



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230976
Nama Lengkap	Galih Pramana Chandra Prasetya
Minggu ke / Materi	11 / Tipe Data Tuple

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

BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

Tuple Immutable

Tuple bisa dikatakan mirip dengan list. Nilai yang disimpan dalam tuple dapat terdiri dari berbagai jenis data dan diindeks dengan bilangan bulat (Integer).

Tuple dan list memiliki perbedaan utama, yaitu sifat tuple yang immutable, artinya elemen-elemen dalam tuple tidak bisa diubah setelah didefinisikan. Berbeda dengan list yang mutable dan elemen-elemennya dapat diubah kapan saja. Selain itu, tuple bisa dibandingkan (compare) dan bersifat hashable, yang memungkinkan mereka digunakan sebagai kunci dalam dictionary di Python, sementara list tidak memiliki sifat tersebut.

Sebuah objek disebut hashable jika memiliki nilai hash yang tetap (menggunakan method `__hash__()`) dan dapat dibandingkan dengan objek lain (menggunakan method `__eq__()` atau `__cmp__()`). Objek hashable yang sama harus memiliki nilai hash yang sama.

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

Sintaks penulisan tuple :

```
t1 = 'a','b','c','d','e'
```

Sintaks lain untuk menulis tuple adalah dengan menggunakan tanda kurung. Misalnya:

```
t2 = ('a','b','c','d','e')
```

Untuk membuat tuple dengan satu elemen, perlu ditambahkan koma di belakang elemen tersebut. Contohnya:

```
single_element_tuple = ('abcde',)
(type(single_element_tuple)
#<type 'tuple'>
```

Jika tidak menambahkan tanda koma, itu tidak akan dianggap sebagai tuple, melainkan sebagai tipe data dari elemen yang ada di dalamnya. Contohnya:

```
single_element = ("hello")
(type(single_element)
# <type 'str'>
```

Kita dapat menggunakan fungsi built-in tuple untuk membuat tuple kosong secara langsung tanpa argumen.

```
my_tuple = tuple()
print(my_tuple)
# ()
```

Jika argumennya berbentuk urutan seperti string, list, atau tuple, fungsi tersebut akan menghasilkan sebuah tuple dengan elemen-elemen yang berurutan.

```
t4 = tuple('dutawacana')
print(type(t4))
# ('d', 'u', 't', 'a', 'w', 'a', 'c', 'a', 'n', 'a')
```

Tuple adalah nama **constructor**, yang tidak dapat digunakan sebagai nama variabel. Sebagian besar operator yang beroperasi pada *list* juga berlaku untuk *tuple*. Tanda kurung kotak [] menunjukkan indeks elemen dalam tuple.

```
t4 = tuple('dutawacana')
print(t4[3])
# a
```

Untuk menampilkan rentang nilai dari elemen tuple dapat menggunakan,

```
t4 = tuple('dutawacana')
print(t4[1:3])
# ('a', 'w')
```

Tuple bersifat immutable (tidak dapat diubah), tapi elemennya bisa diganti.

```
t4 = tuple('dutawacana')
t4 = ('D',) + t4[3:]
print(t4)
# ('D', 'a', 'c', 'a', 'n', 'a')
```

Membandingkan Tuple

Operator perbandingan memungkinkan kita untuk membandingkan dua nilai atau lebih. Kemampuan ini juga berlaku untuk struktur data sekuensial seperti tuple, list, dictionary, dan set.

Operator perbandingan pada sekuensial bekerja dengan membandingkan elemen pertama dari setiap sekuensial. Jika elemen pertama sama, operator perbandingan akan membandingkan elemen berikutnya. Proses ini akan terus berulang sampai ditemukan elemen yang berbeda. Jika semua elemen sama, maka sekuensial dianggap sama.

Saat menggunakan fungsi sort pada Python, elemen pertama dari setiap item dalam data yang akan diurutkan akan dibandingkan terlebih dahulu. Jika elemen pertama sama, maka elemen kedua akan dibandingkan, dan seterusnya. Proses ini berlanjut hingga semua elemen telah dibandingkan. Fitur DSU (***Decorate, Sort, Undercorate***) memungkinkan pengurutan multi-level ini dengan cara yang fleksibel dan efisien.

1. ***Decorate***: Membuat daftar tuple dengan key pengurutan di depan elemen dalam urutan (sekuensial).
2. ***Sort***: Mengurutkan daftar tuple dengan fungsi sort bawaan Python.
3. ***Undercorate***: Mengambil elemen dari urutan (sekuensial) yang telah diurutkan.

Contoh:

```
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)
# t.sort(reverse=False)
urutan = list()
for length, kata in t:
    urutan.append(kata)

print(urutan)
#['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.

Penugasan Tuple

Salah satu fitur unik Python adalah kemampuannya menggunakan tuple di sisi kiri pernyataan penugasan. Ini memungkinkan penetapan beberapa variabel di sisi kiri secara berurutan.

Misalnya, ketika kita memiliki dua daftar elemen yang terurut, kita bisa menetapkan elemen pertama dan kedua dari urutan tersebut ke dalam variabel x dan y dalam satu pernyataan penugasan. Kemudian Python akan menterjemahkan sintaks tuple dalam langkah-langkah sebagai berikut:

```
m = [ 'have', 'fun' ]
x = m[0]
y = m[1]
print(x)
# 'have'
print(y)
# 'fun'
```

Tuple yang digunakan pada contoh di atas berada di sisi kiri dari pernyataan penugasan tanpa menggunakan tanda kurung. Berikut adalah contoh yang sama dengan menggunakan tanda kurung (parentheses).

```
m = [ 'have', 'fun' ]
(x, y) = m
x = m[0]
y = m[1]
print(x) # 'have'
print(y) # 'fun'
```

Tuple memungkinkan pula untuk menukar nilai variable dalam satu statement. Sebagai contoh, **a, b = b, a** merupakan statement tuple. Bagian kiri merupakan tuple dari variable dan bagian kanan merupakan tuple dari expressions. Tiap nilai pada bagian kanan diberikan/ditugaskan ke masing-masing variable di sebelah kiri. Semua expresions pada bagian kiri dievaluasi sebelum diberikan penugasan. Perlu diperhatikan bahwa jumlah variable antara sisi kiri dan sisi kanan harus sama.

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, g, t = email.split('.')
print(username)
# didanendya
print(domain)
# ti.ukdw.ac.id
```

Dictionaries and Tuple

Dictionary di Python memiliki metode bawaan bernama **items()**. Metode ini mengembalikan nilai list yang berisi pasangan kunci-nilai dari dictionary. Setiap elemen dalam list yang dihasilkan adalah tuple yang terdiri dari dua elemen:

1. Elemen pertama adalah **kunci** dari pasangan.
2. Elemen kedua adalah **nilai** dari pasangan.

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

Seperti dictionary biasa, item yang dihasilkan dari metode **items()** tidak memiliki urutan. Namun, karena list yang dihasilkan adalah list of tuple, tuple itu sendiri memiliki kemampuan untuk dibandingkan,

sehingga urutannya dapat diubah dengan operasi **sort**. Mengubah dictionary menjadi list of tuple merupakan cara untuk menampilkan isi dictionary yang diurutkan berdasarkan kuncinya.

Multipenugasan dengan dictionaries

Kita dapat menggunakan **items()**, **tuple assignment**, dan **for** untuk mengiterasi dictionary dan menampilkan kunci dan nilai dalam satu loop.

Mengurutkan dictionary berdasarkan nilainya (bukan kuncinya) dengan membuat daftar tuple (nilai, kunci), mengurutkannya, dan membalik urutannya.

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

l.sort(reverse=True)
print(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.

Kata yang sering muncul

Pada bagian ini kita akan mencoba menampilkan kata yang sering muncul pada text (dapat menggunakan file berformat txt).

```
import string
fhand = open('romeofull.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
# urutkan dictionary by value
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)
```

```
# 61 romeo
# 56 juliet
# 14 thee
# 11 above
# 10 love
# 8 name
# 7 again
# 6 night
# 6 me
# 6 here
```

Program ini dimulai dengan membaca file dan menghitung kemunculan kata-kata. Hasilnya disimpan dalam dictionary yang memetakan setiap kata ke jumlah kemunculannya dalam dokumen. Dictionary ini kemudian diubah menjadi list tuple (nilai, kunci) dan diurutkan dalam urutan terbalik. Urutan diubah karena nilai pertama (jumlah kemunculan) digunakan untuk perbandingan. Jika ada beberapa kata dengan jumlah kemunculan yang sama, urutan abjad berdasarkan kunci (kata) digunakan untuk membedakannya. Pada bagian akhir, program melakukan iterasi 10 kali, di mana setiap iterasi mencetak kata dan jumlah kemunculannya dari list yang berisi 10 kata yang paling sering muncul.

Tuple sebagai kunci dictionaries

Kita dapat menggunakan tuple sebagai kunci dictionary karena tuple dapat di-hash, sedangkan list tidak.

Kunci komposit (tuple) berguna untuk struktur data kompleks, seperti direktori telepon (nama belakang, nama depan ke nomor telepon). Hal ini memungkinkan pencarian mudah dengan nama. Penulisan pernyataan penugasan dalam dictionary sebagai berikut:

- `directory[last,first] = number`

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

Looping pada direktori dengan kunci tuple:

1. **Inisialisasi variabel:** Menetapkan elemen tuple (nama belakang, nama depan) sebagai variabel.
2. **Iterasi:** Mengulangi setiap tuple dalam direktori.
3. **Pencetakan:** Mencetak nama dan nomor telepon yang sesuai dengan tuple saat ini.

```
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
```


BAGIAN 2: LATIHAN MANDIRI (60%)

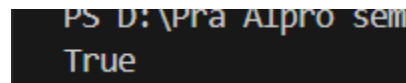
Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

SOAL 1

SC :

```
#latman 11.1
tA = (90,90,90,90)
tB = tA[0]
for i in tA:
    if i == tB:
        tC = ("True")
    else:
        print(False)
        exit()
print(tC)
```

Output:

A screenshot of a terminal window with a black background. The text 'PS D:\PRA Alpro sem' is visible on the first line, and 'True' is printed on the second line.

Penjelasan:

- Membuat variable **tA** yang berisi empat elemen dengan nilai 90.
- Selanjutnya, nilai pertama dari tuple **tA** disalin ke variabel **tB**.
- Program melakukan iterasi melalui setiap elemen di dalam tuple **tA**.
- Dalam setiap iterasi, dicek apakah nilai saat ini sama dengan nilai yang disalin ke **tB**.
- Jika nilai sama dengan **tB**, maka variabel **tC** diatur sebagai string "**True**".
- Jika nilai tidak sama dengan **tB**, maka cetak **False** dan keluar dari program.
- Terakhir, cetak nilai dari variabel **tC**.

Jadi, secara keseluruhan, kode ini menguji apakah semua elemen dalam tuple **tA** memiliki nilai yang sama, dan jika iya, mencetak "**True**".

SOAL 2

Sc :

```
#latman 11.2
nim = input("masukkan nim :")
nama = input("masukkan nama lengkap :")
alamat = input("masukkan alamat :")
print()

print("Data")
print("NIM :", nim)
print("NAMA :", nama)
print("ALAMAT :", alamat)
print()

NIM= tuple(nim)
print("NIM :", NIM)
x = nama.split()
Nama_depan = tuple(x[0][0:])
print("NAMA DEPAN :", Nama_depan)
x = nama.split()
Nama_terbalik = x[::-1]
print("NAMA TERBALIK :", tuple(Nama_terbalik))
```

Output:

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

NIM : (' ', '2', '2', '0', '6', '4', '0', '9', '1')
NAMA DEPAN : ('M', 'a', 't', 'a', 'h', 'a', 'r', 'i')
NAMA TERBALIK : ('Nendya', 'Bhakti', 'Matahari')
```

Penjelasan:

- Meminta input dari pengguna untuk NIM, nama lengkap, dan alamat.
- Mencetak judul "Data".
- Mencetak kembali NIM, nama lengkap, dan alamat yang telah dimasukkan.
- Mengubah NIM menjadi tuple dan mencetaknya kembali.
- Memisahkan nama lengkap menjadi kata-kata individual dan mengambil huruf pertama dari kata pertama sebagai tuple nama depan. Kemudian mencetaknya.

- Membalik urutan kata-kata dalam nama lengkap dan mengonversinya menjadi tuple, kemudian mencetaknya.

SOAL 3

SC

```
# #latman 11.3
namafile = input('Enter file name: ')
domain = dict()
daftar = list()

try:
    handle = open(namafile)
except:
    print('File cannot be opened:', namafile)
    exit()

for line in handle:
    kata = line.split()
    if len(kata) < 2 or kata[0] != "From":
        continue
    post = kata[5].split(":")
    jam = post[0]
    if jam not in domain:
        domain[jam] = 1
    else:
        domain[jam] += 1
for key, val in list(domain.items()):
    daftar.append((key, val))
daftar.sort()
for key, val in daftar:
    print(key, val)
```

Output:

```
Enter 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
```

Penjelasan:

- Meminta user untuk memasukkan nama file.
- Membuat *dictionary* kosong “**domain**” untuk menyimpan frekuensi kemunculan jam.
- Membuat *list* kosong ‘**daftar**’ untuk menyimpan pasangan kunci-nilai dari *dictionary domain*.
- Mencoba membuka file yang diminta. Jika gagal, mencetak pesan kesalahan dan keluar dari program.
- Mengiterasi setiap baris dalam file yang terbuka.
- Memisahkan setiap baris menjadi kata-kata.
- Memeriksa apakah baris tersebut memiliki setidaknya dua kata dan kata pertama adalah "From". Jika tidak, lanjut ke baris berikutnya.
- Jika syarat terpenuhi, mengambil bagian waktu dari baris tersebut (kata ke-6 setelah dipisahkan dengan delimiter spasi).
- Memisahkan waktu menjadi jam.
- Memeriksa apakah jam tersebut sudah ada di dalam kamus domain. Jika tidak, tambahkan dengan nilai 1. Jika sudah ada, tambahkan 1 ke nilai yang sudah ada.
- Mengonversi kamus domain menjadi daftar dan mengurutkannya.
- Mengiterasi melalui setiap pasangan kunci-nilai dalam daftar dan mencetak jam beserta frekuensinya.

Link Github: <https://github.com/GalihPramana/Praktikum-Alpro-71230976.git>