



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230992
Nama Lengkap	Andriano Kurniawan Ladjeba
Minggu ke / Materi	11 / 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.

MATERI 1

Tuple Immutable

Tuple mirip dengan list, dengan indeks bilangan bulat atau integer dan nilai yang dapat disimpan di dalamnya. Satu-satunya perbedaan antara keduanya adalah bahwa tuple tidak dapat dirubah, artinya anggota dalamnya tidak dapat diubah. Tuple memiliki kemampuan untuk dibandingkan (compare) dan dapat dihash, sehingga dapat dimasukkan ke dalam daftar sebagai tombol dalam dictionary python.

Nilai hash yang tidak pernah berubah selama hidupnya (menggunakan method `__hash__()`) dan dapat dibandingkan dengan objek lain (menggunakan method `__eq__()` atau `__cm__()`) disebut objek hashable.

Karena struktur data ini menggunakan nilai hash secara internal, hashingability membuat objek yang dapat digunakan sebagai kamus kunci dan mengatur anggota.

Dalam Python built-in, objek biasanya dapat dihash, tetapi wadah yang tidak dapat diubah, seperti daftar atau kamus, adalah objek yang merupakan instance dari kelas yang didefinisikan pengguna yang dapat dihash secara default; mereka semua memiliki perbandingan yang tidak sama, dan nilai hash mereka adalah `id()`.

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 menambahkan 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()
>>> print(t)
()
```

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

```
>>> t = tuple('dutawacana')
>>> print(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 tuple. 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')
```

Membandingkan Tuple

Operator 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 elemen 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.

Silakan perhatikan contoh berikut untuk memahaminya. Kami memiliki kalimat dan mengurutkan kata-katanya dari yang paling panjang ke yang paling pendek.

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

t.sort(reverse=True)

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

print(urutan)
```

Proses looping pertama akan membuat daftar tuple yang berisi daftar kata yang disesuaikan dengan panjangnya. Jika kondisi sesuai, fungsi sort akan membandingkan elemen pertama dari daftar dengan panjang kata yang ada dan kemudian menuju ke elemen kedua. Urutan secara terbalik (descending) dapat dilakukan dengan menggunakan kata kunci "benar".

Daftar **tuple** akan dibuat dalam looping kedua dalam urutan alfabet yang diurutkan berdasarkan panjangnya. Dalam contoh sebelumnya, empat kata yang termasuk dalam kalimat akan diurutkan secara alfabet terbalik. Dalam list, kata "**what**" akan didahului oleh kata "**soft**".

Output program diatas dapat dilihat dibawah ini:

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

Daftar kata yang muncul diurutkan dari yang paling panjang ke yang paling pendek.

Penugasan Tuple

Python memiliki kemampuan untuk memiliki tuple di sisi kiri statement penugasan, yang memungkinkan untuk menetapkan lebih dari satu variabel secara berurutan di sisi kiri.

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'  
>>>
```

Contoh di atas menggunakan statement penugasan disebelah kiri dan tidak menggunakan tanda kurung. Contoh yang sama dengan menggunakan tanda kurung (parentheses) diberikan di bawah ini.

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

Tuple memungkinkan pula untuk menukar nilai variabel dalam satu statement.

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

Contoh di atas menunjukkan bahwa kedua statemen adalah tuple. Sebuah tuple dari variable di bagian kiri dan sebuah tuple dari ekspresi di bagian kanan. Setiap nilai di bagian kanan diberikan atau ditugaskan ke masing-masing variable di bagian kiri, dan setiap ekspresi di bagian kiri diperiksa sebelum diberi tugas.

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

Dictionaries and Tuple

Dictionaries mempunyai metode yang disebut `items` untuk mengembalikan nilai daftar dari tuple yang masing-masing berisi pasangan nilai dan kunci.

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

Nilai item yang dikembalikan tidak berurutan, seperti yang terlihat dalam dictionary biasa. Bagaimanapun juga, melakukan konversi dictionary dari list tuple menampilkan isi dictionary yang diurutkan berdasarkan kuncinya, karena list tuple membandingkannya sehingga kita dapat melakukan pengurutan (`sort`) pada tuple.

```
>>> 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 diurutkan secara ascending berdasarkan alfabet dan key value.

Multipenugasan dengan dictionaries

Kombinasi antara **items**, **tuple 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, item mengembalikan list dari tuple, dan **key**. **val** adalah penugasan tuple yang melakukan iterasi berulang melalui pasangan key-value pada dictionary.

Key dan **value** akan bergerak maju menuju pasangan key-value berikutnya di dictionary pada setiap iterasi loop. Mereka tetap dalam urutan hash.

Output dari looping diatas

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

Kita dapat mencetak isi kamus yang diurutkan berdasarkan nilai yang disimpan di setiap pasangan nilai kunci jika kedua metode di atas digabungkan.

Untuk melakukannya, langkah pertama adalah membuat list tuple di mana masing-masing tuple beranggotakan (**value, key**). Metode **item** akan memberikan list tuple (**value, key**) dan melakukan pengurutan berdasarkan value, bukan key. Setelah membuat daftar list dengan value-key tuple, langkah berikutnya adalah mengurutkan list tersebut dengan urutan terbalik dan mencetak 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')]
>>>
```

Sangat mudah untuk mengurutkan list dari tuple yang memiliki nilai sebagai elemen pertama dan mendapatkan isi dictionary yang diurutkan berdasarkan nilainya.

Kata yang sering muncul

Kita akan mencoba menampilkan kata-kata yang sering muncul dalam teks Act 2, Scene 2, dari Romeo and Juliet pada bagian ini. Silakan download romeo-full.txt di bagian materi. Kita akan coba membuat program dengan list tuple yang menampilkan sepuluh kata yang paling sering muncul.

```
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
```



```
# 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)
```

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

Untuk membandingkan, nilai pertama digunakan. Jika ada lebih dari satu tuple yang memiliki nilai yang sama, elemen kedua—atau tombol—akan dilihat, sehingga tuple yang memiliki nilai yang sama akan diurutkan berdasarkan urutan abjad sesuai tombol.

Pada bagian terakhir, looping melakukan penugasan ganda sekali lagi dan mencetak sepuluh kata yang paling sering muncul dengan melakukan perulangan pada list **lst[:10]**.

Secara lengkap output dari program diatas akan menampilkan kata yang sesuai untuk frekuensi analisis.

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

Dengan 19 baris program Python yang dipahami, penguraian dan analisis data yang kompleks ini dapat dilakukan. Ini adalah salah satu alasan mengapa python adalah bahasa yang bagus untuk mengeksplorasi data.

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, kemudian 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  
>>>
```

MATERI 2

Penjelasan materi 2, dst... sesuai format ini.

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

Latihan 11.1

The image shows a Python IDE with a dark theme. The editor window displays a Python script named `latihan11.1.py` with the following code:

```
1 def tup(tA):
2     for i in tA :
3         if i != tA[0]:
4             return False
5     return True
6
7 tA = (90,90,90,90)
8 print(tup(tA))
9
10
```

The IDE interface includes a sidebar on the left with tabs for **PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL**, and **PORTS**. The **TERMINAL** tab is active, showing the command prompt output:

```
PS D:\A-Kuliah\Semester 2\Prakkrikum Algoritma dan Pemograman\tugas11\praticei1> & C:/Users/Acer/AppData/Local/Programs/Python/Python310/python.exe "d:/A-Kuliah/Semester 2/Prakkrikum Algoritma dan Pemograman\tugas11\praticei1\latihan11.1.py"
True
PS D:\A-Kuliah\Semester 2\Prakkrikum Algoritma dan Pemograman\tugas11\praticei1>
```

SOAL 2

Latihan 11.2



The screenshot shows a Python IDE with a file named `latihan12.1.py` containing a `bio` function. The function takes a `data` list as input and prints personal information, creating a tuple for `nim`, and processing the `nama` string (lowercase, split, and reversed). The terminal output shows the function being called with a sample data list, resulting in the printed information and the reversed name.

```

1 def bio(data):
2     print("NIM :", data[1])
3     print("NAMA :", data[0])
4     print("ALAMAT :", data[2])
5
6     nim = tuple(data[1])
7     print("NIM:", nim)
8
9     namaDepan = tuple(data[0].split()[0].lower())
10    print("NAMA DEPAN:", namaDepan)
11
12    namaBelakang = data[0].split()[0:]
13    namaBelakang.reverse()
14    namaBelakang = tuple(namaBelakang)

```

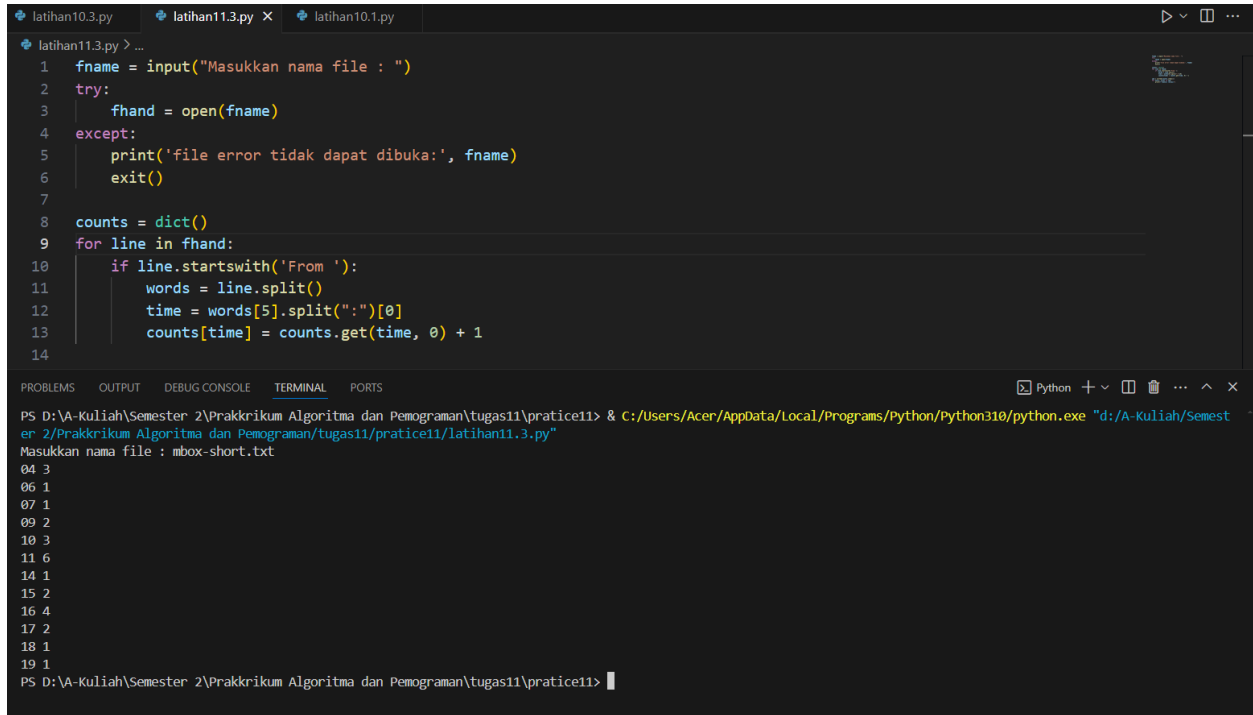
```

PS D:\A-Kuliah\Semester 2\Prakkrikum Algoritma dan Pemograman\tugas11\praticell> & C:\Users\Acer\AppData\Local\Programs\Python\Python310\python.exe "d:/A-Kuliah/Semester 2/Prakkrikum Algoritma dan Pemograman\tugas11\praticell\latihan12.1.py"
NIM : 71230992
NAMA : Andriano Kurniawan Ladjeba
ALAMAT : Sleman DI Yogyakarta
NIM: ('7', '1', '2', '3', '0', '9', '9', '2')
NAMA DEPAN: ('a', 'n', 'd', 'r', 'i', 'a', 'n', 'o')
NAMA TERBALIK: ('Ladjeba', 'Kurniawan', 'Andriano')
PS D:\A-Kuliah\Semester 2\Prakkrikum Algoritma dan Pemograman\tugas11\praticell>

```

SOAL 3

Latihan 11.3



```
latihan11.3.py > ...
1 fname = input("Masukkan nama file : ")
2 try:
3     fhand = open(fname)
4 except:
5     print('file error tidak dapat dibuka:', fname)
6     exit()
7
8 counts = dict()
9 for line in fhand:
10     if line.startswith('From '):
11         words = line.split()
12         time = words[5].split(":")[0]
13         counts[time] = counts.get(time, 0) + 1
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\A-kuliah\Semester 2\Prakkrikum Algoritma dan Pemograman\tugas11\praticel1> & C:\Users\Acer\AppData\Local\Programs\Python\Python310\python.exe "d:/A-kuliah/Semester 2/Prakkrikum Algoritma dan Pemograman\tugas11\praticel1\latihan11.3.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

PS D:\A-kuliah\Semester 2\Prakkrikum Algoritma dan Pemograman\tugas11\praticel1>

Github : <https://github.com/EchoGinDev/tugasAlpro11.git>