



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230977
Nama Lengkap	MICHAEL HOSEA
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.

Materi 1: Pengantar Tuple

Tuple adalah struktur data dalam bahasa pemrograman Python yang mirip dengan list. Namun, tuple bersifat *immutable* atau tidak dapat diubah setelah dibuat. Hal ini berbeda dengan list yang bersifat *mutable* atau dapat diubah. Oleh karena itu, tuple digunakan untuk menyimpan data yang tidak perlu diubah, seperti koordinat, tanggal, atau waktu.

Berikut merupakan cara penulisan tuple:

```
koordinat = (10, 20)
```

```
tanggal = (2023, 4, 8)
```

Tuple dalam Python adalah tipe data yang bisa di-hash dan dibandingkan, sehingga cocok digunakan sebagai kunci dalam dictionary. Objek disebut hashable jika nilai hash-nya tetap sama sepanjang masa hidupnya (menggunakan metode `__hash__()`) dan bisa dibandingkan dengan objek lain (menggunakan metode `__eq__()` atau `__cmp__()`). Objek hashable yang sama harus memiliki nilai hash yang sama. Ini memungkinkan penggunaan objek tersebut sebagai kunci dalam dictionary dan anggota dalam set karena struktur data ini memanfaatkan nilai hash.

Sebagian besar objek bawaan Python bersifat hashable, sedangkan container yang dapat diubah (seperti list atau dictionary) tidak. Objek dari kelas yang didefinisikan oleh pengguna secara default bersifat hashable, dengan nilai hash yang biasanya adalah `id()` objek tersebut, dan mereka dianggap tidak sama kecuali diimplementasikan secara eksplisit.

Materi 1.1: Syntax penulisan Tuple

1. Tanpa tanda kurung

```
angka = 1, 2, 3, 4, 1, 2, 3, 1, 2, 1
```

2. Dengan tanda kurung

```
angka = (1, 2, 3, 4, 1, 2, 3, 1, 2, 1)
```

3. Tuple dengan satu element

```
t1 = ('a',)
print(type(t1)) # Output: <class 'tuple'>
```

Namun, ketika tanpa tanda koma, akan berubah kelas

```
t2 = ('a')
print(type(t2)) # Output: <class 'str'>
```

4. Menggunakan fungsi build-in "Tuple()"

- Tuple kosong:

```
t = tuple()
print(t) # Output: ()
```

- Dari urutan (string, list, atau tuple):

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

Materi 2: Operasi pada Tuple

Tuple mendukung sebagian besar operator yang bekerja pada list. Beberapa contoh operasi dasar pada tuple:

1. Mengakses Elemen

```
t = ('a', 'b', 'c', 'd', 'e')
print(t[0]) # Output: 'a'
```

2. Menampilkan Rentang Elemen:

```
print(t[1:3]) # Output: ('b', 'c')
```

Tuple bersifat immutable, yang berarti elemen-elemen dalam tuple tidak dapat diubah setelah tuple dibuat. Jika mencoba mengubah elemen dalam tuple, akan muncul error:

```
t = ('a', 'b', 'c', 'd', 'e')
t[0] = 'A' # TypeError: 'tuple' object does not support item assignment
```

Namun, kita bisa membuat tuple baru dengan menggantikan elemen tertentu:

```
t = ('A',) + t[1:]
print(t) # Output: ('A', 'b', 'c', 'd', 'e')
```

Catatan Tambahan :

1. Jangan menggunakan nama tuple sebagai nama variabel karena tuple adalah nama constructor bawaan.
2. Sebagian besar operator dan metode yang dapat digunakan pada list juga dapat digunakan pada tuple, kecuali yang mengubah konten (misalnya, `append()`, `remove()`, dll.).

Materi 3.1: Membandingkan dan Mengurutkan Tuple serta Urutan Sekuensial Lainnya dalam Python

Operator perbandingan seperti `<`, `>`, `==`, `!=`, `<=`, dan `>=` dapat digunakan untuk membandingkan tuple dan tipe data sekuensial lainnya seperti list. Cara kerjanya adalah dengan membandingkan elemen pertama dari setiap sekuensial. Jika elemen pertama sama, maka perbandingan berlanjut ke elemen berikutnya, dan proses ini terus berlanjut hingga ditemukan perbedaan.

```
print((0, 1, 2) < (0, 3, 4)) # Output: True
print((0, 1, 2000000) < (0, 3, 4)) # Output: True
```

Materi 3.2: Pengurutan (Sorting) dengan Fungsi `sort()`

Fungsi `sort()` pada Python bekerja dengan cara yang sama. Pertama, elemen diurutkan berdasarkan elemen pertama. Jika elemen pertama sama, pengurutan dilanjutkan ke elemen berikutnya. Fitur ini dikenal sebagai DSU (Decorate, Sort, Undecorate):

1. Decorate – Urutan sekuensial dibangun menjadi daftar tuple dengan satu atau lebih kunci pengurutan sebelum elemen dari urutan (sekuensial).
2. Sort – Daftar tuple diurutkan menggunakan fungsi `sort()` bawaan Python.
3. Undecorate – Melakukan ekstraksi pada elemen yang telah diurutkan pada satu sekuensial.

Contoh DSU:

```
kalimat = 'but soft what light in yonder window breaks'
daf_kata = kalimat.split()

# Decorate – Membuat daftar tuple (panjang kata, kata)
t = []
```

```

for kata in daf_kata:
    t.append((len(kata), kata))

# Sort – Mengurutkan daftar tuple berdasarkan panjang kata (descending)
t.sort(reverse=True)

# Undecorate – Mengekstraksi kata-kata yang telah diurutkan
urutan = []
for length, kata in t:
    urutan.append(kata)

print(urutan)

```

Penjelasan per Langkah:

1. Looping Pertama (Decorate): Membuat daftar tuple yang berisi panjang kata dan kata itu sendiri:

```

t = []
for kata in daf_kata:
    t.append((len(kata), kata))

```

2. Sort: Mengurutkan daftar tuple berdasarkan panjang kata secara descending:

```

t.sort(reverse=True)

```

3. Looping Kedua (Undecorate): Mengekstraksi kata-kata dari tuple yang telah diurutkan berdasarkan panjangnya:

```

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

```

4. Output :

```

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

```

Materi 4.1: Penugasan Tuple

Salah satu fitur unik dari Python adalah kemampuan untuk menggunakan tuple di sisi kiri dari pernyataan penugasan. Hal ini memungkinkan penetapan beberapa variabel secara bersamaan dalam satu pernyataan.

```

m = ['have', 'fun']

```

```
x, y = m
print(x) # Output: 'have'
print(y) # Output: 'fun'
```

Python menerjemahkan penugasan ini menjadi langkah-langkah berikut:

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

Kita bisa juga menggunakan tanda kurung (parentheses) untuk menulis tuple di sisi kiri:

```
m = ['have', 'fun']
(x, y) = m
print(x) # Output: 'have'
print(y) # Output: 'fun'
```

Menukar Nilai Variabel Menggunakan Tuple:

Tuple juga memungkinkan penukaran nilai variabel dalam satu pernyataan:

```
a, b = 1, 2
a, b = b, a
print(a) # Output: 2
print(b) # Output: 1
```

Di sini, kedua sisi dari pernyataan merupakan tuple. Bagian kiri adalah tuple dari variabel, dan bagian kanan adalah tuple dari ekspresi. Setiap nilai di bagian kanan diberikan ke variabel yang sesuai di bagian kiri. Semua ekspresi di bagian kanan dievaluasi sebelum penugasan dilakukan.

Kesalahan Jika Jumlah Variabel Tidak Sama:

Jumlah variabel di sisi kiri harus sama dengan jumlah elemen di sisi kanan. Jika tidak, akan muncul kesalahan:

```
a, b = 1, 2, 3 # ValueError: too many values to unpack
```

Contoh Penugasan dengan Data Sekuensial:

Misalnya, kita ingin membagi alamat email menjadi username dan domain:

```
email = 'didanendya@ti.ukdw.ac.id'
username, domain = email.split('@')
print(username) # Output: 'didanendya'
print(domain)   # Output: 'ti.ukdw.ac.id'
```

Fungsi `split()` mengembalikan dua elemen yang dipisahkan oleh '@', dengan elemen pertama berisi username dan elemen kedua berisi domain.

Materi 5: Dictionaries dengan Tuple

1. Menggunakan `items()` untuk Mengembalikan Daftar Tuple

Dictionary di Python memiliki metode `items()` yang mengembalikan daftar tuple, di mana setiap tuple adalah pasangan key-value.

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

Elemen dalam dictionary tidak diurutkan, tetapi daftar tuple adalah list, sehingga dapat diurutkan. Dengan mengonversi dictionary menjadi list tuple, kita bisa menampilkan isi dictionary yang diurutkan berdasarkan kunci.

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

2. Multipenugasan dengan Dictionary

Kombinasi antara `items()`, penugasan tuple, dan `for` memungkinkan iterasi melalui key dan value dictionary dalam satu loop.

```
for key, val in list(d.items()):
    print(val, key)
# Output:
# 10 a
# 1 b
# 22 c
```

3. Mengurutkan Dictionary Berdasarkan Nilai

Untuk menampilkan isi dictionary yang diurutkan berdasarkan nilai, kita dapat membuat list tuple yang berisi (value, key) dan mengurutkannya.

```

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

```

4. Menampilkan Kata yang Sering Muncul

Kita bisa menampilkan kata yang sering muncul dari teks Romeo and Juliet Act 2, Scene 2. Unduh romeo-full.txt dan gunakan program berikut:

```

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 = 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
```

5. Tuple sebagai Kunci Dictionary

Tuple adalah hashable sedangkan list tidak. Kita bisa menggunakan tuple sebagai composite key dalam dictionary.

```
last = 'nendya'
first = 'dida'
number = '088112266'
directory = dict()
directory[last, first] = number

for last, first in directory:
    print(first, last, directory[last, first])
# Output:
# dida nendya 088112266
```

Dengan menggunakan tuple, kita dapat membuat dictionary yang memetakan pasangan (last-name, first-name) ke nomor telepon.

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

Buatlah program untuk melakukan pengecekan apakah semua anggota yang ada didalam tuple sama.

Contoh:

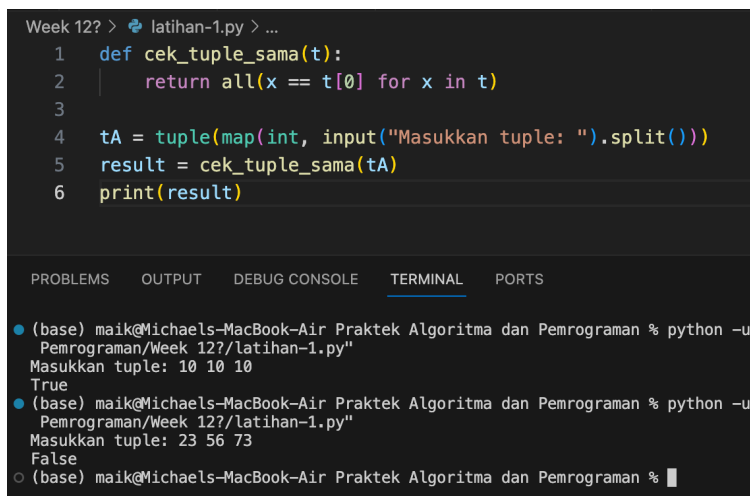
tA= (90, 90, 90, 90)

Output

True

- Source Code dan Output

```
def cek_tuple_sama(t):  
    return all(x == t[0] for x in t)  
  
tA = tuple(map(int, input("Masukkan tuple: ").split()))  
result = cek_tuple_sama(tA)  
print(result)
```



The screenshot shows a code editor with the following Python code:

```
Week 12? > latihan-1.py > ...  
1 def cek_tuple_sama(t):  
2     return all(x == t[0] for x in t)  
3  
4 tA = tuple(map(int, input("Masukkan tuple: ").split()))  
5 result = cek_tuple_sama(tA)  
6 print(result)
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
● (base) maik@Michaels-MacBook-Air Praktek Algoritma dan Pemrograman % python -u  
  Pemrograman/Week 12?/latihan-1.py"  
Masukkan tuple: 10 10 10  
True  
● (base) maik@Michaels-MacBook-Air Praktek Algoritma dan Pemrograman % python -u  
  Pemrograman/Week 12?/latihan-1.py"  
Masukkan tuple: 23 56 73  
False  
○ (base) maik@Michaels-MacBook-Air Praktek Algoritma dan Pemrograman %
```

- Penjelasan

Dalam program ini,

1. Fungsi cek_tuple_sama: Fungsi ini memeriksa apakah semua elemen dalam sebuah tuple memiliki nilai yang sama. Fungsi menggunakan all() untuk mengecek apakah setiap elemen x dalam tuple t sama dengan elemen pertama t[0].

2. Input dan Konversi: Pengguna diminta memasukkan sekumpulan angka yang dipisahkan oleh spasi. Input tersebut diubah menjadi tuple integer menggunakan `map(int, input().split())`.
3. Pengecekan dan Output: Tuple hasil input kemudian diperiksa oleh fungsi `cek_tuple_sama`. Hasil pengecekan (True atau False) dicetak ke layar.

SOAL 2

Buatlah program dengan menggunakan tuple yang dapat melakukan proses seperti pada kasus 11.1, Gunakan data diri anda masing-masing dan lakukan perubahan supaya didapatkan output seperti contoh berikut ini :

Contoh:

```
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')
```

- Source Code dan Output

```
nama = input("Masukkan Nama: ")
nim = input("Masukkan NIM: ")
alamat = input("Masukkan Alamat: ")

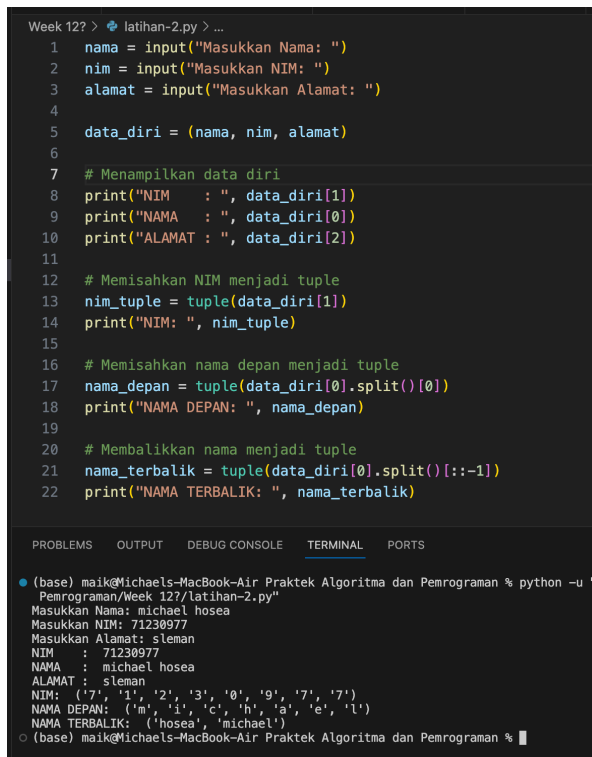
data_diri = (nama, nim, alamat)

# Menampilkan data diri
print("NIM   : ", data_diri[1])
print("NAMA   : ", data_diri[0])
print("ALAMAT : ", data_diri[2])

# Memisahkan NIM menjadi tuple
nim_tuple = tuple(data_diri[1])
print("NIM: ", nim_tuple)

# Memisahkan nama depan menjadi tuple
nama_depan = tuple(data_diri[0].split()[0])
print("NAMA DEPAN: ", nama_depan)
```

```
# Membalikkan nama menjadi tuple
nama_terbalik = tuple(data_diri[0].split()[::-1])
print("NAMA TERBALIK: ", nama_terbalik)
```



The screenshot shows a Python IDE with a file named 'latihan-2.py'. The code defines a tuple 'data_diri' containing name, NIM, and address. It then prints each element, converts NIM and the first part of the name into tuples, and finally prints the name in reverse order as a tuple. The terminal output shows the execution of the program with sample input values: 'michael hosea', '71230977', and 'sleman'.

```
Week 12? > latihan-2.py > ...
1  nama = input("Masukkan Nama: ")
2  nim = input("Masukkan NIM: ")
3  alamat = input("Masukkan Alamat: ")
4
5  data_diri = (nama, nim, alamat)
6
7  # Menampilkan data diri
8  print("NIM      : ", data_diri[1])
9  print("NAMA     : ", data_diri[0])
10 print("ALAMAT    : ", data_diri[2])
11
12 # Memisahkan NIM menjadi tuple
13 nim_tuple = tuple(data_diri[1])
14 print("NIM: ", nim_tuple)
15
16 # Memisahkan nama depan menjadi tuple
17 nama_depan = tuple(data_diri[0].split()[0])
18 print("NAMA DEPAN: ", nama_depan)
19
20 # Membalikkan nama menjadi tuple
21 nama_terbalik = tuple(data_diri[0].split()[::-1])
22 print("NAMA TERBALIK: ", nama_terbalik)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
• (base) maik@Michaels-MacBook-Air: ~ % python -u "
  Pemrograman/Week 12/latihan-2.py"
Masukkan Nama: michael hosea
Masukkan NIM: 71230977
Masukkan Alamat: sleman
NIM      : 71230977
NAMA     : michael hosea
ALAMAT    : sleman
NIM: ('7', '1', '2', '3', '0', '9', '7', '7')
NAMA DEPAN: ('m', 'i', 'c', 'h', 'a', 'e', 'l')
NAMA TERBALIK: ('hosea', 'michael')
• (base) maik@Michaels-MacBook-Air: ~ %
```

- Penjelasan

1. Input Data Diri:
 - Mengambil input nama, NIM, dan alamat dari pengguna.
2. Menyimpan Data dalam Tuple:
 - Menyimpan nama, NIM, dan alamat dalam tuple data_diri.
3. Menampilkan Data Diri:
 - Menampilkan NIM, nama, dan alamat dari tuple data_diri.
4. Memisahkan NIM Menjadi Tuple:
 - Mengubah NIM menjadi tuple karakter dan menampilkannya.
5. Memisahkan Nama Depan Menjadi Tuple:
 - Mengambil nama depan (kata pertama dari nama), mengubahnya menjadi tuple karakter, dan menampilkannya.
6. Membalikkan Nama Menjadi Tuple:
 - Membalik urutan nama (dari nama lengkap) menjadi tuple dan menampilkannya.

SOAL 3

Dengan menggunakan file mbox-short.txt, buatlah program yang dapat membaca log email dan sajikan dalam histogram menggunakan dictionary. Kemudian hitung berapa banyak pesan yang masuk dari email dan sajikan dalam bentuk dictionary.

Silakan cek bagian dibawah ini untuk contoh output dari programnya.

Masukkan nama file : mbox-short.txt

```
{'gopal.ramasammycook@gmail.com': 1, 'louis@media.berkeley.edu': 3,
'cwen@iupui.edu': 5, 'antranig@caret.cam.ac.uk': 1,
'rjlowe@iupui.edu': 2, 'gsilver@umich.edu': 3,
'david.horwitz@uct.ac.za': 4, 'wagnermr@iupui.edu': 1,
'zqian@umich.edu': 4, 'stephen.marquard@uct.ac.za': 2,
'ray@media.berkeley.edu': 1}
```

- Source Code dan Output

```
def count_emails(filename):
    email_counts = {}
    with open(filename, 'r') as file:
        for line in file:
            if line.startswith('From:'):
                email = line.split()[1]
                email_counts[email] = email_counts.get(email, 0) + 1
    return email_counts

filename = input("Masukkan nama file: ")
email_counts = count_emails(filename)
print(email_counts)
```

```
latihan-3.py > menghitung_distribusi_email
1 def menghitung_distribusi_email(filename):
2     try:
3         # Membuka file mbox
4         mbox_file = open(filename, 'r')
5
6         # Membaca setiap baris dalam file mbox
7         lines = mbox_file.readlines()
8
9         # Inisialisasi kamus untuk menyimpan distribusi jam
10        distribution = {}
11
12        # Menghitung distribusi jam
13        for line in lines:
14            if line.startswith('From '):
15                words = line.split()
16                time = words[5].split(':')[0]
17                distribution[time] = distribution.get(time, 0) + 1
18
19        # Menampilkan hasil distribusi jam
20        for hour, count in sorted(distribution.items()):
21            print(hour, count)
22
23        # Menutup file mbox
24        mbox_file.close()
25
26    except FileNotFoundError:
27        print("File not found.")
28
29    # Meminta pengguna untuk memasukkan nama file mbox
30    filename = input("Enter a file name: ")
31
32    # Memanggil fungsi untuk menghitung distribusi jam
33    menghitung_distribusi_email(filename)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python -u "/Users/maik/UKDW/Semester 2/Praktek Algoritma dan Pemrograman/Week 11/alpro-week11
(base) maik@Michaels-MacBook-Air alpro-week11 % python -u "/Users/maik/UKDW/Semester 2/Prakte
han-3.py"
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
(base) maik@Michaels-MacBook-Air alpro-week11 %
```

- Penjelasan

1. Membuka File dan Membaca Baris:

- Fungsi membuka file mbox yang diberikan menggunakan fungsi open() dengan mode 'r' (read) dan kemudian membaca setiap baris dari file tersebut menggunakan metode readlines().

2. Menghitung Distribusi Jam:

- Fungsi memproses setiap baris dalam file mbox yang dimulai dengan 'From '.
- Setiap baris dipisahkan menjadi kata-kata, dan waktu pengiriman email diekstraksi dari kata ke-6.
- Distribusi jam dihitung dengan memperbarui kamus distribution, di mana kunci adalah jam pengiriman email (hanya bagian jam) dan nilainya adalah jumlah email yang dikirim pada jam tersebut. Penghitungan menggunakan metode .get() untuk mengakses nilai saat ini dari kamus, dengan nilai default 0 jika jam belum ada dalam kamus.

3. Menampilkan Hasil:

- Setelah distribusi jam dihitung, hasilnya ditampilkan dalam urutan terurut menggunakan perulangan for. Pemanggilan sorted() digunakan untuk mengurutkan kunci kamus (jam) sebelum ditampilkan.

4. Penanganan Kesalahan:

- Fungsi menggunakan blok try dan except untuk menangani kesalahan yang mungkin terjadi saat membuka file. Jika file tidak ditemukan, fungsi mencetak pesan "File not found."

5. Meminta Nama File dan Memanggil Fungsi:

- Pengguna diminta untuk memasukkan nama file mbox.
- Fungsi `menghitung_distribusi_email` kemudian dipanggil dengan nama file yang dimasukkan oleh pengguna.