VERSION 2.1

30 Agustus 2024

[PEMROGRAMAN FUNGSIONAL]

Modul 1 – Functional Programming using Python

DISUSUN OLEH:

ALVIYA LAELA LESTARI

RAFLI KHARISMA AKBAR

DIAUDIT OLEH:

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

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[PEMROGRAMAN FUNGSIONAL]

PERSIAPAN MATERI

Praktikan diharapkan paham konsep fungsi serta dasar-dasar pemrograman (tipe data, variabel, percabangan, dan perulangan)

TUJUAN PRAKTIKUM

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
3. Mampu mengelola fitur-fitur yang ada dalam paradigma pemrograman fungsional menggunakan Python

PERSIAPAN SOFTWARE/APLIKASI

- Laptop/Komputer
- Windows/Linux/Mac OS
- Pycharm/Google Collab /Jupyter Notebook
- Source Code Google Collab Modul 1

1. Functional Programming Paradigm

Pada semester sebelumnya, kalian sudah mempelajari terkait Pemrograman dasar, Pemrograman berbasis objek (OOP), Pemrograman lanjut, dan Struktur Data. Pada pemrograman OOP, kalian berfokus pada objek sebuah *class* dalam mengembangkan sebuah program. Berbeda dengan pemrograman fungsional, seperti namanya, paradigma program ini membahas lebih dalam terkait fungsi dari sebuah program dengan berbagai fitur atau karakteristik khusus didalamnya.

Paradigma Pemrograman Fungsional (*Functional Programming Paradigm)* adalah suatu pendekatan dalam pemrograman yang **berfokus pada penggunaan fungsi** (*Function)* sebagai komponen utama dalam menulis program. Secara sederhana, dalam pemrograman fungsional, kita menulis fungsi untuk melakukan komputasi dan kemudian sebagai pengguna, kita memanggil fungsi tersebut agar berguna bagi kita.

1.1 Fitur Pemrograman Fungsional

Sebuah program dalam pemrograman fungsional dapat dianggap sebagai **kumpulan fungsi yang menerima input dan menghasilkan output** tanpa memiliki keadaan (*Stateless*). Apa maksudnya? jadi maksud dari *stateless* disini adalah untuk input yang sama akan selalu memberikan hasil yang sama pula, tanpa mempengaruhi variabel atau data di luar fungsi tersebut. Istilah ini disebut **Pure Function** (fungsi murni) yang akan kita pelajari lebih detail pada modul berikutnya.

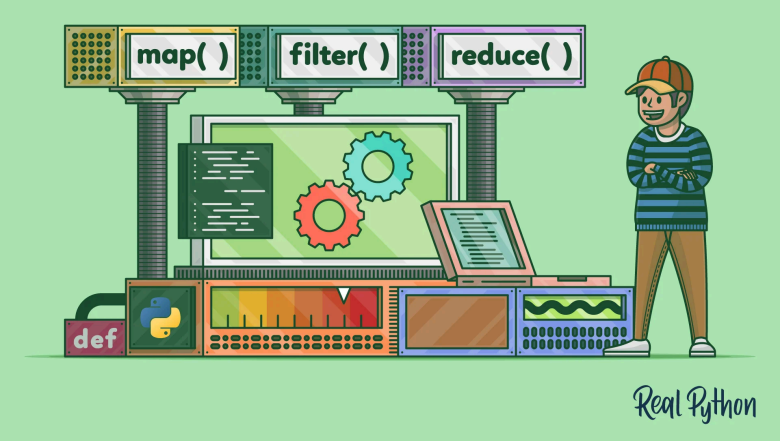
Selain *Pure Function,* terdapat beberapa fitur kunci pada paradigma fungsional seperti penggunaan ***Recursion*** (Rekursi) sebagai pengganti looping program, ***High-Order-Functions, Lambda Functions, Map, Filter, dll.*** Pada praktikum Pemrograman Fungsional ini kita akan menggunakan bahasa ***Python*** walaupun konsep Pemrograman Fungsional sendiri bisa diimplementasikan di bahasa pemrograman lain seperti *Javascript, Haskell, Ruby dll.*

1.2 Mengenal Lebih Dalam Functional Programming

Pemrograman fungsional adalah gaya pemrograman **deklaratif** yang berfokus pada *"apa yang harus diselesaikan"* daripada *"bagaimana cara menyelesaikannya."* Dalam gaya ini, kita menggunakan **ekspresi** dan fungsi murni (**pure**) yang selalu

menghasilkan output yang sama untuk input yang sama, tanpa mengubah keadaan program (tidak ada efek samping).

Ini berbeda dengan pemrograman imperatif yang memberikan instruksi langkah demi langkah tentang bagaimana menyelesaikan suatu tugas, mirip dengan memberikan resep memasak yang mendetail. Sebaliknya, pemrograman deklaratif seperti memesan kue di kafe, di mana kita hanya menyebutkan jenis kue yang kita inginkan tanpa menjelaskan cara membuatnya. Pemrograman deklaratif cenderung lebih mudah dibaca dan dipelihara karena beroperasi pada tingkat abstraksi yang lebih tinggi, sehingga mengurangi kompleksitas dan detail implementasi.



Animasi Functional Programming (Python)

2. Tools

Beberapa tools yang akan kita gunakan dalam praktikum Pemrograman Fungsional kali ini akan dijelaskan dalam poin poin berikut :

2.1 Python

Python adalah bahasa pemrograman yang fleksibel dengan pendekatan fungsional dan semantik dinamis. Python terkenal dengan sintaksisnya yang mudah dipelajari, serta memiliki pustaka standar yang besar dan beragam. Fitur-fitur menarik seperti pengelolaan data dan memori otomatis, serta pembaruan modul yang konsisten, menjadikannya pilihan favorit bagi banyak *developer*. Python dapat digunakan untuk berbagai aplikasi, mulai dari pengembangan web, analisis data, kecerdasan buatan, hingga otomasi sistem. Bahasa ini dapat berjalan di berbagai sistem operasi, termasuk Linux, Windows, Mac OS, dan Android. Jangan lupa untuk mendownload *Python* jika

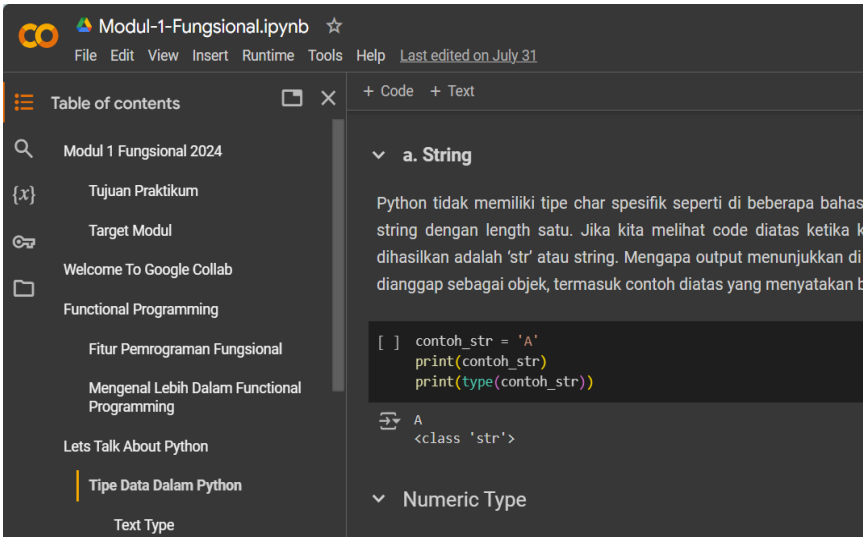
ingin menjalankannya di lokal komputer masing-masing, jika terjadi kendala bisa segera hubungi asisten ya!.

2.2 Pycharm (*Optional*)

PyCharm adalah Integrated Development Environment (IDE) yang dikembangkan oleh JetBrains khusus untuk pengembangan Python. PyCharm menyediakan berbagai fitur yang dirancang untuk meningkatkan produktivitas pengembang Python, termasuk pelengkapan otomatis kode, debugging, pengujian unit, integrasi dengan sistem kontrol versi, dan dukungan untuk kerangka kerja web seperti Django dan Flask. Jadi bagi kalian yang sudah terbiasa dengan lingkungan seperti Intellij bisa menggunakan Pycharm. Untuk instalasi Pycharm sendiri bisa menggunakan link panduan disini.

2.3 Google Collab

Collaboratory, atau disingkat "Collab", adalah produk dari Google Research. Collab memungkinkan siapa saja untuk menulis dan mengeksekusi kode python arbitrer melalui browser, dan sangat cocok untuk pembelajaran mesin, analisis data, dan pendidikan. Secara lebih teknis, Collab adalah layanan notebook Jupyter yang dihosting yang tidak memerlukan penyiapan untuk digunakan, sekaligus memberikan akses gratis ke sumber daya komputasi termasuk GPU. Oleh karena itu Collab sangat direkomendasikan untuk kalian yang memiliki komputer / laptop dengan spesifikasi yang rendah. Untuk mengaksesnya pun cukup mudah seperti menggunakan produk Google pada umumnya, kalian hanya perlu login dan membuat notebook baru. Link collab bisa diakses disini



Contoh tampilan *Google Collab*

2.4 Visual Studio Code

Untuk yang satu ini mungkin kalian sudah tidak asing, text editor untuk semua programmer di seluruh dunia. Sedikit catatan, untuk bisa menjalankan python di Visual Studio Code pastikan kalian menginstal package, extensi, ataupun library yang mendukung karena tidak seperti IDE dan Google Collab, VScode membutuhkan konfigurasi manual agar bisa dijalankan. (**note: jika sudah menginstal namun tidak bisa dijalankan coba untuk menambahkan directory bin ke dalam environment variables di PC kalian masing-masing*)

3. Let's Talk About Python

Python sendiri merupakan bahasa pemrograman yang termasuk ke dalam kategori high level programming language. Pada dasarnya Python didesain dengan fokus pada keterbacaan kode dan sintaks yang mudah dipahami, sehingga memudahkan pengembang dalam menulis dan memelihara program. Pembahasan dan Pengenalan tentang python akan dijelaskan poin-poin sebagai berikut :

3.1 Tipe Data dan Variabel Dalam Python

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

3.1.1 Apa itu Variable dalam Python?

Anggap saja variabel adalah sebuah keranjang, tempat di mana kita bisa memasukkan sesuatu di dalamnya, yaitu data. Di python, kita bisa memasukkan tipe data apa saja ke dalam keranjang (variabel) tanpa harus mendefinisikan tipe datanya terlebih dahulu (hal ini berbeda dengan beberapa bahasa pemrograman lain yang mengharuskan kita mendefinisikan tipe data terlebih dahulu). Sebuah variabel dapat berganti tipe data sedinamis mungkin. Perhatikan dan jalankan baris kode di bawah agar kalian lebih memahami penggunaan variabel pada python!

```python
var = 212
print(var)
print(type(var))

# redefine variable

var = "Variable baru coy"
print(var)
print(type(var))
```

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 memiliki jenis. 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, karena tipe data yang spesifik menentukan nilai yang dapat kita tetapkan dan operasi yang bisa kita lakukan. Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan 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 setiap 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.

```
1 contoh_str = 'A'
2 print(contoh_str)
3 print(type(contoh_str))
4
5 # Output:   A
6 # Output:   <class 'str'>x
```

Jika kita melihat *code* diatas, ketika kita mencoba untuk membungkus menggunakan type maka output yang dihasilkan adalah 'str' atau string. Mengapa output menunjukkan *<class>* di awal ketika ingin mengetahui tipe data? Karena dalam python semua data dianggap sebagai objek, termasuk contoh diatas yang menyatakan bahwa A merupakan objek dari kelas 'str'.

b. Concat String

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

```
1 kata = "I love you"
2 kata2 = ", and you love me back 💔"
3
4 concat = kata + kata2
5 # contoh concat
6 print(concat)
```

B. Numeric Type

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

a. Int

```
1 contoh_int = 10
2 print(type(contoh_int))
3
4 # Output : <class 'int' >
```

Seperti pada bahasa pemrograman pada umumnya tipe data **int** merepresentasikan bilangan bulat (**integer**). Beberapa bahasa pemrograman seperti C/C++ atau java memiliki tipe data dengan ukuran yang berbeda seperti *short* untuk bilangan bulat dengan ukuran lebih kecil dari int, dan long untuk bilangan bulat dengan ukuran lebih besar dari int. Namun, pada Python, ukuran tipe data int akan disesuaikan secara dinamis berdasarkan nilai bilangan yang kita berikan. Kita bisa menggunakan perintah berikut untuk mengeceknya:

```
1 import sys
2 print(sys.getsizeof(1))
3 print(sys.getsizeof(12345234525232523))
4
5 # Output : 28
6 # Output : 32
```

Semakin besar value dari sebuah **int** maka jumlah byte juga semakin besar, sesuai dengan kebutuhan memori dari value tersebut. 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
136    func def    does not include default args and other attrs
```

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.

```python
i = 42
s = "3.14159"
print(float(i))  # Output: 42.0
print(float(s))  # Output: 3.14159
```

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.

```python
z1 = 2 + 3j
z2 = 1 - 1j
print(z1 + z2)  # Output: (3+2j)
print(z1 - z2)  # Output: (1+4j)
print(z1 * z2)  # Output: (5+1j)
print(z1 / z2)  # Output: (-0.5+2.5j)
```

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. 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. Ini adalah kumpulan elemen yang berurutan (ter-*index*) dan dapat diubah (*mutable*). Berikut adalah beberapa poin penting tentang list dalam Python:

1) *Mutable* (Dapat Diubah)

Kita tetap bisa melakukan perubahan pada data dalam list karena sifat dari list yang dapat diubah/*mutable*.

```
1 cth_list = [1 ,2 ,3]
2 cth_list[0] = 10 # Mengubah elemen
3 cth_list.append(4) # Menambah elemen
4
5 print(cth_list) # Output [10, 2, 3, 4]
```

2) *Ordered* (Berurutan)

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

```
1 cth_list = [1, 2, 3]
2 print(cth_list[0])  # Output: 1
3 print(cth_list[1])  # Output: 2
```

3) Heterogeneous (Banyak Tipe Data)

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

```
1 cth_list = [1, 'hello', 3.14, True]
```

4) *Allow Duplicates* (Duplikasi)

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

```
1 cth_list = ["apple", "banana", "cherry",
  "apple", "cherry"]
2 print(cth_list)
```

Masih ada beberapa ciri yang dimiliki oleh List seperti indexing dan 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 tidak dapat diubah (immutable). Tuple serupa dengan list, namun perbedaan utamanya adalah elemen dalam tuple tidak dapat diubah setelah didefinisikan. Tuple menggunakan tanda kurung '()', sedangkan list menggunakan tanda kurung siku '[]'. Membuat tuple sangat mudah; kita hanya perlu menuliskan nilai-nilai yang dipisahkan oleh koma. Di bawah ini pengelompokkan ciri dari sebuah tuple :

1) Immutable (Tidak dapat diubah)

Ketika tuple sudah berhasil dibuat, kita tidak bisa mengubah, menambah atau menghapus elemen-elemen di dalamnya.

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

2) Heterogeneous (Banyak Tipe Data)

Tuple mengizinkan untuk memiliki data yang berbeda tipe seperti List.

```
1 cth_tuple = ("abc", 34, True, 40, "Male")
2 print(cth_tuple)
```

3) *Ordered* (Berurutan)

Tuple bersifat ordered seperti List yaitu untuk mengaksesnya kita bisa menggunakan index masing-masing data.

```
1 cth_tuple = (10, 20, 30, 40, 50)
2 print(my_tuple[0])  # Output: 10
3 print(my_tuple[1])  # Output: 20
4 print(my_tuple[-1])  # Output: 50
5 print(my_tuple[-2])  # Output: 40
```

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 dicontoh 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:

```python
r = range(10)
print(r[2])  # Output: 2
print(r[2:5])  # Output: range(2, 5)
```

3) Iteration

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

```python
for i in range(5):
    print(i) # Output: 0 1 2 3 4
```

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 perlu mengasosiasikan kunci unik dengan nilai tertentu. Berikut merupakan ciri dari dictionary

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

Berikut adalah contoh deklarasi tipe data dict:

```python
thisdict = {
  "Name": "Widodo",
  "NIM" : "202110370311402",
  "Semester" : 7
}

```

dan beberapa operasi yang bisa dilakukan dengan tipe data dictionary:

```python
# contoh changeable and nested dictionary
thisdict = {
  "Name": "Widodo",
  "NIM" : "202110370311402",
  "Semester" : 7,
  "Teman" : {
      "Nama" : "Sutadi",
      "Umur" : 55
  },
  "Identitas Ortu": {
      "Nama": "Rian Firjatullah",
      "Umur": 60
  }
}

# casting objek ke dalam variable
var = thisdict["Identitas Ortu"]["Nama"]

print("Temanku adalah", thisdict["Teman"]["Nama"])
print("Nama orang tua adalah", var)
print(thisdict)
```

**note* : buka collab untuk output dan contoh lebih dalam

E. Boolean Type

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

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

3.1.3 Casting Tipe Data dalam Python

Kadangkala ada saatnya kita perlu menentukan tipe pada variabel. Ini dapat dilakukan dengan cara casting. Python sesungguhnya adalah bahasa berorientasi objek, dan karenanya menggunakan kelas untuk menentukan tipe data, termasuk tipe primitifnya. Oleh karena itu, casting dalam python dilakukan menggunakan fungsi konstruktor. Mari kita simak beberapa contoh dari casting di bawah ini :

```python
x = 1
y = 10.657
z = [10,9,8,7]

print(type(x)) # Output : <class 'int'>

print(type(y), " : " , y) # Output : <class 'float'> : 10.657

print(type(z)) # Output : <class 'list'>

# Casting variable x ke dalam str
x = str(x)

# Casting variable y ke dalam int
y = int(y)

# Casting variable z ke dalam tuple
z = tuple(z)

print(type(x)) # Output : <class 'str'>

print(type(y), " : ", y) # Output : <class 'int'> : 10

print(type(z)) # Output : <class 'tuple'>

```

Kalian bisa melihat lebih detail dan mencobanya secara langsung di dalam google collab modul 1.

3.2 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 :

```python
# if condition

amIHappy = "Yes"

if amIHappy == "Yes":
  print("Yeepe i hope you always happy!")
```

```python
# if else condition

areYouHappy = "No"

if areYouHappy == "Yes" or areYouHappy == "yes":
  print("Yeepe i hope you always happy!")
else:
  print("Fa inna ma'al usri yusro")
```

```python
# if elseif else condition

areYouOkay = "IDK"

if areYouOkay == "Yes":
  print("Yeepe i hope you always happy!")
elif areYouOkay == "IDK":
  print("You have something say to me?")
else:
  print("Fa inna ma'al usri yusro")
```

Jalankan code di collab untuk melihat output gengs!

3.3 Perulangan Dalam Python

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

3.3.1 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:

```python
# for loop pada sequence type

friendList = ["Sutadi", "Andri", "Boy", "Bagus", "Widodo"]
for x in friendList:
  print(x)

```

```
# for loop pada string

for x in "Zaid":
  print(x)

# break keyword

for x in friendList:
  print(x)
  if x == "Bagus":
    print("Cut off dulu yaa")
    break

```

Buka collab untuk mengeksplor contoh-contoh implementasi lainnya untuk for-loop. Selain dengan data sequence tersebut diatas, kita juga bisa menggunakan for dengan fungsi range().

3.3.2 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
```

3.3.3 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.

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


print(factorial(5))
```

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 detil pada modul selanjutnya.

3.4 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 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 untuk 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”. Untuk lebih memahami mengenai function pada python, kalian dapat menjalankan program dibawah ini:

```python
def firstExample():
    print("Hello it's me")

firstExample()

# Output : Hello it's me

def secondExample(name, hobbies):
    print("Hi, iam " + name + " and my hobby is " + hobbies)

```

```
11 secondExample("Widodo", "Fishing")
12
13 # Output : Hi, iam Widodo and my hobby is
   Fishing
```

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. Kok bisa???

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

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

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:

1. Input: tambahkan parameter sebagai input data-data yang dibutuhkan oleh fungsi.
2. Output: tambahkan return value sebagai hasil keluaran dari fungsi.
3. 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 sebagai berikut:

```
 1 # deklarasi fungsi menggunakan parameter (reusability)
 2 def secondExample(name, hobbies):
 3     # Menambahkan return sebagai output
 4     return ("Hi, iam " + name + " and my hobby is " + hobbies)
 5
 6 # Assign function ke dalam variabel
 7 callback = secondExample("Widodo", "Fishing")
 8
 9
10 print(callback)
11
12 # Output : Hi, iam Widodoand my hobby is Fishing
```

Kalian bisa melakukan hal yang sama untuk fungsi lainnya.

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

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

3.5 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. Namun, dalam 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 justru dalam python indentasi atau spasi digunakan untuk menentukan **scope dan jangkauan code**. Jadi dalam python kitak tidak menggunakan kurung kurawal dalam membatasi scope. Lebih jauh akan dibahas di collab.

CODELAB

Codelab dikerjakan di Google Collab masing-masing!

CODELAB 1

Dengan data yang telah disediakan, pisahkan antara nilai int, float, dan string dengan ketentuan sebagai berikut:

1. Data float disimpan dalam tipe tuple
2. Data string disimpan ke dalam list

3. Data int disimpan dalam dictionary, dimana data satuan, puluhan dan ribuan akan dimasukkan ke dalam dictionary sebagai value.
4. Kemudian tampilkan hasil nya seperti pada contoh.

Silahkan buka Collab untuk mendapatkan data dan potongan code untuk diselesaikan.

CODELAB 2

1. Buatlah sebuah program untuk memisahkan data input user menjadi bilangan genap dan ganjil.
2. Data input dipisah dengan spasi, data genap akan dimasukkan ke dalam list dan data ganjil akan dimasukkan ke dalam tuple. (Gunakan casting untuk menyamakan tipe data)
3. Tampilkan hasil yang telah dibuat.
4. Kalian bisa mengakses Collab untuk mendapatkan potongan code.

CODELAB 3

```
nilai_mahasiswa = {
    "example": {"example1": 80, "example2": 75, "example3": 85}
}
```

1. Buatlah sebuah kamus (nested dictionary) yang menyimpan data nilai untuk 5 mahasiswa dengan 3 mata kuliah berbeda.
2. Buatlah fungsi untuk menghitung rata-rata nilai setiap mahasiswa.
3. Buatlah fungsi untuk menghitung rata-rata nilai seluruh mahasiswa.

TUGAS

SOAL A (INTERMEDIATE)

Buatlah sebuah CRUD sederhana dengan requirements berikut :

1. Register
   - User bisa register menggunakan NIM dan passwordnya
   - Ketika berhasil register maka data user yang baru sudah tersimpan (bisa login)
   - Ketika berhasil register maka user baru diarahkan untuk mengisi biodata profil dan friends list (bisa diberikan menu skip)

2. Login
   - User yang sudah terdaftar bisa melakukan login dan mengakses menu
3. User Menu
   - Hanya mampu melihat profilnya dan datanya sendiri
   - User bisa menambahkan data profil yang sebelumnya di skip atau list temannya
   - User bisa mengedit data profilnya sendiri atau list temannya
   - User bisa menghapus data profil atau list temannya
4. Menerapkan konsep dictionary dan list untuk menyimpan data.
   - Pastikan bahwa key dari setiap dictionary berkaitan sesuai dengan data dari akun masing2, misal : akun dengan nim 402 hanya bisa mengakses dictionary profile dan friendlist dari 402)
5. **Contoh** penyimpanan Data :

```
account = {'yourNIM':'yourPassword'}
profil  = {'yourNIM': {'name':'ucup','role':'user','email':'example@mail.com'}}
friends = {'yourNIM':['teman1','teman2']}
```

**disclaimer : di atas hanya contoh saja, kamu bisa membuat yang lebih dari contoh tersebut, selama tidak kurang/downgrade dari contoh.*

SOAL B (ADVANCED)

Buatlah CRUD sesuai dengan requirements berikut:

NO.	Requirements	Description
1.	Main Duty	• Kembangkanlah tugas soal A untuk membuat sebuah CRUD dengan tema bebas (*usahakan untuk tidak duplikat agar memudahkan penilaian*)
2.	Struktur Data	• Menggunakan 3 tipe data yaitu tuple, list dan dictionary **(wajib)** • Mekanisme dan bentuk struktur data dibebaskan • Setiap 3 jenis tipe data (list, tuple, dictionary) **diwajibkan** untuk bisa melakukan CRUD
3.	Login	• Menggunakan fitur login dengan akun dan struktur data yang sama seperti soal A yaitu NIM dan Password dimana NIM ini akan dijadikan ***key*** untuk mengaitkan dengan struktur data yang lain (misal data peminjaman

		atau pembelian)
4.	Keterkaitan Key	• Jika tema yang anda pilih mengharuskan untuk mengisi data peminjam/pemesan/pembeli maka wajib untuk menggunakan keterkaitan key (layaknya primary key dan foreign key)
5.	Validasi dan verifikasi	• Buat code kalian clean dan tidak mudah ter-executed ketika salah input atau error lainnya
6.	Kreatif	• Kreatifitas dalam program maupun konsep merupakan keharusan sebagai ciri khas pribadi (agar program tidak identik).

KRITERIA & DETAIL PENILAIAN

KETERANGAN		MAX POIN	PROGRAM IDENTIK
Codelab	Codelab 1, 2, & 3 masing-masing 5 poin	15	15
Tugas A Intermediate	Pemahaman	50	42
	Code/Kelengkapan Fitur	15	0
Tugas B Advanced	Pemahaman	60	50
	Code/Kelengkapan Fitur	25	0
Total Tugas A + Codelab		**80**	**57**
Total Tugas B + Codelab		**100**	**65**

**)Note: Program Identik berarti program sama persis dengan praktikan lain sehingga yang dinilai hanya pemahaman terhadap materi (code tidak mendapat bobot nilai sama sekali).*

***)Poin diatas merupakan poin maksimal yang bisa diperoleh. Asisten bisa memberikan nilai dibawah itu jika dirasa praktikan tidak maksimal saat demo (kurangnya pemahaman tentang apa yang di demokan).*