
# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

| | |
|---|---|
| **NIM** | **71230982** |
| **Nama Lengkap** | **Rainie Fanita Chrisabel Hadisantoso** |
| **Minggu ke / Materi** | **11 / Tipe Data Tuples** |

**SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.**

**SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.**

**PROGRAM STUDI INFORMATIKA**
**FAKULTAS TEKNOLOGI INFORMASI**
**UNIVERSITAS KRISTEN DUTA WACANA**
**YOGYAKARTA**
**2024**

# MATERI : Tipe Data Tuples

Pada pertemuan materi ke 11 degan materi tuple, berikut beberapa topik yang ingin dicapai melalui pembelajaran ini: 1. Menjelaskan tuple, 2. Membandingkan isi tuple, 3. Melakukan perintah pada tuple, 4. Hubungan tuple dan dictionary, 5. Melakukan banyak perintah dengan dictionary, dan 6. Tuple sebagai key dictionary.

Hal pertama yang perlu dibahas adalah perbedaan antara tuple, list dan string.

1. String
   - Hanya menyimpan data berupa teks
   - Bersifat tidak dapat diubah atau immutable
   - Ditandai dengan tanda kutip “ ” atau ‘ ‘
   - Dapat diakses dan dimanipulasi dengan metode string dalam Python

```
string = "Hello, World!"
```

2. List
   - Dapat menyimpan berbagai tipe data yang berbeda
   - Bersifat dapat diubah atau mutable
   - Ditandai dengan tanda kurung siku []
   - Isi dapat diubah, ditambah, dihapus

```
lists = ["H", "e","l","l","o",",","," ","W","o","r","l","d","!"]
```

3. Tuple
   - Dapat menyimpan berbagai tipe data yang berbeda
   - Bersifat tidak dapat diubah atau immutable
   - Ditandai dengan tanda kurung ()
   - Dapat diakses dan dimanipulasi, digunakan ketika data tidak perlu berubah.

```
tup1 = "H", "e","l","l","o",",","," ","W","o","r","l","d","!"
tup2 = ("H", "e","l","l","o",",","," ","W","o","r","l","d","!")
```

| | list | tuple | string |
|---|---|---|---|
| *syntax* | a comma-separated list of values inside square brackets [] | possibly inside round brackets () | letters inside quotation marks “” |
| *what kind of values?* | all kinds, even different data types within a sequence | | only characters and strings |
| *mutable (i.e. values can be modified afterwards)* | yes, you can replace single elements with the slice and the bracket operator | no, but you can replace it with another one using the same name | |
| *nesting* | yes (i.e. list or tuple within a list or tuple) | | no |

## Tuple Immutable

Tuple mirip dnegan list. Keduanya dapat menyimpan nilai tipe data apapun dan memiliki indeks sunutk uruta  elemen-elemennya. Yang menjadi perbedaan antara keduanya adalah, list bersifat mutable atau dapat diubah, sedangkan tuple bersifat tidak dapat diubah atau immutable. Tuple tidak dapat diubah namin dapat dibandingkan dengan elemen-elemen lainnya. Selain itu, karena tidak dapat diubah tuple menjadi bersifat hashable sehingga dapat diubah menjadi list dan digunakan sebagai key pada dictionary dalam Python.

Hashability adalah sebutan sebuah data yang tidak dapat diubah-ubah sehingga dapat digunakan sebagai key dalam sebuah dictionary, selain itu data juga dapat dibandingkan dengan data-data lainya yang juga bernilai sama dengan data tersebut. Untuk hash dapat digunakan metode hash() pada data. Sedangkan untuk membandingkan data-data dapat digunakan metode eq() atau cm().

Penulisan tuple sebagai berikut:

```
tup1 = "H", "e","l","l","o",",","  ","W","o","r","l","d","!"
tup2 = ("H", "e","l","l","o",",","  ","W","o","r","l","d","!")
```

Jika hanya 1 elemen maka harus ditambahkan kome dibelakang elemennya. Jika tidak ada tanda koma dibelakang elemen maka akan dianggap sebagai string.

```
tup3 = ("Rain")
type(tup3)       #str
tup4 = ("Rain", )
type(tup4)       #tuple
```

Untuk mendefinisikan tipe data tuple, dapat digunakan fungsi build-in Python tuple().

```
tup = tuple()
type(tup)       #tuple
t = tuple("Hello, World!")
print(t)
#output : ("H", "e","l","l","o",",","  ","W","o","r","l","d","!")
```

Sebagian besar operator-operator untuk list dapat digunakan oleh tuple. Seperti penggunaan indeks untuk memanggil elemen dalam tuple. Penggunaan indeks pada tuple sama persis dengan cara-cara penggunaan indeks pada list.

## Membandingkan Tuple

Operator perbandingan dapat digunakan pada tuple juga. Caranya adalah elemen pertama menjadi patokan perbandingan untuk elemen-elemen lainnya. Jika kedua elemen sama maka akan lanjut ke elemen berikutnya da seterusnya hingga semua elemen habis atau ditemukan ada perbedaan diantara elemen-elemennya.

```
(0,1,2) < (3,4,5)   #True
(0,2,3) > (3,4,5)   #False
(0,1,2) == (0,1,2)  #True
```

Fungsi sort juga memiliki proses kerja yang sama. Fungsi sort juga mengurutkan elemen berdasarkan elemen pertama. Namun juga dapat dilakukan penrgurutan berdasarkan elemen lainya. Fungsi-fungsi ini disebut dengan Decorate, Sort, Undecorate atau disingkat DSU.

1. Decorate: Membangun daftar tuple dengan urutan sekuensial dengan beberapa key urutan sebelum daftar sekuensial tuple
2. Sort: mengubah tuple menjadi list kemudian mengurutkannya dari kecil ke besar, kalau mau sebaliknya perlu menggunakan perintah tambahan.
3. Undercorate: Mengeluarkan salah satu elemen dari daftar elemen yang sudah diututkan secara sekuensial.

## Penugasan Tuple

Kita dapat mendefinisikan tuple disisi kiri dari statement penugasan. Sehingga dapat diberikan lebih dari satu variabel yang didefinisikan disebelah kiri. Dan variabel tersebut akan dihubungkan dengan data dalam tuple sehigga menjadi key elemen tuple.

```
nama = ("Rainie", "Fanita", "Chrisabel","Hadisantoso")
a, b, c, d = nama
print(a)    #Rainie
print(b)    #Fanita
print(c)    #Chrisabel
print(d)    #Hadisantoso
```

Sebetulnya cara kerja penyimpanan data list dengan variabel-variabel disisi kiri adalahsebagai berikut:

```
nama = ["Rainie", "Fanita", "Chrisabel","Hadisantoso"]
a = nama[0]
b = nama[1]
c = nama[2]
d = nama[3]

print(a)    #Rainie
print(d)    #Hadisantoso
```

Untuk melakukan penugasan tersebut juga bisa digunakan dengan menggunakan tanda kurung. Hasil tanpa tanda kurung dan dengan tanda kurung memiliki outcome yang sama.

Selain itu bisa juga melakukan penukaran nilai variabel menggunakan penugasan variabel yang banyak.

```
a, b, c, d = d, c, b, a
print(a)    #Hadisantoso
print(b)    #Chrisabel
print(c)    #Fanita
print(d)    #Rainie
```

Penugasan dengan berbagai variabel daat dilakukan dengan syarat jumlah variabel yang ditugaskan sama dengan jumlah elemen-elemen yang ada dalam tuple. Jika jumlah variabel dan elemen tidak sama, maka akan terjadi error.

```
a, b = 1, 2, 3
#ValueError: too many values to unpack
```

Bisa juga dilakukan penugasan berbagai variabel pada sebuah string. Hal tersebut dapat dilakukan dengan pembagian data yang diinginkan, dan jumlah variabelnya menyesuaikan dengan kebutuhan pembagian.

```
email = "rainiefch@gmail.com"
username, domain = email.split("@")

print(username)     #rainiefch
print(domain)       #gmail.com
```

## Dictionaries and Tuple

Seperti yang sudah kita ketahui, dictionaries berisikan items. Tiap items terdiri dari key dan value disebut juga key-value pair. Key bertindak sebagai indeks, sedangkan value adalah nilai dari key. Untuk mengubah dictionary menjadi list berisi tuple, dapat digunakan perintah list()

```
dic = {"a":1 , "b":3, "c":2}
tup = list(dic.items())
print(tup)  #tup [("a",1), ("c",2), ("b",3)]
```

Namun dapat dilihat bahwa penyimpanan items tidak dilakukan secara berurutan. Sehingga urutan menjadi acak, tidak sesuai dengan urutan key yang digunakan, yang dalam hal ini adalah urutan alfabet. Karena itu perlu digunakan sort untuk mengurutkan ietsm berdasarkan key.

```
dic = {"a":1 , "b":3, "c":2}
tup = list(dic.items())
print(tup)  #tup [("a",1), ("c",2), ("b",3)]
tup.sort()
print(tup)  #tup [("a",1), ("b",3), ("c",2)]
```

Urutan elemen tuple dalam list dilakukan secara ascending secara default. Untuk secara descending perlu diguanakn perintah tambahan: Descending = True.

```
tup.sort(reversed = True)
print(tup)  #tup [("c",2), ("b",3), ("a",1)]
```

Artinya reverse = True adalah perintah dilakukan secara terbalik adalah benar. Sehingag ya diurutkan secara terbalik.

## Multipenugasan dengan dictionaries

Kita dapat mengkombinasikan items, tuple assignments dan for dalam sebuah kode. Bentuknya adalah, setiap key dan value pada sebuah dictionary dilakukan perulangan.

```
for key, val in list(d.items()):
    print(val, key)
```

Dari kode diatas, dilakukan perulangan untuk tiap pasangan key dan value dalam list berisikan items dictionary. Untuk tiap perulangan ditampilkan pasangan key dan value yang menjadi objek perulangan saat itu. Hasil dari kode diatas sebagai berikut:

```
a 1
b 3
c 2
```

## Kata yang sering muncul

Kode dibawah ini mengkombinasikan berbagai fungsi dan penugasan yang sudah dipelajari untuk menghitung jumlah kata yang muncul dalam sebuah file berisikan skrip romeo dan juliet. Namun dibatasi hanya menampilkan 10 kata yang paling sering muncul saja.

```
import string
fhand = open('romeo-full.txt')
counts = dict()
for line in fhand:
    line = line.translate(str.maketrans('', '', string.punctuation))
    line = line.lower()
    words = line.split()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1

lst = list()
for key, val in list(counts.items()):
    lst.append((val, key))

lst.sort(reverse=True)

for key, val in lst[:10]:
    print(key, val)
```

Penjelasan kode diatas sebagia berikut:

1. Program membuka file 'romeo-full.txt' untuk dibaca.
2. Membuat kamus (dictionary) kosong bernama 'counts' untuk menyimpan jumlah kemunculan setiap kata.
3. Program membaca setiap baris dalam file.
4. Setiap baris diubah menjadi lowercase dan tanda baca dihilangkan menggunakan metode translate() dan maketrans() dari modul string.
5. Setiap baris kemudian dibagi menjadi kata-kata dengan menggunakan metode split().
6. Program melakukan iterasi pada setiap kata dalam baris tersebut.
7. Untuk setiap kata, program memeriksa apakah kata tersebut sudah ada dalam kamus 'counts'. Jika belum ada, kata tersebut ditambahkan ke kamus dengan nilai 1. Jika sudah ada, nilai kemunculan kata tersebut ditambah 1.
8. Setelah selesai membaca seluruh baris, sebuah list bernama 'lst' dibuat.
9. Setiap pasangan kunci-nilai dari kamus 'counts' (kemunculan kata dan kata itu sendiri) diubah menjadi tupel dan dimasukkan ke dalam list 'lst'.
10. List 'lst' diurutkan secara menurun berdasarkan nilai kemunculan kata.
11. Hasilnya, sepuluh kata dengan frekuensi kemunculan tertinggi ditampilkan bersama dengan jumlah kemunculannya.

Hasil output dari data diatas adalah sebagai berikut:

```
61 i
42 and
40 romeo
34 to
34 the
32 thou
32 juliet
30 that
29 my
24 thee
```

## Tuple sebagai kunci dictionaries

Seperti yang sudah kita ketahui tuple itu hasbale dan list tidak. Jadi kita dapat menggunakan tuple sebagai composite key untuk digunakan pada sebuah dictionary. Contohnya pada pembuatan direktori telepon.

Dalam pembuatan direktori telepon, kita menggunakan pasangan last-name dan first-name sebagai kunci untuk memetakan ke nomor telepon dalam dictionary. Dengan menggunakan tuple sebagai kunci, kita dapat menyimpan informasi yang terkait dalam satu entitas hashable. Misalkan kita memiliki variabel last, first, dan nomor, kita dapat menetapkan nomor telepon ke direktori dengan menggunakan pernyataan seperti ini:

```
directory[last, first] = number
```

Pada looping for yang terkait dengan dictionary, kita dapat mengakses setiap kunci (pasangan last-name, first-name) dan mencetak nama dan nomor telepon yang sesuai:

```
for last, first in directory:
    print(first, last, directory[last,first])
```

Dengan cara ini, kita mengakses elemen dari setiap tuple kunci dan mencetak nama dan nomor telepon yang sesuai.

# BAGIAN 2: LATIHAN MANDIRI (60%)

## SOAL 11.1

Buatlah program untuk membandingkan isi sebuah tuple apakah sama semua atau tidak.

a. Source code

```
def comper(tup):
    hasil = all(isi == tup[0] for isi in tup)
    print(hasil)

tA = (90, 90, 90, 90)
comper(tA)
tB = (1, 2, 3, 4)
comper(tB)
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads\File\
1.py"
True
False
```

c. Penjelasan

Def comper untuk membandingkan data-data dalam sebuah tuple. Caranya variabel hasil untuk mengecek dengan fungsi all yaitu berlaku semua. Untuk isi dalam tuple adalah sama dengan data tuple pertama atau indeks ke 0. Kemudian diluar disajikan data dan dipanggil fungsi comper. Hasilnya adalah True atau False

## SOAL 11.2

Buat program menggunakan tuple untuk menampilkan data diri: NIM, nama depan, dan nama panjang terbalik.

a. Source code

```
data = ("Rainie Fanita Chrisabel Hadisantoso", "71230982", "Godean, DIY")

nim = tuple(data[1])
nama_depan = tuple(data[0].split()[0])
nama_terbalik = tuple(data[0].split()[::-1])
alamat = tuple(data[2])

print(f"NIM : {nim}")
print(f"NAMA DEPAN : {nama_depan}")
print(f"NAMA TERBALIK : {nama_terbalik}")
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads\File\
2.py"
NIM : ('7', '1', '2', '3', '0', '9', '8', '2')
NAMA DEPAN : ('R', 'a', 'i', 'n', 'i', 'e')
NAMA TERBALIK : ('Hadisantoso', 'Chrisabel', 'Fanita', 'Rainie')
```

c. Penjelasan

Pertama diberikan variabel data bertipe tuple berisi NIM, nama panjang dan alamat.Data NIM dalam tuple diambil dengan indeks 1 dan disimpan dalam variabel nim. Data nama depan diambil dari indeks ke 0 dan di split hanya mengambil indeks 0 dari tuple nama. Untuk nama terbalik diambil dari indeks data ke 0 dan kemudian displit dari urutan belakang dengan cara indeks -1 hingga 0. Kemudian masing-masing variabel dengan data yang dibutuhkan ditampilkan.

## SOAL 11.3

Membuat program untuk menghitung jumlah email terkirim tiap jamnya dalam file mbox-short.txt.

a. Source code

```
jumlah = dict()
file = input("Masukkan nama file: ")
try:
    f = open(file)
except:
    print("File tidak dapat dibuka")
    exit()

for baris in f:
    if baris.startswith("From "):
        waktu = baris.split()[5]
        jam = waktu.split(':')[0]
        jumlah[jam] = jumlah.get(jam, 0) + 1

urut = sorted(jumlah.items())

for jam, jumlah in urut:
    print(jam, jumlah)
```

b. Output

```
PS C:\Users\asus\Downloads\File> python -u "c:\Users\asus\Downloads\File\
3.py"
Masukkan nama file: mbox-short.txt
04 3
06 1
07 1
09 2
10 3
11 6
14 1
15 2
16 4
17 2
18 1
19 1
```

c. Penjelasan

Pertama dibuat variabel jumlah dengan tipe dictionary untuk menyimpan data jam dan julahnya. Jadi nanti bentuk isi tuplenya adalah [ jam1: jumlah , jam2:jumlah, dst]. Kemudian input nama file yang mau di buka. Jika bisa dibuka maka lanjut jika tidak aka akan menampilkan pesan error dan program dihentikan. Jika bisa dibuka dilakukan perulangan untuk tiap baris dalam file teks. Jika baris file teks diawali dengan “From” maka akan dijalankan percabangan. Variabel waktu berisi isi baris dipisah-pisah dengan split() di indeks ke 5, karena jam adalah kata ke 6 dalam baris. Variabel waktu untuk menyimpan jam saja dengan cara memanggil string dalam waktu kemudian dipisah berdasarkan “:” dan hanya diambil string pertama atau indeks ke 0. Lalu untuk menghitung jumlah digunakan dictionary jumlah, untuk tiap jma yang didapat akan dimasukkan kedalam dictionary dengan cara menghitung jumlah yang sudah ada dalam dictionary kemudian ditambah 1. Baru dictionary jumlah diubah menjadi tuple dengan fungsi sorted() berdasarkan items didalamnya. Tuple jam dan jumlah disimpan dalam variabel urut, kemudian ditampilkan dengan perulangan untuk tiap jam dan jumlah didalam variabel urut.

Link Github:

https://github.com/rainiefch/Praktikum-Algoritma-Pemrograman_71230982/tree/12cb6d39cfc353f7878004a0da9d222d162d4361/Latihan%2011