MINGGU: 4

(1). LIST pada Python

List adalah struktur data pada python yang mampu menyimpan lebih dari satu data, seperti array

A. Cara Membuat List di Python

List dapat kita buat seperti membuat variabel biasa, namun nilai variabelnya diisi dengan tanda kurung siku (`[]`).

Contoh:

```
# Membuat List kosong
warna = []

# Membuat list dengan isi 1 item
hobi = ["membaca"]
```

Apabila list-nya memiliki lebih dari satu isi, maka kita bisa memisahnya dengan tanda koma.

Contoh:

```
buah = ["jeruk", "apel", "mangga", "duren"]
```

Jenis data apa saja yang boleh diisi ke dalam List?

list dapat diisi dengan tipe data apa saja, string, integer, float, double, boolean, object, dan sebagainya.

Kita juga bisa mencampur isinya.

Contoh:

```
laci = ["buku", 21, True, 34.12]
```

Ada empat jenis tipe data pada list `laci`:

1. `"buku"` adalah tipe data string;
2. `21` adalah tipe data integer;
3. `True` adalah tipe data boolean;
4. dan `34.12` adalah tipe data float.

B. Cara Mengambil Nilai dari List

Setelah kita tahu cara membuat dan menyimpan data di dalam *List*, mari kita coba mengambil datanya.

List sama seperti array, list juga memiliki nomer indeks untuk mengakses data atau isinya.

Nomer indeks *list* selalu dimulai dari nol (`0`).

Nomer indeks ini yang kita butuhkan untuk mengambil isi (item) dari *list*.

Contoh:

```
# Kita punya list nama-nama buah
buah = ["apel", "anggur", "mangga", "jeruk"]

# Misanya kita ingin mengambil mangga
# Maka indeknya adalah 2
print (buah[2])
```

Akan menghasilkan output:

```
"mangga"
```

Latihan 1: Membuat Program dengan List

Untuk memantapkan pemahaman, silahkan coba latihan berikut.

1. Buat sebuah list untuk menyimpan kenalanmu
2. Isi list sebanyak 5
3. Tampilkan isi list indeks nomer 3
4. Tampilkan semua teman dengan perulangan
5. Tampilkan panjang list

Mari kita coba…

```
# Buat list untuk menampung nama-nama teman
my_friends = ["Anggun", "Dian", "Agung", "Adi", "Adam"]

# Tampilkan isi list my_friends dengan nomer indeks 3
print ("Isi my_friends indeks ke-3 adalah: {}".format(my_friends[3]))

# Tampilkan semua daftar teman
print ("Semua teman: ada {} orang".format(len(my_friends)))
for friend in my_friends:
    print (friend)
```

Pada kode di atas, kita menggunakan fungsi `len()` untuk mengambil panjang *list*.

Hasil outputnya:

```
Isi my_friends indeks ke-3 adalah: Adi
Semua teman: ada 5 orang
Anggun
Dian
Agung
Adi
Adam
```

C. Mengganti Nilai List

List bersifat *mutable*, artinya isinya bisa kita ubah-ubah.

Contoh:

```
# list mula-mula
buah = ["jeruk", "apel", "mangga", "duren"]
# mengubah nilai index ke-2
buah[2] = "kelapa"
print(buah)
```

Maka `"mangga"` akan diganti dengan `"kelapa"`.

```
["jeruk", "apel", "kelapa", "duren"]
```

D. Menambahkan Item List

Metode *(method)* atau fungsi yang bisa digunakan untuk menambahkan isi atau item ke List:

1. `append(item)` menambahkan item dari belakang.
2. `insert(index, item)` menambahkan item dari indeks tertentu

Contoh:

```
#list mula-mula
buah = ["jeruk", "apel", "mangga", "duren"]
# Tambahkan manggis
buah.append("manggis")
print(buah)
```

Hasilnya `"manggis"` akan ditambahkan setelah item terakhir.

```
["jeruk", "apel", "mangga", "duren", "manggis"]
```

Metode yang kedua menggunakan *method* `insert()` untuk menambahkan item pada indeks tertentu.

Contoh:

```
#list mula-mula
buah = ["jeruk", "apel", "mangga", "duren"]
buah.insert(2, "duren")
print(buah)
["jeruk", "apel", "mangga", "duren", "manggis"]
```

Hasilnya `"duren"` akan ditambahkan setelah mangga.

```
["jeruk", "apel", "mangga", "duren", "manggis"]
```

Latihan 2: Membuat Program dengan List

Sekarang mari kita coba membuat program dengan memanfaatkan *method* `prepend()` dan `append()`.

Silahkan langsung di ketik dan dicoba.

```
# Membuat list kosong untuk menampung hobi
hobi = []
stop = False
i = 0

# Mengisi hobi
while(not stop):
    hobi_baru = input("Inputkan hobi yang ke-{}: ".format(i))
    hobi.append(hobi_baru)

    # Increment i
    i += 1

    tanya = input("Mau isi lagi? (y/t): ")
    if(tanya == "t"):
        stop = True


# Cetak Semua Hobi
print ("=" * 10)
print ("Kamu memiliki {} hobi".format(len(hobi)))
for hb in hobi:
    print ("- {}".format(hb))
```

Coba eksekusi dan inputkan sebuah nilai.

```
Inputkan hobi yang ke-0: mincing
Mau isi lagi? (y/t): y
Inputkan hobi yang ke-1: baca
Mau isi lagi? (y/t): y
Inputkan hobi yang ke-2: computer
Mau isi lagi? (y/t): t
==========

Kamu memiliki 3 hobi
- mincing
- baca
- komputer
```

E. Menghapus Item di List

Untuk menghapus salah satu isi dari *List*, kita bisa menggunakan perintah `del`.

Perintah `del` akan menghapus sebuah variabel dari memori.

Contoh:

```
# Membuat List
todo_list = [
    "Balajar Python",
    "Belajar Java",
    "Belajar Android",
    "Belajar Database",
    "Belajar Web"
]

# Misalkan kita ingin menghapus "Belajar Database"
# yang berada di indeks ke-3
del todo_list[3]

print (todo_list)
```

Hasilnya, `"Belajar Database"` akan dihapus:

```
['Balajar Python', 'Belajar Java', 'Belajar Android', 'Belajar Web']
```

Selain menggunakan perintah `del`, kita juga bisa menggunakan *method* `remove()` dengan paramter item yang akan dihapus.

Contoh:

```
# mula-mula kita punya list
a = ["a", "b", "c", "d"]
# kemudian kita hapus b
a.remove("b")

print (a)
```

Hasilnya:

```
["a", "c", "d"]
```

F. Memotong list

Seperti string, list juga dapat dipotong-potong.

Contoh:

```
# Kita punya list warna
warna = ["merah", "hijau", "kuning", "biru", "pink", "ungu"]

# Kita potong dari indeks ke-1 sampai ke-5
print (warna[1:5])
```

Hasilnya:

```
['hijau', 'kuning', 'biru', 'pink']
```

G. Operasi List

Ada beberapa operasi yang bisa dilakukan terhadap List, diantaranya:

- Penggabungan (+)
- Perkalian (*)

Contoh:

```
# Beberapa list lagu
list_lagu = [
    "No Women, No Cry",
    "Dear God"
]

# playlist lagu favorit
playlist_favorit = [
    "Break Out",
    "Now Loading!!!"
```

```
]

# Mari kita gabungkan keduanya
semua_lagu = list_lagu + playlist_favorit

print (semua_lagu)
```

Hasilnya:

```
['No Women, No Cry', 'Dear God', 'Break Out', 'Now Loading!!!']
```

Sedangkan untuk operasi perkalian hanya dapat dilakukan dengan bilangan.

Contoh:

```
# playlist lagu favorit
playlist_favorit = [
    "Break Out",
    "Now Loading!!!"
]

# ulangi sebanyak 5x
ulangi = 5

now_playing = playlist_favorit * ulangi

print (now_playing)
```

Hasilnya:

```
['Break Out', 'Now Loading!!!', 'Break Out', 'Now Loading!!!', 'Break Out',
'Now Loading!!!', 'Break Out', 'Now Loading!!!', 'Break Out', 'Now
Loading!!!']
```

H. List Multi Dimensi

Pada contoh-contoh di atas, kita hanya membuat list satu dimensi saja.

List dapat juga memiliki lebih dari satu dimensi atau disebut dengan multi dimensi.

List multi dimensi biasanya digunakan untuk menyimpan struktur data yang kompleks seperti tabel, matriks, graph, tree, dsb.

Contoh:

```
# List minuman dengan 2 dimensi
list_minuman = [
    ["Kopi", "Susu", "Teh"],
    ["Jus Apel", "Jus Melon", "Jus Jeruk"],
    ["Es Kopi", "Es Campur", "Es Teler"]
]

# Cara mengakses list multidimensi
# misalkan kita ingin mengambil "es kopi"
print (list_minuman[2][0])
```

Angka dua `2` pada kode di atas, menujukan indeks list yang akan kita akses. Kemudian setelah dapat list-nya baru kita ambil isinya.

Hasil outputnya:

```
"Es Kopi"
```

Bagaimana kalau kita ingin menampilkan semua isi dalam list multi dimensi?

Gampang…

Tinggal gunakan perulangan bersarang.

```
# List minuman dengan 2 dimensi
list_minuman = [
    ["Kopi", "Susu", "Teh"],
    ["Jus Apel", "Jus Melon", "Jus Jeruk"],
    ["Es Kopi", "Es Campur", "Es Teler"]
]

for menu in list_minuman:
    for minuman in menu:
        print (minuman)
```

Hasilnya:

```
Kopi
Susu
Teh
Jus Apel
Jus Melon
Jus Jeruk
Es Kopi
Es Campur
Es Teler
```

(2). TUPLE pada Python

Sebuah tupel adalah urutan objek Python yang tidak berubah. Tupel adalah urutan, seperti daftar. Perbedaan utama antara tupel dan daftarnya adalah bahwa tupel tidak dapat diubah tidak seperti List Python. Tupel menggunakan tanda kurung, sedangkan List Python menggunakan tanda kurung siku.

Membuat tuple semudah memasukkan nilai-nilai yang dipisahkan koma. Secara opsional, Anda dapat memasukkan nilai-nilai yang dipisahkan koma ini di antara tanda kurung juga. Sebagai contoh :

```
#Contoh sederhana pembuatan tuple pada bahasa pemrograman python

tup1 = ('fisika', 'kimia', 1993, 2017)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
```

Tupel kosong ditulis sebagai dua tanda kurung yang tidak berisi apa-apa, contohnya : tup1 = (); Untuk menulis tupel yang berisi satu nilai, Anda harus memasukkan koma, meskipun hanya ada satu nilai, contohnya : tup1 = (50,) Seperti indeks String, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya

A. Akses Nilai Dalam Tuple Python

Untuk mengakses nilai dalam tupel, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Sebagai contoh :

```
#Cara mengakses nilai tuple

tup1 = ('fisika', 'kimia', 1993, 2017)
tup2 = (1, 2, 3, 4, 5, 6, 7 )

print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])
```

Setelah Anda mengeksekusi kode diatas, hasilnya akan seperti dibawah ini :

```
tup1[0]: fisika
tup2[1:5]: (2, 3, 4, 5)
```

B. Update Nilai Dalam Tuple Python

Tuple tidak berubah, yang berarti Anda tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru seperti ditunjukkan oleh contoh berikut.

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

# Aksi seperti dibawah ini tidak bisa dilakukan pada tuple python
# Karena memang nilai pada tuple python tidak bisa diubah
# tup1[0] = 100;

# Jadi, buatlah tuple baru sebagai berikut
tup3 = tup1 + tup2
print (tup3)
```

Hasilnya

```
(12, 34.56, 'abc', 'xyz')
```

C. Hapus Nilai Dalam Tuple Python

Menghapus elemen tuple individual tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tupel lain dengan unsur-unsur yang tidak diinginkan dibuang.

Untuk secara eksplisit menghapus keseluruhan tuple, cukup gunakan del statement. Sebagai contoh

```
tup = ('fisika', 'kimia', 1993, 2017)
print(tup)

# hapus tuple dengan statement del
del tup

# lalu buat kembali tuple yang baru dengan elemen yang diinginkan
tup = ('Bahasa', 'Literasi', 2020)
print("Setelah menghapus tuple :", tup)
```

Hasilnya

```
('fisika', 'kimia', 1993, 2017)
Setelah menghapus tuple : ('Bahasa', 'Literasi', 2020)
```

D. Operasi Dasar Pada Tuple Python

Tupel merespons operator + dan * sama seperti String; Mereka berarti penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah tupel baru, bukan string.

Sebenarnya, Tuple merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada Tuple python

Python Expression	Hasil	Penjelasan
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
(‘Halo!’,) * 4	(‘Halo!’, ‘Halo!’, ‘Halo!’, ‘Halo!’)	Repetition

3 in (1, 2, 3)	True	Membership
for x in (1,2,3) : print (x, end = ‘ ‘)	1 2 3	Iteration

E. Indexing, Slicing dan Matrix Pada Tuple Python

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti pada String, dengan asumsi masukan berikut

Dengan asumsi input berikut : `T = ('C++', 'Java', 'Python')`

Python Expression	Hasil	Penjelasan
`T[2]`	`'Python'`	Offset mulai dari nol
`T[-2]`	`'Java'`	Negatif: hitung dari kanan
`T[1:]`	`('Java', 'Python')`	Slicing mengambil bagian

F. Fungsi Build-in Pada Tuple Python

Python menyertakan fungsi built-in sebagai berikut

Python Function	Penjelasan
`cmp(tuple1, tuple2)`	# Tidak lagi tersedia dengan Python 3
`len(tuple)`	Memberikan total panjang tuple.
`max(tuple)`	Mengembalikan item dari tuple dengan nilai maks.
`min(tuple)`	Mengembalikan item dari tuple dengan nilai min.
`tuple(seq)`	Mengubah seq menjadi tuple.

(3). DICTIONARY pada Python

A. Apa itu Dictionary pada Python?

Dictionary adalah stuktur data yang bentuknya seperti kamus. Ada kata **kunci** kemudian ada **nilaninya**. Kata kunci harus unik, sedangkan nilai boleh diisi denga apa saja.

Contoh:

```
aku = {
    "nama": "Petani Kode",
    "url:" "https://www.ppnp.ac.id"
}
```

Pada contoh di atas kita membuat sebuah *Dictionary* bernama `aku` dengan isi data nama dan URL. `nama` dan `url` adalah kunci *(key)* yang akan kita gunakan untuk mengakses nilai di dalamnya.

Inilah perbedaanya dibandingkan *list* dan *tuple*. *Dictionary* memiliki kunci berupa teks—bisa juga angka—sedangkan *list* dan *tuple* menggunakan indeks berupa angka saja untuk mengakses nilainya.

Dalam bahasa pemrograman lain (seperti PHP), *Dictionary* juga dikenal dengan sebutan **asosiatif array**.

B. Membuat Dictionary

Hal yang wajib ada di dalam pembuatan *Dictionary* adalah:

- nama dictionary,
- *key*,
- *value*,
- buka dan tutupnya menggunakan kurung kurawal.

Antara key dan value dipisah dengan titik dua (`:`) dan apabila terdapat lebih dari satu item, maka dipisah dengan tanda koma (`,`).

Contoh satu item:

```
nama_dict = {
    "key": "value"
}
```

Contoh tiga item:

```
nama_dict = {
```

```
    "key1": "value",
    "key2": "value",
    "key3": "value"
}
```

Isi dari *Dictionary* dapat berupa:

- String
- Integer
- Objek
- List
- Tuple
- Dictionary
- dsb.

Contoh:

```
pak_tani = {
    "nama": "Petani Kode",
    "umur": 22,
    "hobi": ["coding", "membaca", "cocok tanam"],
    "menikah": False,
    "sosmed": {
        "facebook": "petanikode",
        "twitter": "@petanikode"
    }
}
```

Mari kita lihat isi dari *Dictionary* di atas:

- `nama` berisi string `"Petani Kode"`
- `umur` berisi integer `22`
- `hobi` berisi list dari string
- `menikah` berisi boolean `False`
- dan `sosmed` berisi Dictionary

B. Menggunakan Konstruktor

Selain menggunakan cara di atas, kita juga bisa membuat *Dictionary* dari constructor `dict()` dengan parameter *key* dan *value*.

Contoh:

```
warna_buah = dict(jeruk="orange", apel="merah", pisang="kuning")
```

Maka akan menghasilkan *dictionary* seperti ini:

```
{'jeruk': 'orange', 'pisang': 'kuning', 'apel': 'merah'}
```

C. Mengakses Nilai Item dari Dictionary

Kita sudah tahu cara membuat *Dictionary*, sekarang bagaimana cara mengaksesnya?

Cara mengaksesnya sama seperti *list*. Namaun kunci yang digunakan bukan angka, melainkan *keyword* yang sudah kita tentukan di dalam *Dictionary*-nya.

Contoh:

```
# Membuat Dictionary
pak_tani = {
    "nama": "Petani Kode",
    "umur": 22,
    "hobi": ["coding", "membaca", "cocok tanam"],
    "menikah": False,
    "sosmed": {
        "facebook": "petanikode",
        "twitter": "@petanikode"
    }
}

# Mengakses isi dictionary
print("Nama saya adalah %s" % pak_tani["nama"])
print("Twitter: %s" % pak_tani["sosmed"]["twitter"])
```

Maka akan menghasilkan:

```
Nama saya adalah Petani Kode
Twitter: @petanikode
```

Selain dengan cara di atas, kita juga bisa mengambil nilai *Dictionary* dengan *method* `get()`.

Contoh:

```
print(pak_tani.get("nama"))
```

Hasilnya:

```
Petani Kode
```

D. Menggunakan Perulangan

Untuk mencetak semua isi *Dictionary*, kita bisa menggunakan perulangan seperti ini:

```
# Membuat dictionary
web = {
    "name": "petanikode",
    "url": "https://www.ppnp.ac.id",
    "rank": "5"
}
```

```
# Mencetak isi dictionary dengan perulangan
for key in web:
    print(web[key])
```

Hasilnya:

```
petanikode
5
https://www.ppnp.ac.id
```

Kita juga bisa melakukannya seperti ini:

```
web = {
    "name": "petanikode",
    "url": "https://www.ppnp.ac.id",
    "rank": "5"
}

for key, val in web.items():
    print("%s : %s" % (key, val))
```

Hasilnya:

```
name : petanikode
rank : 5
url : https://www.ppnp.ac.id
```

D. Mengubah Nilai Item Dictionary

Dictionary bersifat *mutable*, artinya nilainya dapat kita ubah-ubah. Untuk mengubah nilai *Dictionary*, kita bisa lakukan seperti ini:

```
nama_dic["kunci"] = "Nilai Baru"
```

Contoh:

```
# membuat dictioanary
skill = {
    "utama": "Python",
    "lainnya": ["PHP","Java", "HTML"]
}

# Mencetak isi skill utama
print(skill["utama"])

# mengubah isi skill utama
skill["utama"] = "Rust"

# Mencetak isi skill utama
print(skill["utama"])
```

Maka akan menghasilkan:

```
Python
Rust
```

E. Menghapus Item dari Dictionary

Untuk menghapus nilai *Dictionary*, kita bisa menggunakan perintah `del` dan method `pop()`.

Method `pop()` adalah *method* yang berfungsi untuk mengeluarkan item dari *dictionary* sedangak fungsi `del` adalah fungsi untuk menghapus suatu variabel dari memori.

Contoh menghapus dengan `del`:

```
del skill["utama"]
skill
{'lainnya': ['PHP', 'Java', 'HTML']}
```

Contoh menghapus dengan method `pop()`:

```
skill.pop("utama")
'Rust'
skill
{'lainnya': ['PHP', 'Java', 'HTML']}
```

…atau bila kita ingin menghapus semuanya sekaligus, kita bisa menggunakan *method* `clear()`.

Contoh:

```
skill.clear()
```

F. Menambahkan Item ke Dictionary

Kita bisa menggunakan method `update()` untuk menambahkan isi ke Dictionary. Parameternya berupa *Dictionary*.

Selain berfungsi untuk menambahkan, method ini juga berfungsi untuk mengubah nilai dictionary apabila kunci yang dimasukkan sudah ada di dalamnya.

Contoh:

```
# membuat dictionary user
user = {
    "name": "petanikode"
}
```

```
# menambahkan password
user.update({"password": "akucintakamu123"})

print(user)

# update name
user.update({"name": "peternaklinux"})

print(user)
```

Hasilnya:

```
{'name': 'petanikode', 'password': 'akucintakamu123'}
{'name': 'peternaklinux', 'password': 'akucintakamu123'}
```

G. Mengambil Panjang Dictionary

Untuk mengambil jumlah data atau panjang *Dictionary*, kita bisa menggunakan fungsi `len()`.

Contoh:

```
# membuat dictonary
books = {
    "python": "Menguasai Python dalam 2028 jam",
    "java": "Tutorial Belajar untuk Pemula",
    "php": "Membuat aplikasi web dengan PHP"
}

# mencetak jumlah data yang ada di dalam dictionary
print("total buku: %d" % len(books))
```

Hasilnya:

```
total buku: 3
```