11. Tipe Data Tuples

11.1 Tujuan Praktikum

Setelah mempelajari Bab ini, mahasiswa diharapkan dapat:

1. Dapat menjelaskan tentang tuple.
2. Dapat membandingkan tuple.
3. Dapat memberikan penugasan pada tuple.
4. Dapat menjelaskan hubungan antara tuple dan dictionary.
5. Dapat memberikan multi penugasan dengan dictionary.
6. Dapat menggunakan tuple sebagai kunci dalam dictionary.

11.2 Alat dan Bahan

Praktikum ini membutuhkan perangkat komputer yang memiliki spesifikasi minimum sebagai berikut:

1. Terkoneksi ke Internet dan dapat mengunduh package-package Python.
2. Mampu menjalankan sistem operasi Windows 10 atau Ubuntu Linux.

Perangkat lunak yang diperlukan untuk mendukung praktikum ini adalah sebagai berikut:

1. Python 3.7 atau 3.8 yang terinstall menggunakan Anaconda atau Installer Python lainnya.
2. Web Browser (Mozilla Firefox, Microsoft Edge atau Google Chrome).
3. Command Prompt (jika menggunakan Windows).
4. Terminal (jika menggunakan Linux).
5. Editor Python (Visual Studio Code, PyCharm, Spyder atau editor-editor lainnya yang mendukung Python).

11.3 Materi

11.3.1 *Tuple Immutable*

Tuple bisa dikatakan menyerupai list. Nilai yang disimpan dalam *tuple* dapat berupa apapun dan diberikan indeks bilangan bulat *(integer)*. Perbedaan utama adalah sifat *tuple* yang *immutable*, yaitu dimana anggota dalam *tuple* tidak bisa dirubah. Tuple sendiri dapat dibandingkan *(compare)* dan bersifat *hashable* sehingga dapat dimasukkan dalam *list* sebagai *key* pada *dictionary* python.

Sebuah objek disebut *hashable* jika memiliki nilai *hash* yang tidak pernah berubah selama hidupnya (menggunakan method **__hash__**()), dan dapat dibandingkan dengan benda-benda lain (menggukanan method **__eq__**() atau **__cm__**()). Obyek *Hashable* membandingkan yang sama dengan memiliki nilai hash yang harus sama.

Hashability membuat sebuah objek yang dapat digunakan sebagai kamus kunci dan mengatur anggota, karena struktur data ini menggunakan nilai *hash* secara internal.

Objek pada pyhton *built-in* biasanya *hashable*, sementara tidak bisa berubah wadah (seperti daftar atau kamus) adalah. Objek yang merupakan *instance* dari kelas yang didefinisikan pengguna yang *hashable* secara default; mereka semua membandingkan yang tidak sama, dan mereka nilai hash adalah **id**().

selengkapnya : https://docs.python.org/3/glossary.html

Sintaks penulisan *tuple* :

```
>>> t = 'a','b','c','d','e'
```

Sintaks lain penulisan *tuple* dapat menggunakan tanda kurung:

```
>>> t = ('a','b','c','d','e')
```

Tuple dengan satu element, ditambahkan koma (**,**) dibelakang penulisan *tuple*.

```
>>> t1 = ('a',)
>>> type(t1)
<type 'tuple'>
```

Jika tidak menanbahkan tanda koma, akan dianggap sebagai *string*.

```
>>> t2 = ('a')
>>> type(t2)
<type 'str'>
```

Model sintaks lain dengan menggunakan fungsi ***built-in*** *tuple*. Ketika tidak diisi dengan argument apapun, secara langsung akan membentuk *tuple* kosong.

```
>>> t = tuple()
>>> pint(t)
()
```

Jika argumennya berupa urutan (*string, list,* atau *tuple*), akan mengembalikan nilai *tuple* dengan elemen-elemen yang berurutan.

```
>>> t = tuple('dutawacana')
>>> pint(t)
('d','u','t','a','w','a','c','a','n','a')
```

Tuple merupakan nama dari ***constructor***, kita tidak bisa menggunakannya sebagai nama variabel. Sebagian besar operator yang bekerja pada *list* bekerja juga pada *tupple*. Tanda kurung kotak [ ] menandakan indeks element dari *tuple*.

```
>>> t = ('a', 'b', 'c', 'd', 'e')
>>> print(t[0])
'a'
```

Untuk menampilkan rentang nilai dari element *tuple* dapat menggunakan:

```
>>> print(t[1:3])
('b', 'c')
```

Tuple bersifat *immutable* artinya element pada *tuple* tidak dapat berubah. Jika anda berusaha mengubah *tuple*, maka akan didapatkan error.

```
>>> t[0] = 'A'
TypeError: object doesn't support item assignment
```

Element pada *tuple* tidak dapat dirubah tetapi dapat diganti dengan elemen lain.

```
>>> t = ('A',) + t[1:]
>>> print(t)
('A', 'b', 'c', 'd', 'e')
```

11.3.2 Membandingkan *Tuple*

Opertor perbandingan *(compare)* dapat bekerja pada *tuple* dan model sekuensial lainnya (*list, dictionary,set*). Cara kerjanya dengan membandingkan elemen pertama dari setiap sekuensial yang ada. Jika ditemukan adanya kesamaan, akan berlanjut ke elemen berikutnya. Proses ini berlangsung terus menerus hingga ditemukan adanya perbedaan.

```
>>> (0, 1, 2) < (0, 3, 4)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
```

Fungsi ***(sort)*** pada python bekerja dengan cara yang sama. Tahap pertama akan melakukan pengurutan berdasarkan elemen pertama. Pada kasus tertentu akan mengurutkan berdasarkan elemen kedua dan sebagainya. Fitur ini disebut dengan DSU – ***(Decorate, Sort, Undercorate)***

1. ***Decorate*** – urutan (sekuensial) membangun daftar *tuple* dengan satu atau lebih *key* pengurutan sebelum eleman dari urutan (sekuensial).
2. ***Sort*** – *list tuple* menggunakan *sort* (fungsi bawaan di python).
3. ***Undercorate*** – melakukan ekstraksi pada elemen yang telah diurutkan pada satu sekuensial.

Untuk memahaminya, silakan perhatikan contoh dibawah ini. Kita mempunyai sebuah kalimat dan akan melakukan pengurutan kata dalam kalimat tersebut dari yang paling panjang ke yang paling pendek.

```
kalimat = 'but soft what light in yonder window breaks'
dafkata = kalimat.split()
t = list()
for kata in dafkata:
    t.append((len(kata), kata))

t.sort(reverse=True)

urutan = list()
for length, kata in t:
    urutan.append(kata)

print(urutan)
```

Looping yang pertama akan membuat daftar *tuple*, yang berisi daftar kata sesuai dengan panjangnya. Fungsi ***sort*** akan membandingkan elemen pertama dari list dari panjang kata yang ada dan kemudian akan menuju ke elemen kedua jika kondisi sesuai. *Keyword reverse=True* digunakan untuk melakukan urutan secara terbalik *(descending)*.

Looping kedua akan membangun daftar *tuple* dalam sesuai urutan alfabet diururtkan berdasarkan panjangnya. Pada contoh diatasm Empat kata dalam kalimat akan diurutkan secara alfabet terbalik. Kata *"what"* akan muncul sebelum *"soft"* didalam *list*.

Output program diatas dapat dilihat dibawah ini:

```
['yonder', 'window', 'breaks', 'light', 'what',
'soft', 'but', 'in']
```

Kata-kata yang muncul diurutkan sebagai *list* dengan urutan panjang kata dari yang paling panjang ke kata yang paling pendek.

11.3.3 Penugasan *Tuple*

Salah satu fitur unik dari Python adalah kemampuannya untuk memiliki *tuple* disisi kiri dari statement penugasan. Hal ini mengijinkan untuk menetapkan lebih dari satu variabel pada sisi sebelah kiri secara berurutan.

Contohnya ketika ada dua daftar elemen yang merupakan urutan. Kita akan mencoba menetapkan elemen pertama dan kedua dari urutan tersebut kedalam variabel *x* dan *y* dalam satu *statement*.

```
>>> m = [ 'have', 'fun' ]
>>> x, y = m
>>> x
'have'
>>> y
'fun'
>>>
```

Python akan menterjemahkan sintaks *tuple* dalam langkah-langkah sebagai berikut:

```
>>> m = [ 'have', 'fun' ]
>>> x = m[0]
>>> y = m[1]
>>> x
'have'
>>> y
'fun'
>>>
```

Tuple yang digunakan pada contoh diatas menggunakan statement penugasan disebelah kiri dan tidak menggunakan tanda kurung. Berikut ini merupakan contoh yang sama dengan menggunakan tanda kurung *(parentheses)*.

```
>>> m = [ 'have', 'fun' ]
>>> (x, y) = m
>>> x
'have'
>>> y
'fun'
 >>>
```

Tuple memungkinkan pula untuk menukar nilai variabel dalam satu statement.

```
>>>> a, b = b, a
```

Dari contoh diatas, dapat dilihat bahwa kedua statemnt merupakan *tuple*. Bagian kiri merupakan *tuple* dari variable dan bagian kanan merupakan tuple dari *expresions*. Tiap nilai pada bagian kanan diberikan/ditugaskan ke masing-masing variable di sebelah kiri. Semua *expresions* pada bagian kiri dievaluasi sebelum diberikan penugasan.

Yang perlu diperhatikan adalah jumlah variabel antara sisi kiri dan sisi kanan harus sama.

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

Pada sisi sebelah kanan biasanya terdapat data sekuensial *string, list* atau *tuple*. Sebagai contoh, kita akan membagi alamat email menjadi *user name* dan *domain* seperti berikut ini.

```
>>> email = 'didanendya@ti.ukdw.ac.id'
>>> username, domain = email.split('@')
```

Nilai yang dikembalikan dari **split** terdiri dari dua elemen yang dipisahkan tanda "@', elemen pertama berisi **username** dan elemen selanjutnya berisi **domain**.

```
>>> print(usernam)
didanendya
>>> print(domain)
ti.ukdw.ac.id
```

11.3.4 *Dictionaries and Tuple*

Dictionaries mempunyai metode yang disebut **items** untuk mengembalikan nilai *list* dari *tuple* dimana tiap *tuple*-nya merupakan *key-value pair* (pasangan kunci dan nilai).

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = list(d.items())
>>> print(t)
[('b', 1), ('a', 10), ('c', 22)]
```

Seperti pada *dictionary* umumnya, *item* yang dikembalikan nilainya tidak berurutan. Bagaimanapun juga, *list* dari *tuple* adalah ***list***, dan tuple membandingkannya sehingga kita dapat melakukan pengurutan *(sort)* pada *tuple*. Melakukan konversi *dictionary* dari *list tuple* merupakan cara untuk menampilkan isi *dictionary* yang diurutkan berdasarkan kuncinya.

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = list(d.items())
>>> t
[('b', 1), ('a', 10), ('c', 22)]
>>> t.sort()
>>> t
[('a', 10), ('b', 1), ('c', 22)]
```

List yang muncul diurukan secara *ascending* berdasarkan alfabet dan *key value*.

11.3.5 Multipenugasan dengan *dictionaries*

Kombinasi antara ***items, tupple assignment*** dan ***for*** akan menghasilkan kode tertentu pada *keys* dan *values* dari *dictionary* dalam satu loop.

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

Pada bagian looping ini, terdapat 2 iterasi variabel karena *item* mengembalikan *list* dari *tuple* dan ***key***. ***val*** merupakan penugasan *tupple* yang melakukan iterasi berulang melalui pasangan *key-value* pada *dictionary*.

Pada setiap iterasi yang melalui loop, baik ***key*** dan ***value*** akan bergerak maju menuju pasangan *key-value* berikutnya pada *dictionary* (masih dalam urutan *hash*).

Output dari looping diatas

```
10 a
22 c
1 b
```

Jika dilakuan penggabungan dari kedua teknik diatas, kita dapat melakukan pencetakan isi *dictionary* yang diurutkan berdasarkan nilai yang disimpan di setiap pasangan *key-value*.

Untuk melakukannya langkah pertama dengan membuat *list tuple* di mana masing-masing *tuple* beranggotakan ***(value,key)***. Method ***item*** akan memberikan list *tupel* ***(value,key)*** dan akan melakukan pengurutan berdasarkan *value*, bukan *key*. Setelah daftar *list* dengan *value-key* tuple terbentuk, langkah selanjutnya mengurutkan *list* tersebut dengan urutan terbalik dan melakukan pencetakan *list* baru yang disortir.

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> l = list()
>>> for key, val in d.items() :
...     l.append( (val, key) ) ...
>>> l
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> l.sort(reverse=True)
>>> l
[(22, 'c'), (10, 'a'), (1, 'b')]
>>>
```

Dengan melakukan penyusunan *list* dari *tuple* yang memiliki *value* sebagai elemen pertama, akan mudah untuk mengurutkan *list* dari *tuple* tersebut dan mendapatkan isi *dictionary* yang diurutkan berdasarkan nilainya.

11.3.6 Kata yang sering muncul

Pada bagian ini kita akan mencoba menampilkan kata yang sering muncul pada text dari *Romeo and Juliet Act 2, Scene 2*. Silakan download romeo-full.txt pada bagian materi. Kita akan coba membuat program menggunakan *list* dari *tuple* untuk menampilkan 10 kata yang paling sering muncul.

```
1  import string
2  fhand = open('romeo-full.txt')
3  counts = dict()
4  for line in fhand:
5      line = line.translate(str.maketrans('', '', string.punctuation))
6      line = line.lower()
7      words = line.split()
8      for word in words:
9          if word not in counts:
10             counts[word] = 1
11         else:
12             counts[word] += 1
13
14 # urutkan dictionary by value
15 lst = list()
16 for key, val in list(counts.items()):
17     lst.append((val, key))
18
19 lst.sort(reverse=True)
20
21 for key, val in lst[:10]:
22     print(key, val)
```

Bagian pertama dari program digunakan untuk membaca file dan melalukan komputasi terhadap *dictionary* yang memetakan tiap kata ke sejumlah kata dalam dokumen yang tidak berubah. Dengan menambahkan *print out* ***counts***, list tuple *(val, key)* dibuat dan diurutkan dalam list dengan urutan terbalik.

Nilai pertama akan digunakan dalam perbandingan. Jika ada lebih dari satu *tupel* dengan nilai yang sama, kemudian akan melihat ke elemen kedua (*key*), sehingga *tuple* di mana nilainya yang sama akan diurutkan berdasarkan urutan abjad sesuai *key*.

Pada bagian akhir, ada looping yang melakukan iterasi penugasan ganda dan mencetak 10 kata yang paling sering muncul dengan melakukan perulangan pada list **lst[:10]:**.

Secara lengkap output dari program diatas akan memampilkan kata yang sesuai untuk frekuesni analisis.

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

Penguraian dan analisis data yang kompleks ini dapat dilakukan dengan 19 baris program python dipahami. Hal tersebut merupakan salah satu alasan mengapa python adalah pilihan yang baik sebagai bahasa untuk mengeksplorasi informasi.

11.3.7 Tuple sebagai kunci *dictionaries*

Tuple merupakan *hashable* dan *list* tidak. Ketika kita ingin membuat *composite key* yang digunakan dalam *dictionary*, kita dapat menggunakan tuple sebagai *key*.

Misalnya menggunakan *composite key* jika ingin membuat direktori telepon yang memetakan dari pasangan *last-name*, *first-name* ke nomor telepon Dengan asumsi bahwa kita telah mendefinisikan variabel *last*, *first*, dan nomor. Penulisan pernyataan penugasan dalam *dictionary* sebagai berikut:

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

Expression yang ada didalam kurung kotak adalah *tuple*. Langkah selanjutnya dengan memberikan penugasan *tuple* pada looping **for** yang berhubungan dengan dictionary.

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

Looping yang berhubungan dengan *keys* pada direktori diatas merupakan *tuple*. Elemen dari masing-masing *tuple* ditetapkan pertama kali, kemudaian dilakukan pencetakan nama dan nomor telepon yang sesuai.

```
>>> last = 'nendya'
>>> first = 'dida'
>>> number = '088112266'
>>> directory = dict()
>>> directory[last, first] = number
```

```
>>> for last, first in directory:
...     print(first, last, directory[last,first])
...
dida nendya 088112266
>>>
```

11.4 Kegiatan Praktikum

Kasus 11.1 Buatlah program yang dapat melakukan proses berikut:

1. Membuat dan mencetak tuple.
2. Membagi tuple kedalam string.
3. Membuat tuple yang berisi string dari sebuah kata.
4. Membuat tuple yang berisi semua, kecuali huruf pertama dari sebuah string
5. Mencetak tuple secara terbalik

```
Contoh:

kota:  ('Jakarta', 'Jogja', 'Surabaya')
kota[0]:  Jakarta
kota[1]:  Jogja
kota[2]:  Surabaya

str1:  Jakarta
str2:  Jogja
str3:  Surabaya

tpl:  ('B', 'e', 'l', 'a', 'j', 'a', 'r', ' ', 'P', 'y', 't', 'h', 'o', 'n')

tpl1:  ('e', 'l', 'a', 'j', 'a', 'r', ' ', 'P', 'y', 't', 'h', 'o', 'n')

urutan:  ('Jaka', 'Joko', 'Jono')
urutan_terbalik:  ('Jono', 'Joko', 'Jaka')
```

Pembahasan Kasus 11.1

Program diatas merupakan fungsi dasar tuple pada pemrograman python. Untuk lebih jelasnya silakan perhatikan kode program berikut ini.

```
'''
1) Membuat dan mencetak tuple.
'''
kota = ("Jakarta", "Jogja", "Surabaya")
print("kota: ", kota)
print("kota[0]: ", kota[0])
print("kota[1]: ", kota[1])
print("kota[2]: ", kota[2])
print() # prints baris

'''
2) Membagi tuple kedalam string.
```

```
'''
str1, str2, str3 = kota
print("str1: ", str1)
print("str2: ", str2)
print("str3: ", str3)
print()

'''
3) Membuat tuple yang berisi string dari sebuah kata.
'''
tpl = tuple("Belajar Python")
print("tpl: ", tpl)
print()

'''
4) Membuat tuple yang berisi semua, kecuali huruf pertama dari sebuah string
'''
# direct string
tpl1 = tuple("Belajar Python"[1:])
print("tpl1: ", tpl1)


'''
5) Mencetak tuple secara terbalik
'''
name = ("Jaka", "Joko", "Jono")
# using slicing technique
rev_name = name[::-1]
rev2= name[::-2]
print("urutan: ", name)
print("urutan_terbalik: ", rev_name)
```

Kasus 11.2 Buatlah program yang dapat menghitung jumlah commit yang dilakukan oleh akun email tertentu. Gunakan file mbox-short.txt atau mbox.txt

```
Enter a file name: mbox-short.txt
cwen@iupui.edu 5

Enter a file name: mbox.txt
zqian@umich.edu 195
```

Pembahasan Kasus 11.2

Untuk mengerjakan kasus 11.2 beberapa langkah yang harus dilakukan adalah :

1. Membaca file text
2. Parsing **From** dan line dan hitung pesan yang ada dalam dictionary.
3. Buat list untuk mencetak tuple (email, jumlah).
4. Gunakan urutan terbaik untuk mencetak email yang paling banyak melakukan commit.

`Gunakan pola : From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008 untuk parsingnya.`

Berdasarkan pengecekan yang telah kita lakukan diatas, dapat dibuat kode programmnya sebagai berikut:

```
dic_email = dict()            # variables
lst = list()

fname = input('Nama File: ')
try:
    fhand = open(fname)
except FileNotFoundError:
    print('File tidak bisa dibuka:', fname)
    quit()

for baris in fhand:
    kata = baris.split()
    if len(kata) < 2 or kata[0] != 'From':
        continue
    else:
        if kata[1] not in dic_email:
            dic_email[kata[1]] = 1        # entri 1
        else:
            dic_email[kata[1]] += 1       # +1

for key, val in list(dic_email.items()):
    lst.append((val, key))                # isi daftar dengan nilai kunci dict

lst.sort(reverse=True)                    # sort nilai tertinggi

for key, val in lst[:1]:                  # tampilkan nilai pertama (terbesar)
    print(val, key)
```

Silahkan lihat hasil programnya!

11.5 Latihan Mandiri

Latihan 11.1 Buatlah program untuk melakukan pengecekan apakah semua anggota yang ada didalam tuple sama.

```
Contoh:
tA= (90, 90, 90, 90)

Output
True
```

■

Latihan 11.2 Buatlah program dengan menggunakan tuple yang dapat melalakukan proses seperti pada kasus 11.1, Gunakan data diri anda masing-masing dan lakukan perubahan supaya didapatkan output seperti contoh berikut ini :

```
Contoh:
Data:  ('Matahari Bhakti Nendya', '22064091', 'Bantul, DI Yogyakarta')

NIM    :  22064091
NAMA   :  Matahari Bhakti Nendya
ALAMAT :  Bantul, DI Yogyakarta

NIM:  ('2', '2', '0', '6', '4', '0', '9', '1')

NAMA DEPAN:  ('a', 't', 'a', 'h', 'a', 'r', 'i')

NAMA TERBALIK:  ('Nendya', 'Bhakti', 'Matahari')
```

■

Latihan 11.3 Buatlah program untuk menghitung distribusi jam dalam satu hari dimana ada pesan yang diterima dari setiap email yang masuk. Gunakan file mbox-short.txt untuk sebagai datanya. Berikut ini adalah contoh output dari programmnya.

```
Contoh:
Enter a file name: 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
```

■