Fakultas Informatika
School of Computing
Telkom University

CII-1A3- PENGENALAN PEMROGRAMAN

PERTEMUAN KE-8

TIPE DATA TERSTRUKTUR

Struktur data

- Arti : Tipe data yang dapat menyimpan **banyak** nilai dan memiliki struktur tertentu
- Tujuan : Untuk menyimpan dan memanipulasi **sejumlah** data
- Built-in Python :
  1) list,
  2) tuple,
  3) set,
  4) dictionary.

LIST

Karakteristik list

- Dapat menyimpan elemen dengan tipe data yang berbeda-beda
- Mempunyai index (posisi) → tersimpan terurut berdasarkan index
- List dan elemen (anggota dari list) bersifat **mutable**
  - List dapat diubah (ditambah, dikurangi)
  - Elemen dapat diganti
- Elemen list boleh redundan (nilai elemen yang sama)

1. Membuat list

<my_list> = [**<nilai>**]
<my_list> = list([**<nilai>**])

a. Pemisah elemen dari list adalah tanda koma

```
list1 = [5,6,7]
print(list1)
```

```
[5, 6, 7]
```

b. Memungkinkan pendefinisian list di dalam list

```
list2 = list([[1,2,3],10])
print(list2)
```

?????

2. Mengakses elemen list

<my_list> [**<index>**]

a. Index berupa integer yang dimulai dari 0

list1 = [5,6,7] → print(**list1**[**0**])

```
5
```

b. Menampilkan elemen terakhir pada list bersarang

list2 = list([[1,2,3],10]) → print(**list2**[**0**][-1])

```
3
```

A. Kesalahan pengaksesan elemen list

- List yang belum didefinisikan
- Index list melebihi jumlah elemen yang terdefinisi di dalam list
- Index tidak bertipe data integer

B. Mencari index jika diketahui nilai

<my_list>.index(**<nilai>**)

- Jika nilai ada di dalam list

list1 = [5,6,7] → print(**list1.**index**(6)**)

```
1
```

- Jika nilai tidak ditemukan pada list

list2 = list([[1,2,3],10]) → print(**list2.**index**(2)**)

```
print(list2.index(2))
ValueError: 2 is not in list
```

C. Cek keanggotaan list

<nilai> in **<my_list>** Return boolean

- Return Boolean
  - True → jika elemen ada di dalam list
  - False → jika elemen tidak ada di dalam list

list1 = [5,6,7] → print(**5** in **list1**) `True`

list2 = list([[1,2,3],10]) → print(**[1,2]** in **list2**) `False`

D. Iterasi elemen list

for **<my_var>** in **<my_list>**:

- Iterasi setiap elemen pada list menggunakan loop

```
list1 = [5,6,7]
for x in list1:
        print(x)
```

```
5
6
7
```

```
list2 = list([[1,2,3],10])
sumY = 0
for y in list2:
        if isinstance (y, int):
                sumY = sumY+y
print(sumY)
```

```
10
```

E. Slice pada list

<my_list>[<start>:<end>]

- Potongan elemen pada list, mengakses beberapa elemen sekaligus dari <start> hingga <end>-1

list1 = [5,6,7]

print(list1[:1])

```
[5]
```

Mulai : paling awal

print(list1[-2:])

```
[6, 7]
```

Hingga : paling akhir

print (list1[0:3])

```
[5, 6, 7]
```

3. Menambah elemen list

A. Sebuah elemen di akhir list

<my_list>.append(**<nilai>**)

list1 = [5,6,7]

list1.append(8)

```
[5, 6, 7, 8]
```

list1.append([1,2,3])

```
[5, 6, 7, [1, 2, 3]]
```

list1.append('hi')

```
[5, 6, 7, 'hi']
```

3. Menambah elemen list

B. Beberapa elemen di akhir list

<my_list>.extend(**<iterable>**)

list1 = [5,6,7]

list1.extend(8)

```
    list1.extend(8)
TypeError: 'int' object is not iterable
```

list1.extend([1,2,3])

```
[5, 6, 7, 1, 2, 3]
```

list1.extend('hi')

```
[5, 6, 7, 'h', 'i']
```

iterable : object yang bisa diakses elemennya satu persatu
(list, str, tuple, dict, file)

3. Menambah elemen list

C. Sebuah elemen pada posisi tertentu

<my_list>.insert(**<index>, <nilai>**)

list1 = [5,6,7]

list1.insert(1,3)

```
[5, 3, 6, 7]
```

list1.insert(-1,3)

```
[5, 6, 3, 7]
```

list1.insert(-2,‘hi’)

```
[5, 'hi', 6, 7]
```

4. Mengubah elemen list

<my_list>[<index>]=<nilai>

- Mutable → re-assignment

list1 = [5,6,7]
list2 = list([[1,2,3],10])

list1=list2
print(list1)

```
[[1, 2, 3], 10]
```

list1[0]=list2
print(list1)

```
[[[1, 2, 3], 10], 6, 7]
```

list1[-2:]=[10]
print(list1)

```
[5, 10]
```

5. Menghapus elemen list

A. Berdasarkan nilai → **sebuah** elemen yang **pertama** kali ditemukan

<my_list>.remove(**<nilai>**)

list2 = list([10,[1,2,3],10])

list2.remove([1,2])
print(list2)

```
    list2.remove([1,2])
ValueError: list.remove(x): x not in list
```

list2.remove(10)
print(list2)

```
[[1, 2, 3], 10]
```

list2.remove(10,10)
print(list2)

```
    list2.remove(10,10)
TypeError: remove() takes exactly one argument (2 given)
```

5. Menghapus elemen list

B. Berdasarkan index → default yaitu index terakhir

<my_list>.pop(**<index>**)

list2 = list([10,[1,2,3],10])

list2.pop(1)
print(list2)

```
[10, 10]
```

z=list2.pop()
print(z,list2)

```
10 [10, [1, 2, 3]]
```

z=list2.pop(3)
print(z,list2)

```
    z=list2.pop(3)
IndexError: pop index out of range
```

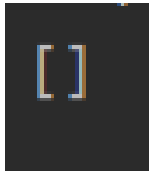
5. Menghapus elemen list

C. Semua elemen

<my_list>.clear()

```
list2 = list([10,[1,2,3],10])

list2.clear()
print(list2)
```



6. *Built-in function* pada list

<my_list>.sort()	Pengurutan -Ascending
<my_list>.sort(reverse=True)	Pengurutan -Descending

- Tipe data dari elemen list harus sama

```
list2 = list([10,[1,2,3],10])
list2.sort()
print(list2)
```

```
    list2.sort()
TypeError: '<' not supported between instances of 'list' and 'int'
```

```
list2 = list([2,22,1,10])
list2.sort(reverse=True)
print(list2)
```

```
[22, 10, 2, 1]
```

6. *Built-in functions* pada list (2)

len(**<my_list>**)	Banyaknya elemen di dalam list

- Tipe data boleh berbeda

```
list2 = list([10,[1,2,3],10])
print('Banyak elemen dari list2= ', len(list2))
```

```
Banyak elemen dari list2= 3
```

max(**<my_list>**)	Nilai tertinggi dari elemen di dalam list
min(**<my_list>**)	Nilai terendah dari elemen di dalam list
sum(**<my_list>**)	Jumlah dari elemen di dalam list

- Tipe data harus integer

```
list2 = list([2,22,1,10])
print(sum(list2),min(list2),max(list2))
```

```
35 1 22
```

TUPLE

Karakteristik tuple

- Dapat menyimpan elemen dengan tipe data yang berbeda-beda
- Mempunyai index (posisi) → tersimpan terurut berdasarkan index
- Elemen tuple bersifat **immutable**
  - Tuple tidak dapat diubah (ditambah, dikurangi)
  - Elemen tidak dapat diganti (kecuali elemennya bersifat mutable → elemen tersebut dapat diganti)
- Elemen tuple boleh redundan (nilai elemen yang sama)

1. Membuat tuple

<my_tuple>=tuple(**<nilai>**)	1 argumen, iterable
<my_tuple>= (**<nilai>,**)	beberapa elemen, tidak iterable

```
tuple1= tuple('halo')
print(tuple1)
```

```
('h', 'a', 'l', 'o')
```

```
tuple2 = ('halo',)
print(type(tuple2), tuple2)
```

```
<class 'tuple'> ('halo',)
```

```
tuple3 = 'halo',(1,2),[1,'saya’]
print(type(tuple3), tuple3)
```

```
<class 'tuple'> ('halo', (1, 2), [1, 'saya'])
```

```
>>> tuple2 = ('halo')
>>> print(type(tuple2), tuple2)
<class 'str'> halo
```

2. Mengakses elemen tuple

- Sama dengan mengakses elemen pada list
  - index, cek keanggotaan, slicing dan iterasi

```
tuple3 = 'halo',(1,2),[1,'saya']
print(tuple3[-1])
```

```
[1, 'saya']
```

```
print(tuple3[1][0])
```

```
1
```

```
print(tuple3[2][0:1])
```

```
[1]
```

```
print(tuple3.index((1,2)))
```

```
1
```

```
print('saya' in tuple3[2])
```

```
True
```

3. Mengubah menambah menghapus

- Tidak bisa dimodifikasi → immutable
- Bisa dimodifikasi hanya pada elemen yang mutable

```
tuple3 = ‘halo’,(1,2),[1,’saya’]
tuple3[2] = [1,2]
print(tuple3)
tuple3[2][0] = [1,2]
print(tuple3)
tuple3[2].insert(0,[1,2,’ok’])
print(tuple3)
del tuple 3
print(tuple3)
```

```
    tuple3[2]=[1,2]
TypeError: 'tuple' object does not support item assignment
```

```
('halo', (1, 2), [[1, 2], 'saya'])
```

```
('halo', (1, 2), [[1, 2, 'ok'], [1, 2], 'saya'])
```

```
   print(tuple3)
NameError: name 'tuple3' is not defined
```

4. *Built-in* function & fitur pada tuple

<my_tuple>.count(**<nilai>**)

- Banyaknya elemen **<nilai>** pada **<my_tuple>**

```
tuple2 = tuple([1,1,2,3,1])
print(tuple2.count(1))
```

```
3
```

- *Double assignment*

```
(a,b) = ((11,12),tuple(['name',1,2]))
print(type(a),a,type(b),b)
```

```
<class 'tuple'> (11, 12) <class 'tuple'> ('name', 1, 2)
```

SET

Karakteristik set

- Dapat menyimpan elemen dengan tipe data yang berbeda-beda **namun elemennya harus immutable**
- Tidak mempunyai index (posisi) → tersimpan tidak memiliki urutan
- Set bersifat mutable
  - Set dapat diubah (ditambah, dikurangi)
  - Elemen tidak dapat diganti
- Elemen set **tidak** boleh redundan (nilai elemen harus unik)

1. Membuat set

<my_set>=set(**<nilai>**)	1 item, iterable
<my_set>= {**<nilai>,**}	beberapa item, tidak iterable

```
set1 = set()
print(type(set1))
```

```
<class 'set'>
```

```
set1 = {}
print(type(set1))
```

```
<class 'dict'>
```

```
set1 = {'halo'}
print(set1)
```

```
{'halo'}
```

```
set1 = set('halo')
print(set1)
```

```
{'o', 'h', 'a', 'l'}
```

```
set2 = set(['halo',(1,2),[1,'saya']])
print(set2)
```

```
    set2 = set(['halo',(1,2),[1,'saya']])
TypeError: unhashable type: 'list'
```

```
set2 = set(['halo',(1,2),1,'saya'])
print(set2)
```

```
{'halo', 1, (1, 2), 'saya'}
```

2. Mengakses elemen set

- Cek keanggotaan dan iterasi
  - Tidak memiliki index

```
set1 = set('halo')
print('h' in set1)
```

```
True
```

```
set2 = set(['halo',(1,2),1,'saya'])
for x in set2:
print(x)
```

```
1
(1, 2)
saya
halo
```

3. Menambah elemen set

<my_set>.add(**<nilai>**)	1 item , tidak iterable
<my_set>. update(**<nilai>**)	Beberapa item, iterable

- Elemen yang dientry harus **immutable** (~~list~~)

```
set1 = set('halo')
set1.add('here')
print(set1)
```

```
{'h', 'l', 'here', 'a', 'o'}
```

```
set2 = set(['halo',(1,2),1,'saya'])
set2.update((1,4),'oke')
print(set2)
```

```
{'e', 1, 3, (1, 2), 4, 'k', 'halo', 'o', 'saya'}
```

```
set1.add(tuple([2,5]))
print(set1)
```

```
{'l', 'h', 'here', 'a', 'o', (2, 5)}
```

4. Mengubah elemen set

- Set bersifat mutable
- Elemen set bersifat immutable
  - Elemen tidak bisa diupdate / diubah

5. Menghapus elemen set

A. Berdasarkan nilai → jika tidak ditemukan menimbulkan exception

<my_set>.remove(**<nilai>**)

```
set1 = set('halo')
set1.remove('o')
print(set1)
```

```
{'l', 'h', 'a'}
```

```
set2 = {'halo'}
try:
    set2.remove('o')
except Exception as
er:
    print('error: ',er)
```

```
error:  'o'
```

5. Menghapus elemen set

B. Berdasarkan nilai → jika tidak ditemukan **tidak** menimbulkan exception

<my_set>.discard(**<nilai>**)

```
set1 = set('halo')
set1.discard('o')
print(set1)
```

```
{'l', 'a', 'h'}
```

```
set2 = {'halo'}
set2.discard('o')
print(set2)
```

```
{'halo'}
```

5. Menghapus elemen set

C. 1 elemen acak & mengembalikan nilai yang dihapus

<my_set>.pop()

```
set1 = set('halo')
d = set1.pop()
print(d,', isi set = ', set1)
```

```
o , isi set =  {'a', 'l', 'h'}
```

DICTIONARY

Karakteristik dictionary

- Tersimpan dalam bentuk pasangan **key** dan **value/nilai**
  - **Key** = harus immutable dan unik
  - **Value/nilai** = mutable maupun immutable
- Tidak memiliki index → tidak ada urutan tertentu
- Mutable
  - Bisa ditambah atau dikurangi

1. Membuat dictionary

<my_dict> = {**<key1>** :**<nilai1>**, **<key2>** :**<nilai2>**,…}
<my_dict> = dict([(**<key1>**,**<nilai1>**),(**<key2>**,**<nilai2>**),…])

```
dict1={'k1':255,'k2':311,'k3':111}
print(dict1)
```

```
{'k1': 255, 'k2': 311, 'k3': 111}
```

```
dict2 = dict([(1,12),(5,60),(1,12)])
print(dict2)
```

```
{1: 12, 5: 60}
```

```
dict1={['k1']:255,['k2']:311,['k3']:111}
print(dict1)
```

```
    dict1={['k1']:255,'k2':311,'k3':111}
TypeError: unhashable type: 'list'
```

```
dict1={'k1':255,{'k2',1}:[311,122]}
print(dict1)
```

```
    dict1={'k1':255,{'k2',1}:[311,122]}
TypeError: unhashable type: 'set'
```

Set → mutable

2. Mengakses elemen dictionary

- A. Akses dengan key

<my_dict>[**<key>**]	Jika **<key>** tidak ada → exception
<my_dict>.get(**<key>**)	Jika **<key>** tidak ada → return **<None>**

```
dict1={'k1':255,'k2':[311,122],'k3':111}
print(dict1['k2’][0])
```

```
311
```

```
print(dict1.get('k2’)[1])
```

```
122
```

```
print(dict1[‘k5’])
```

```
    print(dict1['k5'])
KeyError: 'k5'
```

2. Mengakses elemen dictionary

- B. Cek keanggotaan dictionary

<key> in **<my_dict>**	Boolean

```
dict1={'k1':255, ('k2',1):[311,122],'k3':111}
print('k2' in dict1)
```

```
False
```

```
print(tuple(['k2',1]) in dict1)
```

```
True
```

```
print(('k2',1) in dict1)
```

```
True
```

2. Mengakses elemen dictionary

- C. Iterasi elemen dictionary

for **<my_key>** in **<my_dict>**:
for **<my_key>**, **<my_val>** in **<my_dict>**.items():

```
dict1={'k1':255, ('k2',1):[311,122],'k3':111}
for x in dict1:
    print(x)
```

```
k1
('k2', 1)
k3
```

```
for x,y in dict1.items():
    print(x,' bernilai ',y)
```

```
k1  bernilai  255
('k2', 1)  bernilai  [311, 122]
k3  bernilai  111
```

3. Menambah elemen & mengganti nilai

<my_dict>[<key>] = <nilai>

- Jika **<key>** belum ada
  - penambahan sebagai elemen baru
- Jika **<key>** sudah ada
  - Update / ganti dengan **<nilai>** yang baru

```
dict1={'k1':255, ('k2',1):[311,122],'k3':111}
dict1['k2']='saya'
print(dict1)
```

```
{'k1': 255, ('k2', 1): [311, 122], 'k3': 111, 'k2': 'saya'}
```

```
dict1['k1']='saya'
print(dict1)
```

```
{'k1': 'saya', ('k2', 1): [311, 122], 'k3': 111}
```

4. Menghapus elemen dictionary

A. Berdasarkan key tertentu

<my_dict>.pop(**<key>**)

Jika **<key>** tidak ada → exception

```
dict1={'k1':255, ('k2',1):[311,122],'k3':111}
dict1.pop('k1')
print(dict1)
```

```
{('k2', 1): [311, 122], 'k3': 111}
```

```
dict1.pop('k2')
print(dict1)
```

```
    dict1.pop('k2')
KeyError: 'k2'
```

4. Menghapus elemen dictionary

- B. Sebuah elemen acak

<my_dict>.popitem()

- Jika **<my_dict>** kosong → exception
- Return item yang dihapus

```
dict1={'k1':255, ('k2',1):[311,122],'k3':111}
temp = dict1.popitem()
print(temp,' dihapus dari ',dict1)
```

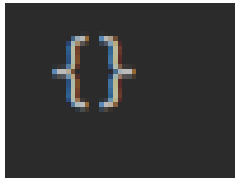
```
('k3', 111)  dihapus dari  {'k1': 255, ('k2', 1): [311, 122]}
```

4. Menghapus elemen dictionary

C. Semua elemen

<my_dict>.clear()

```
dict1={'k1':255,
('k2',1):[311,122],'k3':111}
dict1.clear()
print(dict1)
```



5. *Built-in function* pada dictionary

<my_dict> = {}.fromkeys([**<key>**], **<nilai_default>**)

- Pembuatan dictionary baru dengan deretan kunci [**<key>**] dan memiliki nilai default **<nilai_default>** pada semua **<key>**
- Jika **<nilai_default>** tidak diisi maka otomatis akan bernilai None

```
dict1={}.fromkeys([1,2,3],'ok')
print(dict1)
```

```
{1: 'ok', 2: 'ok', 3: 'ok'}
```

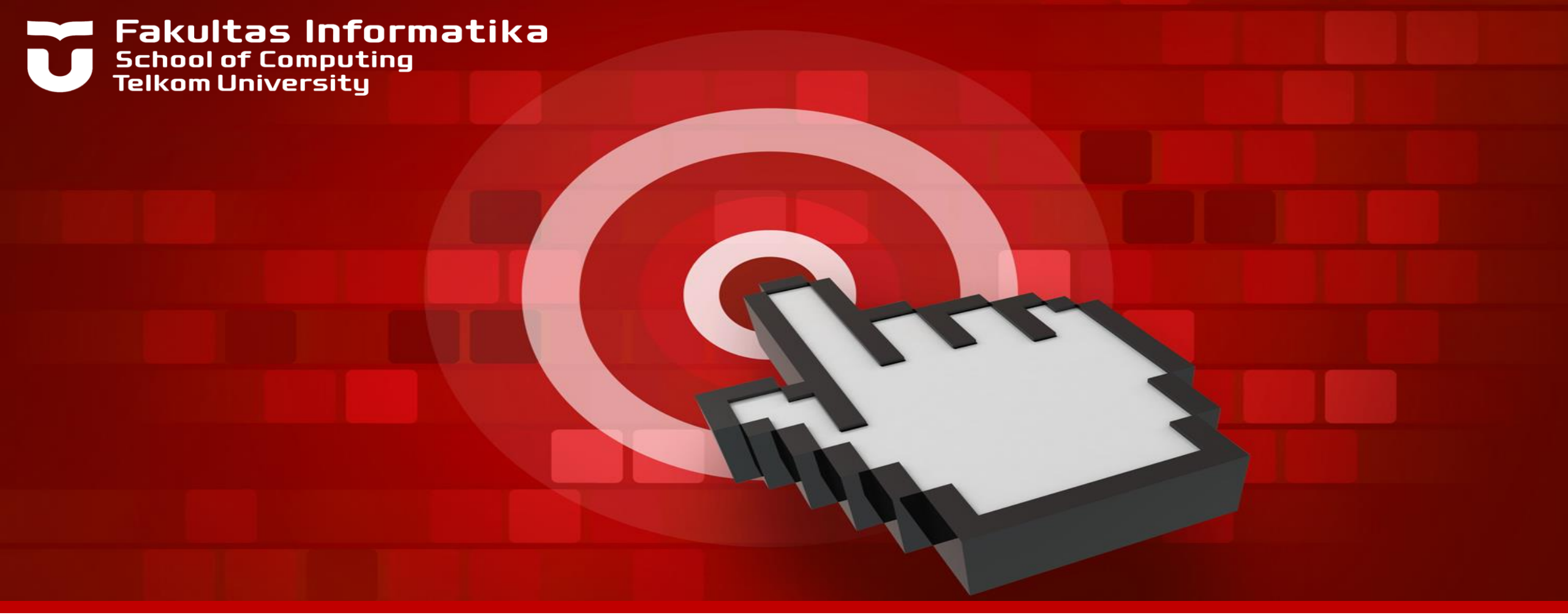
```
dict1={}.fromkeys([1,2,3],)
print(dict1)
```

```
{1: None, 2: None, 3: None}
```

Rangkuman

- List : [1,2,3,…] atau list([1,2,3,..])
- Tuple : (1,2,3,…) atau tuple([1,2,3,…])
- Set : {1,2,3,…} atau set([1,2,3,…])
- Dictionary : {1:’v1’ , 2:’v2’, 3:’v3’,…} atau dict([(1,’v1’),(2,’v2’),(3,’v3’,…)])

Fakultas Informatika
School of Computing
Telkom University



THANK YOU