Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**<71230989>**
Nama Lengkap	**<Yohanes Nevan Adventus Wibawa>**
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%)

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 mirip dengan list karena keduanya dapat menyimpan nilai apa pun dan diindeks dengan bilangan bulat. Namun, perbedaan utamanya terletak pada sifat tuple yang tidak dapat diubah (immutable). Ini berarti bahwa setelah sebuah tuple dibuat, nilainya tidak dapat diubah lagi. Tuple juga dapat dibandingkan dan bersifat hashable, yang berarti dapat digunakan sebagai kunci dalam dictionary Python.

Objek disebut hashable jika nilai hashnya tetap tidak berubah selama hidupnya dan dapat dibandingkan dengan objek lain. Kemampuan hashability ini memungkinkan objek untuk digunakan sebagai kunci dalam kamus Python, karena penggunaan nilai hash secara internal. Objek bawaan Python umumnya hashable, sementara wadah yang tidak dapat diubah (seperti list atau dictionary) tidak. Objek yang merupakan instance dari kelas yang didefinisikan pengguna secara default hashable; mereka membandingkan yang tidak sama dan nilai hashnya adalah id() objek tersebut.

```
t1 = 'a','b','c','d','e'
print(type(t1))
```

Output =
```
<class 'tuple'>
```

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

```
t3 = tuple("dutawacana")
t3 = ("b",) +t3[0:]
print(type(t3))
print(t3)
```

Output =
```
<class 'tuple'>
('b', 'd', 'u', 't', 'a', 'w', 'a', 'c', 'a', 'n', 'a')
```

Kode tersebut mengubah string "dutawacana" menjadi sebuah tuple yang berisi setiap karakter dalam string tersebut, sehingga menghasilkan ('d', 'u', 't', 'a', 'w', 'a', 'c', 'a', 'n', 'a'). Kemudian, elemen 'b' ditambahkan di awal tuple ini, menghasilkan tuple baru ('b', 'd', 'u', 't', 'a', 'w', 'a', 'c', 'a', 'n', 'a'). Setelah itu, tipe dari t3 dicetak, yang menunjukkan bahwa t3 adalah <class 'tuple'>, dan isi dari t3 juga dicetak, yang menunjukkan tuple yang telah dimodifikasi. Hasil akhir yang ditampilkan adalah tipe tuple dan isinya yaitu ('b', 'd', 'u', 't', 'a', 'w', 'a', 'c', 'a', 'n', 'a').

Membandingkan Tuple

Operator perbandingan (compare) dapat digunakan pada tuple dan struktur data sekuensial lainnya seperti list, dictionary, dan set. Cara kerjanya adalah dengan membandingkan elemen pertama dari setiap sekuens. Jika elemen pertama sama, perbandingan berlanjut ke elemen berikutnya, dan proses ini terus berlanjut hingga ditemukan perbedaan. Fungsi sort pada Python bekerja dengan prinsip yang sama. Pertama-tama, pengurutan dilakukan berdasarkan elemen pertama. Jika ada elemen yang sama, pengurutan dilanjutkan ke elemen kedua, dan seterusnya. Fitur ini dikenal dengan nama DSU (Decorate, Sort, Undecorate).

```
t4 = (0, 1, 2) < (0, 3, 4)
t5 = (0, 1, 2000000) < (0, 3, 4)
print(t4)
print(t5)
```

Output =
```
True
True
```

1. **Decorate:** Membangun daftar tuple dari urutan (sekuensial) dengan menambahkan satu atau lebih key pengurutan sebelum elemen dari urutan tersebut.
2. **Sort:** Mengurutkan daftar tuple menggunakan fungsi sort bawaan Python.
3. **Undecorate:** Mengekstrak elemen yang telah diurutkan kembali ke dalam urutan sekuensial.

Sebagai contoh, kita dapat menggunakan teknik ini untuk mengurutkan kata-kata dalam sebuah kalimat berdasarkan panjang kata, dari yang terpanjang hingga yang terpendek.

```
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)
```

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

Looping pertama akan membuat daftar tuple yang berisi kata-kata beserta panjangnya. Fungsi sort akan membandingkan elemen pertama dari daftar berdasarkan panjang kata, dan jika ada kesamaan, akan melanjutkan ke elemen kedua. Penggunaan keyword reverse=True memastikan pengurutan dilakukan secara menurun (descending). Looping kedua akan membangun daftar tuple dalam urutan alfabet berdasarkan panjang kata. Pada contoh di atas, empat kata dalam kalimat akan diurutkan dalam urutan alfabet terbalik. Misalnya, kata "what" akan muncul sebelum "soft" dalam daftar.

Penugasan Tuple

Salah satu fitur unik Python adalah kemampuannya untuk memiliki tuple di sisi kiri dari pernyataan penugasan. Ini memungkinkan penetapan beberapa variabel sekaligus secara berurutan di sisi kiri. Misalnya, jika ada dua daftar elemen yang berurutan, kita bisa menetapkan elemen pertama dan kedua dari urutan tersebut ke dalam variabel x dan y dalam satu pernyataan.

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

Output =
```
have
fun
```

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.

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

print(username)
print(domain)
```

Output =
```
yohanes.nevan
ti.ukdw.ac.id
```

Dictionaries and Tuple

Dictionaries memiliki metode yang disebut `items`, yang mengembalikan daftar tuple, di mana setiap tuple merupakan pasangan kunci dan nilai (key-value pair). Seperti dictionary pada umumnya, item yang dikembalikan tidak berurutan. Namun, karena daftar tuple adalah sebuah list dan tuple dapat dibandingkan, kita bisa mengurutkan (sort) tuple tersebut. Mengonversi dictionary menjadi daftar tuple adalah cara untuk menampilkan isi dictionary yang diurutkan berdasarkan kuncinya, seperti contoh dibawah ini.

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

Output =
```
[('a', 10), ('b', 1), ('c', 22)]
```

Multipenugasan Dengan Dictionaries

Menggabungkan metode items, penugasan tuple, dan perulangan for memungkinkan kita untuk mengakses kunci dan nilai dari dictionary dalam satu loop. Dalam looping ini, terdapat dua variabel iterasi karena items mengembalikan daftar tuple, dan key, val adalah penugasan tuple yang berulang melalui pasangan kunci-nilai pada dictionary. Pada setiap iterasi melalui loop, baik kunci (key) maupun nilai (value) akan maju ke pasangan kunci-nilai berikutnya dalam dictionary (masih dalam urutan hash). Output dari looping ini akan menampilkan setiap pasangan kunci dan nilai secara berurutan.

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

Output =
```
10 a
1 b
22 c
```

Dengan menggabungkan kedua teknik di atas, kita dapat mencetak isi dictionary yang diurutkan berdasarkan nilai yang disimpan di setiap pasangan kunci-nilai. Langkah pertama adalah membuat daftar tuple di mana masing-masing tuple berisi (value, key). Metode items akan memberikan daftar tuple (key, value), dan kita akan membaliknya menjadi (value, key) kemudian mengurutkannya berdasarkan nilai. Setelah daftar dengan tuple value-key terbentuk, langkah selanjutnya adalah mengurutkan daftar tersebut dalam urutan terbalik dan mencetak daftar baru yang telah diurutkan.

Kata yang 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)
```

Output =

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

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]:.

Tuple sebagai kunci dictionaries

Tuple bersifat hashable, sedangkan list tidak. Ketika kita ingin membuat kunci komposit yang digunakan dalam dictionary, kita dapat menggunakan tuple sebagai kuncinya. Misalnya, untuk membuat direktori telepon yang memetakan pasangan nama belakang (last-name) dan nama depan (first-name) ke nomor telepon, kita bisa menggunakan tuple sebagai kunci. Dengan asumsi kita sudah mendefinisikan variabel last, first, dan number, kita bisa menuliskan pernyataan penugasan dalam dictionary sebagai berikut:

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

Langkah selanjutnya adalah menggunakan tuple dalam looping for yang berhubungan dengan dictionary.

```
last = 'nevan'
first = 'yohanes'
number = '71230989'
directory = dict()
directory[last, first] = number
for last, first in directory:
    print(first, last, directory[last,first])
```

Output =

```
yohanes nevan 71230989
```

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

```
# 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()  # mencetak baris kosong untuk pemisah

# Membagi tuple ke dalam string
str1, str2, str3 = kota
print("str1: ", str1)
print("str2: ", str2)
print("str3: ", str3)
print()  # mencetak baris kosong untuk pemisah

# Membuat tuple yang berisi string dari sebuah kata
tpl = tuple("Belajar Python")
print("tpl: ", tpl)
print()  # mencetak baris kosong untuk pemisah

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

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

Output =

```
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')
```

Kode diatas menjalankan beberapa operasi dengan tuple. Pertama, membuat dan mencetak tuple `kota` yang berisi nama-nama kota, serta mencetak elemen individualnya. Kedua, membagi elemen-elemen tuple `kota` ke dalam variabel `str1`, `str2`, dan `str3`, lalu mencetaknya. Ketiga, mengubah string "Belajar Python" menjadi tuple `tpl` yang berisi setiap karakter, lalu mencetaknya. Keempat, membuat tuple `tpl1` dari string yang sama namun tanpa huruf pertama, dan mencetaknya. Terakhir, mencetak tuple `name` yang berisi beberapa nama dalam urutan terbalik (`rev_name`) dan urutan terbalik dengan langkah dua (`rev2`).

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

```
# Membuat dictionary dan list
dic_email = dict()
lst = list()

# Meminta pengguna untuk memasukkan nama file
fname = input('Nama File: ')
try:
    fhand = open(fname)
except FileNotFoundError:
    print('File tidak bisa dibuka:', fname)
    quit()

# Memproses setiap baris dalam file
```

```
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 pertama
        else:
            dic_email[kata[1]] += 1  # menambah hitungan

# Mengubah dictionary menjadi list tuple dan mengurutkannya
for key, val in dic_email.items():
    lst.append((val, key))  # isi daftar dengan nilai kunci dict

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

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

Output =
```
Nama File: mbox-short.txt
cwen@iupui.edu 5
```

Kode ini membuat sebuah dictionary dic_email dan sebuah list lst untuk menyimpan data sementara. Pengguna diminta memasukkan nama file, kemudian program mencoba membuka file tersebut. Jika file tidak ditemukan, program menampilkan pesan kesalahan dan menghentikan eksekusi. Program membaca setiap baris dalam file, membagi baris menjadi kata-kata, dan memeriksa apakah baris tersebut dimulai dengan 'From' dan memiliki setidaknya dua kata. Jika ya, program menambahkan alamat email (kata kedua) ke dalam dictionary dic_email atau memperbarui hitungannya jika sudah ada. Setelah memproses semua baris, program mengonversi dictionary menjadi daftar tuple yang berisi frekuensi dan alamat email, lalu mengurutkannya secara menurun. Akhirnya, program menampilkan alamat email yang paling sering muncul beserta jumlah kemunculannya.

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

```python
def cek(t):
    return len(set(t)) == 1

#contoh True
tA = (90,90,90,90)
print(cek(tA))

#contoh False
tb = (90,90,60,90)
print(cek(tb))
```

Output =

```
True
False
```

Fungsi cek(t) digunakan untuk memeriksa apakah semua elemen dalam tuple t adalah sama. Fungsi ini bekerja dengan mengonversi tuple menjadi set, yang secara otomatis menghapus elemen duplikat, kemudian memeriksa apakah panjang set tersebut adalah 1. Jika panjangnya 1, itu berarti semua elemen dalam tuple adalah identik. Misalnya, untuk tuple tA yang berisi (90, 90, 90, 90), fungsi cek(tA) akan mengembalikan True karena semua elemen adalah 90. Sebaliknya, untuk tuple tb yang berisi (90, 90, 60, 90), fungsi cek(tb) akan mengembalikan False karena ada elemen yang berbeda (60).

SOAL 2

```python
def data(nama_lengkap, nim, alamat):
    gabung = nama_lengkap, nim, alamat
    print("Data:", tuple(gabung))
    print("\n")
    print("NIM      :", ' '.join(nim))
    print("NAMA     :", ' '.join(nama_lengkap))
    print("ALAMAT   :", alamat)
    print('\n')
    print("NIM:", tuple(nim))
    print("NAMA DEPAN:", tuple(nama_lengkap.split()[0]))
    print("NAMA TERBALIK:", tuple(nama_lengkap.split()[::-1]))

nama_lengkap = 'Yohanes Nevan Adventus Wibawa'
nim = '71230989'
alamat = 'Bantul, Yogyakarta'

data(nama_lengkap, nim, alamat)
```

Output =

```
Data: ('Yohanes Nevan Adventus Wibawa', '71230989', 'Bantul, Yogyakarta')


NIM      : 7 1 2 3 0 9 8 9
NAMA     : Y o h a n e s   N e v a n   A d v e n t u s   W i b a w a
ALAMAT   : Bantul, Yogyakarta


NIM: ('7', '1', '2', '3', '0', '9', '8', '9')
NAMA DEPAN: ('Y', 'o', 'h', 'a', 'n', 'e', 's')
NAMA TERBALIK: ('Wibawa', 'Adventus', 'Nevan', 'Yohanes')
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 11>
```

Fungsi data menerima tiga parameter: nama_lengkap, nim, dan alamat. Fungsi ini menggabungkan ketiga parameter tersebut menjadi satu tuple dan mencetaknya. Selanjutnya, fungsi mencetak NIM, nama lengkap, dan alamat dengan setiap karakter NIM dan nama lengkap dipisahkan oleh spasi. Kemudian, fungsi mencetak NIM sebagai tuple, nama depan sebagai tuple (diambil dari kata pertama nama_lengkap), dan nama lengkap secara terbalik sebagai tuple (urutan kata-kata dalam nama_lengkap dibalik).

Soal 3

```python
def hitung_distribusi_jam(namafile):
    distribusi_jam = {}
    try:
        with open(namafile, 'r') as file:
            for line in file:
                if line.startswith('From '):
                    kata = line.split()
                    waktu = kata[5].split(':')
                    jam = waktu[0]
                    distribusi_jam[jam] = distribusi_jam.get(jam, 0) + 1
    except FileNotFoundError:
        print("File tidak ditemukan")
        return

    print("Distribusi Jam:")
    for jam in sorted(distribusi_jam):
        print(jam, distribusi_jam[jam])

namafile = 'mbox-short.txt'

hitung_distribusi_jam(namafile)
```

Output =

```
Distribusi Jam:
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 C:\Users\LENOVO\Tugas PrakAlPro\Tugas 11>
```

Fungsi `hitung_distribusi_jam` digunakan untuk membaca sebuah file teks dan menghitung distribusi jam dari email yang tercatat dalam file tersebut. Fungsi ini dimulai dengan inisialisasi dictionary kosong `distribusi_jam`.

Kemudian, file dibuka dalam mode baca dan setiap baris dibaca satu per satu. Jika baris diawali dengan 'From ', baris tersebut diproses lebih lanjut. Baris tersebut dipecah menjadi kata-kata, dan waktu (kolom keenam) diambil dan dipecah lagi berdasarkan tanda titik dua untuk mendapatkan jamnya. Jam ini kemudian digunakan untuk memperbarui jumlah kemunculannya dalam dictionary `distribusi_jam`. Jika file tidak ditemukan, fungsi akan mencetak pesan kesalahan dan keluar.

Setelah seluruh file dibaca, fungsi mencetak distribusi jam yang diurutkan berdasarkan jam. Sebagai contoh, jika fungsi ini dipanggil dengan file `mbox-short.txt`, hasilnya adalah distribusi jam email yang diterima, diurutkan berdasarkan jam.

Link Github

Link = https://github.com/YohanesNevan/Tugas-Laporan-Alpro-11.git