PERTEMUAN 2

PRAKTIKUM STRUKTUR DATA

Dosen pengajar : Reny Fitri Yani, S. T, M. T

Disusun oleh:

Ray Dave Adonia Siagian

NIM: 25071103787

UNIVERSITAS RIAU

T.A

2025/2026

I. STUDI KASUS

Studi kasus yang diangkat dalam laporan ini adalah mengenai penggunaan *list*, *tuple*, *sets*, dan *dictionary*. Pada dasarnya, keempat tipe data ini berfungsi untuk menyimpan lebih dari satu elemen yang nantinya akan digunakan dan dimodifikasi. Keempat tipe data ini sangat penting dan kerap digunakan dalam pemrograman, terutama seperti hal menyimpan biodata seseorang seperti nama, NIM, tempat tinggal, dan lain sebagainya.

Dalam file ini juga laporan mengenai latihan serta penjelasan tiap jawaban.

II. PEMBAHASAN

1. List

List adalah salah satu tipe data yang bekerja untuk mengoleksi/menyimpan lebih dari satu nilai yang dideklarasikan dengan simbol []. List tersebut dinamakan "buah" dan memuat 3 string. List bersifat berurutan dan memiliki indeks yang dapat diakses. List dapat diubah dan diganti setelah dibuat.

```
pelangi = ["Merah", "Kuning", "Hijau", "Biru"]
print(pelangi)
pelangi = ["Merah", "Kuning", "Hijau", "Biru", "Hijau"]
print(pelangi)
```

Python juga memungkinkan dua elemen yang sama/duplikasi dalam suatu list.

```
pelangi = ["Merah", "Kuning", "Hijau", "Biru", "Hijau"]
print(len(pelangi))
```

Len() digunakan untuk memeriksa panjang suatu list.

```
pelangi = ["Merah", "Kuning", "Hijau", "Biru", "Hijau"]
angka = [1, 2, 3, 4]
kemungkinan = [True, False]
```

List juga dapat memuat tipe data lainnya seperti integer, boolean, string, dan lainnya. Python menyatakan tipe data list sebagai *list*. `<class 'list'>`

Mengakses List

List diakses menggunakan indeks [*indeks*] yang diawali dari indeks ke-0.

```
print(pelangi[1])
```

Menampilkan:

```
Kuning
```

Indeks dapat diakses dengan angka negatif. List akan menentukan urutan secara mundur.

```
print(pelangi[-1])
```

Menampilkan:

```
Hijau
```

Indeks juga dapat diakses dari rentang tertentu ke indeks sebelum indeks tujuan.

```
print(pelangi[2:4])
```

Menampilkan:

```
['Hijau', 'Biru']
```

List dapat menampilkan dari indeks awal hingga indeks tertentu, tetapi tetap hanya sampai indeks sebelum rentang akhir

```
print(pelangi[:3])
```

Menampilkan:

```
['Merah', 'Kuning', 'Hijau']
```

List dapat menampilkan dari indeks tertentu hingga indeks terakhir.

```
print(pelangi[:3])
```

Menampilkan:

```
['Hijau', 'Biru', 'Hijau']
```

Memeriksa isi List

Kita dapat memeriksa apakah suatu list menyimpan nilai tertentu.

```
rumah = ["Ari", "Beni", "Caca"]
if "Ari" in rumah:
    print("Ari ada dalam rumah.")
```

Dari contoh, untuk memeriksa "Ari" dalam list, gunakan *in* lalu *list* yang ingin diperiksa.

Maka akan menampilkan:

```
Ari ada dalam rumah.
```

Mengganti isi List

Isi dalam suatu *list* dapat diganti berdasarkan indeksnya.

```
rumah = ["Ari", "Beni", "Caca"]
rumah[2] = "Asep"
print(rumah)
```

Dari contoh tersebut, indeks ke-2("Caca") akan diganti dengan "Asep".

Sehingga menampilkan:

```
['Ari', 'Beni', 'Asep']
```

Kita dapat mengganti lebih dari satu elemen dalam suatu *list* dengan rentang indeks tertentu.

```
rumah[0:2] = ["Sari, Jovan"]
```

Sehingga menampilkan:

```
['Sari, Jovan', 'Asep']
```

Kita juga dapat menambahkan dan sekaligus mengganti isi **satu** indeks yang mana akan menggeser indeks berikutnya.

```
rumah[1:2] = ["Hani", "Egi"]
```

Sehingga menampilkan:

```
['Sari, Jovan', 'Hani', 'Egi']
```

Menyisipkan dan Menambah data ke dalam List

Kita dapat menyisipkan elemen baru ke dalam *list* tanpa menghapus elemen yang sudah ada.

```
negara = ["Singapura", "Malaysia", "Filipina"]
negara.insert(1, "Brunei")
```
Maka menampilkan “Erdo” setelah indeks ke-2:
```
['Singapura', 'Brunei', 'Malaysia', 'Filipina']
```

Menambahkan elemen baru dengan append().
```
negara.append("Indonesia")
```
Maka menampilkan setelah indeks terakhir:
```
['Singapura', 'Brunei', 'Malaysia', 'Filipina', 'Indonesia']
```

Kita dapat menambahkan elemen suatu *list* ke *list* lainnya.
```
negara2 = ["India", "Pakistan", "Bangladesh", "Tajikisan"]
negara.extend(negara2)
```
Menampilkan:
```
['Singapura', 'Brunei', 'Malaysia', 'Filipina', 'Indonesia', 'India',
'Pakistan', 'Bangladesh', 'Tajikisan']
```

Menghapus elemen List

Kita dapat menghapus suatu elemen dari *list* dengan menggunakan .remove()
```
provinsi = ["Sumatera Selatan", "Sulawesi Utara", "Kalimantan Barat", "Jawa
Tengah"]
provinsi.remove("Jawa Tengah")
```
Menampilkan:
```
['Sumatera Selatan', 'Sulawesi Utara', 'Kalimantan Barat']
```

Menghapus indeks tertentu dalam *list* dengan pop()
```
provinsi.pop(0)
```
Menampilkan:
```
['Sulawesi Utara', 'Kalimantan Barat']
```

Menghapus indeks terakhir, terjadi apabila menggunakan pop() tanpa memasukkan indeks spesifik.
```
provinsi.pop()
```
Menampilkan:
```
['Sulawesi Utara']
```

Elemen list juga dapat dihapus mengunakan *del*, bahkan *list* itu sendiri.
```
del provinsi[0]
del provinsi
```
Menampilkan:
```
NameError: name 'provinsi' is not defined
```

Loop List

Menampilkan semua elemen dalam *list* satu per satu dengan *for*.
```
hari = ["Senin", "Selasa", "Rabu"]
for x in hari:
```
Menampilkan:
```
Senin
```

```
Selasa
Rabu
```

Perulangan juga dapat dilakukan dengan *while*.

```
while i < len(hari):
    print(hari[i])
```

Menampilkan:

```
Senin
Selasa
Rabu
```

Perulangan dapat dilakukan hanya dengan satu baris kode.

```
hari_lain = [x for x in hari if "S" in x]
```

Jika ada elemen yang memiliki “s”, maka akan disimpan dalam *list* hari_lain

Menampilkan:

```
['Senin', 'Selasa']
```

Mengurutkan List

List dapat diurutkan sesuai abjad.

```
jajan = ["Seblak", "Bakso", "Gorengan", "sate"]
jajan.sort()
```

Menampilkan:

```
['Bakso', 'Gorengan', 'Seblak', 'sate']
```

Dapat juga diurutkan dari belakang/mundur.

```
jajan.sort(reverse = True)
```

atau

```
jajan.reverse()
```

Menampilkan:

```
['sate', 'Seblak', 'Gorengan', 'Bakso']
```

Secara default, urutan *list* dipengaruhi oleh kapitalisasi huruf. Hal ini dapat dibuat agar kapitalisasi tidak berpengaruh.

```
jajan.sort(key = str.lower)
```

Menampilkan:

```
['Bakso', 'Gorengan', 'sate', 'Seblak']
```

Menyalin List

```
bulan = ["Mei", "juni", "Juli", "Agustus"]
bulan_lain = bulan.copy()
```

atau

```
bulan_lain = list(bulan)
```

atau dengan menggunakan tanda ‘:’

```
bulan_lain = bulan[:]
```

Menampilkan:

```
['Mei', 'juni', 'Juli', 'Agustus']
```

2. TUPLE

Tuple adalah tipe data yang, seperti *list*, menyimpan lebih dari satu elemen. Tetapi, *tuple* tidak dapat diubah dan *tuple* itu sendiri berurutan. *Tuple* menggunakan tanda kurung ()

```
negara = ("Serbia", "Georgia", "Armenia")
```

Menampilkan:

```
('Serbia', 'Georgia', 'Armenia')
```

Seperti *list*, *tuple* memiliki indeks dan memungkinkan duplikasi, hanya saja tidak dapat diubah.

```
negara = ("Kanada", "Kanada", "Meksiko", "Amerika Serikat")
```

Menampilkan:

```
('Kanada', 'Kanada', 'Meksiko', 'Amerika Serikat')
```

Len() juga digunakan untuk memeriksa panjang *tuple*.

```
print(len(negara))
```

Menampilkan:

```
4
```

Seperti *list*, *tuple* dapat memuat tipe data apa pun.

Mengakses Tuple

Indeks *Tuple* dapat diakses seperti *list* dengan [].

```
buah = ("Mangga", "Apel", "Sirsak", "Jeruk")
print(buah[1])
```

Menampilkan:

```
Apel
```

Tuple dapat diakses dengan berbagai cara mirip dengan *list*.

```
print(buah[-3])
print(buah[0:2])
print(buah[1:])
print(buah[:2])
```

Menampilkan:

```
Apel
['Mangga', 'Apel']
['Apel', 'Sirsak', 'Jeruk']
['Mangga', 'Apel']
```

Kita juga dapat memeriksa elemen *tuple*.

```
if "Jeruk" in buah:
    print("Saya punya jeruk.")
```

Menampilkan:

```
Saya punya jeruk.
```

Mengganti Tuple

Tuple tidak dapat diganti secara langsung, melainkan mengubahnya menjadi *list*, modifikasi, dan menggantinya kembali menjadi *tuple*.

```
warna = ("Kuning", "Biru", "Ungu")
a = list(warna)
```

```
a.append("Merah")
warna = tuple(a)
```
Menampilkan:
```
('Kuning', 'Biru', 'Ungu', 'Merah')
```

Suatu *tuple* dapat ditambahkan ke *tuple* lain.
```
b = ("Kuning",)
warna += b
print(warna)
```
Menampilkan:
```
('Kuning', 'Biru', 'Ungu', 'Merah', 'Kuning')
```

Tuple dapat dihapus, tetapi elemennya tidak dapat dihapus secara langsung. Jika ingin menghapus elemen *tuple*, konversikan ke *list* terlebih dahulu.
```
a = list(warna)
a.remove("Merah")
warna = tuple(a)
print(warna)
```
Menampilkan:
```
('Kuning', 'Biru', 'Ungu', 'Kuning')
```

Menghapus *tuple* sepenuhnya dengan *del*.
```
del warna
```

Membuka Tuple

Kita dapat meng-*unpack tuple* dengan mendeklarasikan nilai ke variabel(mengikuti indeks).
```
hewan = ("Ayam", "Kadal", "Kelinci")
unggas, reptil, mamalia = hewan

print(unggas)
print(reptil)
print(mamalia)
```
Menampilkan:
```
Ayam
Kadal
Kelinci
```

Kita dapat menggunakan tanda * jika jumlah variabel lebih sedikit dari elemen *tuple*.
```
hewan = ("Ayam", "Kadal", "Kelinci", "Kuda", "Anjing")
(unggas, reptil, *mamalia) = hewan
```
Menampilkan:
```
['Kelinci', 'Kuda', 'Anjing']
```

Loop Tuple

Perulangan *tuple* dapat dilakukan dengan menggunakan *for*.
```
rs = ("dokter", "perawat", "pasien")
for x in rs:
    print(x)
```

Menampilkan:

```
dokter
perawat
pasien
```

Loop dengan indeks.

```
for i in range(len(rs)):
    print(rs[i])
```

Menampilkan:

```
dokter
perawat
pasien
```

Perulangan dengan *while*.

```
x = ("aa", "bb", "cc", "dd")
i = 0
while i < len(x):
    print(x[i])
    i = i + 1
```

Menampilkan:

```
aa
bb
cc
dd
```

Menggabungkan Tuple

```
a = ("A", "B" , "C")
b = (1, 2, 3)

c = a + b
```

Menampilkan:

```
('A', 'B', 'C', 1, 2, 3)
```

```
d = c * 2
```

Menampilkan:

```
('A', 'B', 'C', 1, 2, 3, 'A', 'B', 'C', 1, 2, 3)
```

3. SET

Set adalah tipe data ketiga yang digunakan untuk menyimpan beberapa item dalam satu variabel..

Set adalah kumpulan yang tidak terurut, tidak dapat diubah, dan tidak terindeks. Ditandai dengan simbol {}

```
waktu = {"pagi", "siang", "sore", "malam"}
{'siang', 'malam', 'sore', 'pagi'}
```

Set tidak memungkinkan dua elemen ganda.

Dalam *set*, 1 sama dengan True dan 0 sama dengan False.

Untuk mengukur panjang *set* menggunakan len()

```
print(len(waktu))
```

Menampilkan:

```
4
```

Set memuat berbagai macam tipe data.

Akses Set

```
rumah = {"Adit", "Adiv", "Adinda"}

for x in rumah:
```

Menampilkan:

```
Adiv
Adit
Adinda
```

```
print("Adiv" in rumah)
```

Menampilkan:

```
True
```

```
brand = {"Apple", "Samsung", "Huawei"}
brand.add("Xiaomi")
```

Menampilkan:

```
{'Xiaomi', 'Samsung', 'Apple', 'Huawei'}
```

```
tanaman = {"sawit", "mangga", "kelengkeng"}
tambah_tanaman = ["kelapa", "pinang"]

tanaman.update(tambah_tanaman)
```

Menampilkan:

```
{'pinang', 'kelengkeng', 'kelapa', 'sawit', 'mangga'}
```

Menghapus elemen Set

Kita dapat menggunakan remove() untuk menghapus elemen dalam *set*.

```
negara = {"Mongolia", "Tiongkok", "Jepang"}
negara.remove("Mongolia")
```

Menampilkan:

```
{'Tiongkok', 'Jepang'}
```

discard() juga digunakan untuk menghapus elemen dalam *set*.

```
negara.discard("Tiongkok")
```

Menampilkan:

```
{'Jepang'}
```

pop() juga dapat digunakan, namun menghapus satu elemen acak.

```
negara = {"Mongolia", "Tiongkok", "Jepang"}
negara.pop()
```

Menampilkan:

```
{'Jepang', 'Mongolia'}
```

clear() digunakan untuk mengosongkan *set*.

```
negara = {"Mongolia", "Tiongkok", "Jepang"}
negara.clear()
```

Menampilkan:

```
set()
```

del digunakan untuk menghapus *set*.

```
del negara
```

Menampilkan:

```
NameError: name 'negara' is not defined
```

Loop Set

```
genre = {"horror", "romansa", "komedi"}

for x in genre:
    print(x)
```

Menampilkan:

```
romansa
komedi
horror
```

Menggabungkan Set

Gunakan union() untuk menggabungkan *set*.

```
set1 = {"Rita", "Dewi"}
set2 = {"Erdo", "Gan"}
set3 = {"Adi", "Wiji"}
set_gabungan = set1.union(set2)
```

Menampilkan:

```
{'Dewi', 'Rita', 'Gan', 'Erdo'}
```

union() juga dapat digunakan untuk menggabungkan lebih dari satu set, kecuali elemen yang sudah ada dalam *set* lainnya.

```
set_gabungan2 = set1.union(set2, set3)
```

Menampilkan:

```
{'Wiji', 'Erdo', 'Dewi', 'Adi', 'Gan', 'Rita'}
```

Cara lain untuk menggabungkan *set* yaitu dengan menggunakan simbol " | "

```
set_gabungan = set1 | set2
set_gabungan2 = set1 | set2 | set3
```

Menampilkan output yang sama:

```
{'Gan', 'Rita', 'Erdo', 'Dewi'}
{'Wiji', 'Erdo', 'Dewi', 'Adi', 'Gan', 'Rita'}
```

update() bekerja untuk menyisipkan elemen dari suatu *set* ke *set* lainnya, kecuali elemen yang sudah ada dalam *set* lainnya.

```
x = {"A", "B" , "C"}
y = {1, 2, 3}
```

```
x.update(y)
print(x)
```
Menampilkan:
```
{1, 2, 3, 'B', 'C', 'A'}
```

intersection() mengembalikan nilai yang hanya ditemukan dalam dua *set* yang disatukan.
```
A = {"merah", "kuning", "biru", "putih", "hitam"}
B = {"putih", "hitam"}
C = A.intersection(B)
```
Menampilkan:
```
{'putih', 'hitam'}
```

Fungsi yang sama juga dapat dilakukan dengan simbol &, tetapi perbedaannya intersection() bebas menyatukan tipe data yang berbeda dan tidak dengan menggunakan '&'
```
B = {"putih", "hitam"}
C = A & B
```
Menampilkan:
```
{'hitam', 'putih'}
```

Intersection_update() hanya menyimpan elemen yang sama.
```
B = {"putih", "hitam"}
A.intersection_update(B)
```
Menampilkan:
```
{'hitam', 'putih'}
```

difference() menampilkan elemen yang tidak ditemukan di kedua *set*.
```
A = {"merah", "kuning", "biru", "putih", "hitam"}
B = {"putih", "hitam"}
C = A.difference(B)
```
Menampilkan:
```
{'merah', 'kuning', 'biru'}
```

Fungsi yang sama juga dapat dilakukan dengan simbol -, tetapi perbedaannya intersection() bebas menyatukan tipe data yang berbeda dan tidak dengan menggunakan '-'
```
A = {"merah", "kuning", "biru", "putih", "hitam"}
B = {"putih", "hitam"}
C = A - B
```
Menampilkan:
```
{'merah', 'kuning', 'biru'}
```

difference_update() hanya menyimpan elemen yang tidak sama.
```
A = {"merah", "kuning", "biru", "putih", "hitam"}
B = {"putih", "hitam"}
A.difference_update(B)
```
Menampilkan:
```
{'kuning', 'merah', 'biru'}
```

4. DICTIONARY

Dictionary adalah tipe data yang menyimpan elemen-elemen yang berpasangan. Berurutan, dapat diubah, dan tidak dapat diduplikasi.

```
bio = {
    "nama": "Ahmed",
    "umur": 23,
    "lokasi": "Iraq"
}
```

Menampilkan:

```
{'nama': 'Ahmed', 'umur': 23, 'lokasi': 'Iraq'}
```

Mengakses Dictionary

Kita dapat mengakses elemen dalam *dictionary*.

```
print(bio["nama"])
```

Menampilkan:

```
Ahmed
```

Gunakan len() untuk menentukan berapa banyak elemen yang ada dalam suatu *dictionary*.

```
print(len(bio))
```

Menampilkan:

```
3
```

keys() akan mengembalikan seluruh kata kunci dalam suatu *dictionary*.

```
x = bio.keys()
dict_keys(['nama', 'umur', 'lokasi'])
```

values() akan mengembalikan seluruh nilai kata kunci dalam *dictionary*.

```
y = bio.values()
```

Menampilkan:

```
dict_values(['Ahmed', 23, 'Iraq'])
```

items() akan mengembalikan seluruh elemen sebagai *tuple*.

```
print(z)
```

Menampilkan:

```
dict_items([('nama', 'Ahmed'), ('umur', 23), ('lokasi', 'Iraq')])
```

Mengubah elemen Dictionary

Kita dapat mengubah nilai kata kunci secara langsung.

```
makanan = {
    "nama": "Nasi goreng",
    "stok": 12
}
makanan["stok"] = 11
```

Menampilkan:

```
{'nama': 'Nasi goreng', 'stok': 11}
```

Dictionary juga dapat diperbarui menggunakan update()

```
makanan.update({"stok": 10})
```

Menampilkan:

```
{'nama': 'Nasi goreng', 'stok': 10}
```

Menambahkan elemen Dictionary

```
makanan = {
    "nama": "Nasi goreng",
    "stok": 12
}
makanan["tipe"] = "makanan berat"
```

atau dengan menggunakan *update()*.

```
makanan.update({"harga" : 7000})
```

Menampilkan:

```
{'nama': 'Nasi goreng', 'stok': 12, 'tipe': 'makanan berat', 'harga':
7000}
```

Menghapus elemen Dictionary

pop() digunakan untuk menghapus elemen dalam *dictionary*.

```
makanan = {
    "nama": "Nasi goreng",
    "stok": 12,
    "harga" : 6000
}
makanan.pop("harga")
```

Menampilkan:

```
{'nama': 'Nasi goreng', 'stok': 12}
```

Popitem() digunakan untuk menghapus elemen terakhir.

```
makanan.popitem()
```

Menampilkan:

```
{'nama': 'Nasi goreng'}
```

del digunakan untuk menghapus kata kunci yang spesifik.

```
del makanan["harga"]
```

Menampilkan:

```
{'nama': 'Nasi goreng', 'stok': 12}
```

Clear() digunakan untuk mengosongkan *dictionary* sepenuhnya.

```
makanan = {
    "nama": "Nasi goreng",
    "stok": 12,
    "harga" : 6000
}
for x in makanan:
```

Menampilkan:

```
nama
stok
harga
```

Menampilkan seluruh isi kata kunci dalam *dictionary*.

```
makanan = {
    "nama": "Nasi goreng",
    "stok": 12,
    "harga" : 6000
}
for x in makanan:
```

Menampilkan:

```
Nasi goreng
12
6000
```

Menyalin dalam Dictionary

Menyalin suatu elemen *dictionary* ke yang lainnya dengan mendeklarasikan dict(*yang akan disalin*) ke *dictionary* tujuan

```
makanan = {
    "nama": "Nasi goreng",
    "stok": 12,
    "harga" : 6000
}
makanan2 = dict(makanan)
```

Menampilkan:

```
{'nama': 'Nasi goreng', 'stok': 12, 'harga': 6000}
```

Nested Dictionary

```
harta = {
    "orang1" : {
        "nama" : "Nag",
        "uang" : 100
    },
    "orang2" : {
        "nama" : "Odre",
        "uang" : 50
    },
    "orang3" : {
        "nama" : "Otnay",
        "uang" : 0
    }
}
```

Menampilkan:

```
{'orang1': {'nama': 'Nag', 'uang': 100}, 'orang2': {'nama': 'Odre',
'uang': 50}, 'orang3': {'nama': 'Otna
y', 'uang': 0}}
```

Mengaksesnya dengan nama *dictionary*, lalu bagian terluar *dictionary*.

```
print(harta["orang3"]["uang"])
```

Menampilkan:

```
0
```

Melakukan perulangan dengan items().

```
for x, s in harta.items():
    print(x)

    for y in s:
        print(y + ':', s[y])
```

Menampilkan:

```
orang1
nama: Nag
uang: 100
orang2
nama: Odre
uang: 50
orang3
nama: Otnay
uang: 0
```

3. PENJELASAN TUGAS

1. Diberikan sebuah list angka:
   `angka = [10, 20, 30, 40, 50]`
   1. Tambahkan angka 60 ke dalam list.
   2. Hapus angka 20 dari list.
   3. Tampilkan angka tertinggi dan terendah
   4. Hitung rata-rata angka setelah perubahan data
   5. Tampilkan seluruh isi list setelah perubahan.

1. Deklarasikan terlebih dahulu

```
angka = [10, 20, 30, 40, 50]
```

1.1 Tambah angka 60

```
angka.append(60)
print(angka)
```

Gunakan append() untuk menambahkan elemen baru, masukkanb 60

1.2 Hapus angka 20 dari list

```
angka.pop(1)
print(angka)
```

pop() digunakan untuk menyingkirkan 20(yang merupakan indeks ke-1) dari list angka

1.3 Tampilkan angka tertinggi dan terendah

```
terendah = min(angka)
tertinggi = max(angka)
```

Gunakan max() untuk mencari nilai terbesar dan min() untuk mencari nilai terkecil.

1.4 Hitung rata-rata

```
x = sum(angka)/len(angka)
```

Gunakan sum() untuk menjumlahkan seluruh integer dalam angka yang nantinya akan dibagikan dengan len() atau panjang list angka

1.5 Tampilkan seluruh isi setelah perubahan

```
print("List akhir:", angka)
print("Tertinggi:", tertinggi)
print("Terendah:", terendah)
print("Rata-rata:", x)
```

Menggunakan print() untuk melihat perubahannya sekarang.

2. Diberikan sebuah tuple data mahasiswa:
   `mahasiswa = ("A001", "Budi", "Informatika")`
   1. Tampilkan nama mahasiswa dari tuple tersebut.
   2. Tampilkan seluruh isi tuple menggunakan perulangan for.
   3. Jelaskan satu alasan mengapa tuple tidak bisa diubah.

2. Deklarasi

```
mahasiswa = ("A001", "Budi", "Informatika")
```

2.1 Tampilkan nama

```
print("Nama mahasiswa: ", mahasiswa[1])
```

2.2 Tampilkan seluruh isi dengan perulangan for

```
for isi in mahasiswa:
    print(isi)
```

2.3 Alasan kenapa tuple tidak bisa diubah

Tuple tidak bisa diubah karena isinya tidak dapat diubah, hapus, atau diganti. Hal ini berguna untuk menjaga keamanan.

3. Diberikan dua set mata kuliah pilihan:
   `kelas_A = {"Struktur Data", "Basis Data", "AI", "Pemrograman Web"}`
   `kelas_B = {"Struktur Data", "Machine Learning", "AI", "Cloud Computing"}`
   1. Tentukan mata kuliah yang diambil oleh kedua kelas.
   2. Tentukan mata kuliah yang hanya diambil kelas A.
   3. Tentukan seluruh mata kuliah unik yang diambil oleh kelas A dan B.

3. Deklarasi

```
kelas_a = {"Struktur Data", "Basis Data", "AI", "Pemrograman Web"}
kelas_b = {"Struktur Data", "Machine Learning", "AI", "Cloud Computing"}
```

3.1 Mata kuliah yang diambil oleh kedua kelas

```
kedua_kelas = kelas_a & kelas_b
```

Menggunakan simbol & untuk mencari persamaan di antara kedua set mata kuliah.

3.2 Mata kuliah yang hanya diambil oleh kelas A

```
a = kelas_a - kelas_b
```

Menggunakan simbol – untuk mencari mata kuliah A yang tidak ditemukan di kelas B.

3.3 Seluruh mata kuliah unik

```
semua = kelas_a | kelas_b
```

Menggunakan simbol – untuk mencari perbedaan di antara kedua set mata kuliah yang berikutnya digabung.

4. Sebuah data mahasiswa disimpan dalam bentuk dictionary:

```
mahasiswa = {
    "A001": {"nama": "Budi", "prodi": "Informatika",
"ipk": 3.45},
    "A002": {"nama": "Siti", "prodi": "Sistem
Informasi", "ipk": 3.20},
    "A003": {"nama": "Andi", "prodi": "Informatika",
"ipk": 3.75}
}
```

1. Tampilkan nama mahasiswa yang memiliki IPK di atas 3.5.
2. Hitung rata-rata IPK seluruh mahasiswa.
3. Tambahkan satu data mahasiswa baru ke dalam dictionary tersebut.

4. Deklarasi

```
mahasiswa = {
    "A001": {"nama": "Budi", "prodi": "Informatika", "ipk": 3.45},
    "A002": {"nama": "Siti", "prodi": "Sistem Informasi", "ipk": 3.20},
    "A003": {"nama": "Andi", "prodi": "Informatika", "ipk": 3.75}
}
```

4.1 Mahasiswa dengan IPK di atas 3,5

```
for atas in mahasiswa.values():
    if atas["ipk"] > 3.5:
        print(atas["nama"])
```

Mahasiswa.values() mengambil data semua mahasiswa.
Atas bekerja untuk menyimpan identitas mahasiswa dengan IPK di atas 3,5.
Print(atas[“nama”}) merupakan syarat, bila terdapat mahasiswa dengan IPK di atas 3,5 maka nama tersebut disimpan dalam variabel **atas** lalu ditampilkan.

4.2 Rata-rata IPK seluruh mahasiswa

```
total = 0

for data in mahasiswa.values():
    total += data["ipk"]

rata = total / len(mahasiswa)
print("IPK Rata-rata:", rata)
```

Awalnya **total** dideklarasikan 0.
For data in mahasiswa.values() menjalankan looping untuk memeriksa semua data mahasiswa.
total += data[“ipk”] menggabungkan seluruh nilai IPK **mahasiswa** ke **total**.
Rata = total/len(mahasiswa) menghitung rata-rata dengan memasukkan **total** yang sudah dibagi dengan jumlah data yang diambil dari **mahasiswa**.

4.3 Tambahkan satu data mahasiswa baru

```
mahasiswa["A004"] = {
    "nama": "Adit",
    "prodi": "Teknik Lingkungan",
    "ipk": 3.60
}
```

Mahasiswa[“A004”] sebagai key baru.
“nama”, “prodi”, “ipk” mengikuti pola yang sudah ada.
“Adit”, “Teknik Lingkunagn”, 3.60 mengisi informasi/menambahkan value