Modul Praktikum

Algoritma Pemrograman Dasar

7

ASCii

“LIST DAN TUPLE”

“LIST”

7.1.1. Tujuan :

- Mahasiswa dapat memahami apa itu list
- Mahasiswa dapat memanggil dan memanipulasi list serta elemennya
- Mahasiswa dapat memahami konsep list multi dimensi
- Mahasiswa dapat mengimplementasikan list pada program

7.1.2. Penjelasan

List adalah salah satu tipe data kolektif yang ada di Python, mirip dengan array di bahasa pemrograman lain namun lebih fleksibel.

7.1.3. Definisi List

List dapat menyimpan berbagai data sekaligus seperti string, integer, float, dan sebagainya. Kita bahkan dapat menyimpan list di dalam sebuah list yang disebut dengan *nested list* atau list multi-dimensi.

List didefinisikan menggunakan tanda kurung siku [ ] dan di setiap anggota dari data tersebut dipisahkan oleh tanda koma.

7.1.4. Sifat List

Sifat-sifat list bisa didaftar seperti ini:

- **List bersifat dinamis**, dalam artian list dapat berubah-ubah isinya. elemen-elemen di dalam list dapat ditambah maupun dihapus.
- **List dapat berisi tipe data apapun**, elemen-elemen di dalam list yang sama dapat memiliki tipe data yang berbeda. Bahkan, elemen di dalam list dapat bertipe data list yang biasa disebut *nested list*.
- **List bersifat terurut**, elemen-elemen di dalam list terurut berdasarkan indeksnya, dimana elemen pertama di dalam sebuah list berindeks 0, elemen kedua berindeks 1, dan seterusnya. Karena indeks tersebut, meskipun elemen-elemen di dalam list itu sama, urutannya berbeda dan tidak dianggap sama dari perspektif python.

7.1.5. Cara Membuat List di Python

List dapat kita buat seperti mendeklarasikan variabel, namun nilai variabelnya diisi dengan tanda kurung siku [ ] dan setiap elemen dipisahkan koma.

Contoh :

```
#berikut list nama mahasiswa

nama_mhs = [“Celio”, “Afrizal”, “Farrel”, “Ghazali”]
```

Bentuk list di dalam python tidak harus linear. Jika elemennya sedikit, kita dapat membuat list seperti ini;

Contoh:

```
nama_mhs = [
                “Celio”,
                “Afrizal”,
                “Farrel”,
               “Ghazali”
                        ]
```

Namun, jika elemen di dalam list jumlahnya banyak, atau list akan dioperasikan contoh diatas tidak disarankan.

Sesuai dengan sifat kedua list. elemen-elemen di dalam list dapat memiliki tipe data yang berbeda satu sama lain.

Contoh:

```
#berikut adalah mendefinisikan list tas

tas = ["buku", 32, True, 3.14, [“IF”, 24]]
```

Ada lima jenis tipe data pada list diatas:

- “buku” merupakan tipe data string
- 32 merupakan tipe data integer
- True merupakan tipe data boolean
- 3.14 merupakan tipe data float
- [“IF”, 24] merupakan tipe data list

7.1.6. Cara menampilkan list

List, secara keseluruhan, dapat kita panggil menggunakan nama list itu saja sama seperti kita memanggil nilai dari variabel biasa.

Contoh:

```
#menggunakan list tas dicontoh sebelumnya
tas = ["buku", 32, True, 3.14, ["IF", 24]]
print(tas)
```

output:

```
["buku", 32, True, 3.14, ["IF", 24]]
```

Namun ada kalanya kita hanya ingin memanggil satu elemen saja dari sebuah list. dalam kasus ini, kita dapat menggunakan indeks dari elemen yang kita panggil dengan format nama_list[indeks].

Contoh :

```
#berikut adalah contoh menggunakan list tas
tas = ["buku", 32, True, 3.14, ["IF", 24]]
#disini kita melakukan pemanggilan pada list “tas” untuk elemen yang pertama
#karena indels pada python di mulai dari 0, maka kita masukkan angka 0 pada bagian indeks
print(tas[0])
print(tas[4])
```

output :

```
buku

[“IF”, 24]
```

7.1.7. Cara memanipulasi list

List bersifat dinamis, kita dapat menambahkan elemen baru ke dalam list yang sudah ada, mengubah elemen yang telah ada dan menghapus elemen dari list.

7.1.7.1 Menambah elemen ke dalam list

Untuk menambah elemen baru ke dalam list yang sudah ada, kita dapat menggunakan fungsi .append() dari python. formatnya: nama_list.append(elemen baru)

Contoh:

```
#list mata kuliah yang sulit di semester 1

matkul = [“Kalkulus”, “Fisika Dasar”, “Pengantar Teknologi Informasi”]
print(matkul)
#tampilkan list sebelum di append dengan elemen baru

#kita akan menambahkan mata kuliah “Logika Informatika” ke dalam list di atas

matkul.append(“Logika Informatika”)
print(matkul)
```

output:

```
[“Kalkulus”, “Fisika Dasar”, “Pengantar Teknologi Informasi”]
[“Kalkulus”, “Fisika Dasar”, “Pengantar Teknologi Informasi”, “Logika Informatika”]
```

Jika kita perhatikan, elemen baru yang ditambahkan kedalam sebuah list menggunakan fungsi .append() selalu berada di indeks terakhir list tersebut. Lantas, bagaimana caranya jika kita ingin menambahkan elemen baru tersebut di indeks tertentu?

Selain .append(), untuk menambahkan elemen baru kita juga dapat menggunakan fungsi .insert(). Fungsi inilah yang dapat menambahkan, dalam kasus ini menyisipkan, elemen baru ke indeks tertentu suatu list. formatnya: nama_list.insert(indeks, elemen baru)

Contoh

```
#list mata kuliah yang sulit semester 1

matkul = [“Kalkulus”, “Fisika Dasar”, “Pengantar Teknologi Informasi”]
print(matkul)
#tampilkan list sebelum insert elemen baru

#kita akan menambahkan mata kuliah “Logika Informatika” ke dalam list di atas

matkul.insert(1, “Logika Informatika”)
print(matkul)
```

output:

```
[“Kalkulus”, “Fisika Dasar”, “Pengantar Teknologi Informasi”]
[“Kalkulus”, “Logika Informatika”, “Fisika Dasar”, “Pengantar Teknologi Informasi”]
```

Dapat dilihat kita berhasil menambahkan elemen baru ke indeks pertama dari list matkul. Perlu diingat dengan menyisipkan suatu elemen baru, elemen setelahnya akan bergeser.

7.1.7.2 Mengubah elemen list

Elemen dari list yang sudah ada dapat diubah dengan cara memanggil list dan indeks dari elemen yang ingin kita ubah lalu menggunakan "=" untuk mendefinisikan dengan elemen baru. Esensi dari metode ini adalah menimpa elemen yang sudah ada dengan elemen baru.

Contoh:

```
#list nama program studi di fakultas teknik baru

prodi = ["Informatika", "Sistem Informasi", "Arsitektur", "Teknik Lingkungan"]
print(prodi)
#kita akan mengubah "Arsitektur" menjadi "Teknik Arsitektur"

prodi[2] = "Teknik Arsitektur"
print(prodi)
```

output:

```
["Informatika", "Sistem Informasi", "Arsitektur", "Teknik Lingkungan"]
["Informatika", "Sistem Informasi", "Teknik Arsitektur", "Teknik Lingkungan"]
```

7.1.7.3 Menghapus elemen list

Elemen dari list dapat kita hapus dengan menggunakan salah satu cara yang umum dalam python, yaitu del lalu nama list dan indeks dari elemen yang ingin kita hapus. formatnya: del nama_list[indeks].

Contoh:

```
#list nama program studi di fakultas teknik baru

prodi = [“Informatika”, “Sistem Informasi”, “Arsitektur”, “Teknik Lingkungan”]
print(prodi)
#kita akan menghapus “Teknik Arsitektur”

del prodi[2]
print(prodi)
```

output:

```
[“Informatika”, “Sistem Informasi”, “Teknik Arsitektur”, “Teknik Lingkungan”]
[“Informatika”, “Sistem Informasi”, “Teknik Lingkungan”]
```

7.1.8 *Slicing* list

Slicing atau memotong list adalah aksi memilih sebuah set elemen dari list. Sebelumnya kita sudah tau bagaimana cara memanggil satu elemen dari sebuah list, disini kita memanggil lebih dari 1 elemen dari sebuah list dengan cara memotong dan mengambil elemen-elemen tersebut. Untuk itu, kita menggunakan operator ':'. formatnya: nama_list[*start*:*stop*:*step*]

dimana;

- *start* adalah indeks dari elemen awal dari deret elemen yang ingin kita ambil
- *stop* adalah indeks dari elemen terakhir dari deret elemen yang ingin kita ambil
- *step* merupakan langkah antar elemen yang dipilih (*step* terhitung dari elemen yang terpilih, bukan elemen setelahnya, jika ingin melangkahi 1 elemen diantara elemen yang terpilih *step* = 2)

Contoh 1:

```
#list nama program studi di fakultas teknik baru

prodi = [“Informatika”, “Sistem Informasi”, “Teknik Arsitektur”, “Teknik Lingkungan”, “Teknik Pertambangan”, “Teknik Elektro”, “Teknik Industri”, “Teknik SIpil”, “Teknik Geologi”, “Teknik Kimia”]

#menggunakan konsep start dan stop, kita akan mengambil prodi Informatika sampai Teknik Elektro

prodi(prodi[0:6])
```

output:

```
['Informatika', 'Sistem Informasi', 'Teknik Arsitektur', 'Teknik Lingkungan', 'Teknik Pertambangan', 'Teknik Elektro']
```

Contoh 2:

```
#penggunaan step

prodi = [“Informatika”, “Sistem Informasi”, “Teknik Arsitektur”, “Teknik Lingkungan”, “Teknik Pertambangan”, “Teknik Elektro”, “Teknik Industri”, “Teknik Sipil”, “Teknik Geologi”, “Teknik Kimia”]

#kita ingin menampilkan semua elemen di dalam list dengan jarak 1 elemen diantaranya
print(prodi[::2], “\n”)

#kita akan mengambil prodi Sistem Informasi sampai Teknik Geologi dengan 1 langkah setiap elemen di antaranya
print(prodi[1:8:2])
```

output:

```
['Informatika', 'Teknik Arsitektur', 'Teknik Pertambangan', 'Teknik Industri', 'Teknik Geologi']

['Sistem Informasi', 'Teknik Lingkungan', 'Teknik Elektro', 'Teknik Sipil']
```

7.1.9 Operasi List

List dapat dioperasikan dengan dua operator, yaitu pertambahan (+) dan perkalian (*)

Pertambahan (+)

Contoh:

```
#list mobil bahan bakar listrik
bensin = ["avanza", "honda", "yamaha"]

#list mobil listrik
listrik = ["tesla", "SAIC"]

 #menggabungkan listrik dan bensin
semua = bensin+listrik

#menampilkan list semua
print(semua)
```

output:

```
['avanza', 'honda', 'yamaha', 'tesla', 'SAIC']
```

Perkalian (*)

Contoh:

```
#list mobil bahan bakar bensin
bensin = ["avanza", "honda", "yamaha"]

#menampilkan bensin sebanyak 3 kali
 print(bensin*3)
```

output:

```
['avanza', 'honda', 'yamaha', 'avanza', 'honda', 'yamaha', 'avanza', 'honda', 'yamaha']
```

7.1.10 List di dalam list (*nested* list)

Telah disinggung di bab sebelumnya, bahwa list dapat berisi tipe data apapun bahkan list itu sendiri. Lalu, bagaimana konsepnya? Membuat list di dalam list membuat list terluar menjadi dua dimensi atau lebih, tentu cara pemanggilan elemen nya sedikit berbeda. Kita dapat membayangkan list dua dimensi atau lebih ini seperti matriks dalam matematika:

Contoh:

```
#list barang

barang = [ [“Sapatu”, “Kaus Kakhi”, “Sarung”], [“Pulpen”, “Pensil”, “Laptop”] ]

#memanggil list secara keseluruhan untuk melihat bagaimana outputnya
print(barang, “\n”)

#memanggil indeks pertama
print(barang[0], “\n”)

#kali ini kita ingin menampilkan elemen “Pulpen”, kita perlu mengakses indeks dua kali, yang pertama untuk memilih list dengan elemen yang kita inginkan dan yang kedua indeks elemen itu sendiri

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

output:

```
[ [“Sapatu”, “Kaus Kakhi”, “Sarung”], [“Pulpen”, “Pensil”, “Laptop”] ]

[“Sapatu”, “Kaus Kakhi”, “Sarung”]

Pulpen
```

7.1.10.1 Perulangan dalam *nested* list

Jika kita ingin mengambil semua elemen di dalam kedua list tanpa terpisahkan list tersebut tidak mungkin kita panggil satu-satu dengan metode diatas. Untuk itu, kita dapat memanfaatkan perulangan **for**;

Contoh:

```
#list barang
barang = [ [“Sapatu”, “Kaus Kakhi”, “Sarung”], [“Pulpen”, “Pensil”, “Laptop”] ]

#perulangan for untuk mendapatkan semua elemen
for i in barang:

    for j in i :

        print (j)

#i dan j merupakan variabel sementara / temporary, kita dapat menggantinya dengan apa saja asal sesuai dengan syarat nama variabel
```

output:

```
Sapatu
Kaus Kakhi
Sarung
Pulpen
Pensil
Laptop
```

Studi Kasus:

Buatlah sebuah list yang terdiri dari 10 elemen, untuk nama list dan elemennya bebas.

1. Tampilkan elemen list dari indeks 1 sampai 7!
2. Tampilkan hanya elemen list dengan indeks genap! (indeks 0 tidak termasuk)
3. Tampilkan 5 elemen list dari belakang!

"Tuple"

7.2.1. Tujuan :

- Mahasiswa dapat memahami Tuple
- Mahasiswa dapat menggunakan Tuple pada program
- Mahasiswa dapat memahami nested Tuple
- Mahasiswa dapat mengunpacking tuple

7.2.2. Penjelasan

Tuple pada python adalah struktur data yang digunakan untuk menyimpan sekumpulan data. Tuple bersifat immutable, artinya isi tuple tidak bisa kita ubah dan hapus. Namun, dapat kita isi dengan berbagai macam nilai objek. Tuple adalah salah satu struktur data di Python yang mampu menyimpan sekumpulan nilai dalam satu variabel.

7.2.2 Cara membuat Tuple

Cara membuat tuple dapat dibuat dengan tanda kurung seperti ini :

```
Tuple = (123, 433, 696, 993, "Hello World")
```

juga dapat dibuat tanpa tanda kurung :

```
Tuple2 = 123, 433, 696, 993, "Hello World"
```

7.2.3. Membuat Tuple kosong

Kita juga dapat membuat tuple kosong tanpa isi, dapat di tuliskan seperti ini :

#Membuat Tuple kosong

```
kosong = ( )
```

Lalu dapat membuat Tuple dengan di buat berisi satu (singleton), maka kita harus menambahkan kutip 1 ataupun kutip 2 .

Contoh :

```
#Membuat tuple

tuple = "aldy"

tuple1 = ("aldy")
```

7.2.4. Mengakses Tuple

Sama pada list yang dapat di panggil, Tuple juga dapat di akses menggunkan cara yang sama.

Contoh :

```
#Mendeklarasikan Tuple

nama = ("rizky", "abdullah", "reza")

#Mengakses nilai Tuple

print (nama[1])

print (nama[2])

print (nama[3])
```

output :

```
risky

Abdullah

reza
```

7.2.5. Menampilkan Tuple berurut

Pada Tuple kita dapat menampilkannya berturut seperti, jika kita memiliki 10 tipe data pada Tuple dan kita menampilkan 0-6 saja maka kita menggunakan tuple[0:6]

Contoh :

```
#Mendeklarasikan Tuple
mahasiswa = (69, "Informatika", "2209106044", "Aldy septian ")
#Kita panggil tuple mahasiswa dari data ke 1 sampai ke 3
#Kita akan menampilkan dari index 0 sampai ke index 2
print(mahasiswa[0:3])
```

Keluaran :

```
(69, 'Informatika', '2209106044')
```

7.2.6. *Nested* tuple

Sama seperti list, tuple juga dapat berisi tuple / tuple di dalam tuple.

Contoh :

```
#deklarasikan tuple
tuple = (69, "Informatika", "2209106044", "Aldy septian")
tuple1 = (44, "Informatika", "2209106064", "Abdullah")


#mendeklarasikan tuple di dalam tuple
tuple2 = (tuple, tuple1)
#menampilkan tuple di dalam tuple
print(tuple2)
```

output:

```
((69, 'Informatika', '2209106044', 'Aldy septian'), (44,
'Informatika', '2209106064', 'Abdullah'))
```

7.2.7. Sequence Unpacking

Proses pembuatan tuple dapat disebut sebagai packing, sedangkan Ketika kita mengambil tuple dapat disebut unpacking.

Contoh :

```
#mendeklarasikan tuple
mahasiswa = (69, "Informatika", "2209106044", "Aldy septian ")
#lalu kita unpacking
absen, prodi, nim, nama = mahasiswa
#maka ketiga variabel tersebut akan berisi data dari tuple mahasiswa
#menampilkan variabel
print(absen)
print(prodi)
print(nim)
print(nama)
```

output :

```
69
Informatika
2209106044
Aldy septian
```

Studi kasus

Contoh Studi Kasus Perintah Program Aplikasi tuple Anggota suatu organisasi dengan perintah sebagai berikut :

1. Buat sebuah tuple untuk menyimpan Anggota AI
2. Isi tuple sebanyak 5
3. Tampilkan isi tuple indeks nomor 3
4. Tampilkan semua anggota AI dengan perulangan
5. Tampilkan panjang tuple

untuk menyelesaikan studi kasus di atas dapat di gunakan program di bawah :

```
#mendeklarasikan tuple
anggota = ("aldy", "rizky ", "abdullah", "milos", "reza")
#menampilkan index ke 3
print(f"anggota AI yang ketiga adalah {anggota[3]}")
#menampilkan semua anggota dengan perulangan
for i in anggota :
      print(i)
#menampilkan banyak anggota
print("banyak anggota AI sebanyak : %d" %len(anggota))
```

output :

```
anggota AI yang ketiga adalah milos
aldy
rizky
abdullah
```

```
milos

reza

banyak anggota AI sebanyak : 5
```