Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**Josephine Marcelia**
Nama Lengkap	**71231005**
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

BAGIAN 1: MATERI MINGGU INI (40%)

Tuple Immutable

Tuple bisa dikatakan menyerupai list.Namun perbedaannya terletak pada sifat tupel yang tidak dapat diubah (immutable), artinya setelah sebuah tuple dibuat, isi di dalamnya tidak dapat diubah atau dimodifikasi. Sedangkan list bersifat dapat diubah (mutable), artinya kita dapat menambahkan, menghapus, atau mengubah item di dalam list.

Penulisan tuple :

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

Jika ingin membuat tuple yang hanya berisi satu elemen, kita dapat menambahkan koma setelah elemen tersebut.
Namun jika kita tidak menambahkan koma, maka akan dianggap sebagai string seperti berikut:

```
tuple1 = ('x',)
print(type(tuple1))

tuple2 = ('x')
print(type(tuple2))
```

Output :

```
<class 'tuple'>
<class 'str'>
```

Jika argumen berupa data berurutan (seperti string, list, atau tuple), fungsi tuple() akan menghasilkan tuple dengan elemen-elemen berurutan.

Contoh:

```
t = tuple('josephine')
print(t)
```

Output :

```
('j', 'o', 's', 'e', 'p', 'h', 'i', 'n', 'e')
```

Tuple adalah nama constructor sehingga kita tidak bisa menggunakannya sebagai nama variabel.
Operator yang berfungsi pada list juga berfungsi pada tuple. Anda dapat mengakses elemen dengan tanda kurung kotak [ ] :

```
t = ('p','y','t','h','o','n')
print(t[0])
```

Output :
```
p
```

Untuk mencetak rentang nilai, kita dapat menggunakan tanda (:) dalam indeks :

```
print(t[2:5])
```

Output

```
('t', 'h', 'o')
```

Karena sifatnya yang immutable, maka elemen dalam tuple tidak bisa diubah. Jika kita mencoba mnegubahnya, akan terjadi error :

```
t[0] = 'B'
```

Output :

```
    t[0] = 'B'
    ~^^^
TypeError: 'tuple' object does not support item assignment
```

Namun, kita dapat membuat tuple baru dengan mengganti elemen:

```
t = ('B',) + t[1:]
print(t)
```

Output:

```
('B', 'y', 't', 'h', 'o', 'n')
```

Membandingkan Tuple

Tuple dan tipe data sekuensial lainnya seperti list, dictionary, dan set dapat kita dibandingkan menggunakan operator perbandingan. Proses perbandingannya kita mulai dengan membandingkan elemen pertama dari setiap urutan, kemudian berlanjut ke elemen berikutnya jika ada kesamaan. Proses ini akan berlanjut sampai perbedaannya ditemukan.

Contoh:

```
t1 = (0, 1, 2)
t2 = (3, 4, 5)

hasil = t1 > t2
print(hasil)
```

Output :

```
False
```

Fungsi pengurutan dalam Python bekerja dengan cara yang mirip, dimulai dengan mengurutkan berdasarkan elemen pertama, kemudian beralih ke elemen berikutnya jika diperlukan. Proses ini dikenal sebagai DSU (Decorate, Sort, Undecorate):

1. **Decorate**: Membuat daftar tuple dengan panjang kata sebagai kunci urutan sebelum elemen.
2. **Sort**: Mengurutkan list tuple berdasarkan panjang kata.
3. **Undecorate**: Mengambil elemen yang sudah diurutkan dalam daftar.

Contoh :
Program pengurutan kata dalam kalimat dari panjang terpanjang ke terpendek

```
kalimat = 'The beauty of nature will never fade'
daftar_kata = kalimat.split()
urutan = sorted(daftar_kata, key=len, reverse=True)
print(urutan)
```

Output :

```
['beauty', 'nature', 'never', 'will', 'fade', 'The', 'of']
```

Dalam program diatas, pertama-tama kita perlu memecah kalimat 'The beauty of nature will never fade' menjadi daftar kata menggunakan metode split(). Kita kemudian mengurutkan daftar kata tersebut berdasarkan panjang kata menggunakan fungsi sorted(), di mana kita menentukan panjang kata dengan parameter key=len dan mengatur urutan secara menurun dengan reverse=True. Hasilnya akan mencetak pengurutan kata dari yang terpanjang ke terpendek seperti output diatas.

Penugasan Tuple

Pada Python, kita bisa menggunakan tuple di sisi kiri dalam penugasan variabel. Ini memungkinkan kita menetapkan beberapa variabel sekaligus dalam satu pernyataan. Misalnya, kita dapat menetapkan elemen pertama dan kedua dari sebuah daftar ke variabel x dan y dalam satu pernyataan:

```
kata = ['bright', 'future']
k1, k2 = kata
print(k1)
print(k2)
```

Output :

```
bright
future
```

Pada contoh diatas, dapat kita lihat bahwa python menguraikan sintaks tuple tersebut dengan menetapkan k1 sebagai elemen pertama dari kata dan k2 sebagai elemen kedua.

Tuple juga memungkinkan kita menukar nilai dua variabel dengan satu pernyataan:

```
k1 = "bright"
k2 = "future"

k1, k2 = k2, k1
print(k1)
print(k2)
```

Output :

```
future
bright
```

Namun, jumlah variabel di sisi kiri dan kanan harus sama, jika tidak akan terjadi ValueError. Kita juga bisa membagi string seperti alamat email menjadi username dan domain menggunakan metode split( ).

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

Output :

```
josephinemrc
ti.ukdw.ac.id
```

Dictionaries and Tuple

Kita dapat menggunakan metode items pada Dictionaries untuk mendapatkan list dari tuple, dimana setiap tuplenya adalah pasangan kunci-nilai. Sebagai contoh:

```
d = {'c': 10, 'b': 22, 'a':9 }

t = list(d.items())
print("Sebelum diurutkan:", t)

t.sort()
print("Setelah diurutkan:", t)
```

Output :

```
Sebelum diurutkan: [('c', 10), ('b', 22), ('a', 9)]
Setelah diurutkan: [('a', 9), ('b', 22), ('c', 10)]
```

Pada contoh kita bisa menggunakan fungsi sort( ) sehingga data yang sebelumnya acak diurutkan secara ascending berdasarkan alfabet dan kuncinya.

Multipenugasan dengan dictionaries

Ketika kita menggabungkan items, tupple assignment, dan loop, kita bisa membuat kode yang efisien untuk mengakses keys dan values dari dictionary secara bersamaan. Dalam jenis loop ini, setiap langkah mengambil pasangan key-value dari dictionary dan mengolahnya. Outputnya adalah nilai diikuti oleh kunci, sesuai dengan urutan yang ditentukan.

Misalnya, dengan menggunakan teknik ini, kita dapat mencetak isi dictionary yang diurutkan berdasarkan nilai yang disimpan di dalamnya. Pertama, kita membuat list tuple di mana setiap tuple memiliki elemen (value, key). Metode items memberikan list tuple (value, key) dan melakukan pengurutan berdasarkan value. Setelah kita memiliki list tuple value-key, langkah selanjutnya adalah mengurutkan list tersebut secara terbalik dan mencetaknya.

Contoh :

```
d = {'j': 10, 'k': 15, 'l': 5}

l = [(val, key) for key, val in d.items()]
print("Sebelum diurutkan:", l)

l.sort(reverse=True)
print("Setelah diurutkan:", l)
```

Output :

```
Sebelum diurutkan: [(10, 'j'), (15, 'k'), (5, 'l')]
Setelah diurutkan: [(15, 'k'), (10, 'j'), (5, 'l')]
```

Dengan menyusun list dari tuple yang memiliki value sebagai elemen pertama, kita dapat dengan mudah mengurutkan list tersebut dan mendapatkan isi dictionary yang diurutkan berdasarkan nilai.

Kata yang sering muncul

Pada bagian ini, kita akan membuat program untuk menampilkan 10 kata yang paling sering muncul dalam teks Romeo and Juliet Act 2, Scene 2. Program ini akan membaca teks, menghitung frekuensi kemunculan setiap kata, dan mengurutkannya berdasarkan frekuensinya. Outputnya adalah 10 kata yang paling sering muncul beserta jumlah kemunculannya.

```
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 = [(val, key) for key, val in counts.items()]
lst.sort(reverse=True)

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

Output :

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

Pada program diatas, kita perlu menggunakan import string karena kita menggunakan fungsi-fungsi dari modul string untuk melakukan beberapa operasi pada teks yang dibaca dari file.

Kita juga menggunakan fungsi translate() dari modul string untuk menghapus tanda baca dari setiap baris teks. Fungsi ini menerima sebuah tabel translasi yang berisi aturan untuk mengganti karakter dengan karakter lain atau menghapusnya. Kita menggunakan str.maketrans('', '', string.punctuation) untuk membuat tabel translasi yang menghapus semua tanda baca dari teks.

Tuple sebagai kunci dictionaries

Tuple digunakan sebagai kunci dalam dictionary karena tuple bersifat hashable, berbeda dengan list yang tidak bisa digunakan sebagai kunci. Contoh penggunaannya adalah saat kita ingin membuat composite key, seperti dalam membuat direktori telepon yang memetakan pasangan last-name, first-name ke nomor telepon. Dalam contoh ini, tuple digunakan sebagai kunci dalam dictionary, misalnya directory[last, first] = number.
Dalam loop for yang terkait dengan dictionary, kita menggunakan tuple sebagai kunci. Elemen dari masing-masing tuple ditetapkan terlebih dahulu, kemudian dilakukan pencetakan nama dan nomor telepon yang sesuai.

Misalnya:

```
last = 'marcelia'
first = 'josephine'
number = '087886522738'
directory = dict()
directory[last, first] = number
for last, first in directory:
    print(first, last, directory[last,first])
```

Output :

```
josephine marcelia 087886522738
```

Pada contoh di atas kita menciptakan sebuah direktori kontak menggunakan dictionary Python. Tuple (last, first) kita gunakan sebagai kunci dalam dictionary untuk menyimpan nomor telepon. Kemudian, melalui loop for, informasi kontak ditampilkan dengan mencetak nama dan nomor telepon yang sesuai.

BAGIAN 2: LATIHAN MANDIRI (60%)

Latihan 11.1

```
def cek(tup):
    return all(elem == tup[0] for elem in tup)
tA = (90, 90, 90, 90)
print(cek(tA))
```

Output :

```
True
```

Penjelasan :
Pada program diatas, saya menggunakan fungsi return all(elem == tup[0] for elem in tup) untuk memeriksa apakah setiap elemen dalam tuple tup sama dengan elemen pertama tuple (tup[0]).

Program tersebut melakukan iterasi melalui semua elemen dalam tuple tup, membandingkan setiap elemen dengan elemen pertama tuple. Jika semua elemen sama dengan elemen pertama, maka program akan mengembalikan nilai True, yang menunjukkan bahwa semua elemen dalam tuple tersebut sama. Sebaliknya, jika terdapat setidaknya satu elemen yang tidak sama dengan elemen pertama, maka program akan mengembalikan nilai False.

Latihan 11.2

```
data = ('Matahari Bhakti Nendya', '22064091', 'Bantul, DI Yogyakarta')

nama = data[0]
nim = data[1]
alamat = data[2]

nim_tup = tuple(nim)
nama_depan = nama.split()[0]
nama_depan_tup = tuple(nama_depan[1:])
nama_terbalik_tup = tuple(nama.split()[::-1])

print(f"Data: {data}\n")
print(f"NIM : {nim}")
print(f"NAMA : {nama}")
print(f"ALAMAT : {alamat}\n")
print(f"NIM: {nim_tup}\n")
print(f"NAMA DEPAN: {nama_depan_tup}\n")
print(f"NAMA TERBALIK: {nama_terbalik_tup}\n")
```

Output :

```
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')
```

Penjelasan :

Dalam kode di atas, saya menggunakan indeks data[0], data[1], dan data[2] untuk mengambil data NIM, nama, dan alamat yang akan dicetak melalui fungsi print.

Kemudian, pada variabel nim_tup, saya menggunakan fungsi tuple untuk mengonversi setiap karakter dalam string nim menjadi elemen-elemen terpisah.

Untuk variabel nama_depan, saya menggunakan split untuk membagi string nama menjadi beberapa bagian berdasarkan spasi. Kemudian, dengan menggunakan indeks [0], saya mengambil elemen pertama dari list tersebut, yaitu "Matahari".

Setelah penggunaan split, saya membentuk variabel nama_depan_tup dengan menggunakan tuple untuk mengonversi karakter-karakter setelah karakter pertama dari nama depan menjadi tuple. Saya juga menggunakan slicing [1:] untuk mengambil karakter kedua hingga terakhir dari nama depan, yaitu "atahari".

Pada variabel nama_terbalik_tup, saya menggunakan split untuk membagi string nama menjadi beberapa bagian berdasarkan spasi. Kemudian, dengan menggunakan [::-1], saya membalik urutan elemen-elemen dalam list tersebut. Setelah itu, dengan fungsi tuple(), saya mengonversi hasil pembalikan tersebut menjadi sebuah tuple.

Setelah itu, saya menggunakan fungsi print untuk mencetak hasil output NIM, nama depan, dan nama terbalik seperti yang ditunjukkan dalam output di atas.

Latihan 11.3

```
file_name = input("Enter a file name: ")

try:
    with open(file_name, 'r') as file_handle:
        hour_counts = dict()

        for line in file_handle:
            if line.startswith('From '):
                time = line.split()[5]
                hour = time.split(':')[0]
                hour_counts[hour] = hour_counts.get(hour, 0) + 1

        for hour in sorted(hour_counts):
            print(hour, hour_counts[hour])

except FileNotFoundError:
    print("File cannot be opened:", file_name)
```

Output :

```
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
```

Kode with open(file_name, 'r') saya gunakan untuk membuka file yang telah diinputkan user dalam mode baca. Kemudian, saya membuat dictionary kosong dengan nama hour_counts yang akan digunakan untuk menyimpan jumlah kemunculan setiap jam. Lalu, menggunakan loop for, saya membaca setiap baris dalam file.

Saya menggunakan if statement untuk melakukan pengecekan apakah baris dimulai dengan teks 'From '. Jika benar, baris tersebut dianggap sebagai baris yang mengandung informasi waktu. Kemudian, dengan menggunakan split, saya membagi baris menjadi beberapa bagian berdasarkan spasi, dan mengambil bagian ke-6 (indeks 5), yang berisi informasi waktu.

Selanjutnya, untuk mengambil jam dari informasi waktu yang didapatkan dengan memisahkan jam dan menit, saya menggunakan split(':') dan mengambil bagian pertama hasil pemisahan tersebut. Setelah

itu, saya memperbarui dictionary hour_counts dengan jumlah kemunculan jam yang dihitung sejauh ini, menggunakan get() untuk mendapatkan nilai saat ini dan menambahkannya dengan 1. Jika jam tersebut belum ada dalam dictionary, maka nilai defaultnya adalah 0.

Setelah selesai membaca file, saya menggunakan loop for untuk mencetak hasil penghitungan jumlah pesan untuk setiap jam. Hasilnya diurutkan secara alfabetis berdasarkan jamnya.

Pada blok except, jika file yang dimasukkan pengguna tidak ditemukan, akan dicetak pesan error.

Link Github :
https://github.com/Josephinemrc/Tugas11-PrakAlpro.git