Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230985**
Nama Lengkap	**TOMAS BECKET**
Minggu ke / Materi	**11 / Tipe Data Tupple**

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%)

MATERI 1

TUPPLE IMMUTABLE

Tuple adalah salah satu tipe data bawaan dalam Python yang digunakan untuk menyimpan sekumpulan objek. Tuple mirip dengan daftar (list) dalam hal bisa menyimpan berbagai jenis data, namun ada perbedaan mendasar antara tuple dan list, yaitu immutability (ketidakberubahan). Sehingga jika dijelaskan lebih detail arti dari Tupple Immutable yaitu :

- Tuple: Sebuah tuple adalah kumpulan data yang terurut dan tidak dapat diubah setelah didefinisikan. Tuple dideklarasikan menggunakan tanda kurung biasa () dan elemen-elemen di dalamnya dipisahkan oleh koma.
- Immutable: Immutability berarti bahwa setelah tuple dibuat, nilai-nilai di dalamnya tidak dapat diubah, ditambah, dihapus, atau dimodifikasi dengan cara apa pun. Ini berbeda dengan list, yang bersifat mutable dan memungkinkan perubahan.

MATERI 2

Contoh source code Tupple Imutable

```
1   # IMMUTABLE TUPPLE
2   t =  'a','b','c','d','e'
3   t = ('a','b','c','d','e')
4   print(t[0])
```

Pada source code diatas sintaks penulisan tupple bisa menggunakan ‘ ‘ atau ‘ ‘ dan menggunakan tanda kurung dan penambahan koma sehingga menciptakan tupple. Karena jika tidak ditambah koma maka akan dianggap sebagai string.

Contoh seperti Source code dari module 12 Tuple :

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

MATERI 3

MEMBANDINGKAN TUPLE

Operator perbandingan dalam Python dapat bekerja pada tipe data sekuensial seperti tuple, list, dan string. Perbandingan dilakukan elemen per elemen, dimulai dari elemen pertama. Jika elemen pertama sama, perbandingan dilanjutkan ke elemen berikutnya hingga ditemukan perbedaan atau hingga semua elemen habis dibandingkan.

Proses pengurutan dalam Python, terutama dengan menggunakan metode DSU (Decorate-Sort-Undecorate), adalah teknik yang sangat berguna ketika kita ingin mengurutkan data berdasarkan lebih dari satu kriteria.

1. Decorate: Membangun daftar tuple yang menggabungkan kunci pengurutan dengan elemen dari sekuensial asli.
2. Sort: Mengurutkan daftar tuple menggunakan metode pengurutan bawaan Python.
3. Undecorate: Mengekstrak elemen asli dari tuple yang sudah diurutkan.

```
# MEMBANDINGKAN TUPPLE
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 pan jangnya. 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 berda sarkan panjangnya. Pada contoh diatasm Empat kata dalam kalimat akan diurutkan secara alfabet

terbalik. Kata "what" akan muncul sebelum "soft" didalam list

MATERI 4

PENUGASAN TUPLE

```
# PENUGASAN TUPPLE
m = ['have', 'fun', 'asik']
x, y = m
print(x)
print(y)
```

Dalam Python, kita dapat menggunakan tuple unpacking untuk menetapkan nilai ke beberapa variabel dalam satu pernyataan penugasan. Ini adalah fitur yang sangat berguna dan membuat kode lebih bersih dan lebih mudah dibaca. Contoh ketika source code diatas memanggil print x dan y maka yang muncul yaitu have dan fun saja.

MATERI 5

PENUGASAN TUPLE 2

```
# DICTIONARIES AND TUPLE
email = 'tomas.becket@ti.ukdw.ac.id'
username, domain = email.split('@')
print(username)
print(domain)
```

Contohnya seperti source code diatas menunjukan email = 'tomas.becket@ti.ukdw.ac.id'. dan terdapat 2 variabel baru yaitu username dan domain dimana pada variabel email akan melakukan split jika terdapat @ pada string tersebut sehingga ketika di print username dan domain akan terpisah

MATERI 6

DICTIONARIES DAN TUPLE

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

dictionary memiliki metode items() yang mengembalikan daftar tuple, di mana setiap tuple adalah pasangan kunci-nilai dari dictionary tersebut. Metode ini sangat berguna untuk iterasi dan manipulasi data dalam dictionary.

Secara default, dictionary dalam Python tidak menjamin urutan item yang disimpan. Namun, kita dapat mengurutkan item dari dictionary berdasarkan kunci atau nilai dengan menggunakan beberapa metode. Salah satu cara yang umum adalah dengan mengonversi dictionary menjadi

list of tuples, kemudian mengurutkan list tersebut, dan akhirnya menampilkannya atau memprosesnya lebih lanjut.

```
>>> 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)]
```

Contoh Source code pada module 12 Tuple

MATERI 7

KATA YANG SERING MUNCUL

Pada source code dibawah untuk menampilkan kata-kata yang paling sering muncul dalam teks "Romeo and Juliet Act 2, Scene 2", kita perlu melakukan beberapa langkah:

1. Membaca file teks.
2. Menghitung frekuensi kemunculan setiap kata.
3. Mengonversi hasilnya menjadi list of tuples.
4. Mengurutkan list berdasarkan frekuensi kemunculan.
5. Menampilkan 10 kata yang paling sering muncul.

```
# KATA YANG SERING MUNCUL
import string
fhand = open('rome-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(count.items()):
    lst.append((val, key))
```

Bagian pertama dari program membaca file dan melakukan komputasi untuk membuat dictionary yang memetakan setiap kata ke jumlah kemunculannya dalam dokumen. Setelah

menghitung kemunculan kata, dictionary diubah menjadi list of tuples (nilai, kunci) dan diurutkan dalam urutan terbalik.

Proses pengurutan menggunakan nilai pertama (frekuensi) sebagai dasar perbandingan. Jika ada lebih dari satu tuple dengan nilai yang sama, maka elemen kedua (kunci) akan digunakan untuk mengurutkan tuple tersebut berdasarkan abjad.

Di bagian akhir, program melakukan iterasi dengan penugasan ganda untuk mencetak 10 kata yang paling sering muncul. Ini dilakukan dengan mengulangi list lst[:10].

Sehingga hasil output yang muncul yaitu :

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

MATERI 8

TUPLE SEBAGAI KUNCI DICTIONARIES

Tuple bersifat hashable sedangkan list tidak. Ketika kita ingin membuat kunci gabungan (composite key) yang digunakan dalam dictionary, kita dapat menggunakan tuple sebagai kunci. Misalnya, jika kita ingin membuat direktori telepon yang memetakan pasangan last-name dan first-name ke nomor telepon, kita dapat menggunakan tuple sebagai kunci dalam dictionary. Asumsikan bahwa kita telah mendefinisikan variabel last, first, dan nomor.

Contoh pada source code dibawah

```
# TUPLE SEBAGAI DICTIONAIRIES
last = 'tomas'
first = 'becket'
number = '081227860696'
directory = dict()
directory[last, first] = number
for last, first in directory:
    print(first, last, directory[last,first])
```

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

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

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

Looping yang berkaitan dengan kunci (keys) pada dictionary di atas menggunakan tuple. Setiap elemen dari tuple tersebut ditetapkan terlebih dahulu, kemudian dilakukan pencetakan nama dan nomor telepon yang sesuai dengan tuple tersebut.

LINK GITHUB :

https://github.com/TomasBeckett/PrakAlpro12.git

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

1.1 SOURCE CODE

```python
def anggota(t):
    return all(x == t[0] for x in t)

tA = (90, 90, 90, 90)
tB = (90, 90, 91, 90)
hasil1 = anggota(tA)
hasil2 = anggota(tB)
print(hasil1)
print(hasil2)
```

1.2 PENJELASAN

Fungsi anggota akan menerima satu parameter t yang merupakan tuple. Menggunakan all() untuk memeriksa apakah semua elemen di dalam tuple sama dengan elemen pertama t[0].

all() akan mengembalikan True jika semua kondisi di dalamnya terpenuhi, dan False jika ada kondisi yang tidak terpenuhi dan Fungsi kemudian mengembalikan nilai True atau False sesuai dengan hasil pengecekan.

```python
tA = (90, 90, 90, 90)
tB = (90, 90, 91, 90)
```

Source code di atas memperlihatkan bagaimana fungsi ini digunakan dengan dua tuple yang berbeda untuk memeriksa apakah semua elemennya sama.

1.3 HASIL OUTPUT

```
True
False
```

SOAL 2

2.1 SOURCE CODE

```python
def proses_data(data_diri):
    nama, nim, alamat = data_diri

    print(f"NIM : {nim}")
    print(f"NAMA : {nama}")
    print(f"ALAMAT : {alamat}")

    nim_tuple = tuple(nim)
    print(f"NIM: {nim_tuple}")

    nama_depan = tuple(nama.split()[0])
    print(f"NAMA DEPAN: {nama_depan}")

    nama_terbalik = tuple(nama.split()[::-1])
    print(f"NAMA TERBALIK: {nama_terbalik}")

data = ('Tomas Becket', '71230985', 'Jakarta')

proses_data(data)
```

2.2 PENJELASAN

```python
def proses_data(data_diri):
```

proses_data(data_diri): Fungsi ini menerima satu parameter, yaitu data, yang merupakan tuple yang berisi nama, NIM, dan alamat.

```python
nama, nim, alamat = data_diri
```

nama, nim, alamat = data: Memisahkan elemen-elemen dari tuple data ke dalam variabel nama, nim, dan alamat.

print: Menampilkan informasi dasar dari data.

```python
nim_tuple = tuple(nim)
print(f"NIM: {nim_tuple}")
```

nim_tuple = tuple(nim): Memisahkan NIM menjadi tuple yang terdiri dari karakter-karakter NIM.

```python
nama_depan = tuple(nama.split()[0])
print(f"NAMA DEPAN: {nama_depan}")
```

nama_depan = tuple(nama.split()[0]): Mengambil nama depan dan memisahkannya menjadi tuple karakter-karakter.

```
nama_terbalik = tuple(nama.split()[::-1])
print(f"NAMA TERBALIK: {nama_terbalik}")
```

nama_terbalik = tuple(nama.split()[::-1]): Membalikkan urutan nama dan memisahkannya menjadi tuple.

2.3 HASIL OUTPUT

```
NIM : 71230985
NAMA : Tomas Becket
ALAMAT : Jakarta
NIM: ('7', '1', '2', '3', '0', '9', '8', '5')
NAMA DEPAN: ('T', 'o', 'm', 'a', 's')
NAMA TERBALIK: ('Becket', 'Tomas')
```

SOAL 3

3.1 SOURCE CODE

```
def jampesan():
    nama_file = input("Enter a file name: ")

    try:
        with open(nama_file, 'r') as file:
            distribusi_jam = {}

            for baris in file:
                if baris.startswith('From '):
                    kata_kata = baris.split()
                    waktu = kata_kata[5]
                    jam = waktu.split(':')[0]
                    distribusi_jam[jam] = distribusi_jam.get(jam, 0) + 1

            for jam in sorted(distribusi_jam.keys()):
                print(f"{jam} {distribusi_jam[jam]}")

    except FileNotFoundError:
        print(f"File {nama_file} tidak ditemukan.")

jampesan()
```

3.2 PENJELASAN

Source code diatas menggunakan pemanggilan fungsi jampesan() dimana pada line kedua akan

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

input("Enter a file name: "): Meminta pengguna memasukkan nama file.

```
with open(nama_file, 'r') as file:
```

with open(nama_file, 'r') as file: Membuka file dalam mode baca.

```
distribusi_jam = {}
```

distribusi_jam = {}: Inisialisasi dictionary untuk menyimpan distribusi jam.

```
for baris in file:
```

for baris in file: Membaca file baris per baris.

```
if baris.startswith('From '):
    kata_kata = baris.split()
    waktu = kata_kata[5]
    jam = waktu.split(':')[0]
    distribusi_jam[jam] = distribusi_jam.get(jam, 0) + 1
```

if baris.startswith('From '): Mencari baris yang berisi informasi waktu penerimaan email. Dan kata_kata = baris.split(): Memisahkan baris menjadi kata-kata. Kemudian waktu = kata_kata[5]: Mengambil waktu dari kolom kelima dan jam = waktu.split(':')[0]: Memisahkan jam dari waktu. Sehingga untuk Menghitung jumlah email yang diterima pada setiap jam dapat dengan distribusi_jam[jam] = distribusi_jam.get(jam, 0) + 1:

```
for jam in sorted(distribusi_jam.keys()):
    print(f"{jam} {distribusi_jam[jam]}")
```

for jam in sorted(distribusi_jam.keys()): Mengurutkan jam sebelum menampilkan hasil.

```
print(f"{jam} {distribusi_jam[jam]}")
```

print(f"{jam} {distribusi_jam[jam]}"): Menampilkan jam dan jumlah email.

```
try:
```

```
except FileNotFoundError:
    print(f"File {nama_file} tidak ditemukan.")
```

try-except: Menangani pengecualian jika file tidak ditemukan.

3.3 HASIL 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
```

LINK GITHUB :

https://github.com/TomasBeckett/PrakAlpro12.git