VERSI 2.2

AGUSTUS 2025

PEMROGRAMAN FUNGSIONAL

MODUL 1 - Functional Programming using Python

DISUSUN OLEH:

FERA PUTRI AYU L., S.KOM., M.T.

ALFI AULIA AZZAHRA

RAHMATUN NIKMAH

TIM LABORATORIUM INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PENDAHULUAN

TUJUAN

1. Sub-CPMK 7: Mahasiswa mampu mendesain program dengan teknik yang tepat untuk menyelesaikan masalah dengan menggunakan paradigma pemrograman fungsional (P6)

TARGET MODUL

1. Menguasai konsep Function dalam Paradigma Fungsional
2. Menguasai konsep Tipe Data Sequence pada Python

PERSIAPAN

1. Laptop/Komputer
2. Python, Google Collab/VS Code/Jupyter Notebook
3. *Source Code Modul 1*

KEYWORDS

Functional Programming Paradigm, Python, Function, Variable, Data Type, Sequence Type

TABLE OF CONTENTS

1. FUNCTIONAL PROGRAMMING PARADIGM

Di semester sebelumnya kita sudah belajar beberapa pemrograman, seperti Pemrograman Dasar, PBO (Pemrograman Berorientasi Objek), Pemrograman Lanjut, dan Struktur Data. Nah, kali ini kita akan mempelajari *Pemrograman Fungsional* , yaitu cara menulis program dengan berfokus pada fungsi.

Sederhananya, dalam pemrograman fungsional kita membuat fungsi-fungsi yang bertugas melakukan suatu perhitungan. Setelah itu, fungsi tersebut bisa kita panggil kembali kapanpun dibutuhkan.

1.1. TENTANG PARADIGMA FUNGSIONAL

Pemrograman fungsional termasuk gaya pemrograman deklaratif. Fokusnya ada pada “apa yang ingin dicapai”, bukan “bagaimana cara mencapainya”. Kebalikan dari gaya pemrograman deklaratif adalah pemrograman imperatif. Dalam pemrograman imperatif, kita menuliskan instruksi langkah demi langkah agar komputer tahu persis bagaimana sebuah tugas harus dilakukan. Artinya, fokus utamanya ada pada proses atau “bagaimana cara mencapainya”. Gaya pemrograman inilah yang selama ini kita pelajari di semester sebelumnya.

Sementara itu, dalam pemrograman deklaratif (termasuk fungsional), perhatian lebih ditekankan pada hasil akhir yang diinginkan, tanpa perlu menjelaskan detail prosesnya. Sebagai ilustrasi:

- Pemrograman imperatif itu seperti menulis resep masakan lengkap, langkah demi langkah.
- Pemrograman deklaratif itu seperti memesan kue di kafe — kita hanya menyebutkan kue apa yang diinginkan, tanpa perlu tahu cara membuatnya.

Karena lebih deklaratif, kode dalam pemrograman fungsional biasanya lebih singkat dan mudah dibaca, sebab kita tidak perlu repot mengurusi detail teknis yang rumit.

1.2. FITUR PEMROGRAMAN FUNGSIONAL

Program fungsional bisa dianggap sebagai kumpulan fungsi yang menerima input lalu menghasilkan output, tanpa menyimpan atau mengubah data di luar fungsi tersebut (*stateless*). Artinya, jika inputnya sama, hasilnya juga akan selalu sama. Fungsi seperti ini disebut fungsi murni (*Pure Function*). Selain itu, ada beberapa ciri penting lain dalam paradigma ini, misalnya:

- *Recursion* (rekursif) sebagai pengganti perulangan (looping)

- *High-Order Function* (fungsi yang bisa menerima atau mengembalikan fungsi lain)
- *Lambda Function* (fungsi tanpa nama)
- Pemanfaatan beberapa *Functional Tools* (seperti fungsi Map, Filter, Reduce)

Hal-hal inilah yang nantinya akan kita pelajari dalam praktikum pemrograman fungsional. Tapi tenang, bertahap, tidak semuanya sekarang. Okay..

2. TOOLS

Konsep-konsep paradigma fungsional seperti pure function, lambda function, dan lain-lain yang akan kita bahas di modul selanjutnya juga banyak diterapkan pada beberapa bahasa seperti: JavaScript, Haskell, Ruby, dan sebagainya. Namun, dalam praktikum ini, kita akan menggunakan Python untuk menerapkan konsep pemrograman fungsional.

Kenapa python? Karena python adalah bahasa pemrograman yang fleksibel dengan dukungan paradigma fungsional serta semantik dinamis. Bahasa ini terkenal dengan sintaksisnya yang sederhana dan mudah dipelajari, serta memiliki pustaka standar yang luas.

Python mendukung berbagai kebutuhan, mulai dari pengembangan web, analisis data, kecerdasan buatan (artificial intelligence), hingga otomasi sistem. Selain itu, Python dapat berjalan di berbagai sistem operasi seperti Linux, Windows, macOS, bahkan Android.

Untuk praktikum ini, pastikan Python sudah terinstal di komputer masing-masing. Kalau ada kendala instalasi, bisa cek Modul 0 atau langsung tanya asisten yaa... Berikut adalah beberapa tools yang bisa dipakai dalam praktikum Pemrograman Fungsional ini, bebas terserah mau pakai yang mana:

- Visual Studio Code
- Google colab
- Jupyter Notebook

Tapi, paling direkomendasikan pakai Google Colab sih karena simpel, nggak perlu instal apa-apa, dan *source code* modul juga sudah disediakan dalam format Colab jadi tinggal pencet-pencet *run* aja. Eits, tapi walaupun gitu tetap siapin juga Visual Studio Code di laptop masing-masing ya, buat jaga-jaga kalau tiba-tiba dibutuhin. Dah intinya, instalasi lengkapnya ada di Modul 0.

3. LET'S TALK ABOUT PYTHON

Pada bagian ini kita tidak lagi membahas Python sebagai *tools,* tetapi langsung masuk ke dasar-dasar pemrograman dengan Python. Fokus kita adalah mengenal konsep dasar Python, mulai dari variabel, tipe data, hingga struktur kode sederhana.

Python dikenal sebagai bahasa dengan sintaks yang ringkas dan mudah dibaca, sehingga sangat cocok dipakai untuk belajar pemrograman. Melalui poin-poin berikut, kita akan memahami bagaimana Python mengelola data, membuat percabangan, melakukan perulangan, hingga membuat fungsi.

3.1. Tipe Data dan Variabel dalam Python

Sebelum kita mulai praktik, kita samakan dulu frekuensi pemahaman kita.

3.1.1. Apa itu Variabel dalam Python

Anggap saja variabel adalah sebuah keranjang. Tempat untuk kita meletakkan sesuatu (data) di dalamnya. Di python, kita bisa memasukkan data tipe apa saja ke dalam keranjang (variabel) tanpa harus mendefinisikan tipenya terlebih dahulu. Hal ini berbeda dengan beberapa bahasa pemrograman lain umumnya, yang mengharuskan kita mendefinisikan tipe data di awal (seperti int, string, float, dll). Sebuah variabel di python dapat berganti tipe data sedinamis mungkin. Perhatikan dan jalankan baris kode di bawah agar kalian lebih memahami penggunaan variabel pada python!

```
#deklarasi Variabel

nama = "Tulis Namamu Disini"
NIM = 202210370311000
ipk = 7.00
```

Pastikan kalian sudah menjalankan (run) cell code #deklarasi di atas sebelum *run cell* #cetak dibawah berikut!

Cell ini dapat di run lagi setelah kalian melakukan experimen dengan cell #redefine dan perhatikan bedanya. Jangan lupa untuk melaporkan hasilnya!

```
#cetak isi dari variabel

print("Nama \t:", nama,"\t| Tipe data: ", type(nama))
print("NIM \t:", NIM, "| Tipe data\t:", type(NIM))
print("IPK \t:", ipk, "\t| Tipe data:", type(ipk))
```

📌 Latihan 1

```
#redefine variable

NIM = '202210370311000'
ipk = False

# PERCOBAAN1
```

3.1.2. Bagaimana dengan Tipe Data dalam Python?

Tipe data –sesuai namanya– adalah jenis dari suatu data. Setiap data memiliki nilai, dan setiap nilai dikategorikan ke dalam beberapa jenis/tipe. Memahami tipe data di Python adalah aspek dasar dalam pemrograman, karena dapat membantu kita mengenali jenis data yang digunakan, ukurannya, serta fungsi-fungsi yang terkait.

Pengetahuan tentang tipe data ini dapat membuat proses pengkodean lebih efisien. Tipe data yang spesifik menentukan nilai yang dapat kita tetapkan hingga operasi-operasi (terhadap data) yang bisa kita lakukan. Python sendiri mempunyai tipe data yang cukup unik bila dibandingkan dengan bahasa pemrograman yang lain. Diantaranya adalah sebagai berikut :

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

Mari kita bahas lebih lanjut untuk beberapa tipe data yang ada di python, *check this out*!

A. Text Type

a. String

Python tidak memiliki tipe char spesifik seperti di beberapa bahasa pemrograman lain seperti C atau Java. Di Python, satu karakter adalah string dengan length satu.

```
contoh_str = 'A'
print(contoh_str)
print(type(contoh_str))
```

```
A
<class 'str'>
```

Pada kode di atas, variabel `contoh_str` berisi satu karakter 'A'. Ketika dijalankan `print(contoh_str)`, hasil yang ditampilkan adalah huruf "A". Selanjutnya, jika menggunakan fungsi `type(contoh_str)`, Python akan menampilkan output: <class 'str'>. Hal ini menunjukkan bahwa nilai `'A'` pada variabel `contoh_str` dianggap sebagai objek dari kelas string (`str`), bukan `char` seperti pada beberapa bahasa pemrograman lain. Dengan kata lain, di Python semua data diperlakukan sebagai objek, termasuk string.

b. Concat String

Di bawah ini merupakan contoh concat atau penggabungan 2 string dari variabel yang berbeda ke dalam 1 variabel.

```
kata = "I love you"
kata2 = ", and you love me back 💗"

# contoh concat
concat = kata + kata2
print(concat)
```

```
I love you, and you love me back 💗
```

B. Numeric Type

Berikut merupakan beberapa contoh dari tipe data *numeric* yang ada di python.

a. Int

```
contoh_int = 10
print(contoh_int)
print(type(contoh_int))
```

```
10
<class 'int'>
```

Pada Python, tipe data int digunakan untuk merepresentasikan bilangan bulat (integer), misalnya -3, 0, 10, atau 1000. Di bahasa pemrograman lain seperti C/C++ atau Java, tipe data bilangan bulat biasanya memiliki ukuran tertentu yang dibedakan lagi menjadi short, int, atau long. Setiap jenisnya punya batas nilai yang bisa disimpan.

Namun, Python lebih fleksibel karena tipe data int akan menyesuaikan ukuran secara otomatis sesuai dengan nilai yang diberikan. Artinya, kita tidak perlu menentukan apakah sebuah bilangan itu masuk ke dalam kategori short, int, atau long. Python akan mengatur kebutuhan memorinya sendiri. Perhatikan kode berikut:

📌 Latihan 2

```
import sys
print(sys.getsizeof(-123456789))
print(sys.getsizeof(1234567890))

# PERCOBAAN2
```

Semakin besar value dari sebuah int maka jumlah byte juga semakin besar, sesuai dengan kebutuhan memori dari value tersebut. *For your information*, berikut sedikit keterangan tentang *size*/ukuran suatu penyimpanan data di python 3.

```
Bytes  type        scaling notes
28     int         +4 bytes about every 30 powers of 2
37     bytes       +1 byte per additional byte
49     str         +1-4 per additional character (depending on max width)
48     tuple       +8 per additional item
64     list        +8 for each additional
224    set         5th increases to 736; 21nd, 2272; 85th, 8416; 341, 32992
240    dict        6th increases to 368; 22nd, 1184; 43rd, 2280; 86th, 4704; 171st, 9320
136    func def    does not include default args and other attrs
```

Sebagai perumpamaan, tipe data pada Python bisa diibaratkan seperti wadah air yang elastis. Di bahasa lain, ukuran wadah sudah ditentukan: ada gelas kecil, sedang, dan besar. Kalau jumlah air lebih banyak dari ukuran gelas, maka lebihan airnya tidak akan bisa tersimpan (tumpah/hilang/loss). Sedangkan di Python, wadahnya bisa otomatis membesar sesuai banyaknya air, sehingga angka berapapun tetap bisa ditampung.

Kesimpulannya, tipe data int pada Python bersifat dinamis dan fleksibel. Kita bisa menyimpan bilangan bulat kecil hingga bilangan yang

sangat besar tanpa perlu khawatir menentukan ukuran tipe datanya terlebih dahulu.

b. Float

Di Python, tidak ada tipe data *double* yang terpisah seperti di beberapa bahasa pemrograman lain (misalnya C atau Java). Sebaliknya, Python menggunakan tipe data float untuk merepresentasikan bilangan desimal, misalnya 3.14, -0.5, atau 2.0.

```
i = 42
s = "3.14159"
print(float(i))   # konversi dari int ke float
print(float(s))   # konversi dari string ke float

42.0
3.14159
```

Kode di atas menunjukkan bahwa kita bisa mengubah tipe data lain menjadi float. Misalnya, angka bulat 42 dapat diubah menjadi 42.0, atau string "3.14159" dapat diubah menjadi bilangan desimal 3.14159. Hal ini dikenal dengan istilah **casting**

Apakah float juga memerlukan *byte* yang lebih besar jika angkanya semakin besar, sebagaimana tipe data int? Untuk mengetahuinya, lakukan eksperimen sederhana dengan memodifikasi nilai dari dua data berikut:

📌 Latihan 3

```
#coba kalian modifikasi nilai dari data1 & data2 berikut, lalu amati hasilnya
data1 = 1.123456789012345678
data2 = 100000.123456789012345678

print('nilainya= ', data1)
print('tipenya = ', type(data1))
print('sizenya = ', sys.getsizeof(data1))

print('nilainya= ', data2)
print('tipenya = ', type(data2))
print('sizenya = ', sys.getsizeof(data2))

# PERCOBAAN3
```

📌 Pertanyaan Latihan 3

Jalankan dan perhatikan hasil output dari kode di atas! Apakah ukuran byte (sizenya) berubah ketika nilai float semakin besar? Mengapa hasilnya seperti itu? Bagaimana jika dibandingkan dengan ukuran byte pada tipe int sebelumnya?

c. Complex

Python memiliki tipe data complex untuk merepresentasikan bilangan kompleks. Bilangan kompleks terdiri dari bagian real dan bagian imajiner, yang masing-masing adalah bilangan float. Di Python, Anda bisa membuat bilangan kompleks menggunakan notasi a + bj, di mana a adalah bagian real dan b adalah bagian imajiner

```
z1 = 2 + 3j
z2 = 1 - 1j
print(z1 + z2)
print(z1 - z2)
print(z1 * z2)
print(z1 / z2)
```

```
(3+2j)
(1+4j)
(5+1j)
(-0.5+2.5j)
```

Tipe ini akan sangat membantu jika kita perlu untuk menyimpan sebuah rumus aljabar dalam program kita. Tidak hanya itu, bahkan kita bisa melakukan operasi aritmatika terhadapnya.

C. Sequence Type

Sequence adalah kumpulan nilai yang diakses melalui indeks dan memiliki beberapa tipe data utama, seperti array. Namun, di Python, tipe data sequence sedikit berbeda dari array pada umumnya. Data sequence lebih dinamis dibandingkan array, kita bisa menambahkan elemen sebanyak yang kita inginkan tanpa batasan ukuran seperti pada array yang terbatas karena indeks [..] . Selain itu, sequence juga dapat menyimpan elemen dengan tipe data yang berbeda. Mari kita pelajari beberapa jenis sequence berikut ini:

a. List

List dalam Python adalah salah satu tipe data sequence yang paling fleksibel dan sering digunakan. List merupakan kumpulan elemen yang berurutan/terindex (*ordered*) dan dapat diubah (*mutable*). Karena fleksibilitas nya, list sangat sering dipakai dalam pemrograman sehari-hari. Berikut beberapa karakteristik dari tipe data list:

1. *Mutable* (Dapat diubah)

   List bersifat *mutable*, artinya kita bisa menambah, menghapus, atau mengubah elemen di dalamnya setelah list dibuat. Jalankan sel kode berikut secara berurutan!

```
contoh_list1 = [1, 2]

#operasi penambahan list menggunakan method append
contoh_list1.append(3)

#cetak isi list
print(contoh_list1)
```

```
[1, 2, 3]
```

```
#operasi penambahan list dengan list
contoh_list2 = contoh_list1 + [4, 5, 6, 7]
print(contoh_list2)
```

```
[1, 2, 3, 4, 5, 6, 7]
```

2. *Ordered* (Berurutan)

Maksud dari berurutan disini adalah bahwa untuk mengakses data dalam list akan dimulai berurutan sesuai dengan urutan *index*nya.

Sebelum menjalankan sel kode dibawah ini, pastikan 2 sel kode sebelumnya sudah di run!

```
#akses data berdasarkan urutan index dari depan
print(contoh_list2[0])
print(contoh_list2[1])

#akses data berdasarkan urutan index dari belakang
print(contoh_list2[-1])
print(contoh_list2[-2])
```

```
1
2
7
6
```

3. Heterogeneous (Banyak Tipe Data)

List tidak kaku yang berarti List mampu menyimpan value dengan bermacam-macam tipe data seperti contoh di bawah ini :

```
contoh_list3 = [1, "hello", 3.14, True, 'a']

print(contoh_list3)
```

```
[1, 'hello', 3.14, True, 'a']
```

4. *Allow Duplicates*

List memperbolehkan data yang memiliki value sama karena List dipengaruhi oleh index.

```
contoh_list4 = ["fitra", "adit", "yusuf", "fitra", "maylani"] # fitra ada 2
print(contoh_list4)
```

```
['fitra', 'adit', 'yusuf', 'fitra', 'maylani']
```

Masih ada beberapa ciri yang dimiliki oleh List seperti indexing, *slicing*, *dynamic size* dsb. List juga memiliki method bawaan dari python yang memudahkan kita dalam mengoperasikan data List, silahkan mengunjungi tautan *berikut* untuk lebih detailnya.

b. Tuple

Tuple adalah urutan objek dalam Python yang bersifat *immutable* (tidak dapat diubah). Artinya, setelah sebuah tuple dibuat, isi di dalamnya tidak bisa diubah, ditambah, atau dihapus. Secara bentuk, tuple mirip dengan list, hanya saja tuple menggunakan tanda kurung ( ), sedangkan list menggunakan tanda kurung siku [ ].

1. *Immutable* (Tidak dapat diubah)

   Ketika sebuah tuple sudah berhasil dibuat, maka isinya tidak bisa dimodifikasi. Jika kita mencoba mengubah elemennya, Python akan memberikan error.

📌 Latihan 4

Jalankan dan perhatikan hasil output dari kode berikut! Jelaskan jawabanmu pada laporan!

```
contoh_tuple1 = (1, 2, 3)
contoh_tuple1[0] = 10 # Ini akan error

# PERCOBAAN4a
```

```
#apakah ini juga akan error? coba jelaskan!
contoh_tuple1 = 1, 2, 3
print(contoh_tuple1)
print(type(contoh_tuple1))

# PERCOBAAN4b
```

2. Heterogeneous (Banyak Tipe Data)

   Tuple dapat berisi berbagai macam tipe data sekaligus, mirip seperti list.

```
contoh_tuple2 = ("abc", 34, True, 40, "Male")
print(contoh_tuple2)
```

```
('abc', 34, True, 40, 'Male')
```

3. *Ordered* (Berurutan)

Tuple memiliki urutan tetap. Artinya, kita bisa mengakses elemen-elemen di dalamnya menggunakan indeks, baik dari depan (indeks positif) maupun dari belakang (indeks negatif).

```
contoh_tuple3 = (10, 20, 30, 40, 50)

print(contoh_tuple3[0])   # Elemen pertama
print(contoh_tuple3[1])   # Elemen kedua
print(contoh_tuple3[-1])  # Elemen terakhir
print(contoh_tuple3[-5])  # Elemen pertama juga
```

```
10
20
50
10
```

Untuk mengetahui lebih dalam mengenai tuple, silahkan membuka tautan *berikut.*

c. Range

Range dalam Python adalah tipe data yang digunakan untuk menghasilkan urutan angka. range biasanya digunakan dalam perulangan (*loop*) untuk mengulang sejumlah langkah tertentu. Range menghasilkan urutan angka yang teratur sesuai dengan parameter yang diberikan, namun tidak menyimpan semua angka dalam memori, melainkan menghasilkan angka satu per satu saat dibutuhkan. Ciri - ciri range dapat kita simak pada contoh berikut:

1. *Immutable*

Sama seperti tuple, setelah didefinisikan, objek range tidak dapat diubah. Kita tidak bisa menambah, menghapus, atau mengubah elemen-elemen dalam range.

2. *Indexing* dan *Slicing*

Kita bisa mengakses elemen-elemen tertentu dalam range menggunakan *indeks* dan *slicing,* mirip dengan list. Coba perhatikan dan jalankan kode berikut:

```
r = range (10)
print(r[2])
print(r[2:5])
```

```
2
range(2, 5)
```

3. *Iteration*

Objek range dapat diiterasi menggunakan loop for, yang sangat berguna untuk mengulangi blok kode dengan jumlah langkah yang ditentukan.

```
# range dengan 1 argumen
for i in range(3):
  print(i)
```

```
0
1
2
```

```
# range dengan 2 argumen : argument pertama=start, argumen kedua=end
for i in range(2, 6): # untuk mencetak angka 2 sampai <6
    print(i)
```

```
2
3
4
5
```

```
# range dengan 3 argumen (start, end, increment by)
for i in range(1, 10, 3):
    print(i)
```

```
1
4
7
```

📌 Pertanyaan Latihan 5

Buat sebuah daftar angka ganjil/genap (sesuai NIM) menggunakan range (tanpa if else) dengan batas akhir/end diambil dari 2 digit akhir NIM anda.

Simpan dan tampilkan hasilnya dalam bentuk list (untuk NIM ganjil) dan tuple (untuk NIM genap)! Jika NIM anda <9, silahkan dikali 10.

```
# contoh output untuk nim akhir 001 (ganjil <9 maka end range = 1*10)
# PERCOBAAN5
```

```
[1, 3, 5, 7, 9]
```

D. Mapping Type

a. Dictionary

Dalam Python, kamus atau *dictionary* adalah tipe data yang digunakan untuk menyimpan nilai-nilai dalam format pasangan kunci-nilai (key-value). Dictionary sangat berguna ketika kita ingin mengasosiasikan kunci unik dengan nilai tertentu. Berikut merupakan ciri dari dictionary :

- Kunci tidak boleh duplikat
- *Changeable* (dapat diubah)
- Pengaksesan menggunakan Key

Bagaimana cara membuat *dictionary*? Kode berikut adalah contoh deklarasi tipe data dict:

```
thisdict = {
    "Name" : "Budhe",
    "NIM" : "12345678910",
    "Semester" : 15
}

print(thisdict)
```

```
{'Name': 'Budhe', 'NIM': '12345678910', 'Semester': 15}
```

Beberapa contoh dengan data *dictionary* bisa kalian lihat pada colab

E. Boolean Type

Menyatakan benar *True* yang bernilai 1, atau salah *False* yang bernilai 0

```
a = 5 > 6 # false
b = 6 > 5
print(type(a))
print(a, b)
```

```
<class 'bool'>
False True
```

3.1.3. Casting Tipe data dalam Python

Casting berarti merubah suatu tipe data menjadi tipe lain dengan cara memanggil fungsi konversi bawaan Python, lalu mengirimkan data sebagai parameternya. Dengan casting, kita bisa mengubah int ke string, float ke int, list ke tuple, dan sebagainya. Coba kalian jalankan sel berikut secara berurutan dan amati hasilnya!

```
# Nilai awal
x = 1
y = 10.657
z = [10, 9, 8, 7]

print("=== Sebelum Casting ===")
print("x:", x, "| Tipe:", type(x))
print("y:", y, "| Tipe:", type(y))
print("z:", z, "| Tipe:", type(z))
```

```
=== Sebelum Casting ===
x: 1 | Tipe: <class 'int'>
y: 10.657 | Tipe: <class 'float'>
z: [10, 9, 8, 7] | Tipe: <class 'list'>
```

```
# Melakukan casting
x_str = str(x)       # int ke string
y_int = int(y)       # float ke int (pecahan dibuang)
z_tuple = tuple(z)   # list ke tuple

print("=== Setelah Casting ===")
print("x_str:", x_str, "| Tipe:", type(x_str))
print("y_int:", y_int, "| Tipe:", type(y_int))
print("z_tuple:", z_tuple, "| Tipe:", type(z_tuple))
```

```
=== Setelah Casting ===
x_str: 1 | Tipe: <class 'str'>
y_int: 10 | Tipe: <class 'int'>
z_tuple: (10, 9, 8, 7) | Tipe: <class 'tuple'>
```

📌 Pertanyaan Latihan 6

Lakukan casting NIM kalian dengan tipe data string menjadi int dan float! Misalnya ubah string "123" menjadi integer dan float.

Apa yang terjadi jika kita mencoba mengubah string "hello" menjadi integer atau float? dan bagaimana hasilnya dan mengapa

3.2. Indentasi

Indentasi dalam konteks pemrograman adalah pengaturan spasi di awal baris kode yang digunakan untuk menandai struktur dan hierarki blok kode (seperti awal dari sebuah paragraf). Dalam beberapa bahasa pemrograman, indentasi hanya digunakan untuk meningkatkan keterbacaan kode.

Pada python, indentasi memiliki peran yang sangat penting karena digunakan untuk mendefinisikan blok kode, seperti di dalam fungsi, loop, kondisi, dan lain-lain. Tidak seperti banyak bahasa pemrograman lainnya yang menggunakan tanda kurung kurawal { } untuk menandai blok kode, Python menggunakan indentasi untuk tujuan ini.

Intinya, di dalam python kita tidak bisa asal menggunakan ‘tab’ atau spasi untuk mengatur baris agar terlihat rapi. Indentasi atau spasi dalam python digunakan untuk menentukan scope dan jangkauan code.

Kalian bingung? Coba lakukan eksperimen berikut:

```
# indentasi yang salah
x = 10

if x > 5:
print("x lebih besar dari 5")
```

```
  File "/tmp/ipython-input-2560958080.py", line 4
    print("x lebih besar dari 5")
    ^
IndentationError: expected an indented block after 'if' statement on line 3
```

```
# indentasi yang benar
x = 10

if x > 5:
    print("x lebih besar dari 5")
```

```
x lebih besar dari 5
```

Biasanya, indentasi ini diperlukan saat ditemukan adanya tanda titik dua (:) dalam program. Hal ini merupakan pengganti tanda kurung kurawal { } pada bahasa program umumnya (seperti pada C/Java). Karena di python, kurung kurawal { } merupakan identitas dari sebuah data dictionary, bukan blok kode.

3.3. Percabangan Dalam Python

Sama seperti bahasa pemrograman lainnya, python juga memiliki percabangan seperti if, else, elseif. Namun, python tidak memiliki percabangan *switch case*. Penerapan percabangan ini mirip dengan bahasa pemrograman yang telah kalian pelajari. Untuk lebih memahaminya, kalian bisa menjalankan baris program dibawah ini

```
# if condition

nilai = 80

if nilai >= 75:
    print("Selamat! Kamu lulus.")
```

```
Selamat! Kamu lulus.
```

Dan contoh yang lain bisa kalian lihat di *colab*

3.4. Perulangan Dalam Python

Pada python, kita bisa melakukan perulangan dengan beberapa cara diantaranya:

3.4.1. Perulangan For

Perulangan for pada python adalah perintah yang digunakan untuk melakukan iterasi dari sebuah nilai sequence atau data koleksi seperti *List, Tuple, String, range* dan lainnya. For pada python memiliki perilaku yang berbeda dengan for pada kebanyakan bahasa pemrograman yang lain, karena pada python ia sangat berkaitan dengan data sequence atau data kolektif. Mungkin kalau dibandingkan dengan bahasa lain, for pada python lebih dikenal sebagai foreach. Perhatikan contoh berikut:

```
# for loop pada sequence type

aktivitasKu = ["Bangun tidur", "Sholat", "Kuliah", "Jajan", "Turu"]

# for loop pada list (sequence type)
for aktivitas in aktivitasKu:
  print(aktivitas)
```

```
Bangun tidur
Sholat
Kuliah
Jajan
Turu
```

Buka *colab* untuk mengeksplor contoh-contoh implementasi lainnya untuk for-loop.

3.4.2. Perulangan While

Pada python, *while loop* mirip dengan *while loop* yang ada pada bahasa pemrograman lainnya. Namun, tidak ada *do-while* dalam python. Tidak seperti for loop yang memiliki aturan khusus. *While loop* pada python cukup mudah dipahami. Kalian bisa mencoba langsung *while loop* pada python dengan menjalankan program dibawah ini.

```
i = 1

while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

3.4.3. Fungsi Rekursif

Fungsi rekursif adalah fungsi yang melakukan perulangan dengan mengacu pada dirinya sendiri. Dalam paradigma pemrograman fungsional, rekursi adalah cara penting untuk melakukan iterasi data, menggantikan penggunaan loop konvensional.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

print(factorial(5))
```

```
120
```

Rekursi dalam paradigma fungsional mempromosikan penggunaan fungsi murni dan menghindari perubahan keadaan (state) yang tidak perlu. Fungsi murni adalah fungsi yang hasilnya hanya bergantung pada inputnya dan tidak mengubah variabel di luar fungsi tersebut. Terkait hal ini akan kita bahas lebih detail pada modul selanjutnya.

3.5. Function Dalam Python

Jika pada paradigma pemrograman sebelumnya (OOP-Java) kalian lebih familiar dengan istilah method, maka pada pemrograman fungsional dengan python kali ini kita akan banyak menggunakan function (ee, jangan salah, di python juga ada method kok, cek cek *disini* untuk tau perbedaan keduanya).

Python sendiri memiliki banyak *built-in function*, diantaranya yang sering kita gunakan, antara lain *print()* untuk mencetak objek, *len()* berfungsi mengembalikan (*return*) panjang suatu objek, *type()* berfungsi mengembalikan (*return*) tipe data dari suatu variabel, dan masih banyak lagi yang lainnya. Sedangkan untuk membuat suatu *function*, perlu didefinisikan dengan keyword "def"--stand for *define*.

Berikut adalah salah dua contoh deklarasi serta eksekusi fungsi:

```
#cara mendefinisikan fungsi menggunakan keyword 'def'
def firstExample():
    print("Hello World")

#cara memanggil/mengeksekusi fungsi
firstExample()
```

```
#fungsi dengan parameter
def secondExample(name, praktikum):
    print("Rama dan " + name + " sedang praktikum " + praktikum)

secondExample("Wempy", "P. Fungsional")
```

Fungsi-fungsi diatas adalah deklarasi fungsi standar pada python. Meski kita sudah membangun program kita dalam satuan fungsi-fungsi, bukan berarti kita sudah mengimplementasikan paradigma fungsional. Seperti halnya fungsi-fungsi diatas, fungsi-fungsi tersebut belum memenuhi paradigma fungsional.

Seperti apakah fungsi dalam paradigma fungsional itu? Mari kita cek kembali materi topik 1 tentang definisi fungsi dalam paradigma fungsional:

Program dalam paradigma fungsional dapat dianggap sebagai kumpulan fungsi yang menerima input dan menghasilkan output tanpa mempengaruhi keadaan (*stateless*) dan tidak memiliki efek samping apapun (*pure function*).

Kita bisa memperbaiki dan menyusun kode dengan pendekatan lebih fungsional untuk mencapai manfaat yang ditawarkan oleh paradigma fungsional. Bagaimana caranya? Pastikan fungsi yang kita bangun memenuhi standar berikut:

- *Input*: tambahkan parameter sebagai input data-data yang dibutuhkan oleh fungsi.
- *Output*: tambahkan return value sebagai hasil keluaran dari fungsi.
- *Pure*: tidak memiliki efek samping berarti fungsi tidak mengakses atau mengubah variabel global atau data-data lain di luar fungsi.

Dengan demikian, kita dapat menghasilkan program yang lebih elegan, mudah dipahami, dan mudah dikelola. Mari kita perbaiki dan susun ulang kode fungsi sebelumnya dengan pendekatan fungsional seperti berikut:

```
# deklarasi func menggunakan parameter (reusability)
def secondExample(name, praktikum):
    # Menambahkan return sebagai output
    return "Rama dan " + name + " sedang praktikum " + praktikum

# Assign function ke dalam variabel: untuk menyimpan hasil kembalian/return dari fungsi
callback = secondExample("Zaky", "Jarkom")

print(callback)
```

```
Rama dan Zaky sedang praktikum Jarkom
```

```
# Deklarasi fungsi sebagai ekspresi
def multiply(a,b):
    return a * b

result = multiply(10,22)
print(result)
```

```
220
```

CODELAB

CODELAB dikerjakan dalam bentuk laporan (disertai penjelasan) dengan merujuk pada *template jawaban*

CODELAB 1

Lengkapi program dibawah ini untuk memisahkan data harga yang diinput oleh user menjadi dua kategori:

- Harga murah: kurang dari 50.000, disimpan ke dalam list
- Harga mahal: sama dengan atau lebih dari 50.000 dan disimpan ke dalam tuple
- Input dipisahkan dengan spasi, dan kamu harus melakukan casting sesuai tipe data yang dibutuhkan.

Silahkan buka *collab* untuk mendapatkan potongan code untuk diselesaikan.

CODELAB 2

Sebuah program menghitung total belanja berdasarkan data barang dan harga.

1. Buatlah sebuah dictionary yang menyimpan data nama barang dan harganya.
2. Lengkapi fungsi untuk menghitung total belanja berdasarkan isi dictionary.
3. Tampilkan total belanja.
4. Lengkapi dan perbaiki kode berikut agar dapat berjalan dengan benar.

Silahkan buka *colab* untuk mendapatkan potongan code untuk diselesaikan.

TUGAS

Silakan mengisi tema yang akan kalian gunakan disini *link pengisian tema* GUNAKAN AKUN UMM.

Buatlah sebuah program yang memiliki sistem login dan register serta fitur pengelolaan data. Tema program bebas dan harus sesuai dengan yang telah diisikan pada link yang telah disediakan.

Beberapa ketentuan program diantaranya:

1. Data akun disimpan dalam struktur dictionary, dengan ID sebagai key dan password sebagai value.
2. Data profil setiap akun pengguna disimpan dalam bentuk nested dictionary dengan akun sebagai key dan profil sebagai value nya (isian profil bebas--nama,alamat,hp,dll).
3. Daftar menu aplikasi harus disimpan dalam bentuk tuple untuk diakses kemudian (hint: pakai slicing).

4. Fitur Register: pengguna dapat membuat akun dengan ID (bisa berupa username, NIM, dsb) dan password (bisa kalian tentukan sendiri).
5. Fitur Login: hanya pengguna yang sudah terdaftar yang dapat melakukan login.
6. CRUD (Create, Read, Update, Delete) data hanya bisa dilakukan setelah login.
7. Berikan validasi input menggunakan while atau for agar tidak error jika user salah input.
8. Data Tambahan: siapkan 1 data tambahan sesuai tema yang dipilih untuk diolah dalam program. dengan ketentuan:
    a. Data harus bertipe sequence (Dictionary/List/Tuple)
    b. Jelaskan alasan kalian memilih tipe sequence tertentu untuk data yang kalian pilih
    c. Data yang dikelola terbatas hanya pada data akun yang sedang login saja.

Note: Data yang dikelola harus terkait langsung dengan akun pengguna yang sedang login. Pastikan setiap key pada dictionary sesuai dan hanya dapat diakses oleh akun pengguna yang bersangkutan (misalnya: akun user001 hanya dapat mengakses data yang memiliki key user001).
Contoh output bisa dilihat di *colab*.

RUBRIK PENILAIAN

Komponen Penilaian	Bobot (%)
Codelab	**15**
Tugas	
Kode / Kelengkapan Fitur	30
Originalitas kode	10
Pemahaman	**45**
Total Akhir	**100**