Python Enthusiast

```
bootcamp = ["ParselMouth", "PythonEnthusiast"]
>> for i in bootcamp:
>>     if i:
>>         print("Hello {}".format(i))
...
>> print("let's learn Python Basic")
```

pwcahyo@gmail.com |
muhammadhabibi17@gmail.com

1. *Data Type*

Dalam setiap bahasa pemrograman, sudah sangatlah umum tersedia tipe data. Tipe data tersebut biasanya memiliki keterkaitan dengan definisi variable. Python merupakan bahasa pemrograman dengan karakteristik *Dynamic Typed* seperti yang dibahas pada *Python Characteristic* bagian 12. Didalam bahasa pemrograman Python *programmer* diberi kebebasan untuk mendefinisikan *variable* tanpa harus menuliskan tipe data secara spesifik. Seperti contoh penulisan *variable* dengan tipe data string berikut:

Bahasa Pemrograman Java :

```
String data = "Hello Parselmouth"
```

Bahasa Pemrograman Python

```
data = "Hello Parselmouth"
```

Sementara itu di dalam bahasa pemrograman Python terbagi ke dalam 2 jenis *Data Type* yaitu *primitive data-type* dengan *non primitive data-type* beberapa tipe data yang ada pada bahasa pemrograman Python adalah sebagai berikut:

a. *Primitive data-type*

1) Int

Int (*integer*) merupakan tipe data bilangan bulat. Seperti contoh data 10, maka dapat dilakukan pengecekan sebagai berikut :

```
>>> type(10)
>>> <class 'int'>
```

long int merupakan bilangan yang mempunyai tipe data sama seperti integer (bilangan bulat), hanya saja memiliki cakupan yang lebih panjang, (dalam python 3 long int tetap didefinisikan sebagai int)

2) Float

Float merupakan tipe data bilangan real. Seperti contoh data 30.0, maka dapat dilakukan pengecekan sebagai berikut :

```
>>> type(30.0)
>>> <class 'float'>
```

3) *Complex*

Tipe data *complex* merupakan tipe data untuk bilangan complex, biasanya didefinisikan a+bj (a dan b merupakan bilangan real dan j merupakan bilangan imaginer).

```
>>> type(2+3j)
>>> <class 'complex'>
```

4) *Boolean*

Tipe data *boolean* adalah tipe data yang hanya bernilai *True* atau *False*, atau dapat didefinisikan kedalam bilangan biner 0 (nol) atau 1 (satu), seperti contoh berikut.

```
>>> type(True)
>>> <class 'bool'>
>>> type(False)
>>> <class 'bool'>
```

5) Tipe data String

String merupakan data yang berasal dari kumpulan karakter yang berbentuk teks, untuk sebab itu data string pasti berada diantara tanda petik "". Dengan contoh kalimat "Hello PythonEnthusiast"

```
>>> type("Hello PythonEnthusiast")
>>> <class 'str'>
```

6) Tipe Data None

None digunakan untuk mendefinisikan no value/tidak ada nilai.

```
>>> type("None")
>>> class 'NoneType'>
```

b. Non primitive data-type (data structure)

1) List

List merupakan kumpulan data dari beberapa tipe data (sejenis ataupun bervariasi), definisi list selalu berada diantara tanda kurung siku [*isi list*]. Contoh sebagai berikut.

```
>>> type([1,2,3,4,5])
>>> class 'list'>
>>> type([1,2,3.90,"Hello World",5])
>>> class 'list'>
```

2) Tuple

Tuple merupakan kumpulan data dari beberapa tipe data (sejenis ataupun bervariasi), definisi tuple selalu berada diantara tanda kurung bulat (*isi tuple*). Contoh sebagai berikut.

```
>>> type((1,2,3,4,5))
>>> class 'tuple'>
>>> type((1,2,3.90,"Hello World",5))
>>> class 'tuple'>
```

3) Set

Set merupakan kumpulan data dari beberapa tipe data (sejenis ataupun bervariasi, akan tetapi secara *unique*), definisi set selalu berada diantara tanda kurung kurawal {*isi set*}. Contoh sebagai berikut.

```
>>> type({"banana","apple","banana","grape"})
>>> class 'set'>
>>> print({"banana","apple","banana","grape"})
>>> {"banana","apple","grape"} //unique
```

4) *Dictionary*

Dictionary hampir sama seperti set akan tetapi isi dari *dictionary* terdiri dari *key value pairs*, sehingga setiap value akan memiliki key bisa bertipe (integer/string/float), seperti contoh sebagai berikut.

```
>>> type({1:"banana",2:"apple",3:"banana",4:"grape"})
>>> class 'dict'>
>>> type({"a":"banana","b":"apple","c":"banana","d":"grape"})
>>> class 'dict'>
```

2. Logika kondisional

Penggunakan logika kondisional sudah sangat umum diterapkan pada penulisan kode program untuk membangun sebuah aplikasi atau sistem. Akan tetapi setiap bahasa pemrograman mengakomodasi cara penulisan yang berbeda. Di dalam bahasa pemrograman Python implementasi logikan kondisional dapat dituliskan kedalam variasi penulisan. IF secara konvensional dan IF secara ternary, IF secara konvensional dituliskan dengan penulisan kode standar penulisan IF pada umumnya. Sedangkan IF secara ternary menggunakan penulisan kode yang berbeda (lebih merujuk kepada efisiensi *line code*). Berikut dijelaskan beberapa penggunaan Logika Kondisional pada bahasa Pemrograman Python.

a. IF Tunggal

IF tunggal digunakan pada pendefinisian if secara tunggal dan hanya dalam satu konteks, apabila ada IF tunggal yang lain dan berdiri sendiri, maka dapat disebut dengan IF tunggal yang lain dalam konteks yang berbeda.

1) Contoh pencarian bilangan ganjil

```
>>> X = 3
>>> if (X%2)==1:
>>>     print("Bilangan Ganjil")
```

2) Contoh IF yang memiliki konteks yang berbeda untuk mencari bilangan ganjil dan pencocokan angka 3

```
>>> X = 3
>>> if (X%2)==1:
>>>     print("Bilangan Ganjil")
>>> if X==3:
>>>     print("Angka tiga")
```

b. IF ELIF ELSE

IF ELIF ELSE digunakan pada logika percabangan yang terdiri lebih dari satu kondisi dan semua statement yang didefinisikan masih dalam satu konteks yang sama. Contoh:

```
>>> X = 3
>>> if (float(X).is_integer())!=True:
>>>     print("Bukan bilangan bulat")
>>> elif (X%2)==1:
>>>     print("Bilangan Ganjil")
>>> else:
>>>     print("Bilangan Genap")
```

c. *Ternary* IF

Ternary IF biasanya dipakai untuk menyingkat penulisan logika percabangan. Contoh ternary IF ELSE dari:

```
>>> X = 3
>>> if (X%2)==1:
>>>     print("Bilangan Ganjil")
>>> else:
>>>     print("Bilangan Genap")
```

adalah sebagai berikut:

```
>>> X = 3
>>> print("Bilangan ganjil" if ((X%2)==1) else "Bilangan genap")
```

atau menggunakan tuple:

```
>>> X = 3
>>> print(("Bilangan genap", "Bilangan ganjil")[((X%2)==1)])
```

atau menggunakan dictionary:

```
>>> X = 3
>>> print({True:"Bilangan ganjil", False:"Bilangan genap"}[((X%2)==1)])
```

atau menggunakan lambda function:

```
>>> X = 3
>>> print((lambda:"Bilangan genap", lambda:"Bilangan ganjil")[(X%2)==1]())
```

d. IF AND, OR

1) IF AND

Logika IF AND seringkali dipakai untuk menghubungkan dua buah kondisi atau lebih, dengan syarat dua buah kondisi atau lebih tersebut harus bernilai benar untuk menghasilkan konklusi yang dimaksud. Seperti contoh berikut:

```
>>> X = 3
>>> if ((X%2)==1) AND (X==3):
>>>     print("X adalah bilangan ganjil")
```

2) IF OR

Logika IF OR seringkali dipakai untuk menghubungkan dua buah kondisi atau lebih, dengan syarat salah satu kondisi yang terjadi harus bernilai benar untuk menghasilkan konklusi yang dimaksud. Seperti contoh berikut:

```
>>> X = 3
>>> if ((X%2)==1) OR (X==3):
>>>     print("X adalah bilangan ganjil")
```

e. NESTED IF

Logika Nested If seringkali mensyaratkan ketergantungan pada logika sebelumnya. dapat dikatakan, untuk mencapai persyaratan selanjutnya maka dibutuhkan hasil persyaratan sebelumnya. Seperti contoh berikut ini logika dengan color warna **ungu** memerlukan hasil kondisi logika warna **merah** :

```
>>> X = 3
>>> if ((X%2)==1):
>>>     if (X>10):
>>>             print("X diatas 10")
>>>     elif (X==3):
>>>             print("X adalah 3")
>>>     else:
>>>             print("X dibawah 10")
```

3. Perulangan

Perulangan merupakan logika kode yang seringkali diterapkan selain logika kode percabangan. Disetiap bahasa pemrograman, kode perulangan itu sendiri biasanya memiliki beberapa variasi cara perulangan ataupun penulisan kode. Secara umum perulangan konvensional terbagi menggunakan for atau while. Pada bahasa pemrograman Python dapat ditunjukan sebagai berikut.

a. Range

range merupakan suatu fungsi yang ada didalam bahasa pemrograman python yang sering digunakan untuk menyediakan nilai jarak antara start number sampai stop number, dipengaruhi dengan step number, kemudian akan menyimpan data hasil range kedalam list.

Format penulisan range:

```
range(start, stop, step)          # garis bawah adalah optional
```

Contoh 1: (secara standard, hanya menggunakan nilai stop)

```
>>> print(range(3))
 [0,1,2]
```

Contoh 2: (menggunakan inisialisasi nilai start dan stop)

```
>>> print(range(1,3))
 [1,2]
```

Contoh 3: (menggunakan inisialisasi nilai start, stop dan step)

```
>>> print(range(1,3,+2))
 [1]
```

```
>>> print(range(5,1,-2))
 [5,3]
```

b. FOR

Didalam pemrograman, perulangan seringkali digunakan untuk menyelesaikan kondisi yang berulang dan kondisi tersebut bisa memiliki pola yang sama ataupun berbeda. Perulangan didalam bahasa pemrograman python secara sederhana biasanya mengakomodasi range dapat dilihat sebagai berikut:

```
>>> for i in range(3):
>>>     print(i)
0
1
2
```

BERBEDA dengan

```
>>> for i in range(1,3):
>>>     print(i)
1
2
```

BERBEDA dengan

```
>>> for i in range(1,3,+2):
>>>     print(i)
1
```

c. Perulangan (for) secara bersarang

Didalam pemrograman, perulangan secara bersarang sering dijumpai pada pembentukan matrix (array multidimensi) atau pembentukan pola tertentu. Perulangan secara bertingkat didefinisikan dengan didalam for akan terdapat for atau perulangan lain, seperti contoh berikut ini :

```
>>> for i in range(0,5):
>>>     for j in range(0, i+1):
>>>             print("*", end=" ")
>>>     print()
*
* *
* * *
* * * *
* * * * *
```

d. Perulangan (while)

Perulangan `while` hampir sama dengan `for`, yaitu untuk mengatasi permasalahan berulang, akan tetapi dengan memanfaatkan suatu kondisi, sampai kondisi yang dimaksud terpenuhi. Didalam bahasa pemrograman, perulangan `while` dapat dituliskan sebagai berikut.

```
while kondisi:
        blok_eksekusi
```

contoh pada bahasa pemrograman python:

```
>>> suhu = 0
>>> while suhu < 100:
>>>     print(suhu)
>>>     suhu+=40
0
40
80
```

didalam `while` biasanya ada `counter`, `counter` tersebut digunakan untuk menjalankan `blok_eksekusi` hingga mencapai `kondisi` yang dimaksud, seperti contoh perulangan diatas menggunakan `counter suhu+=40` untuk melakukan `increment` agar `kondisi` terpenuhi.

e. Perulangan (while) secara bersarang

konsep dari perulangan while secara bersarang hampir sama dengan perulangan for secara bersarang, keduanya mengakomodasi perulangan didalam while kemungkinan ada perulangan while lebih dari sama dengan satu atau ada perulangan lain. Seperti contoh berikut, didalam while ada perulangan while.

```
>>> i = 0
>>> while i < 5:
>>>     j = 0
>>>     while j <= i:
>>>             print("*", end=" ")
>>>             j+=1
>>>     print()
>>>     i+=1
```

output program while secara bersarang :

```
*
* *
* * *
* * * *
* * * * *
```

f. BREAK

Fungsi `break` digunakan untuk menghentikan perulangan dengan mengakomodasi `kondisional statement`. Contoh penggunaan `break` pada bahasa pemrograman `python` dapat dilihat sebagai berikut. (menghentikan perulangan apabila angka 3 didapat)

`break` pada `for` :

```
>>> for i in range(0,5):
>>>     if i == 3:
>>>         print("angka 3 didapat")
>>>         break
>>>     print(i)
0
1
2
angka 3 didapat
```

break pada while :

```
>>> i = 0
>>> while i < 5:
>>>     if i == 3:
>>>         print("angka 3 terambil")
>>>         break
>>>     print(i)
>>>     i+=1
1
2
angka 3 didapat
```

g. PASS

fungsi `pass` digunakan untuk membiarkan, atau tidak ada sesuatu yang akan dijalankan, biasanya digunakan untuk inisialisasi kode program untuk terus berjalan, dengan tanpa menghiraukan definisi kode yang ada. Seperti contoh berikut, pass digunakan untuk inisialisasi agar program berjalan pada kode if.

```
>>> for i in range(5):
>>>     if (i%2)==1 or (i==0):
>>>             pass
>>>     else:
>>>             print("{} adalah bilangan genap".format(i))
```

Kemudian apabila kode pas dihilangkan, apa yang terjadi ?

```
>>> for i in range(5):
>>>     if (i%2)==1 or (i==0):
>>>
>>>     else:
>>>             print("{} adalah bilangan genap".format(i))
```

pass digunakan untuk melakukan inisialisasi agar program tetap berjalan, meskipun didalam blok kode tersebut tidak ada blok kode yang dijalankan. Agar program dapat terus berjalan.

h. continue

fungsi continue berbeda dengan pass, continue digunakan untuk melompati proses yang berjalan saat kondisi yang didefinisikan terpenuhi. proses saat itu akan dilompati dan langsung kembali ke perulangan yang berada setelah proses perulangan tersebut, tanpa menghiraukan operasi yang ada. Dalam bahasa pemrograman python continue dapat dilihat sebagai berikut.

```
>>> for i in range(5):
>>>     if i == 3:
>>>             continue
>>>             print("ini continue")
>>>     print("angka sekarang : ",i)
angka sekarang :  0
angka sekarang :  1
angka sekarang :  2
angka sekarang :  4
```

4. Function

Pada bahasa pemrograman python, pendefinisian function selalu menggunakan def diikuti oleh nama function, dilanjutkan dengan pendefinisian parameter yang diapit tanda kurung. Kemudian baris dibawahnya berisi kode function yang digunakan untuk menyelesaikan beberapa tujuan fungsionalitas. Setiap function umumnya memiliki nilai kembalian hasil dari proses yang dihasilkan oleh function. cara mengembalikan

nilai function biasanya menggunakan perintah return. kode penulisan function pada bahasa pemrograman python secara umum dapat dituliskan sebagai berikut :

```
def nama_function(parameter):
        isi kode function….
        return object_variable
```

Pendefinisian function diatas memiliki satu buah parameter, parameter function dapat lebih dari satu (penuliskan dipisahkan dengan koma) seperti contoh dibawah.

```
def nama_function(parameter1, parameter2, parameter3):
        isi kode function….
        return object_variable
```

selain itu function juga bisa tidak memiliki parameter satupun, seperti contoh penulisan function tanpa parameter dibawah.

```
def nama_function():
        isi kode function….
        return object_variable
```

berikut contoh function luas lingkaran, dengan nilai kembalian luas yang ditulis pada bahasa pemrograman python :

```
def luas_lingkaran(r):
        phi = 3.14
        luas = phi*(r**2)
        return luas
```

Variable global, local dan parameter

variable global adalah variable yang dikenali oleh setiap function/method yang didefinisikan. Variable global biasanya didefinisikan di luar function/method. Berbeda dengan variable local, **variable local** adalah kebalikan dari variable global, variable local hanya dikenali pada function/method yang mendefinisikan variable tersebut. **parameter** adalah variable yang berada pada deklarasi function, umumnya parameter merupakan variable yang bersifat local, sehingga pada function tersebut parameter hanya dapat dikenali sebagai variable. Dapat dilihat pada contoh berikut :

```
phi = 3.14

def luas_lingkaran(r):
        luas = phi*(r**2)
        return luas

def keliling_lingkaran(r):
        keliling = phi*(r*2)
        return keliling
```

dari definisi kode diatas, variable **phi** adalah variable global karena didefinisikan diluar fungsi, phi tersebut dapat dikenali pada function luas_lingkaran dan function keliling_lingkaran. Sedangkan variable **luas** dan **keliling** adalah variable local, dikarenakan luas hanya dikenali pada function luas_lingkaran dan keliling hanya dikenali pada function keliling_lingkaran. Setiap function tersebut memiliki satu buah parameter yaitu r, nilai dari **r** hanya dikenali pada masing – masing function (karena r adalah variable local), jadi r pada function luas_lingkaran bisa saja mempunyai nilai yang berbeda dengan r pada keliling_lingkaran.

Latihan:

1. Tuliskan sebuah program yang dapat menerima inputan urutan angka yang dipisahkan dengan koma dari console, buat list dan tuple yang berisi setiap angka tersebut.
   - Misalkan input angka berikut ke program:
     26,37,45,12,79,50
   - Maka, output dari program berupa:
     ['26', '37', '45', '12', '79', '50']
     ('26', '37', '45', '33', '79', '50')

2. Tuliskan sebuah program untuk menyusun setiap nomor ganjil dalam list. list dimasukkan oleh urutan angka yang dipisahkan dengan koma.
   - Misalkan diberikan input ke program sebagai berikut:
     1,2,3,4,5,6,7,8,9
   - Maka, output dari program berupa:
     1,3,5,7,9

3. Tuliskan sebuah program untuk mengurutkan tuple (nama, usia, nilai) dengan urutan naik, dimana nama adalah string, usia dan nilai adalah angka. Tuple dimasukkan dari console. Kriteria pengurutannya adalah:
   a. Sortir berdasarkan nama;
   b. Kemudian urutkan berdasarkan usia;
   c. Kemudian urutkan berdasarkan nilai.

   Prioritasnya adalah nama> usia> nilai.

- Misalkan diberikan input ke program sebagai berikut:
  Cahyo, 20,95
  Dana, 19,90
  Habibi, 17,93
  Habibi, 17,91
- Maka, output dari program berupa:
  [('Cahyo', '20', '95'), ('Dana', '19', '90'), ('Habibi', '17', '91'), ('Habibi', '17 ',' 93 ')]

4. Tuliskan sebuah program untuk menghitung jumlah bersih dari rekening bank berdasarkan log transaksi dari input console. Format log transaksi ditampilkan sebagai berikut:
   D 100
   W 200
   D (*Deposite*) berarti setoran sementara, W (*withdrawal*) berarti penarikan.
   - Misalkan diberikan input ke program sebagai berikut:
     D 400
     D 200
     W 300
     D 100
   - Maka, output dari program berupa:
     400

5. Tuliskan sebuah program yang menerima inputan berupa kalimat dan hitung jumlah huruf dan angka pada kalimat tersebut.
   - Misalkan diberikan input ke program sebagai berikut:
     Apa Kabar? 123
   - Maka, output dari program berupa:
     Huruf: 8
     Angka: 3

6. Tuliskan sebuah program untuk menghitung frekuensi kata-kata dari inputan kalimat. Keluaran akan ditampilkan setelah mengurutkan kunci secara alfanumerik.
   - Misalkan diberikan input ke program sebagai berikut:
     Ada perubahan syntax pada python 2 dan python 3
   - Maka, output dari program berupa:
     2:1
     3:1
     ada:1
     dan:1
     pada:1
     perubahan:1
     python:2
     syntax:1