



# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

<b>NIM</b>	<b>71230971</b>
<b>Nama Lengkap</b>	<b>James Marvin Santoso</b>
<b>Minggu ke / Materi</b>	<b>11 / Tipe Data Tuples</b>

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

### TUPLE IMMUTABLE

Tuple adalah salah satu struktur data dalam Python yang mirip dengan list, namun dengan perbedaan utama yaitu tuple bersifat immutable, artinya setelah tuple dibuat, elemen-elemennya tidak dapat diubah. Immutable berarti bahwa elemen-elemen dalam tuple tidak dapat ditambah, dihapus, atau diubah nilainya setelah tuple dibuat.

Keuntungan menggunakan tuple adalah karena sifatnya yang immutable, sehingga membuat tuple menjadi pilihan yang baik untuk menyimpan sekumpulan data yang tidak perlu diubah. Ini juga membuat tuple lebih efisien secara memori dibandingkan dengan list, karena Python tidak perlu melakukan alokasi ulang memori untuk tuple yang sudah dibuat.

Contoh sederhana penggunaan tuple:

```
# Mendefinisikan tuple
my_tuple = (1, 2, 3, 4, 5)

# Mengakses elemen tuple
print(my_tuple[0]) # Output: 1

# Slicing tuple
print(my_tuple[1:4]) # Output: (2, 3, 4)

# Tuple concatenation
new_tuple = my_tuple + (6, 7, 8)
print(new_tuple) # Output: (1, 2, 3, 4, 5, 6, 7, 8)
```

1. Mendefinisikan tuple: Tuple `my_tuple` dibuat dengan elemen-elemen 1, 2, 3, 4, dan 5.
2. Mengakses elemen tuple: Elemen pertama dari tuple `my_tuple` diakses menggunakan indeks 0 dan dicetak menggunakan fungsi `print`. Outputnya adalah 1.
3. Slicing tuple: Menggunakan slicing untuk mengambil bagian dari tuple. `my_tuple[1:4]` akan menghasilkan tuple baru yang berisi elemen dari indeks 1 hingga indeks 3 (indeks 4 tidak termasuk), yaitu (2, 3, 4). Ini dicetak menggunakan fungsi `print`.
4. Tuple concatenation: Tuple `my_tuple` digabungkan dengan tuple baru (6, 7, 8) menggunakan operator `+`, menghasilkan tuple baru `new_tuple` yang berisi semua elemen dari `my_tuple` dan tuple baru tersebut. Hasilnya adalah (1, 2, 3, 4, 5, 6, 7, 8), yang kemudian dicetak menggunakan fungsi `print`.

## MEMBANDINGKAN TUPLE

Dalam Python, perbandingan antara dua tuple dilakukan dengan membandingkan elemen-elemen mereka secara berurutan. Python membandingkan elemen pertama dari kedua tuple. Jika elemen pertama dari kedua tuple sama, maka akan dibandingkan elemen kedua, dan begitu seterusnya hingga ditemukan elemen yang berbeda atau salah satu tuple habis.

Jika pada suatu titik terdapat perbedaan di antara elemen-elemen yang sedang dibandingkan, hasil perbandingan akan ditentukan oleh nilai elemen tersebut. Misalnya, jika pada posisi tertentu elemen tuple pertama lebih kecil dari elemen tuple kedua, maka tuple pertama dianggap lebih kecil secara keseluruhan.

Contoh sederhana membandingkan tuple, sebagai berikut:

```
tuple1 = (1, 2, 3)
tuple2 = (1, 2, 4)

# Membandingkan tuple1 dan tuple2
if tuple1 < tuple2:
    print("tuple1 lebih kecil dari tuple2")
elif tuple1 == tuple2:
    print("tuple1 sama dengan tuple2")
else:
    print("tuple1 lebih besar dari tuple2")

# Output: tuple1 lebih kecil dari tuple2

tuple3 = (1, 2)
tuple4 = (1, 2, 3)

# Membandingkan tuple3 dan tuple4
if tuple3 < tuple4:
    print("tuple3 lebih kecil dari tuple4")
else:
    print("tuple3 tidak lebih kecil dari tuple4")

# Output: tuple3 lebih kecil dari tuple4
```

Dalam contoh pertama, karena elemen ketiga dari tuple1 lebih kecil dari elemen ketiga tuple2, maka tuple1 dianggap lebih kecil secara keseluruhan. Sedangkan dalam contoh kedua, tuple3 dianggap lebih kecil dari tuple4 karena tuple3 adalah awalan dari tuple4.

## PENUGASAN TUPLE

Dalam Python, tuple adalah tipe data yang bersifat immutable, artinya setelah tuple dibuat, elemen-elemennya tidak dapat diubah. Namun, Anda masih dapat melakukan penugasan (assignment) tuple untuk mengatur ulang nilai-nilai dalam tuple atau mengekstrak nilai-nilai dari tuple ke dalam variabel terpisah. Berikut adalah contoh penugasan tuple dalam Python:

```
# Mendefinisikan tuple
person = ("John", 30, "New York")

# Mengekstrak nilai dari tuple ke dalam variabel terpisah
name, age, city = person

# Menampilkan nilai variabel
print("Nama:", name) # Output: Nama: John
print("Usia:", age) # Output: Usia: 30
print("Kota:", city) # Output: Kota: New York
```

Dalam contoh ini, kita mendefinisikan tuple `person` yang berisi informasi tentang seseorang (nama, usia, kota). Kemudian, kita menggunakan penugasan tuple untuk mengekstrak nilai-nilai tersebut ke dalam variabel terpisah `name`, `age`, dan `city`, dan kemudian menampilkan nilai-nilai tersebut.

## DICTIONARIES DAN TUPLE

Dictionaries dan tuple adalah dua struktur data yang berbeda dalam Python, dengan kegunaan dan karakteristik yang berbeda pula.

1. Dictionaries (Kamus):
  - Dictionaries adalah struktur data yang memetakan kunci (key) ke nilai (value).
  - Kunci dalam sebuah dictionary harus unik dan dapat berupa string, angka, atau tuple yang bersifat immutable.
  - Nilai dalam dictionary dapat berupa objek Python apa pun.
  - Dictionaries ditulis dalam kurung kurawal `{}` dengan format `key: value`.
2. Tuple:
  - Tuple adalah struktur data yang menyimpan urutan elemen yang tidak dapat diubah (immutable).
  - Elemen dalam tuple dapat berupa objek Python apa pun.
  - Tuple ditulis dalam tanda kurung biasa `()` dengan elemen-elemen dipisahkan oleh koma.

Perbedaan utama antara dictionaries dan tuple adalah bahwa dictionaries digunakan untuk menyimpan data dengan kunci-nilai yang bersifat dinamis dan dapat diubah, sedangkan tuple digunakan untuk menyimpan sekumpulan data yang bersifat statis dan tidak dapat diubah setelah tuple dibuat.

## MULTIPENUGASAN DENGAN DICTIONARIES

Multipenugasan (multiple assignment) dengan dictionaries dalam Python memungkinkan Anda untuk menugaskan nilai-nilai dari beberapa kunci dalam dictionary ke dalam variabel secara bersamaan. Hal ini mempermudah dalam mengakses dan menggunakan nilai-nilai tersebut dalam kode

Berikut adalah contoh penggunaannya:

```
# Mendefinisikan dictionary
person = {
    "name": "John",
    "age": 30,
    "city": "New York",
    "country": "USA"
}

# Multipenugasan dengan dictionaries
name, age, city, country = person["name"], person["age"], person["city"],
person["country"]

# Menampilkan nilai variabel
print("Nama:", name)      # Output: Nama: John
print("Usia:", age)       # Output: Usia: 30
print("Kota:", city)      # Output: Kota: New York
print("Negara:", country) # Output: Negara: USA
```

Pada contoh ini, kita memiliki dictionary person yang berisi informasi tentang seseorang, termasuk nama, usia, kota, dan negara. Dengan menggunakan multiple assignment, kita menugaskan nilai-nilai dari kunci-kunci dalam dictionary tersebut ke dalam variabel name, age, city, dan country. Hal ini memungkinkan kita untuk dengan mudah mengakses nilai-nilai tersebut dan membuat kode lebih mudah dibaca.

## TUPLE SEBAGAI KUNCI DICTIONARIES

Di Python, struktur data dictionary digunakan untuk mengaitkan nilai (value) dengan kunci (key) tertentu. Salah satu fitur menarik dari dictionary adalah kemampuannya untuk menggunakan tuple sebagai kunci. Ini terjadi karena tuple bersifat immutable (tidak dapat diubah), sehingga aman digunakan sebagai kunci dalam struktur data yang membutuhkan kunci yang stabil.

```
# Mendefinisikan dictionary dengan tuple sebagai kunci
data = {
    ('John', 'Doe'): 25,
    ('Jane', 'Smith'): 30,
    ('Alice', 'Brown'): 35
}

# Mengakses nilai menggunakan tuple sebagai kunci
print(data[('John', 'Doe')]) # Output: 25
print(data[('Jane', 'Smith')]) # Output: 30
```

Dalam contoh ini, kita menggunakan tuple (nama\_depan, nama\_belakang) sebagai kunci dalam dictionary data. Kita dapat mengakses nilai dalam dictionary dengan menggunakan tuple sebagai kunci, serta melakukan operasi seperti penambahan, penghapusan, dan pengubahan nilai menggunakan tuple sebagai kunci.

Penggunaan tuple sebagai kunci dalam dictionary dapat sangat berguna dalam situasi di mana kita perlu menggunakan beberapa nilai yang saling terkait sebagai kunci dan ingin memastikan bahwa kunci tersebut tidak berubah setelah ditetapkan.

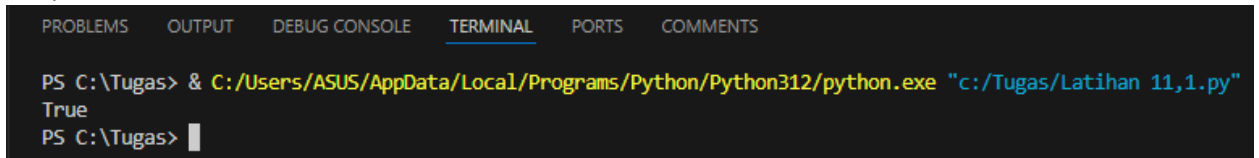
## BAGIAN 2: LATIHAN MANDIRI

### SOAL 11.1

- Source Code :

```
def cek_sama(t):  
    return len(set(t)) == 1  
  
tA = (90, 90, 90, 90)  
  
if cek_sama(tA):  
    print("True")  
else:  
    print("False")
```

- Output :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS  
  
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 11,1.py"  
True  
PS C:\Tugas> █
```

- Penjelasan :

Kode tersebut memiliki dua bagian utama. Pertama, ada definisi fungsi `cek_sama(t)` yang menggunakan set untuk memeriksa apakah semua elemen dalam tuple `t` sama. Jika panjang set dari `t` (yaitu setelah menghilangkan elemen yang duplikat) adalah 1, maka fungsi mengembalikan `True`, yang menunjukkan bahwa semua elemen dalam tuple sama. Kedua, kode menginisialisasi tuple `tA` dengan empat elemen yang semuanya bernilai 90. Kemudian, kode memanggil fungsi `cek_sama(tA)` dan jika hasilnya `True`, mencetak "True", jika tidak, mencetak "False". Dalam kasus ini, karena semua elemen dalam tuple `tA` memiliki nilai yang sama (90), fungsi `cek_sama(tA)` mengembalikan `True`, sehingga hasilnya mencetak "True".

## SOAL 11.2

- Source Code :

```
data_diri = ('Matahari Bhakti Nendya', '22064091', 'Bantul, DI Yogyakarta')
nama, nim, alamat = data_diri

print("Data:", data_diri)
print("NIM :", nim)
print("NAMA :", nama)
print("ALAMAT :", alamat)

nim_tuple = tuple(nim)
nama_depan_tuple = tuple(nama.split()[0])
nama_terbalik_tuple = tuple(reversed(nama.split()))

print("NIM:", nim_tuple)
print("NAMA DEPAN:", nama_depan_tuple)
print("NAMA TERBALIK:", nama_terbalik_tuple)
```

- Output :



```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 11,2.py"
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: ('M', 'a', 't', 'a', 'h', 'a', 'r', 'i')
NAMA TERBALIK: ('Nendya', 'Bhakti', 'Matahari')
PS C:\Tugas>
```

- Penjelasan :  
Kode tersebut pertama-tama menginisialisasi tuple `data_diri` yang berisi informasi nama, NIM, dan alamat seseorang. Kemudian, kode tersebut meng-ekstrak nilai-nilai dari tuple tersebut ke dalam variabel `nama`, `nim`, dan `alamat`, dan menampilkannya dengan menggunakan fungsi `print`. Selanjutnya, kode tersebut membuat tuple baru `nim_tuple` yang berisi karakter-karakter dari NIM, `nama_depan_tuple` yang berisi karakter-karakter dari nama depan, dan `nama_terbalik_tuple` yang berisi kata-kata dari nama yang telah dibalik urutannya. Dalam contoh ini, nama "Matahari Bhakti Nendya" akan diubah menjadi tuple ('y', 'a', 'd', 'n', 'e', 'N', 'i', 't', 'k', 'a', 'h', 'B', ' ', 'i', 'r', 'a', 'h', 'a', 't', 'a', 'M') untuk `nama_terbalik_tuple`.



### SOAL 11.3

- Source Code :

```
nama_file = input("Masukkan nama file: ")

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

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

        for jam, jumlah in sorted(distribusi_jam.items()):
            print(jam, jumlah)

except FileNotFoundError:
    print("File tidak ditemukan.")
except Exception as e:
    print("Terjadi kesalahan:", e)
```

- Output :

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 11,3.py"
Masukkan nama file: 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
PS C:\Tugas> █
```

- Penjelasan :

Kode tersebut untuk memasukkan nama file, kemudian membaca file tersebut untuk mengidentifikasi baris yang dimulai dengan 'From ', memecah baris-baris tersebut untuk mendapatkan jam pengiriman email, menghitung distribusi jumlah email yang dikirim pada setiap jam, dan mencetak hasilnya setelah diurutkan berdasarkan jam, dengan penanganan kasus jika file tidak ditemukan atau terjadi kesalahan selama proses tersebut.

- Link GitHub :

<https://github.com/Jamesmarvins/Tugas-Alpro-11.git>