARUN**	CHA
1202170250	**RAHADIAN ALDI NUGROHO**	ALD
1202184117	**RAJA NANDA SATRYA**	JAX
1202184062	**RISKI ANANDA WIDIYA PUTRI**	RAA
1202183322	**RIZALRASYD DWISELIA RIDWANAH**	ZAY
1202184019	**SHAHNAZ KAMILAH**	SAZ
1202184209	**TITISARI RAMADHANE**	TRD
1202184224	**TODOAN JEHEZKIEL SANTANA MUARA S.**	TDS
1202181065	**ZAHWA ALIFAH AMMATULLAH**	ZAW

Peraturan Praktikum

1. Tidak diperbolehkan menggunakan nilai praktikum tahun sebelumnya tanpa ada per*set*ujuan koordinator dosen mata kuliah praktikum yang bersangkutan dengan kepala urusan laboratorium.
2. Jika praktikan mengajukan izin (sakit dan anggota keluarga meninggal), maka surat perizinan diberikan ke pihak Laboratorium maksimal 6 hari *set*elah jadwal praktikum masing-masing.
3. Izin lomba atau penugasan institusi tidak berlaku, kecuali sudah terdapat dispensasi di Igracias. NB: Dilampirkan screenshoot dispensasi igracias.
4. **Seragam praktikum**
   a. Bagi mahasiswa tidak diperkenankan menggunakan celana berbahan jeans/ chino (tidak strech) saat praktikum.
   b. Bagi mahasiswi diwajibkan menggunakan rok panjang tidak ketat saat praktikum
   c. Jika ditermukan praktikan menggunakan hal-hal terkait, maka diperkenankan untuk mengganti dengan dresscode yang sesuai dengan peraturan Universitas Telkom dan tidak ada penambahan waktu.

      NB: Untuk dresscode hari Senin dapat menggunakan kemeja merah telkom atau kemeja putih polos, dresscode hari Selasa sampai Rabu menggunakan kemeja putih. Untuk dresscode hari Kamis s/d Sabtu praktikan tetap diwajibkan menggunakan kemeja formal berkancing depan dari atas sampai bawah dan berkerah (bukan kerah sanghai dan bukan polo) dan celana atau rok bahan berwarna hitam/biru gelap.

Rundown Praktikum

Daftar Isi

Tim Penyusun 1

Peraturan Praktikum 2

Rundown Praktikum 3

Daftar Isi 4

1. Pengenalan *Data Structure* 5

1.1. *List* [ ] 5

1.1.1. Mengakses dan Memanipulasi Elemen *List* [ ] 6

1.2. *Tuple* ( ) 10

1.2.1. Mengakses Elemen *Tuple* ( ) 11

1.3. *Set* { } 11

1.3.1. Memanipulasi Elemen *Set* { } 13

1.4. *Dictionary* { } 16

1.4.1. Mengakses dan Memanipulasi Elemen *Dictionary* { } 16

2. *Tips* dan *Trick* 19

2.1. Menampilkan Elemen Data menggunakan *FOR LOOP* 19

2.1.2. *List* [ ], *Tuple* ( ), dan *Set* { } 19

2.1.2. *Dictionary* { } 20

2.2 Metode Lainnya 21

3. Studi Kasus 22

3.1 Contoh Studi Kasus Struktur Data *List* 22

3.2 Contoh Studi Kasus Struktur Data *Tuple* 23

3.3 Contoh Studi Kasus Struktur Data *Set* 24

3.4 Contoh Studi Kasus Struktur Data *Dictionary* 25

Referensi 27

1. Pengenalan *Data Structure*

List, tuple, dictionary, dan *set* merupakan sebuah struktur data. Struktur data sendiri merupakan kumpulan data atau suatu kelompok data dengan tata letak yang berisi kolom-kolom data. Dengan adanya struktur data kita dapat menyimpan sekumpulan data hanya dengan satu variabel di waktu yang sama. Baik *list, tuple, dictionary* dan *set* setiap elemennya dipisahkan dengan tanda koma (**,**).

1.1. *List* [ ]

List adalah struktur data yang berisi elemen yang berurut / tersusun dengan jelas. Tiap elemen (anggota) *list* memiliki *index* sesuai dengan urutannya. *Index* dimulai dari 0.

List bisa berisi elemen dengan tipe yang sama maupun berbeda. Untuk mendeklarasikan *list*, digunakan tanda kurung siku **[ ]** dan masing-masing elemennya dipisahkan oleh tanda koma.

Berikut merupakan cara penulisan *list*:

Source Code :

```
list1 = ["apel","jeruk","manggis","durian","salak"] #Contoh list dengan tipe data sama
list2 = ["apel", 12] #Contoh data list dengan tipe data berbeda
print("Isi list1 :",list1) #Menampilkan list1
print("Isi list2 :",list2) #Menampilkan list2
print("Index ke-0 pada list1 :",list1[0]) #Menampilkan data di list1 index ke-0
print("Index ke-1 sampai 3 pada list1:",list1[1:4]) #Menampilkan data di list1 index ke-1 sampai 3
```

Output :

```
Print Output:

Isi list1 : ['apel', 'jeruk', 'manggis', 'durian', 'salak']
Isi list2 : ['apel', 12]
Index ke-0 pada list1 : apel
Index ke-1 sampai 3 pada list1: ['jeruk', 'manggis', 'durian']
```

1.1.1. Mengakses dan Memanipulasi Elemen *List* [ ]

Kita dapat menggunakan metode bawaan (*built-in*) dari *python* seperti yang tertera pada tabel di bawah ini:

Metode	Syntax	Kegunaan
Append	*nama_list.append(value)*	Menambahkan nilai baru kedalam index yang terdapat pada *list*
Insert	*nama_list.insert(index, value)*	Menyisipkan nilai baru pada posisi tertentu kedalam index yang terdapat pada *list*
Extend	*nama_list.extend(variabel_list)*	Menggabungkan beberapa *list* kedalam suatu *list*
Clear	*nama_list.clear()*	Menghapus semua elemen secara keseluruhan
Count	*nama_list.count(value)*	Mengembalikan jumlah elemen yang sesuai dengan *value* pada *list*
Index	*nama_list.index(value)*	Mengembalikan *index* dari elemen pertama yang sama dengan *value*
Reverse	*nama_list.reverse()*	Membalikkan urutan *value* pada struktur data
Sort	*nama_list.sort(nama_list, reverse=False)*	Mengurutkan *value* pada struktur data
Copy	*list_one = list_two*	Menyalin data dari *list_two* kedalam *list_one*

Berikut contoh penggunaan metode *append, insert,* dan *extend* pada list:

Source Code :

Penggunaan metode append, insert, extend

```
list1 = ['apel','jeruk','nanas']
print("Isi list1 sebelum ditambah dengan metode append :",list1)
list1.append('durian') #Menambahkan dengan metode append
print("Isi list1 setelah ditambah dengan metode append :",list1)
print("-+"*45,'\n')

list2 = ['salak','mangga','anggur']
print("Isi list2 sebelum ditambah dengan metode insert :",list2)
list2.insert(2,'manggis')    #Menambahkan dengan metode insert
                             #Menyisipkan manggis pada posisi index ke-2
print("Isi list2 setelah ditambah dengan metode insert :",list2)
print("-+"*45,'\n')

list3 =['nangka','pisang','kelapa']
print("Isi list3 sebelum ditambah dengan metode extend :",list3)
list3.extend(list2)
print("Menggabungkan isi list3 dengan list2 :",list3)
```

Output :

```
Isi list1 sebelum ditambah dengan metode append : ['apel', 'jeruk', 'nanas']
Isi list1 setelah ditambah dengan metode append : ['apel', 'jeruk', 'nanas', 'durian']
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Isi list2 sebelum ditambah dengan metode insert : ['salak', 'mangga', 'anggur']
Isi list2 setelah ditambah dengan metode insert : ['salak', 'mangga', 'manggis', 'anggur']
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Isi list3 sebelum ditambah dengan metode extend : ['nangka', 'pisang', 'kelapa']
Menggabungkan isi list3 dengan list2 : ['nangka', 'pisang', 'kelapa', 'salak', 'mangga', 'manggis', 'anggur']
```

Kita dapat menambahkan metode *append dengan* inputan dari *user*:

Source Code :

Penggunaan metode append dengan inputan dari user

```
list1 = ['buah','sayur','bunga']
print("Isi list1 sebelum ditambahkan indexnya :",list1)
print("Panjang list1 sebelum ditambahkan :",len(list1))
tambahIndex = input("Apa yang ingin anda tambahkan? ")
list1.append(tambahIndex) #Menambahkan isi pada list1
print("Index berhasil ditambahkan\n")
print("Isi list1 telah ditambah",list1)
print("Panjang list1 setelah ditambah :",len(list1))
```

Output :

```
Isi list1 sebelum ditambahkan indexnya : ['buah', 'sayur', 'bunga']
Panjang list1 sebelum ditambahkan : 3
Apa yang ingin anda tambahkan? biji-bijian
Index berhasil ditambahkan

Isi list1 telah ditambah ['buah', 'sayur', 'bunga', 'biji-bijian']
Panjang list1 setelah ditambah : 4
```

Pengisian *list* juga dapat menggunakan metode for *loop* seperti ini :

Source Code :

```
list1= []
panjangList = int(input("Masukkan panjang list : ")) #Banyak data yang ingin dimasukan
for i in range (0,panjangList):
    inputan=input("masukkan index : ") #Data yang ingin dimasukkan
    list1.append(inputan)
print("\nMenampilkan isi List")
for i in range (0,panjangList):
    print("Isi index ke-",i,": ",list1[i]) #Menampilakan data dengan for loop
```

Output :

```
Masukkan panjang list : 3
masukkan index : Pertama
masukkan index : Kedua
masukkan index : Ketiga

Menampilkan isi List
Isi index ke- 0 :  Pertama
Isi index ke- 1 :  Kedua
Isi index ke- 2 :  Ketiga
```

Untuk menghapus elemen dari *list*, kita dapat menggunakan metode ***pop*****,** dan ***remove*****.**

Metode	Syntax	Kegunaan
Pop	*nama_list.pop( )*	Menghapus elemen pada urutan terakhir pada *list*
Pop	*nama_list.pop(index)*	Menghapus elemen pada urutan tertentu pada *list*
Remove	*nama_list.remove(value)*	Menghapus elemen sesuai dengan nama value yang kita berikan

Berikut merupakan penggunaan metode *pop* dan *remove*:

Source Code :

```
list2 = ['salak','mangga','anggur']
print("Isi list2 sebelum dihapus dengan fungsi pop :",list2)
list2.pop() #Menghapus dengan fungsi pop()
print("Isi list2 setelah dihapus dengan fungsi pop :",list2)
print('\n',"-+"*45,'\n')

list1 = ['apel','jeruk','nanas']
print("Isi list1 sebelum elemen index ke-1 dihapus dengan fungsi pop(index) :",list1)
list1.pop(1) #Menghapus dengan fungsi pop(index) pada index ke-1
print("Isi list1 setelah elemen index ke-1 dihapus dengan fungsi pop(index) :",list1)
print('\n',"-+"*45,'\n')

list3 =['nangka','pisang','kelapa']
print("Isi list3 sebelum dihapus dengan fungsi remove :",list3)
list3.remove('nangka') #Menghapus dengan fungsi remove
print("Isi list3 setelah dihapus dengan fungsi remove :",list3)
```

Output :

```
Isi list2 sebelum dihapus dengan fungsi pop : ['salak', 'mangga', 'anggur']
Isi list2 setelah dihapus dengan fungsi pop : ['salak', 'mangga']

 -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Isi list1 sebelum elemen index ke-1 dihapus dengan fungsi pop(index) : ['apel', 'jeruk', 'nanas']
Isi list1 setelah elemen index ke-1 dihapus dengan fungsi pop(index) : ['apel', 'nanas']

 -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Isi list3 sebelum dihapus dengan fungsi remove : ['nangka', 'pisang', 'kelapa']
Isi list3 setelah dihapus dengan fungsi remove : ['pisang', 'kelapa']
```

1.2. *Tuple* ( )

Tuple dalam *python* adalah struktur data yang digunakan untuk menyimpan data konstan atau statis. Tuple bersifat *immutable*, artinya **elemen *tuple* tidak bisa kita ubah dan hapus**. Namun, dapat kita isi dengan berbagai macam nilai dan objek hanya pada saat inisialisasi.

Tuple dideklarasikan dengan menggunakan tanda kurung **( )** dan elemennya dipisahkan oleh tanda koma. *Tuple* berguna untuk data yang dimaksudkan tidak diubah isinya. Misalnya *tuple* komposisi warna untuk putih adalah (255,255,255).

Berikut merupakan cara penulisan *tuple*:

Source Code :

```python
tuple1 = (15,27,33,47) #tuple berisi angka
tuple2 = ('FRI','FTE','FKB','FIT','FIF') #Tuple berisi string
tuple3 = (['DASPRO','EAD','BPAD'],['MANPROSI','GARTEK','MARKETING']) #Tuple berisi list
print("Isi tuple1 :",tuple1) #Menampilkan tuple1
print("Isi tuple2 :",tuple2) #menampilkan tuple2
print("Isi tuple3 :",tuple3) #menampilkan tuple3
print("Index ke-1 pada tuple3 :",tuple3[1]) #menampilkan tuple3 index ke-1
```

Output :

```
Isi tuple1 : (15, 27, 33, 47)
Isi tuple2 : ('FRI', 'FTE', 'FKB', 'FIT', 'FIF')
Isi tuple3 : (['DASPRO', 'EAD', 'BPAD'], ['MANPROSI', 'GARTEK', 'MARKETING'])
Index ke-1 pada tuple3 : ['MANPROSI', 'GARTEK', 'MARKETING']
```

1.2.1. Mengakses Elemen *Tuple* ( )

Berikut beberapa metode bawaan (*built-in*) *python* untuk mengakses isi *Tuple*.

Metode	Syntax	Kegunaan
Count	*tuple.count(value)*	Mengembalikan jumlah elemen yang sesuai dengan *value* pada *tuple*
Index	*tuple.index(value)*	Mengembalikan *index* dari elemen pertama yang sama dengan *value*

Berikut merupakan contoh penggunaan metode *count()* dan *index()* pada *tuple*:

Source Code :

```
tuple1 = ('H','A','P','P','Y',' ','C','O','D','I','N','G')
print("Output metode count :",tuple1.count('D'))
print("Output metode index :",tuple1.index('D'))
```

Output :

```
Output metode count : 1
Output metode index : 8
```

1.3. *Set* { }

Set adalah salah satu struktur data di *python* yang tidak berurut **(*unordered*)**. *Set* memiliki elemen yang unik **(tidak ada duplikasi)**. Jika kita meletakkan dua elemen yang sama di dalam *set*, maka otomatis *set* akan menghilangkan yang salah satunya.

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

Set dibuat dengan meletakkan elemen-elemennya **ke dalam fungsi *set***-nya dan dipisahkan menggunakan tanda koma. Kita juga bisa membuat *set* dari *list* dengan memasukkan *list* ke dalam fungsi ***set()***.

Set bisa berisi data campuran, baik *integer, float, string*, dan lain sebagainya. Akan tetapi *set* tidak bisa berisi *list*, *set*, dan *dictionary*.

Ciri khas yang dimiliki oleh *set { }* :

Operasi	Kegunaan
setA | setB *setA.union(setB)*	Mengembalikan nilai set yang merupakan *union* (gabungan) dari set A dan set B
setA & setB *setA.intersection(setB)*	Mengembalikan nilai set yang merupakan *intersection* (irisan) dari set A dan set B
setA - setB *setA.difference(setB)*	Mengembalikan selisih nilai dari set A dan set B (elemen yang termasuk di A, tapi tidak termasuk di B)
setA ^ setB *setA.symmetric_difference(setB)*	Mengembalikan nilai komplemen dari set A dan set B (elemen milik A atau B yang tidak sama)
setA <= setB *setA.issuperset(setB)*	Mengembalikan *True* apabila A merupakan subset dari B
setA >= setB *setA.issuperset(setB)*	Mengembalikan *True* apabila B merupakan subset dari A
setA.isdisjoint(setB)	Mengembalikan *True* apabila set B tidak mengandung set A, dan *False* apabila set B mengandung set A

Berikut merupakan cara penulisan *set*:

Source Code :

```
set1 = {1,2,3}
print("Isi Set1 :",set1)

#Dengan menggunakan fungsi set()
set2 = set(['bunga','sayur','buah'])
print("\nIsi Set2 :",set2)

# set data campuran
set3 = {1, 2.0, "Python", (3,4,5)}
print("\nIsi Set3 :",set3)

# bila kita mengisi duplikasi, set akan menghilangkan salah satu
# output: {1,2,3}
set4 = {1,2,2,3,3,3}
print("\nIsi Set4 :",set4)

# set tidak bisa berisi anggota list
# contoh berikut akan muncul error TypeError
set5 = {1,2,[3,4,5]}
print("\nIsi Set5 :",set5)
```

Output set1 – set4 :

```
Isi Set1 : {1, 2, 3}

Isi Set2 : {'buah', 'sayur', 'bunga'}

Isi Set3 : {1, 2.0, 'Python', (3, 4, 5)}

Isi Set4 : {1, 2, 3}
```

Output set5 :

```
Traceback (most recent call last):
  File "e:/xXMASBROOXx/PYTHON/BelajarPython/NoEnter.py", line 100, in <module>
    set5 = {1,2,[3,4,5]}
TypeError: unhashable type: 'list'
```

1.3.1. Memanipulasi Elemen *Set* { }

Set bersifat *mutable*. Tapi, karena *set* adalah struktur data tidak berurut atau tidak tersusun dengan rapi (*unordered*), maka *set* tidak memiliki *index* dan *set* tidak mendukung metode *slicing*.

Terdapat beberapa metode untuk mengubah isi *set*, antara lain:

Metode	Syntax	Kegunaan
Add	*nama_set.add(value)*	Menambah satu anggota ke dalam *set*
Update	*nama_set.update([value1, value2])*	Menambahkan beberapa elemen sekaligus ke dalam *set.* **List dan Tuple* bisa digunakan sebagai masukan dari metode *update*().

Berikut merupakan penggunaan metode *add* dan *update*:

Source Code :

```
set1={3,6,7,8,1}
print("Isi set1 sebelum ditambah dengan add :",set1)
set1.add(11)
set1.add(20)
print("Isi set1 setelah ditambah dengan add :",set1)
listA = [5,4,3,9,10]
set1.update(listA)
print("Isi set1 setelah ditambah dengan update :",set1)
```

Output :

```
Isi set1 sebelum ditambah dengan add : {1, 3, 6, 7, 8}
Isi set1 setelah ditambah dengan add : {1, 3, 6, 7, 8, 11, 20}
Isi set1 setelah ditambah dengan update : {1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 20}
```

Kita bisa menghapus elemen *set* dengan menggunakan metode ***discard()*** dan ***remove()***.

Metode	Syntax	Kegunaan
Discard	*set.discard(value)*	Menghapus anggota *set* namun tidak akan memunculkan *error* bila anggota yang ingin dihapus ternyata tidak ada di dalam *set*

Remove	*set.remove(value)*	Kebalikan dari *discard,* akan memunculkan *error* apabila anggota yang dihapus tidak ada di dalam *set*

Berikut merupakan contoh penggunaan metode *discard* dan *remove* pada *set*:

Source Code :

```
# Membuat set baru
set1 = {1, 2, 3, 4, 5}
print("Isi awal set1 :",set1)

# Menghapus 4 dengan discard
set1.discard(4)
print("Isi set1 setelah Discard angka 4 : ",set1)

# Menghapus 5 dengan remove
set1.remove(5)
print("Isi set1 setelah Remove angka 5 : ",set1)

# Jika anggota yang mau dihapus tidak ada dalam set
# Discard tidak akan memunculkan error
set1.discard(6)
```

Output :

Print Output:

```
Isi awal set1 : {1, 2, 3, 4, 5}
Isi set1 setelah Discard angka 4 :  {1, 2, 3, 5}
Isi set1 setelah Remove angka 5 :  {1, 2, 3}
```

Set1.discard*(6) tidak menampilkan *error* walaupun 6 tidak ada pada *set1

1.4. *Dictionary* { }

Dictionary adalah struktur data yang tiap elemennya terdiri dari pasangan kunci-nilai (*key-value*), key di *dictionary* itu harus ***static* atau *immutable* atau tetap (tidak bisa diubah)**, dan key di *dictionary* selain *string* juga dapat berupa *integer*. Mirip dengan kamus dimana ada kata ada arti. *Dictionary* umumnya dipakai untuk data yang besar dan untuk mengakses anggota data secara acak. *Index* pada *dictionary* adalah *key*-nya.

Dictionary dideklarasikan dengan menggunakan tanda kurung kurawal **{ }**, dimana elemennya memiliki bentuk *"key" : "value"* dan tiap elemen dipisah tanda koma. Kunci dan nilainya bisa memiliki tipe sembarang. Untuk mengakses nilai dari anggota *dictionary*, kita dapat menggunakan *key*-nya.

Berikut merupakan cara penulisan *dictionary*:

Source Code :

```
#Membuat Dictionary
skill = {
    "utama" : "Python", #"utama" sebagai Key dan "Python" sebagai value
    "lainnya" : ["PHP","JAVA","C++", "JS"]
}

#Mencetak isi key "utama"
print("Isi key utama :",skill["utama"])
print("isi key lainnya :",skill["lainnya"])
print("Isi key lainnya pada index ke-2 :",skill["lainnya"][2])
```

Output :

```
Isi key utama : Python
isi key lainnya : ['PHP', 'JAVA', 'C++', 'JS']
Isi key lainnya pada index ke-2 : C++
```

1.4.1. Mengakses dan Memanipulasi Elemen *Dictionary* { }

Dictionary bersifat *mutable*. Kita bisa menambahkan atau mengubah nilai dari elemennya menggunakan operator penugasan. Bila *key*-nya sudah ada, maka nilainya yang akan diupdate. Bila *key*-nya belum ada, maka akan ditambahkan sebagai *key* baru.

Berikut merupakan contoh penggunaan metode meng-*update* dan menambah elemen pada *dictionary*:

Source Code :

```
dict1 = {'nama':'Gani', 'usia':35}
print("Isi dict1 : ")
print(dict1)

# mengupdate nilai
dict1['usia'] = 36
print("\nMengupdate usia : ",
    "\nIsi key usia setelah diupdate : ",dict1["usia"])

# menambah anggota
dict1['alamat'] = 'Medan'
print("\nMenambah anggota/key baru :",
    "\nIsi dict1 setelah ditambah anggota/key baru : ")
print(dict1)
```

Output :

```
Isi dict1 :
{'nama': 'Gani', 'usia': 35}

Mengupdate usia :
Isi key usia setelah diupdate :  36

Menambah anggota/key baru :
Isi dict1 setelah ditambah anggota/key baru :
{'nama': 'Gani', 'usia': 36, 'alamat': 'Medan'}
```

Kita dapat menghapus elemen tertentu pada *dictionary* dengan menggunakan metode ***pop()***, ***popitem(),*** dan ***clear()***.

Metode	Syntax	Kegunaan
Pop	*nama_dictionary.pop(value)*	Menghapus elemen dengan mengembalikan kunci dari elemen tersebut
Popitem	*nama_dictionary.popitem()*	Menghapus elemen terakhir dari *dictionary*

Clear	*nama_dictionary.clear()*	Menghapus semua elemen secara keseluruhan dalam *dictionary*

Berikut merupakan contoh penggunaan metode ***pop(), popitem(),* dan *clear()*** pada *dictionary*:

Source Code :

```
pangkat = {1:1, 2:4, 3:9, 4:16, 5:25, "6":36}
print("Isi pangkat :",pangkat)
# menghapus anggota tertentu
print("\nMenghapus key 3 dengan pop() dan mengembalikan valuenya :",pangkat.pop(3))

# menghapus anggota secara acak
print("\nMenghapus anggota secara acak dengan popitem() :",pangkat.popitem())
print("\nIsi pangkat setelah dihapus secara acak :",pangkat)

# menghapus semua anggota
pangkat.clear()
print("\nMenghapus semua anggota pangkat dengan clear :")
# Error karena pangkat sudah dihapus
print("\nIsi pangkat setelah dihapus dengan clear :",pangkat)
```

Output :

```
Isi pangkat : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, '6': 36}

Menghapus key 3 dengan pop() dan mengembalikan valuenya : 9

Menghapus anggota secara acak dengan popitem() : ('6', 36)

Isi pangkat setelah dihapus secara acak : {1: 1, 2: 4, 4: 16, 5: 25}

Menghapus semua anggota pangkat dengan clear :

Isi pangkat setelah dihapus dengan clear : {}
```

2. *Tips* dan *Trick*

2.1. Menampilkan Elemen Data menggunakan *FOR LOOP*

2.1.2. *List* [ ], *Tuple* ( ), dan *Set* { }

Menggunakan *for loop* untuk menampilkan elemen dari *list*, *tuple*, dan *set*.

Berikut merupakan cara penulisannya:

Source Code :

```
list1 = ["matahari","bulan","bumi","manusia"]
tuple1 = ("Bojongsoang","Sukapura","Sukabirus")
set1 = {"Jawa Barat","Jawa Tengah","Jawa Timur"}

print("Isi dari list1 : ")
for elemen in list1:
    print(elemen,end=" ")

print("\nIsi dari tuple1 : ")
for elemen in tuple1:
    print(elemen, end=" ")

print("\nIsi dari set1 : ")
for elemen in set1:
    print(elemen,end=" ")
```

Output :

```
Isi dari list1 :
matahari bulan bumi manusia
Isi dari tuple1 :
Bojongsoang Sukapura Sukabirus
Isi dari set1 :
Jawa Timur Jawa Tengah Jawa Barat
```

2.1.2. *Dictionary* { }

Menggunakan *for loop* untuk menampilkan elemen dari *dictionary* **{ }.**

Berikut merupakan cara penulisan *dictionary*:

Source Code :

```
print(">> CARA 1 <<")
sebuahDict = {"Nama":"Budi","Umur":50,"Berat":70}
for key,val in sebuahDict.items():
    print("keynya = ", key)
    print("valnya = ", val,end=" ")
    print("\n")

#Atau dapat ditulis seperti ini juga
print(">> CARA 2 <<")
dict2 = {"Nama":"Budi","Umur":50,"Berat":70}
for key,val in dict2.items():
    print(key,val)
```

Output :

```
>> CARA 1 <<
keynya =  Nama
valnya =  Budi

keynya =  Umur
valnya =  50

keynya =  Berat
valnya =  70

>> CARA 2 <<
Nama Budi
Umur 50
Berat 70
```

2.2 Metode Lainnya

Terdapat beberapa metode yang bisa digunakan dalam struktur data:

Metode	Kegunaan	*List*	*Tuple*	*Set*	*Dictionary*
sort( )	Mengurutkan data				
count(value)	Menghitung banyaknya *value* dalam struktur data				
index(value)	Mencari *index* berdasarkan *value*				
reverse( )	Membalikkan urutan elemen pada struktur data				
update( )	Menambahkan lebih dari satu data ke dalam struktur data				

Keterangan warna:

- Merah = Tidak tersedia / tidak dapat digunakan
- Hijau = Tersedia / dapat digunakan

3. Studi Kasus

3.1 Contoh Studi Kasus Struktur Data *List*

Dalam sebuah kelompok mentoring Alpro terdapat 10 mahasiswa, Shahnaz selaku asisten dalam kelompok tersebut ingin membuat sebuah program yang dapat menyimpan nama-nama anggota kelompok mentoring dalam sebuah struktur data *list*. Bantulah Shahnaz untuk membuat program tersebut!

Contoh *Output* :

```
        PROGRAM DATA MAHASISWA MENTORING ALPRO
Masukkan Nama Mahasiswa ke-1 : Fauzi
Masukkan Nama Mahasiswa ke-2 : Daniel
Masukkan Nama Mahasiswa ke-3 : Todo
Masukkan Nama Mahasiswa ke-4 : Aziz
Masukkan Nama Mahasiswa ke-5 : Dede
Masukkan Nama Mahasiswa ke-6 : Ghufron
Masukkan Nama Mahasiswa ke-7 : Ilma
Masukkan Nama Mahasiswa ke-8 : Rizky
Masukkan Nama Mahasiswa ke-9 : Ekky
Masukkan Nama Mahasiswa ke-10 : Faiz
--------------------------------------------------

Daftar mahasiswa Alpro : ['Fauzi', 'Daniel', 'Todo', 'Aziz', 'Dede', 'Ghufron', 'Ilma', 'Rizky', 'Ekky', 'Faiz']
```

Source Code :

```
print('\tPROGRAM DATA MAHASISWA MENTORING ALPRO')

#inisialisasi List
nama_mahasiswa_alpro = []
for i in range(10):
    #memasukkan inputan user sekaligus menambahkan ke List
    nama_mahasiswa_alpro.append(input("Masukkan Nama Mahasiswa ke-"+str(i+1)+" : "))

print('-'*50)
#menampilkan List
print("\nDaftar mahasiswa Alpro :",nama_mahasiswa_alpro)
```

3.2 Contoh Studi Kasus Struktur Data *Tuple*

Jody merupakan seorang *Software Developer*, ia sedang membuat sebuah program yang dapat mengkonversi urutan bulan menjadi nama bulan secara berurutan. Dimana jika pengguna program konversi tersebut menginputkan nomor maka akan mencetak nama bulannya.

Contoh *Output* :

```
        PROGRAM KONVERSI BULAN
Bulan ke : 5
Mei
```

```
        PROGRAM KONVERSI BULAN
Bulan ke : 14
Gak ada Broo...
```

Source Code :

```
print("\tPROGRAM KONVERSI BULAN")

#inisialisasi Tuple
nama_bulan = ('Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus', 'September', 'Oktober', 'November', 'Desember')

#menginputkan urutan bulan
urutan_bulan = int(input("Bulan ke : "))

#mengecek inputan user (harus diantara 1 dan 12)
if(1 <= urutan_bulan <= 12):
    #mencetak nama bulan
    print(nama_bulan[urutan_bulan-1])

else:
    #jika inputan user bukan diantara 1 dan 12
    print('Gak ada Broo...')
```

3.3 Contoh Studi Kasus Struktur Data *Set*

Seorang pedagang buah bernama Ifen selalu kesusahan ketika mengecek stok buah yang ada di tokonya, akhirnya Ifen pun meminta tolong kepada Anda untuk membuatkan sebuah program untuk mengecek stok barang di tokonya.

Contoh *Output* :

```
        PROGRAM CEK STOK BUAH
Stok buah yang tersedia :
nanas, mangga, naga, pisang, apel, semangka,
Cari buah apa? : apel
Buah apel TERSEDIA
```

```
        PROGRAM CEK STOK BUAH
Stok buah yang tersedia :
mangga, nanas, pisang, semangka, apel, naga,
Cari buah apa? : manggis
Buah manggis TIDAK TERSEDIA
```

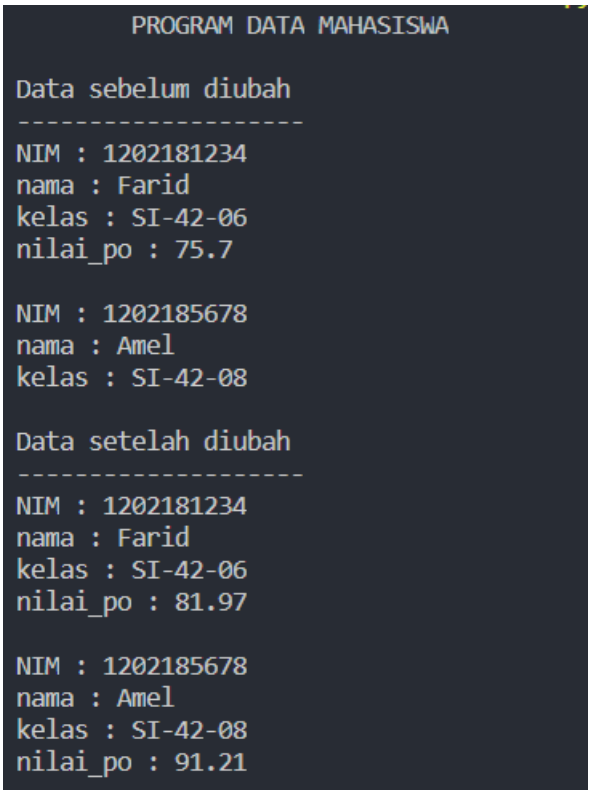
Source Code :

```
print('\tPROGRAM CEK STOK BUAH')

#inisilasisasi Set, masukkan stok buah
stok_buah = {'pisang', 'apel', 'nanas', 'mangga', 'semangka', 'naga'}

#menampilkan Set stok_buah dengan Perulang
print('Stok buah yang tersedia : ')
for elemen in stok_buah:
    print(elemen, end=', ')
print()
#memasukkan inputan user
cari_buah = input('Cari buah apa? : ')

#cek apakah inputan user ada di Set stok_buah
if cari_buah in stok_buah:
    #print tersedia jika ada
    print('Buah', cari_buah, 'TERSEDIA')

else:
    #print tidak tersedia jika tidak ada
    print('Buah', cari_buah, 'TIDAK TERSEDIA')
```

3.4 Contoh Studi Kasus Struktur Data *Dictionary*

Buatlah sebuah program yang dapat menampung data mahasiswa berdasarkan NIM yang didalamnya terdapat nama, kelas, dan nilai Perilaku Organisasi. Tetapi dari data mahasiswa tersebut Amel belum memiliki nilai Perilaku Organisasi. Tambahlah nilai Perilaku Organisasi serta ubahlah nilai Perilaku Organisasi milik Farid dari 75,70 menjadi 81,97.

Contoh *Output* :



Source Code :

```
print('\tPROGRAM DATA MAHASISWA\n')

#inisialisasi Dictionary, masukkan data mahasiswa
data_mahasiswa = {
    '1202181234' : {
        'nama' : 'Farid',
        'kelas' : 'SI-42-06',
        'nilai_po' : 75.70
    },
    '1202185678' : {
        'nama' : 'Amel',
        'kelas' : 'SI-42-08',
    }
}


print("Data sebelum diubah")
print('-'*20)
for key, value in data_mahasiswa.items():       #
    print('NIM :',key)                          #
    for key2 in value:                          #Menampilkan isi Dictionary dengan perulangan FOR
        print(key2, ':', value[key2])           #
    print()                                     #

#menambahkan nilai_po kepada Amel berdasarkan key-nya yaitu 1202185678
data_mahasiswa['1202185678']['nilai_po']=91.21

#mengubah nilai_po pada Farid berdasarkan key-nya yaitu 1202181234
data_mahasiswa['1202181234']['nilai_po']=81.97

print("Data setelah diubah")
print('-'*20)
for key, value in data_mahasiswa.items():       #
    print('NIM :',key)                          #
    for key2 in value:                          #Menampilkan isi Dictionary dengan perulangan FOR
        print(key2, ':', value[key2])           #
    print()                                     #
```

Referensi

https://kopiding.in/penerapan-*list*-*dictionary*-*tuple*-dan-*set*-pada-*python*/

https://www.codepolitan.com/mengenal-*list*-dan-berbagai-operasinya-di-*python*

https://devtips.glcnetworks.com/macam-macam-removing-konten-*list*-pada-*python*/

https://www.it-swarm.asia/id/*python*/perbedaan-antara-del-hapus-dan-pop-pada-daftar/1068202240/amp/

https://www.*python*indo.com/*tuple*/

https://www.*python*indo.com/*set*/

https://www.*python*indo.com/*dictionary*/

https://snakify.org/en/lessons/sets/