BAB 4

Struktur Data dan Fungsi

4.1 Struktur Data List

List adalah struktur data pada python yang mampu menyimpan lebih dari satu data, seperti array. List dapat dibuat seperti membuat variabel biasa, namun nilai variabelnya diisi dengan tanda kurung siku ([]).Llist dapat diisi dengan tipe data apa saja, string, integer, float, double, boolean, object, dan sebagainya.

Contoh:

```
# Membuat List kosong
jenis = []
# Membuat list dengan isi 1 item
item = ["buku"]
```

Apabila list-nya memiliki lebih dari satu isi, maka dapat dipisah dengan tanda koma.

Contoh:

```
item = ["buku", "hp", "laptop", "komputer"]
```

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 dibutuhkan untuk mengambil isi (item) dari list.

Contoh:

```
item = ["buku", "hp", "laptop", "komputer"]
# Misalnya kita ingin mengambil laptop
# Maka indeksnya adalah 2
print(item[2])
```

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

Contoh:

```
#list awal
item = ["buku", "hp", "laptop", "komputer"]
# mengubah nilai index ke-2
item[2] = "mouse"
```

Maka "laptop" akan diganti dengan "mouse".

Terdapat dua metode (method) atau fungsi yang bisa digunakan untuk menambahkan isi atau item ke List:

1. append(item) menambahkan item dari belakang.

   Contoh:

   ```
   #list awal
   item = ["buku", "hp", "laptop", "komputer"]
   # menambah charger ke list
   item.append("charger")
   ```

   Hasilnya "charger" akan ditambahkan setelah item terakhir.

2. insert(index, item) menambahkan item dari indeks tertentu

   Contoh:

   ```
   #list awal
   item = ["buku", "hp", "laptop", "komputer"]
   # menambah charger ke list
   item.insert(2,"charger")
   ```

   Hasilnya "charger" akan ditambahkan setelah item hp..

Untuk menghapus salah satu isi dari List, kita bisa menggunakan perintah ‘del()’.

Contoh:

```
#list awal
item = ["buku", "hp", "laptop", "komputer"]
# menghapus isi indeks-2 yaitu laptop
del(item[2])
```

Hasilnya, "Laptop" akan dihapus.

Selain menggunakan perintah ‘del()’, dapat menggunakan method ‘remove()’ dengan parameter item yang akan dihapus.

Contoh:

```
#list awal
item = ["buku", "hp", "laptop", "komputer"]
# menghapus item pada list yaitu laptop
item.remove("laptop")
```

Seperti string, list juga dapat dipotong-potong.

Contoh:

```
#list awal
item = ["buku", "hp", "laptop", "komputer"]
# Kita potong dari indeks ke-1 sampai ke-3
print(item[1:3])
```

Hasilnya, ['hp', 'laptop'].

Ada beberapa operasi yang bisa dilakukan terhadap List, diantaranya:

1. Penggabungan (+)

   Contoh:

   ```
   # Item 1
   item = ["buku", "hp", "laptop", "komputer"]
   # Item 2
   item_terpakai = ['Sepeda', "Mobil", "Motor"]
   # Mari kita gabungkan keduanya
   semua_item = item + item_terpakai
   print(semua_item)
   ```

2. Perkalian (*)

   Operasi perkalian hanya dapat dilakukan dengan bilangan.

   Contoh:

   ```
   # Item
   item = ["buku", "hp", "laptop", "komputer"]
   # Penggandaan
   gandakan = 5
   # Mari kita gandakan
   item_tergandakan = item * gandakan
   print(item_tergandakan)
   ```

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, dan lain-lain.

Contoh:

```
List_item = [
            ["buku", "hp", "laptop", "komputer"],
            ["Sepeda", "Motor", "Bis"],
            ["Penghapus", "Penggaris", "Kereta"]
]
print(List_item[0][2])
```

Hasilnya, “Laptop”

4.2 Struktur Data Tuple

Tuple dalam Python adalah stuktur data yang digunakan untuk menyimpan sekumpulan data. Tupe bersifat immutable, artinya isi tuple tidak bisa kita ubah dan hapus. Namun, dapat kita isi dengan berbagai macam nilai dan objek.

Tuple biasanya dibuat dengan tanda kurung seperti ini:

```
tuple = (1234, 4321, 'Halo')
```

atau, tanpa tanda kurung seperti ini:

```
tuple = 1234, 4321, 'Halo'
```

jika ingin membuat sebuah tuple tanpa isi, kita bisa menuliskannya seperti ini:

```
# Membuat tuple kosong
kosong = ()
```

Untuk membuat Tuple yang hanya berisi satu (singleton), maka kita harus manambahkan tanda koma di belakangnnya.

Contoh:

```
# membuat tuple
tuple1 = ('Isi',)
tuple2 = "isi sama",
```

Sama seperti List, Tuple juga memiliki indeks untuk Mengakses item di dalamnya. Indeks Tuple dan list selalu dimulai dari nol 0. Di Tuple juga kita bisa melakukan *slicing* atau memotong. Cara untuk mengakses item dan memotong sama dengan cara pada struktur data List.

Untuk mengambil panjang atau jumlah item di dalam Tuple, kita bisa menggunakan fungsi ‘len()’.

Contoh:

```
# Membuat Tuple
hari = ('Senin', 'Selasa', 'Rabu', 'Kamis', 'Jum\'at', 'Sabtu',
'Minggu')

# Mengambil panjang tuple hari
print("Jumlah hari: %d" % len(hari))
```

Tuple juga dapat di nested, artinya Tuple bisa diisi dengan Tuple.

Contoh:

```
tuple1 = "Halo", "aku", "ngecrushin"
tuple2 = "kamu", "dari", "dulu"
tuple3 = (tuple1, tuple2)  # <- nested tuple
```

Tuple juga bisa diisi dengan objek apapun seperti list, dictionary, object, dan lain-lain.

Proses pembuatan Tuple bisa disebut sebagai *packing*, sementara untuk mengambil (ekstrak) seluruh isinya disebut *unpacking*.

Contoh:

```
# mula-mula kita buat tuple seperti ini
web = 123, "HaloGit", "https://www.halogit.com"

# lalu di-unpacking
id_web, nama, url = web

# maka sekarang tiga variabel tersebut akan bernilai
# sesuai yang ada di dalam tuple
# mari kita cetak
print(id_web)
print(nama)
print(url)
```

Dengan melakukan *unpacking*, isi tuple akan di-*copy* ke variabel. Lalu dengan variabel kita bisa melakukan apapun, seperti mengubah isinya. Karena variabel bersifat *mutable*.

4.3 Struktur Data Dictionary

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

Contoh:

```
aku = {
   "nama": "Iqbal",
   "hobi": "mencintai"
}
```

Hal yang wajib ada di dalam pembuatan Dictionary adalah:

1. nama dictionary
2. key
3. value
4. buka dan tutupnya menggunakan kurung kurawal
5. Antara key dan value dipisah dengan titik dua (:) dan apabila terdapat lebih dari satu item, maka dipisah dengan tanda koma (,)

Contoh:

```
nama_dict = {
   "key1": "value",
   "key2": "value",
   "key3": "value"
}
```

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

Contoh:

```
item_kuliah = dict(laptop="Asus", tas="seadanya", sepatu="hasil
looting")
```

Cara mengaksesnya item dari dictionary sama seperti list. Nama kunci yang digunakan bukan angka, melainkan keyword yang sudah ditentukan di dalam Dictionary-nya.

Contoh:

```
item_kuliah    =    dict(laptop="Asus",    tas="seadanya",    sepatu="hasil
looting")
# Mengakses isi dictionary
print("Laptop saya adalah %s" % item_kuliah["laptop"])
```

Selain dengan cara di atas, dapat juga mengambil nilai Dictionary dengan method ‘get()’.

Contoh:

```
item_kuliah = dict(laptop="Asus", tas="seadanya", sepatu="hasil looting")
# Mengakses isi dictionary
print(item_kuliah.get("laptop"))
```

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

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

Contoh:

```
item_kuliah = {
   "laptop": "Asus",
   "tas": "seadanya",
   "sepatu": "hasil looting"
}
# mengubah isi laptop
item_kuliah["laptop"] = "Apple"
print(item_kuliah.get("laptop"))
```

Untuk menghapus nilai Dictionary, kita bisa menggunakan perintah del dan method ‘pop()’. Method ‘pop()’ adalah method yang berfungsi untuk mengeluarkan item dari dictionary sedangkan fungsi ‘del()’ adalah fungsi untuk menghapus suatu variabel dari memori.

Contoh menghapus dengan ‘del()’:

```
item_kuliah = {
   "laptop": "Asus",
   "tas": "seadanya",
   "sepatu": "hasil looting"
}
# menghapus laptop
del(item_kuliah["laptop"])
print(item_kuliah)
```

Contoh menghapus dengan ‘pop()’:

```
item_kuliah = {
   "laptop": "Asus",
   "tas": "seadanya",
   "sepatu": "hasil looting"
}
# menghapus laptop
item_kuliah.pop("laptop")
print(item_kuliah)
```

bila ingin menghapus semuanya sekaligus, kita bisa menggunakan method ‘clear()’.

```
item_kuliah = {
   "laptop": "Asus",
   "tas": "seadanya",
   "sepatu": "hasil looting"
}
# menghapus semuanya
item_kuliah.clear()
print(item_kuliah)
```

Untuk menambahkan isi ke Dictionary dapat menggunakan method update() . Parameternya berupa Dictionary.

Contoh:

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

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

print(user)
```

Untuk mengambil jumlah data atau panjang Dictionary, kita bisa menggunakan fungsi len(). Caranya sama dengan tuple.

4.4 Fungsi dan Prosedur

Pada pembuatan program yang kompleks dan memiliki banyak fitur,diharuskan membuat sebuah fungsi. Dengan fungsi, *programmer* dapat memecah program besar menjadi sub program yang lebih sederhana.

Fungsi pada Python, dibuat dengan kata kunci ‘def’ kemudian diikuti dengan nama fungsinya.

Struktur fungsi:

```
def nama_fungsi():
   print("Halo ini Fungsi")
```

Sama seperti blok kode yang lain, harus memberikan identasi (tab atau spasi 2x) untuk menuliskan isi fungsi.

Contoh:

```
# Membuat Fungsi
def kenalan():
   print("Halo, Aku iqbal")

# Pemanggilan Fungsi
kenalan()
```

Perlu diingat

> Fungsi juga dapat dipanggil pada fungsi lain, bahkan bisa memanggil dirinya sendiri. Fungsi yang memanggil dirinya sendiri, disebut **fungsi rekursif**.

Fungsi juga dapat diberi paramater. Parameter adalah variabel yang menampung nilai untuk diproses di dalam fungsi.

Contoh:

```
def salam(isi):
   print(isi)
salam("halo")
```

Pada contoh diatas adalah fungsi dengan parameter isi.

Jika menginginkan parameter lebih dari satu, dapat dipisah dengan koma.

Contoh:

```
# Membuat fungsi dengan parameter
def luas_segitiga(alas, tinggi):
   luas = (alas * tinggi) / 2
```

```
    print("Luas segitiga: %f" % luas)
# Pemanggilan fungsi
luas_segitiga(1, 2)
```

Fungsi yang tidak mengembalikan nilai biasanya disebut dengan prosedur. Terdapat juga fungsi yang mengembalikan nilai. Cara mengembalikan nilai adalah menggunakan kata kunci return lalu diikuti dengan nilai atau variabel yang akan dikembalikan

Contoh:

```
def luas_persegi(sisi):
    luas = sisi * sisi
    return luas

# pemanggilan fungsi
print "Luas persegi: %d" % luas_persegi(2)
```