



Laporan Praktikum Algoritma & Pemrograman

Semester Genap 2024/2025

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.

NIM	<71231046>
Nama Lengkap	<FREIRE HANAN PUTRA>
Minggu ke / Materi	12 / TUPLES

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2025

BAGIAN 1: MATERI MINGGU INI (40%)

MATERI 1 Tuple Immutable

Tuple mirip dengan list karena dapat menyimpan berbagai jenis nilai dan diakses menggunakan indeks bilangan bulat. Namun, perbedaan utamanya adalah bahwa tuple bersifat immutable, artinya isinya tidak bisa diubah setelah dibuat. Selain itu, tuple bisa dibandingkan antar nilainya (compare) dan bersifat hashable, sehingga bisa digunakan sebagai key dalam struktur data dictionary di Python. Sebuah objek disebut hashable jika ia memiliki nilai hash yang tetap sepanjang hidupnya (dihasilkan oleh method `__hash__()`), dan bisa dibandingkan dengan objek lain menggunakan method seperti `__eq__()` atau `__cmp__()`. Objek hashable dapat dibandingkan satu sama lain jika nilai hash-nya sama. Hashability memungkinkan objek untuk digunakan sebagai kunci pada dictionary atau sebagai elemen dalam set, karena struktur data ini mengandalkan sistem hash internal. Secara umum, objek bawaan Python seperti angka dan string adalah hashable, sedangkan struktur data yang bisa diubah (seperti list atau dictionary) tidak hashable. Jika kamu membuat kelas sendiri, objek dari kelas tersebut akan tidak hashable secara default, kecuali kamu mendefinisikan metode `__hash__()` sendiri.

```
1 # Membuat tuple tanpa tanda kurung, Python otomatis mengenali sebagai tuple
2 t = 'a', 'b', 'c', 'd', 'e'
3 print("Tuple t tanpa kurung:", t)
4
5 # Membuat tuple menggunakan tanda kurung (lebih jelas dibaca)
6 t = ('a', 'b', 'c', 'd', 'e')
7 print("Tuple t dengan kurung:", t)
8
9 # Tuple dengan satu elemen harus diakhiri dengan koma
10 t1 = ('a',)
11 print("Tuple t1 (satu elemen):", t1)
12 print("Tipe t1:", type(t1)) # Akan mencetak: <class 'tuple'>
13
14 # Jika tidak memakai koma, maka dianggap sebagai string biasa, bukan tuple
15 t2 = 'a'
16 print("t2 tanpa koma:", t2)
17 print("Tipe t2:", type(t2)) # Akan mencetak: <class 'str'>
18
19 # Membuat tuple kosong menggunakan fungsi bawaan tuple()
20 t_empty = tuple()
21 print("Tuple kosong:", t_empty)
```

```
1 # Membuat tuple dari sebuah string, setiap karakter akan menjadi elemen tuple
2 t_string = tuple('dutaawacana')
3 print("Tuple dari string 'dutaawacana':", t_string)
4
5 # Indexing pada tuple, mengambil elemen pertama (indeks dimulai dari 0)
6 t = ('a', 'b', 'c', 'd', 'e')
7 print("Elemen pertama:", t[0])
8
9 # Slicing tuple, mengambil elemen dari indeks 1 sampai sebelum indeks 3
10 print("Elemen ke-2 dan ke-3:", t[1:3])
11
12 # Tuple bersifat immutable, artinya elemen tidak bisa diubah langsung
13 try:
14     t[0] = 'A' # Ini akan menghasilkan TypeError karena tuple tidak bisa diubah
15 except TypeError as e:
16     print("Error saat mengubah elemen tuple:", e)
17
18 # Untuk 'mengubah' isi tuple, kita buat tuple baru dengan elemen yang diinginkan
19 # Di sini, elemen pertama diganti dengan 'A' dan digabung dengan elemen sisanya
20 t = ('A',) + t[1:]
21 print("Tuple setelah 'mengubah' elemen pertama:", t)
```

MATERI 2 Membandingkan Tuple

Operator perbandingan (comparison) dapat digunakan pada tipe data tuple maupun struktur data sekuensial lainnya seperti list, dictionary, dan set. Mekanismenya dimulai dengan membandingkan elemen pertama dari masing-masing struktur. Jika elemen pertama sama, maka perbandingan dilanjutkan ke elemen berikutnya secara berurutan. Proses ini terus berlangsung sampai ditemukan elemen yang berbeda, yang kemudian menjadi penentu hasil perbandingan.

Fungsi `sort()` dalam Python bekerja dengan pendekatan serupa, yaitu memulai pengurutan berdasarkan elemen pertama dari setiap entri. Jika elemen pertama sama, proses dilanjutkan ke elemen berikutnya untuk menentukan urutan. Dalam kasus yang lebih kompleks, Python menggunakan teknik yang dikenal sebagai **DSU**, singkatan dari **Decorate, Sort, Undecorate**:

1. **Decorate** – Setiap elemen dalam struktur sekuensial dikombinasikan dengan satu atau lebih kunci (key) yang akan digunakan sebagai dasar pengurutan. Biasanya dibentuk sebagai pasangan tuple.
2. **Sort** – Daftar tuple tersebut kemudian diurutkan menggunakan fungsi `sort()` bawaan Python, yang bekerja berdasarkan kunci yang telah ditambahkan sebelumnya.
3. **Undecorate** – Setelah proses pengurutan selesai, elemen asli dari daftar dikembalikan dengan menghilangkan kunci tambahan tersebut.

Contoh praktisnya adalah mengurutkan kata-kata dalam sebuah kalimat berdasarkan panjangnya, dari yang terpanjang hingga yang terpendek.

```
1 kalimat = 'but soft what light in yonder window breaks'
2 dafkata = kalimat.split()
3 t = list()
4 for kata in dafkata:
5     t.append((len(kata), kata))
6     t.sort(reverse=True)
7 urutan = list()
8 for length, kata in t:
9     urutan.append(kata)
10 print(urutan)
```

Pada **looping pertama**, Python akan membuat daftar berupa tuple, di mana setiap tuple terdiri dari panjang kata dan kata itu sendiri. Proses pengurutan dilakukan berdasarkan elemen pertama dari tuple, yaitu panjang kata. Jika panjangnya sama, maka akan dibandingkan elemen kedua (kata itu sendiri). Penggunaan **reverse=True** berfungsi untuk membalik urutan hasil sortir, sehingga urutannya menjadi dari yang paling panjang ke yang paling pendek (descending).

Pada **looping kedua**, daftar kata disusun ulang dalam bentuk tuple dan diurutkan berdasarkan abjad secara terbalik, dengan mempertimbangkan panjang katanya. Dalam kasus tersebut, empat kata terakhir akan diurutkan menurut urutan alfabet secara menurun. Sebagai hasilnya, kata seperti "what" akan muncul sebelum "soft" dalam daftar.

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

List ini menampilkan kata-kata yang telah diurutkan berdasarkan panjangnya dari yang terpanjang hingga yang terpendek.

MATERI 3 Penugasan Tuple

Salah satu keunikan Python adalah kemampuan untuk menggunakan tuple di sisi kiri dari operasi penugasan. Ini memungkinkan kita menetapkan beberapa variabel sekaligus dalam satu baris. Misalnya, ketika kita memiliki list atau tuple berisi beberapa elemen, kita bisa langsung memisahkan elemen-elemen

tersebut ke dalam variabel yang berbeda. Python menerjemahkan proses ini dengan mengambil elemen satu per satu berdasarkan posisi indeksnya.

Contohnya, kita bisa menukar nilai dua variabel hanya dengan satu baris kode menggunakan fitur ini. Python memperlakukan sisi kiri dan kanan dari penugasan sebagai tuple, dan masing-masing nilai di sisi kanan akan diberikan ke variabel di sisi kiri dengan urutan yang sesuai.

Penting untuk diingat bahwa jumlah elemen di kedua sisi harus sama, jika tidak akan terjadi error. Fitur ini sangat berguna, misalnya saat memisahkan username dan domain dari sebuah alamat email menggunakan `split()`.

```
1 # Membuat list dengan dua elemen
2 m = ['have', 'fun']
3
4 # Multiple assignment (penugasan ganda) menggunakan tuple tanpa tanda kurung
5 x, y = m
6 print("x:", x) # Output: 'have'
7 print("y:", y) # Output: 'fun'
8
9 # Penugasan di atas sama dengan menuliskan:
10 x = m[0]
11 y = m[1]
12 print("Ulangi penugasan manual:")
13 print(x, y)
14
15 # Contoh lain: menggunakan tanda kurung untuk menunjukkan tuple secara eksplisit
17 m = ['have', 'fun']
18 (x, y) = m # Ini sama dengan sebelumnya
19 print("Dengan tanda kurung:")
20 print(x, y)
21 print("y:", y)
```

```
1 # Menukar nilai dua variabel dalam satu baris dengan tuple assignment
2 a = 10
3 b = 20
4 print("Sebelum ditukar: a =", a, ", b =", b)
5 a, b = b, a
6 print("Setelah ditukar: a =", a, ", b =", b)
7
8 # Kesalahan jika jumlah variabel tidak sesuai dengan jumlah elemen
9 try:
10     a, b = 1, 2, 3 # ini akan error
11 except ValueError as e:
12     print("Error:", e)
13
14 # Contoh praktis: memisahkan email menjadi username dan domain
15 email = 'didanendya@ti.ukdw.ac.id'
16 username, domain = email.split('@') # split() mengembalikan list ['didanendya', 'ti.ukdw.ac.id']
17
18 print("Username:", username) # Output: 'didanendya'
19 print("Domain:", domain) # Output: 'ti.ukdw.ac.id'
```

MATERI 4 Dictionaries and Tuple

Dictionary di Python memiliki sebuah metode bernama `items()`, yang mengembalikan daftar berisi pasangan tuple. Masing-masing tuple terdiri dari key dan value dari dictionary tersebut. Perlu dicatat bahwa hasil dari `items()` tidak berurutan karena dictionary tidak menjamin urutan penyimpanan berdasarkan input. Namun karena hasil dari `items()` adalah sebuah list, dan elemen-elemennya berupa tuple, kita bisa memanfaatkannya untuk melakukan pengurutan. Tuple mendukung operasi perbandingan, jadi list ini dapat diurutkan. Dengan mengonversi dictionary menjadi list tuple terlebih dahulu, kita bisa menampilkan isi dictionary yang sudah diurutkan berdasarkan kuncinya.

```
1 # Membuat dictionary dengan beberapa pasangan key-value
2 d = {'a': 10, 'b': 1, 'c': 22}
3
4 # Menggunakan metode items() untuk mengambil daftar tuple dari dictionary
5 t = list(d.items()) # Konversi hasil items() menjadi list
6 print("List tuple dari dictionary (sebelum diurutkan):")
7 print(t) # Output: [('b', 1), ('a', 10), ('c', 22)] - tidak berurutan
8
9 # Mengurutkan list dari tuple berdasarkan key (default sort pada elemen pertama dari tuple)
10 t.sort()
11 print("\nList tuple setelah diurutkan berdasarkan key:")
12 print(t) # Output: [('a', 10), ('b', 1), ('c', 22)]
13
14 # Penjelasan tambahan:
15 # - Fungsi sort() akan mengurutkan berdasarkan elemen pertama dari tuple (yaitu key)
16 # - Karena key adalah string, maka urutannya secara alfabet (ascending)
17
18 # Jika ingin mengurutkan berdasarkan value, bisa gunakan parameter key:
19 t_by_value = sorted(d.items(), key=lambda item: item[1])
20 print("\nList tuple diurutkan berdasarkan value (nilai):")
21 print(t_by_value) # Output: [('b', 1), ('a', 10), ('c', 22)]
```

MATERI 5 Multipenugasan dengan Dictionaries

Python memungkinkan kita untuk menggunakan teknik tuple assignment saat melakukan iterasi melalui dictionary. Ketika kita memanggil `items()` pada dictionary, ia akan mengembalikan daftar tuple yang masing-masing berisi pasangan (key, value). Saat melakukan looping, kita dapat langsung memecah setiap pasangan tuple ke dalam dua variabel, misalnya `key` dan `val`. Dengan kombinasi ini, kita bisa dengan mudah mengakses dan memproses key dan value secara bersamaan di dalam loop. Jika kita ingin mengurutkan dictionary berdasarkan nilai (value), kita bisa membuat list baru berisi tuple (value, key), lalu mengurutkannya. Karena tuple dibandingkan dari elemen pertama, urutan akan otomatis berdasarkan nilai

```
1 # Membuat dictionary dengan beberapa pasangan key-value
2 d = {'a': 10, 'b': 1, 'c': 22}
3
4 # Menggunakan metode items() untuk mengambil daftar tuple dari dictionary
5 t = list(d.items()) # Konversi hasil items() menjadi list
6 print(f'list tuple dari dictionary (sebelum diurutkan):')
7 print(t) # Output: [('b', 1), ('a', 10), ('c', 22)] - tidak berurutan
8
9 # Mengurutkan list dari tuple berdasarkan key (default sort pada elemen pertama dari tuple)
10 t.sort()
11 print(f'list tuple setelah diurutkan berdasarkan key:')
12 print(t) # Output: [('a', 10), ('b', 1), ('c', 22)]
13
14 # Penjelasan tambahan:
15 # - Fungsi sort() akan mengurutkan berdasarkan elemen pertama dari tuple (yaitu key)
16 # - Karena key adalah string, maka urutannya secara alfabet (ascending)
17
18 # Jika ingin mengurutkan berdasarkan value, kita gunakan parameter key:
19 t_by_value = sorted(d.items(), key=lambda item: item[1])
20 print(f'list tuple diurutkan berdasarkan value (nilai):')
21 print(t_by_value) # Output: [('b', 1), ('a', 10), ('c', 22)]
```

MATERI 6 Kata Yang Sering Muncul

Program dibawah ini bertujuan untuk menghitung frekuensi kemunculan kata dalam teks drama *Romeo and Juliet* (Act 2, Scene 2) dari file `romeo-full.txt`. Program membaca file, membersihkan tanda baca, mengubah semua kata menjadi huruf kecil, lalu memecah setiap baris menjadi daftar kata. Semua kata disimpan dalam dictionary dengan kata sebagai key dan jumlah kemunculannya sebagai value. Setelah seluruh teks dibaca dan dihitung, dictionary dikonversi menjadi list of tuples dalam format (jumlah, kata), lalu diurutkan menurun berdasarkan jumlah. Kemudian program mencetak 10 kata yang paling sering muncul, disertai jumlah kemunculannya.

```
1 import string # Mengimpor modul string untuk menghapus tanda baca
2
3 # Membuka file teks Romeo and Juliet
4 fhand = open('romeo-full.txt')
5
6 counts = dict() # Dictionary kosong untuk menyimpan jumlah kata
7
8 # Membaca file baris per baris
9 for line in fhand:
10     # Menghapus tanda baca dari setiap baris
11     line = line.translate(str.maketrans('', '', string.punctuation))
12     # Mengubah semua huruf menjadi huruf kecil
13     line = line.lower()
14     # Memecah baris menjadi list kata
15     words = line.split()
16
```

```
1 # Mengisi dictionary dengan kata dan frekuensinya
2 for word in words:
3     if word not in counts:
4         counts[word] = 1 # Jika kata belum ada, masukkan dengan nilai 1
5     else:
6         counts[word] += 1 # Jika sudah ada, tambahkan 1
7
8 # Membuat list tuple (jumlah, kata) untuk proses sorting
9 lst = list()
10 for key, val in counts.items():
11     lst.append((val, key)) # Format (jumlah, kata)
12
13 # Mengurutkan list berdasarkan jumlah (descending)
14 lst.sort(reverse=True)
15
16 # Menampilkan 10 kata teratas yang paling sering muncul
17 for val, key in lst[:10]: # Iterasi hanya 10 item pertama
18     print(val, key)
19
```

MATERI 7 Tuple Sebagai Kunci Dictionaries

Tuple dalam Python bersifat hashable, artinya bisa digunakan sebagai key dalam dictionary. Berbeda dengan list, yang tidak bisa digunakan sebagai key karena mutable. Ini sangat berguna ketika kita ingin menggunakan beberapa nilai (misalnya nama depan dan nama belakang) sebagai satu key tunggal. Contohnya, kita bisa membuat direktori telepon yang memetakan pasangan (nama belakang, nama depan) ke nomor telepon. Dalam iterasi, kita bisa langsung memecah tuple ke dua variabel untuk digunakan dalam proses pencetakan.

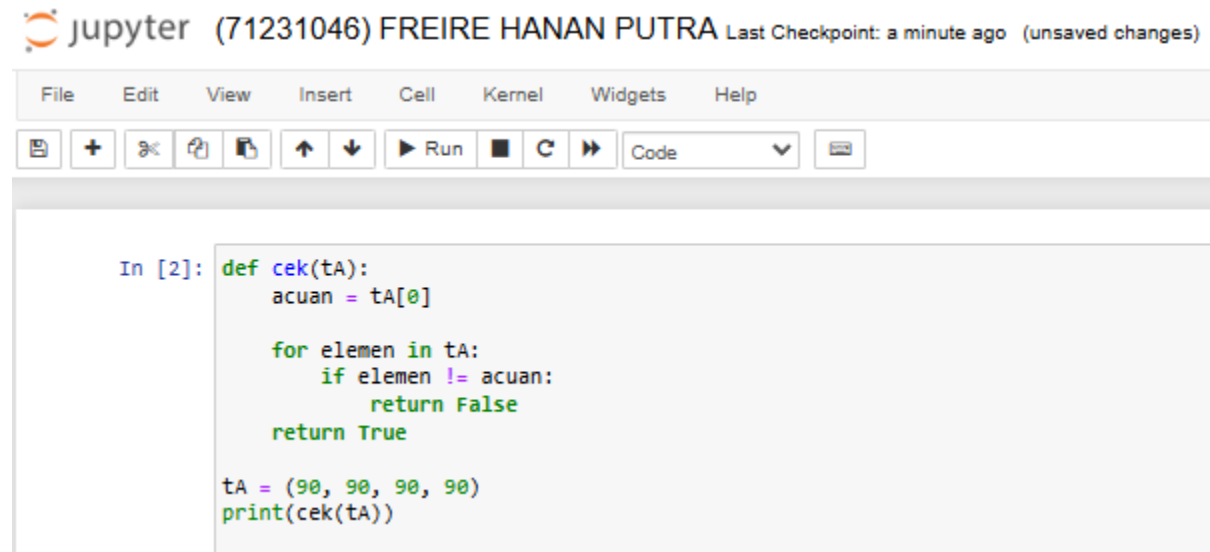
```
1 # Mendefinisikan nilai-nilai untuk nama belakang, nama depan, dan nomor telepon
2 last = 'nendya'      # Variabel last menyimpan string 'nendya' sebagai nama belakang
3 first = 'dida'        # Variabel first menyimpan string 'dida' sebagai nama depan
4 number = '088112266' # Variabel number menyimpan string nomor telepon
5
6 # Membuat dictionary kosong untuk direktori telepon
7 directory = dict()    # Inisialisasi dictionary kosong bernama directory
8
9 # Menggunakan tuple (last, first) sebagai key dalam dictionary
10 directory[last, first] = number
11 # Menyimpan data ke dictionary dengan key berupa tuple (last, first)
12 # Artinya key-nya adalah ('nendya', 'dida'), dan value-nya adalah nomor telepon
13
14 # Melakukan iterasi pada dictionary, key-nya adalah tuple (last, first)
15 for last, first in directory:
16     # Looping akan mengambil setiap key dari dictionary
17     # Setiap key adalah tuple (last, first), yang dipecah menjadi variabel last dan first
18
19     # Menampilkan nama depan, nama belakang, dan nomor telepon
20     print(first, last, directory[last, first])
21 # Mengakses value dictionary dengan key tuple (last, first)
22 # Mencetak: nama depan, nama belakang, dan nomor telepon sesuai pasangan key-value
```

BAGIAN 2: LATIHAN MANDIRI (60%)

Link Github: https://github.com/Freirehnn23/prak_alpro_week13

SOAL 1

Code:



The screenshot shows a Jupyter Notebook interface. At the top, the text "jupyter (71231046) FREIRE HANAN PUTRA" is displayed, followed by "Last Checkpoint: a minute ago (unsaved changes)". Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Under the menu bar is a toolbar with icons for saving, adding cells, undo, redo, and running code. The main area contains a code cell with the following Python code:

```
In [2]: def cek(tA):
        acuan = tA[0]

        for elemen in tA:
            if elemen != acuan:
                return False
        return True

        tA = (90, 90, 90, 90)
        print(cek(tA))
```

Below the code cell, the output "True" is displayed.

Penerapan:

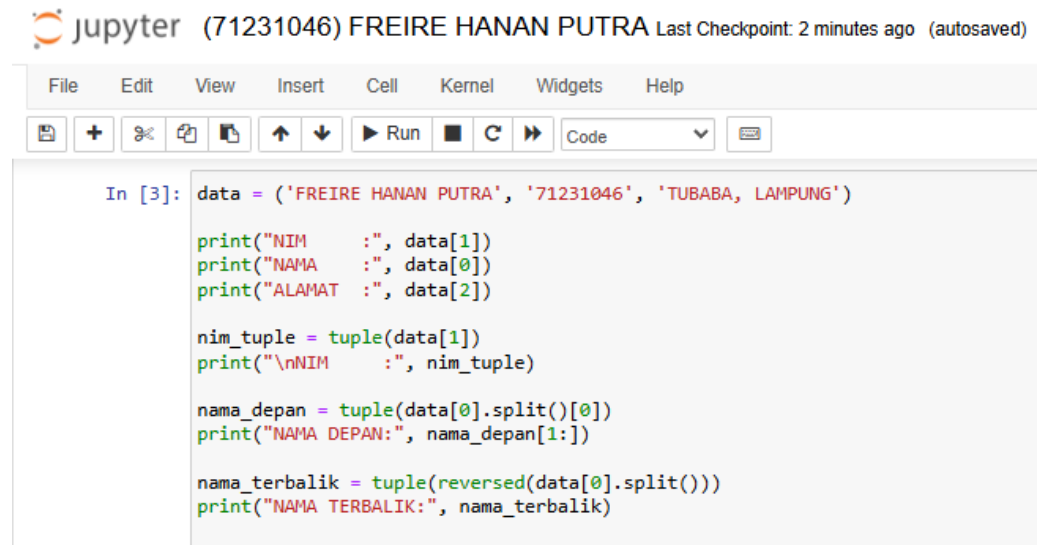
True

Penjelasan:

- Pertama buat fungsi dengan parameter tA
- Setelah itu buat variabel untuk mengakses indeks tA yang paling kiri untuk di looping
- Setelah itu loop setiap elemen di tA untuk mengecek apakah indeks paling kiri = indeks paling kanan
- Setelah itu jika tidak maka return false namun jika benar return true
- Setelah itu tampilkan hasilnya

SOAL 2

Code:



The screenshot shows a Jupyter Notebook window titled "jupyter (71231046) FREIRE HANAN PUTRA" with a status bar indicating "Last Checkpoint: 2 minutes ago (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, undo, redo, and running code. The active code cell contains the following Python code:

```
In [3]: data = ('FREIRE HANAN PUTRA', '71231046', 'TUBABA, LAMPUNG')

print("NIM      :", data[1])
print("NAMA     :", data[0])
print("ALAMAT   :", data[2])

nim_tuple = tuple(data[1])
print("\nNIM      :", nim_tuple)

nama_depan = tuple(data[0].split()[0])
print("NAMA DEPAN:", nama_depan[1:])

nama_terbalik = tuple(reversed(data[0].split()))
print("NAMA TERBALIK:", nama_terbalik)
```

Penerapan:

```
NIM      : 71231046
NAMA     : FREIRE HANAN PUTRA
ALAMAT   : TUBABA, LAMPUNG

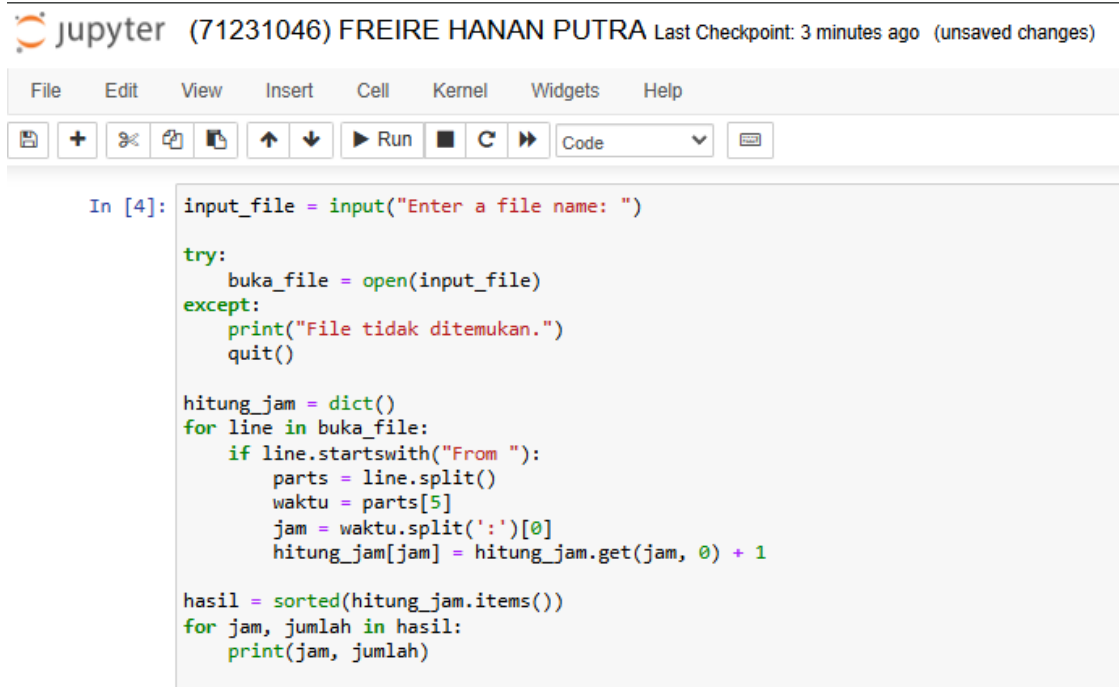
NIM      : ('7', '1', '2', '3', '1', '0', '4', '6')
NAMA DEPAN: ('R', 'E', 'I', 'R', 'E')
NAMA TERBALIK: ('PUTRA', 'HANAN', 'FREIRE')
```

Penjelasan:

- Pertama buat variabel yang berisi list data
- Setelah itu print indeks ke pertama (0) untuk menampilkan NIM
- Lakukan hal yang sama untuk menampilkan nama dan alamat sesuai indeks pada list
- Untuk menampilkan nim sesuai yang diminta, ubah list ke dalam tuple untuk memisahkan setiap karakter pada list yang diakses berdasarkan indeks. Lakukan hal yang sama untuk menampilkan nama yang dipisah per karakter. Namun bedanya adalah pada bagian nama, indeks paling kiri tidak di tampilkan maka menggunakan indeks dengan cara mengakses indeks mulai dari satu indeks dari paling kiri dengan cara [1:] hingga akhir. Gunakan fungsi tuple untuk memisahkan per karakter secara otomatis
- Untuk nama terbalik gunakan fungsi reverse untuk memutar karakte dari paling kiri menjadi paling kanan
- Setelah itu print hasil

SOAL 3

Code:



```
In [4]: input_file = input("Enter a file name: ")

try:
    buka_file = open(input_file)
except:
    print("File tidak ditemukan.")
    quit()

hitung_jam = dict()
for line in buka_file:
    if line.startswith("From "):
        parts = line.split()
        waktu = parts[5]
        jam = waktu.split(':')[0]
        hitung_jam[jam] = hitung_jam.get(jam, 0) + 1

hasil = sorted(hitung_jam.items())
for jam, jumlah in hasil:
    print(jam, jumlah)
```

Penerapan:

```
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
```

Penjelasan:

- Pertama buat try except untuk mengecek apakah file ada atau tidak, jika tidak maka print file tidak ditemukan namun jika ditemukan maka lanjut
- Setelah itu buat dictionary untuk menyimpan hasil looping
- Setelah itu looping setiap line yang ada di file setelah itu mulai cek setiap line yang berawal dari kata from.
- Setelah itu buat sebuah variabel untuk split line yang ada pada file

- Setelah itu akses indeks waktu dan jam di split dengan “:” dengan mengakses indeks 0
- Setelah itu panggil dictionaries dengan mengakses indeks jam yang sudah displit setelah itu ambil nilai jam. Jika nilai jam belum ada di dictionaries maka default nilainya adalah 0
- Setelah itu tambahkan satu ke dalam dictionaries untuk menambahkan dan menandai bahwa jam sudah muncul satu kali. Looping kondisi ini jika baris pada file sudah habis.
- Setelah itu buat variabel hasil untuk mengurutkan hasil looping nya
- Setelah itu print dictionaries dengan mengakses key dan value jam dan jumlah dalam variabel hasil Dimana hasil berisi dictionaries