LAPORAN PRAKTIKUM

STRUKTUR DATA

Pertemuan 2

Disusun oleh :

Nama : Fildza Khusaini

NIM : 25071104589

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

Asisten : 1. Akhlaqul Muhammad Fadwa (2307112834)

2. Rizkillah Ramanda Sinyo (2307126144)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS RIAU

PEKANBARU

GENAP 2025/2026

TUGAS PRAKTIKUM 2: Menjelaskan code yang telah diuji coba

A. Python Dictionaries

1) Membuat dan mencetak Dictionary

```
biodata = {
    "nama" : "fildza",
    "umur" : 18,
    "hobi" : "memasak"
}
print(biodata)

#output
{'nama': 'fildza', 'umur': 18, 'hobi': 'memasak'}
```

dictionary ditulis menggunakan tanda kurung kurawal {}, dan digunakan untuk menyimpan nilai data dalam bentuk pasangan key:value.

2) Item dalam dictionary dapat diakses dengan menggunakan key nya.

```
biodata = {
    "nama" : "fildza",
    "umur" : 18,
    "hobi" : "memasak"
}

print(biodata["nama"])
#output
fildza
```

3) Selain itu, ada juga metode get() yang akan menghasilkan hasil yang sama

```
biodata = {
    "nama" : "fildza",
    "umur" : 18,
    "hobi" : "memasak"
}

print(biodata.get("umur"))
```

```
#output
18
```

4) Dictionary tidak dapat memiliki dua item dengan key yang sama. Jika terdapat key duplikat, nilai dari key duplikat akan menimpa nilai yang sudah ada.

```
contoh = {
    "hari" : "senin",
    "tanggal" : 20,
    "bulan" : 2,
    "tahun" : 2026,
    "tanggal" : 12
}
print(contoh)
#output
{'hari': 'senin', 'tanggal': 12, 'bulan': 2, 'tahun':
2026}
```

Pada contoh di atas, nilai "tanggal" yang awalnya 20 berubah menjadi 12 karena menggunakan key yang sama.

5) Dictionary length
Untuk menghitung berapa banyak item yang berada di dalam dictionary, dapat menggunakan fungsi len()

```
contoh = {
    "hari" : "senin",
    "tanggal" : 20,
    "bulan" : 2,
    "tahun" : 2026,
    "tanggal" : 12
}
print(len(contoh)
#output
4
```

6) Nilai dalam item kamus dapat berupa tipe data apapun

```
bio = {
    "nama": "fildza",
    "umur": 18,
    "status pelajar": True,
    "tanggal lahir": [21, "september", 2007]
}
print(bio)

#output
{'nama': 'fildza', 'umur': 18, 'status pelajar': True,
'tanggal lahir': [21, 'september', 2007]}
```

7) Selain menggunakan tanda kurung kurawal, kita juga bisa membuat dictionary dengan menggunakan kontruktor dict()

```
#cara lain untuk menuliskan dict

bunga = dict(jenis = 'mawar', warna = 'merah', jumlah = 10)

print(bunga)
#output
{'jenis': 'mawar', 'warna': 'merah', 'jumlah': 10}
```

8) Get Keys
Metode keys() akan menampilkan semua key pada dictionary, dan hasilnya akan ikut berubah jika dictionary di ubah

```
bunga = {
    "jenis": "tulip",
    "warna": "kuning",
    "jumlah": "10"
}

x = bunga.keys()
print(x) #sebelum berubah

bunga["pemesan"] = "rania"
print(x) #setelah menambahkan key value baru

#output
dict_keys(['jenis', 'warna', 'jumlah'])
dict_keys(['jenis', 'warna', 'jumlah', 'pemesan'])
```

9) Get Values

Metode values() akan mengembalikan daftar semua nilai di dalam dictionary. Dan daftar nilai juga akan berubah jika diperbarui

```
bunga = {
    "jenis": "tulip",
    "warna": "kuning",
    "jumlah": "10"
}

y = bunga.values()

print(y) #sebelum berubah

bunga["pemesan"] = "fildza"
print(y) #setelah berubah

#output
dict_values(['tulip', 'kuning', '10', 'rania'])
dict_values(['tulip', 'kuning', '10', 'fildza'])
```

10) Get Items

Method items() akan mengembalikan setiap item dalam dictionary dalam bentuk tuple(key, value)

```
bunga = {
    "jenis": "tulip",
    "warna": "kuning",
    "jumlah": "10"
}

print(bunga.items())

#output
dict_items([('jenis', 'tulip'), ('warna', 'kuning'),
('jumlah', '10'), ('pemesan', 'fildza')])
```

11) Mengecek key

Keyword ‘in’ digunakan untuk mengecek keberadaan key dalam dictionary

```
bunga = {
    "jenis": "tulip",
    "warna": "kuning",
    "jumlah": "10"
}
if "warna" in bunga:
    print("ada key 'warna' di dalam dictionary")

#output
ada key 'warna' di dalam dictionary
```

12) Mengubah nilai

Nilai item pada dictionary dapat diubah dengan mengacu pada nama key-nya langsung

```
menu = {
    "nama": "cookies",
    "rasa": "coklat",
    "jumlah": "2"
}
menu["rasa"] = "original"

#output
{'nama': 'cookies', 'rasa': 'original', 'jumlah': '2'}
```

13) Update dictionary

Method update() dapat menambahkkan atau memperbarui data dictionary dari dictionary atau pasangan key value lain.

```
menu = {
    "nama": "cookies",
    "rasa": "coklat",
    "jumlah": "2"
}
menu.update({"varian": "oreo"})

#output
{'nama': 'cookies', 'rasa': 'original', 'jumlah': '2',
'varian': 'oreo'}
```

14) Menghapus Item

Ada beberapa method untuk menghapus item dari dictionary, di antaranya adalah

```
menu = {
    "nama": "cookies",
    "rasa": "coklat",
    "jumlah": "2"
}

#pop() -> akan menghapus item dengan nama key yang ditentukan
menu.pop("rasa")
print(menu)

#output
{'nama': 'cookies', 'jumlah': '2'}
```

```
#popitem() -> menghapus item terakhir yang dimasukkan

menu = {
    "nama": "cookies",
    "rasa": "coklat",
    "jumlah": "2"
}

menu.popitem()
print(menu)

#output
{'nama': 'cookies', 'rasa': 'coklat'}
```

```
#keyword del -> menghapus item dengan nama key yang ditentukan

menu = {
    "nama": "bolu",
    "rasa": "coklat",
    "juumlah": 2
}
del menu["rasa"]

#output
{'nama': 'bolu', 'juumlah': 2}
```

```
#clear() untuk mengosongkan dictionary

menu = {
    "nama": "bolu",
    "rasa": "coklat",
    "juumlah": 2
}
menu.clear()
print(menu)

#output
{}
```

15) loop Dictionaries

Loop pada dictionary secara default menghasilkan key, tetapi bisa juga menampilkan value atau item dengan method tertentu.

```
#loop pada dictionary secara default menghasilkan key

bunga = {
    "jenis": "tulip",
    "warna": "kuning",
    "jumlah": "10"
}
for x in bunga:
    print(x)

#output
jenis
warna
jumlah
```

```
#untuk menampilkan value, dapat menggunakan method tertentu

bunga = {
    "jenis": "tulip",
    "warna": "kuning",
    "jumlah": "10"
}
for x in bunga:
    print(bunga[x])

#output
tulip
kuning
10
```

```
#bisa juga pakai method values()
```

```
bunga = {
    "jenis": "mawar",
    "warna": "pink",
    "jumlah": "21"
}
for x in bunga.values():
    print(x)

#output
mawar
pink
```

```
#kita bisa juga buat perulangan key dan value sekaligus
dengan method items()
```

```
for x, y in bunga.items():
    print(x ,y)

#output
jenis mawar
warna pink
jumlah 21
```

16) Copy dictionaries

gunakan copy() untuk menyalin dictionary agar perubahan pada dictionary asli tidak memengaruhi dictionary salinan

```
#method copy() untuk menyalin dictionary
```

```
menu = {
    "nama": "cookies",
    "rasa": "coklat",
    "jumlah": "2"
}
salinMenu = menu.copy()
print(salinMenu)
```

```
#atau juga bisa pakai fungsi dict()

pesan = {
    "nama": "fildza",
    "pesanan": "susu",
    "jumlah": "1"
}
pesanan = dict(pesan)
print(pesanan)
```

17) Nested Dictionaries

sebuah dictionary dapat berisi dictionary lain, ini disebut nested dictionary (dictionary bersarang)

```
menu = {
"oven1": {"nama": "brownies", "rasa": "coklat", "jumlah": 5},
"oven2": {"nama": "cookies", "rasa": "red velvet", "jumlah":
12}
}
print(menu)

#output
{'oven1': {'nama': 'brownies', 'rasa': 'coklat', 'jumlah': 5},
'oven2': {'nama': 'cookies', 'rasa': 'red velvet', 'jumlah':
12}}

print(len(menu))  #panjangnya 2
```

bisa juga jika ingin menambahkan beberapa dictionary ke dalam sebuah dictionary

```
oven1 = dict(jenis = "bolu", varian = "oreo", jumlah = 5)

oven2 = dict(jenis = "tart", varian = "vanila", jumlah = 5)

totalKue ={
    "oven1" : oven1,
    "oven2" : oven2
}
print(totalKue)

#output
{'oven1': {'jenis': 'bolu', 'varian': 'oreo', 'jumlah': 5},
'oven2': {'jenis': 'tart', 'varian': 'vanila', 'jumlah': 5}}
```

untuk mengakses item dari nested dictionary, bisa dengan menggunakan nama dictionary paling luar terlebih dahulu

```
menu = {
"oven1": {"nama": "brownies", "rasa": "coklat", "jumlah":
5},
"oven2": {"nama": "cookies", "rasa": "red velvet",
"jumlah": 12}
}
print(menu["oven2"]["jumlah"])

#output
12
```

18) loop through nested dictionaries
untuk melakukan perulangan melalui dictionary dengan menggunakan method items()

```
#loop through dictionary
for x, data in menu.items():
    print(x)
    for y in data:
        print(y + ':', data[y])

#output
oven1
nama: brownies
rasa: coklat
jumlah: 5
oven2
nama: cookies
rasa: red velvet
jumlah: 12
```

fungsi items() akan mengambil key dan value. Variabel "x" adalah key dari "menu" (yaitu oven1, oven2). Dan variabel "data" adalah value, yaitu dictionary di dalamnya. Karena "data" adalah dictionary juga, jadi variabel "y" menunjukkan key dari dictionary data ("nama", "rasa", "jumlah"). Dan data[y] adalah value dari key tersebut.

B. Python List

1) Membuat list
list ditulis menggunakan tanda kurung siku [], dan dapat digunakan untuk menyimpann tipe data yang berbeda dalam satu himpunan

```
makanan = ["ikan", "ayam", "pecel"]
print(makanan)
```

2) Untuk mengukur Panjang list, dapat menggunakan fungsi len()

```
minuman = ["teh", "kopi", "susu"]
print(len(minuman))
```

3) Mengakses item list

Untuk mengakses elemen di dalam list, dapat dengan merujuk pada nomor indeknya

```
makanan = ["ikan", "ayam", "pecel"]
print(makanan[0])
```

4) Negative indexing
Mengakses elemen di dalam list dengan menggunakan indeks negatif, yang dimana indeks negatif satu akan mengacu pada item paling akhir

```
boygroup = ["nct", "dream", "127", "wayV", "wish"]
print(boygroup[-1])
#-1 mengacu pada item terakhir, -2 pada item kedua terakhir, dan seterusnya
```

5) Range of indexes
digunakan untuk mengakses atau mengubah beberapa elemen list sekaligus berdasarkan rentang indeks tertentu. Penulisan range of indexes menggunakan aturan range [start:stop]. Elemen yang diakses adalah elemen dari indeks awal (start) sampai sebelum indeks akhir (stop)

```
boygroup = ["nct", "dream", "127", "wayV", "wish", "ateez", "lngshot"]
print(boygroup[2:5])
```

6) Jika rentang nilai awalnya tidak dituliskan, maka range nilai awal akan dimulai dari indeks pertama (nol)

```
boygroup = ["nct", "dream", "127", "wayV", "wish", "ateez", "lngshot"]
print(boygroup[:3])
```

7) Jika tidak menuliskan nilai akhir, maka range akan berlanjut hingga indeks terakhir

```
boygroup = ["nct", "dream", "127", "wayV", "wish", "ateez",
"lngshot"]
print(boygroup[2:])
```

8) Check if item exist
untuk menentukan apakah suatu item tertentu terdapat dalam sebuah list, gunakan keyword in

```
makanan = ["ayam", "ikan", "sayur"]

print("ikan" in makanan) #hasilnya True atau False

#contoh lain
if "ayam" in makanan:
  print("'ayam' ada di dalam list") #jika 'ayam' ada di dalam
list, maka cetak pesan
```

9) Change Value item
Mengganti suatu elemen tertentu dalam sebuah list dapat dilakukan dengan mengacu pada nomor indeknya

```
warna = ["merah", "biru", "kuning", "hijau"]
warna[1] = "jingga" #mengganti elemen pada indeks ke-1 dengan
'jingga'
print(warna)
```

10) Mengganti beberapa elemen list dengan menentukan rentang indeks dan memberikan list sebagai nilai pengganti

```
tempat = ["sungai", "pantai", "gunung", "kota", "pedesaan",
"rumah"]
tempat[1:3] = ["laut", "kebun"]
print(tempat)
```

```
#panjang daftar akan berubah ketika jumlah item yang dimasukkan
tidak sesuai dengan jumlah item yang diganti
warna = ["merah", "biru", "kuning"]
warna[1:3] = ["jingga"] #mengubah nilai kedua dan ketiga dengan
satu nilai
print(warna)
#hasilnya, elemen kedua dan ketiga akan dihapus dan digantikan
oleh satu elemen baru.
```

11) Insert item

insert() digunakan untuk menyisipkan elemen bar uke dalam list tanpa menghapus data yang sudah ada. Penambahan elemen menyebkan elemen lain bergeser ke indeks berikutnya, sehingga elemen dalam list bertambah

```
tempat = ["pantai", "sawit", "gunung"]
tempat.insert(2, "pedesaan")
print(tempat)
#gunung akan pindah ke indeks ke-3, karena pedesaan disisipkan
pada indeks ke-2
#kemudian list akan berubah sisinya menjadi 4 item
```

12) fungsi append() juga dapat digunakan untuk menambahkan item baru ke bagian akhir list

```
nama = ["fildza", "rania", "najwa"]
nama.append("syifa")
print(nama)#syifa akan ditambahkan di bagian akhir list

list akan berubah sisinya menjadi 4 item
```

13) Extend List

untuk menambahkan elemen dari list lain ke list saat ini, gunakan method extend()

```
#extend list
makanan = ["mie", "nasi", "tempe"]
minuman = ["jus", "kopi", "es teh"]
makanan.extend(minuman)

print(makanan)
#method ini tidak harus menambhkan list saja, tapi juga bisa
menambahkan objek iterable apapun
```

14) remove specified item

method remove() digunakan untuk menghapus item tertentu (spesifik)

```
#remove specified item
tempat = ["gunung", "kebun", "sawit"]
tempat.remove("sawit")
print(tempat)
```

```
#contoh lain
makanan = ["telur", "ayam", "gulai", "ikan", "ayam"]
makanan.remove("ayam")
print(makanan)
#yang dihapus hanya 'ayam' dibagian pertama saja.
#karena yang di cek indeks pertama, lalu kedua, begitu selanjutnya
terus sampai akhirnya ketemu yang ingin dihapus, selesai. bagian
setelahnya tidak akan di cek lagi
```

15) fungsi pop() untuk menghapus indeks spesifik

```
#menghapus menggunakan indeks spesifik
minuman = ["eskrim", "sirup", "susu"]
minuman.pop(1)
print(minuman)
```

16) keyword del untuk menghapus indeks yang di acu

```
#keyword del
minuman = ["es teh", "cendol", "es tebak"]
del minuman[0]  #indeks pertama akan di hapus
print(minuman)
```

17) mengosongkan list menggunakan method clear()

```
#clear the list (mengosongkan list)
tempat = ["aula", "kelas", "kantin"]
tempat.clear()
print(tempat)
```

18) loop list
untuk mencetak seluruh elemen list dapat menggunakan perulangan for

```
makanan = ["ayam", "ikan", "sate"]
for x in makanan:      #untuk setiap x di list makanan
  print(x) #cetak elemen satu persatu
```

19) perulangan pada list dengan mengaskses elemen berdasarkan nomor indeksnya

```
tempat = ["rumah", "kelas", "laboraturium"]
for i in range(len(tempat)):
  print(tempat[i])  #menampilkan elemen list sesuai indeks ke-i
```

20) while loop

untuk menampilkan setiap elemen list berdasarkan indeksnya

```
nama = ["fildza", "najwa", "salwa"]
i = 0
while i < len(nama):
  print(nama[i])  #tampilkan elemen list sesuai indeks ke i
  i = i + 1
```

21) List comprehension
perulangan singkat untuk mencetak isi list

```
nama = ["fildza", "najwa", "salwa"]
[print(x) for x in nama]
```

22) List comprehensiom ()

membuat list baru untuk elemen yang memiliki huruf 'a' menggunakan list comprehension dengan for dan if

```
tempat = ["rumah", "kelas", "IC", "kantin", "RS"]
tempatBaru = []

for x in tempat:
  if "a" in x:
    tempatBaru.append(x)

print(tempatBaru)
```

23) Mengatur nilai output list baru

```
tempatBaru = ['Di' +x for x in tempat]
#loop setiap x di list, setiap elemen di list baru diawali teks
"Di"
```

24) Customize sort function

mengurutkan list berdasarkan jarak nilai terhadap angka tertentu

```
def hitungJarak(n):
  return abs(n - 10)

nilai = [90, 40, 54, 12, 23]
nilai.sort(key = hitungJarak) #diurutkan dari selisihnya paling
dekat ke 10
print(nilai)
```

C. Set Dictionaries

1) Membuat set
   Elemen di dalam set tidak berurutan, tidak dapat diakses menggunakan indeks, tidak mengizinkan elemen duplikat. Set ditulis dengan tanda kurung kurawal

```
benda = {"meja", "kipas", "kursi"}

print(benda)
```

2) Dalam satu set, tidak boleh memiliki dua item degan nilai yang sama Jika terdapat elemen yang sama, maka elemen tersebut hanya akan disimpan satu kali.

```
benda = {"piring", "sendok", "garpu", "piring"}
print(benda)
```

3) Item didalam set dapat berupa tipe data apapun

```
nama = {"halo", 123, True}
```

4) Kostruktor set
   selain menggunakan tanda kurung kurawal {}, set juga dapat dibuat dengan konstruktor set. konstruktor set digunakan untuk membuat himpunan (set) baru dari data yg diberikan

```
warna = set(("merah", "pink", "ungu", "biru"))
print(warna)
```

5) Mengakses item
   item pada set tidak dapat diakses dengan merujuk pada indeks atau key, namun kita bisa menggunakan for loop atau keyword in

```
tingkat = {"1A", "1B", "1C", "1D"}

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

6) Memeriksa apakah suatu elemen terdapat di dalam set atau tidak menggunakan keyword in dan not in

```
tingkat= {"1A", "1B", "1C", "1D"}
print("1B" in tingkat)
print("1B" not in tingkat)
```

7) Add item
Setelah set dibuat, itemnya tidak dapat diubah karena ia bersifat mutable. Akan tetapi kita bisa menambahkan item baru ke dalamnya

```
kode = {"me", "ji", "ku"}

kode.add("hi")
print(kode) #set adalah himpunan tidak berurut, jadi item
set dapat muncuk dalam urutan berbeda setiap kali digunakan
```

8) Add sets()
method update() dapat digunakan untuk menambahkan item dari set lain ke set saat ini. bukan hanya set, bisa juga berupa tuple, list dll

```
kodeSet = {"me", "ji", "ku"}
kodeList = ["hi", "bi", "ni", "u"]

kodeSet.update(kodeList)
print(kodeSet)
```

9) Remove item
sama seperti list, set juga dapat menggunakan method remove(), pop(), clear(), dan del. Method pop() akan menghapus item secara acak, dan akan mengembalikan nilai berupa item yang dihapus. Selain menggunakan remove(), kita juga dapat menggunakann discard() untuk menghapus item

```
tingkat= {"1A", "1B", "1C", "1D"}

tingkat.discard("1C")
print(tingkat)
#jika item yang akan dihapus tidak ada, akan muncul error
```

10) python join sets
method union() dan update() untuk menggabungkan semua item dari beberapa himpunan

```
set1 = {"air", "udara", "api"}
set2 = {1, 2, 3}
set3 = {"hutan", "kayu", "daun"}

gabung_set = set1.union(set2, set3)
print(gabung_set)
```

11) method union() juga bisa untuk menggabungkan sebuah himpunan dengan tipe data lain

```
x = {12, 7, 9}
y = ("ayam", "ikan", "tahu")

gabungan = x.union(y)
print(gabungan)
#contoh diatas adalah gabungan set dengan tuple
```

12) Method update()

digunakan untuk menambah semua elemen dari satu set ke set lainnya. Method ini mengubah set asli dan tidak menghasilkan set baru

```
bil1 = {10, 12, 11}
bil2 = {100, 120, 110}

bil1.update(bil2)
print(bil1)
```

13) Intersection
digunakan untuk menyimpan hanya elemen yang sama (duplikat) dari dua set.

```
bil1 = {10, 12, 11}
bil2 = {100, 120, 110}

bil1.update(bil2)
print(bil1)
```

14) mencari elemen yang sama (intersection) dari dua set menggunakan operator &

```
matkul_TIA = {"kwu", "bindo", "bing"}
matkul_TIB = {"kwu", "arsikom", "bindo"}

hasil = matkul_TIA & matkul_TIB
print(hasil)
```

15) Difference
digunakan untuk mengambil elemen yang hanya ada pada set pertama, tetapi tidak ada pada set kedua.

```
matkul_TIA = {"kwu", "bindo", "bing"}
matkul_TIB = {"kwu", "arsikom", "bindo"}

diff = matkul_TIA.difference(matkul_TIB)
print(diff)
```

16) Symmetric difference
digunakan untuk menyimpan elemen yang tidak sama dari dua set, atau dengan kata lain elemen yang hanya ada di salah satu set saja. Method ini akan menghasilkan set baru yang berisi elemen yang tidak terdapat pada kedua set secara bersamaan.

```
himpunan1 = {10, 20, 30, 40}
himpunan2 = {10, 15, 20, 25, 35}

hasil = himpunan1.symmetric_difference(himpunan2)
print(hasil)
```

17) Frozenset
adalah versi set yang bersifat immutable. Digunakan unntuk menyimpan elemen unik, tidak berurutan, dan tidak boleh ada duplikat. Frozenset dapat digunakan ketika membutuhkan struktur data seperti set, tetapi dengan isi yang tidak boleh diubah

```
contoh = frozenset(["apel", "pisang", "mangga"])
print(contoh)
```

D. Tuple Dictionaries

1) Membuat tuple
tuple adalah kumpulan yang terurut dan tidak dapat diubah. Tuple ditulis dengan tanda kurung biasa. Tuple juga dapat berisi berbagai tipe data.

```
data = ("hijau", "kuning", 20, True)
print(data)
```

2) Mengakses item tuple
untuk mengakses item tuple, dapat dengan merujuk pada nomor indeks yang berada di dalam tanda []

```
warna = ("hijau", "kuning", "merah")
print(warna[1])
#tuple juga bisa menggunakan negative indexing, seperti list
```

3) Change Tuple Values
Setelah sebuah tuple dibuat, nilainya tidak dapat diubah. Tuple bersifat immutable. Namun, terdapat cara alternatif (workaround) untuk mengubah nilai tuple, yaitu dengan mengubah tuple menjadi list, melakukan perubahan pada list tersebut, lalu mengubahnya kembali menjadi tuple

```
x = ("merah", "kuning", "hijau")
y = list(x)
y[1] = "jingga"
x = tuple(y)
print(x)
```

4) Add items
sama seperti cara mengubah nilai tuple, untuk menambahkan item ke dalam tuple kita bisa mengubah tuple menjadi list, menambahkan itemnya, kemudian mengubah kembali menjadi tuple

```
buah = ("apel", "pisang")

list_buah = list(buah)
list_buah.append("mangga")

buah = tuple(list_buah)
print(buah)
```

5) Unpacking tuple
saat kita membuat tuple dan langsung mengisinya dengan beberapa nilai, proses ini disebut packing tuple. Jadi unpacking tuple adalah proses mengeluarkan nilai-nilai dalam tuple ke beberapa variable

```
warna = ("merah", "biru", "hijau")
(w1, w2, w3) = warna

print(w1)
print(w2)
print(w3)
```

6) Using arterisk *
kalau jumlah variabel lebih sedikit daripada jumlah isi tuple, kita bisa pakai * supaya sisa nilainya masuk ke list

```
warna = ("merah", "biru", "hijau", "kuning", "ungu")

(warna1, warna2, *sisa_warna) = warna

print(warna1)
print(warna2)
```

7) Loop tuples
untuk loop through a tuple itu sama saja seperti loop through a list.
Loop through the index numbers. Kita bisa melakukan perulangan pada tuple dengan menggunakan nomor indeksnya, dengan menggabungkan fungsi range() dan len()

```
warna = ("merah", "biru", "hijau", "kuning")

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

8) While loop
perulangan pada tuple menggunakan while loop dilakukan dengan memanfaatkan indeks dari setiap elemen tuple. Nilai indeks dimulai dari 0 dan akan terus bertambah hingga mencapai jumlah elemen tuple yang ditentukan oleh fungsi len()

```
warna = ("merah", "biru", "hijau", "kuning")

i = 0
while i < len(warna):
    print(warna[i])
    i = i + 1
```

9) Join two tuples
untuk menggabungkan 2 atau lebih tuple, dapat menggunakan operator +

```
satu = ("bakwan", "pisang", "tahu")
dua = (5, 3, 2)

hasil = satu + dua
print(hasil)
```

10) Multiple tuples
menggandakan isi tuple sebanyak jumlah tertentu dengan menggunakan operator *

```
warna = ("merah", "biru", "hijau")
hasil = warna * 2

print(hasil)
```