MODUL 2
DASAR PYTHON 2

2.1 Tujuan

Setelah Mengikuti Praktikum ini, mahasiswa ini diharapkan dapat:

1. Dapat Memahami Python Operators, List, Tuples, Sets, Dictionaries, If…Else, While Loops, For Loops, Functions.
2. Dapat Mempraktikan Python Operators, List, Tuples, Sets, Dictionaries, If…Else, While Loops, For Loops, Functions.

2.2 Alat & Bahan

1. Laptop
2. Mouse
3. Visual Studio Code
4. Software Python

2.3 Dasar Teori

2.3.1 Software Python

Pengertian Python (bahasa pemrograman) merupakan bahasa pemrograman tinggi yang bisa melakukan eksekusi sejumlah instruksi multi guna secara langsung (interpretatif) dengan metode *Object Oriented Programming* dan juga menggunakan semantik dinamis untuk memberikan tingkat keterbacaan syntax. Sebagai bahasa pemrograman tinggi, python dapat dipelajari dengan mudah karena telah dilengkapi dengan manajemen memori otomatis.

2.3.2 Python Operators

Operator merupakan sebuah simbol khusus yang digunakan dalam pemrograman untuk melakukan operasi tertentu, seperti membuat sebuah alur logika, perhitungan angka dan lain-lain.Operator digunakan untuk melakukan operasi pada variabel dan nilai. Contoh:

```
print(11 + 5)
```

Hasil: 16

Python membagi operator dalam grup berikut:

A. Operasi Aritmatika

Operasi aritmatika digunakan dengan nilai numerik untuk melakukan operasi matematika umum.

Operasi	Name	Example
+	Penambahan	x + y
-	Pengurangan	x - y
*	Perkalian	x * y
/	pembagian	x / y
%	Modulus /sisa bagi	x % y
**	Pangkat	x ** y
//	Pembagian bulat	x//y

B. Operasi Penugasan Python

Operasi penugasan digunakan untuk menetapkan nilai ke variabel. Contoh :

```
x = 5
print(x)
```

Hasil: 5

```
x = 5
x += 3
print(x)
```

Hasil: 8

C. Operator Perbandingan Python

Operator perbandingan digunakan untuk membandingkan dua nilai.

Operasi	Name	Example
= =	setara	x = = y
!=	tidak sama	x != y
>	lebih besar dari	x > y
<	Kurang dari	x < y
>=	Lebih besar dari sama dengan	x >= y
< =	Kurang dari sama dengan	x <= y

D. Python Logical Operators

Operator logis digunakan untuk menggabungkan pernyataan kondisional.

Operasi	Name	Example
and	mengembalikan True jika kedua pernyataan benar	x < 5 and x < 10
or	mengembalikan Benar jika salah satu pernyataan benar	X <5 or X< 4
not	balikkan hasilnya, kembalikan false jika hasilnya benar	Not(x < 5 and x < 10)

E. Python Identity Operators

Digunakan untuk membandingkan dua objek, bukan hanya untuk menentukan apakah nilai mereka sama, tetapi untuk memeriksa apakah mereka adalah objek yang sama dalam lokasi memori yang sama.

Operasi	Name	Example
is	mengembalikan Benar jika kedua Variabel adalah objek yang sama	x is y
Is not	mengembalikan Benar jika kedua variabel bukan objek yang sama	x is not y

F. Python Membership Operators

Operasi keanggotaan digunakan untuk menguji apakah suatu nilai ada dalam koleksi. Ada dua operator keanggotaan yang digunakan dalam Python:

Operasi	Name	Example
in	mengembalikan Benar jika urutan dengan nilai yang ditentukan ada di objek	x in y
Not in	mengembalikan Benar jika urutan dengan nilai yang ditentukan tidak ada dalam objek	x not in y

G. Python Bitwise Operators

Operator bitwise digunakan untuk membandingkan angka (biner).

Operasi	Name	Example
&	AND	Mengatur setiap bit ke 1 jika kedua bit adalah 1
|	OR	Set setiap bit ke 1 jika salah satu dari dua bit adalah 1
^	XOR	Mengatur setiap bit menjadi 1 jika hanya satu dari dua bit yang bernilai 1
~	NOT	Membalikkan semua bit

<<	Zero fill left shift	mengisi shift kiri Geser ke kiri dengan menekan angka nol dari kanan dan biarkan bit paling kiri jatuh
>>	Signed right shift	kanan bertanda Geser ke kanan dengan mendorong salinan bit paling kiri ke dalam dari kiri, dan biarkan bit paling kanan jatuh

2.3.3 Python List

a. List

List digunakan untuk menyimpan beberapa item dalam satu variabel. List adalah salah satu dari 4 tipe data bawaan di Python yang digunakan untuk menyimpan koleksi data, 3 lainnya adalah Tuple, Set, dan Dictionary, semuanya dengan kualitas dan penggunaan yang berbeda. Contoh:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Hasil: `['apple', 'banana', 'cherry']`

a) List Length

Untuk menentukan berapa banyak item yang dimiliki daftar, digunakan fungsi len(). Contoh:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

Hasil: `3`

b. Python Tuple

Tuple digunakan untuk menyimpan beberapa item dalam satu variabel. Tuple adalah salah satu dari 4 tipe data bawaan di Python yang digunakan untuk menyimpan koleksi data, 3 lainnya adalah List, Set, dan Dictionary, semuanya dengan kualitas dan penggunaan yang berbeda. Tuple adalah koleksi yang dipesan dan **tidak dapat diubah**. Tuple ditulis dengan tanda kurung bulat. Contoh:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Hasil: `('apple', 'banana', 'cherry')`

a) Tuple Length

Untuk menentukan berapa banyak item yang dimiliki tupel, gunakan fungsi len(). Contoh:

```
thistuple = tuple(("apple", "banana", "cherry"))
print(len(thistuple))
```

Hasil: 3

2.3.4 Python Set

a. Set

Set digunakan untuk menyimpan beberapa item dalam satu variabel. Set adalah salah satu dari 4 tipe data bawaan di Python yang digunakan untuk menyimpan koleksi data, 3 lainnya adalah List, Tuple, dan Dictionary, semuanya dengan kualitas dan penggunaan yang berbeda. Satu set adalah koleksi yang **tidak berurutan, tidak dapat diubah dan tidak diindeks**. Contoh:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Hasil: {'banana', 'cherry', 'apple'}

b. Get the Length of a Set

Untuk menentukan berapa banyak item yang dimiliki satu set, gunakan fungsi len(). Contoh:

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

Hasil: 3

2.3.5 Python Dictionaries

a. Dictionary

Dictionary digunakan untuk menyimpan nilai data dalam pasangan key:value.

Dictionary adalah koleksi yang dipesan*, dapat diubah, dan tidak mengizinkan duplikat. **Pada Python versi 3.7, Dictionary *dipesan*. Di Python 3.6 dan sebelumnya, kamus *tidak berurutan*.**

Dictionary ditulis dengan tanda kurung kurawal, dan memiliki kunci dan nilai. Contoh:

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
print(thisdict)
```

Hasil: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

b. Dictionary Items

Dictionary Items adalah tersusun, diubah, dan tidak mengizinkan duplikat. Dictionary Items disajikan dalam pasangan kunci:nilai, dan dapat dirujuk dengan menggunakan nama kunci.

2.3.6 Python Conditions and If Statements

Python mendukung kondisi logis yang biasa dari matematika:

1. Sama dengan: a == b
2. Tidak Sama dengan: a != b
3. Kurang dari: a < b

4. Kurang dari sama dengan: a <= b

5. Lebih besar dari: a > b

6. Lebih besar dari sama dengan: a >= b

Kondisi ini dapat digunakan dalam beberapa cara, paling sering dalam "pernyataan if" dan loop. "Pernyataan if" ditulis dengan menggunakan kata kunci if. Contoh:

```
a = 33
b = 200
if b > a:
    print("b lebih dari a")
```

Hasil: `b lebih dari a`

a. Elif

Kata kunci elif adalah cara pythons untuk mengatakan "jika kondisi sebelumnya tidak benar, maka coba kondisi ini". Contoh:

```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Hasil: `a and b are equal`

b. Else

Kata kunci else digunakan dalam struktur percabangan if untuk menangkap semua kondisi yang tidak tertangkap oleh pernyataan if atau elif sebelumnya. Ini memberikan alternatif atau aksi yang akan dilakukan ketika semua kondisi di atasnya bernilai False.. Contoh:

```
a   = 200
b   = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

Hasil: `a is greater than b`

c. And

Kata kunci and adalah operator logis yang digunakan untuk menggabungkan dua atau lebih pernyataan kondisional. Contoh:

```
a = 200
b = 33
c = 500
```

```
if a > b and c > a:
    print("Both conditions are True")
```

Hasil: `Both conditions are True`

d. Or

Kata kunci or adalah operator logis yang digunakan untuk menggabungkan pernyataan kondisional. Contoh:

```
a   = 200
b   = 33
c = 500
if a > b or a > c:
    print("At least one of the conditions is True")
```

Hasil: `At least one of the conditions is True`

2.3.7 Python Loops

Python memiliki dua perintah loop primitif:

1. while loop
2. for loop

a. The While Loop

Dengan menggunakan **loop while**, kita dapat mengeksekusi serangkaian pernyataan selama suatu kondisi bernilai True. Loop ini akan terus berulang hingga kondisi yang telah ditentukan.

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

Hasil:

```
1
2
3
4
5
```

b. For Loops

For loops digunakan untuk mengulangi suatu urutan (baik berupa daftar, tupel, kamus, himpunan, atau string).

Dengan For Lopps kita dapat mengeksekusi serangkaian pernyataan, satu kali untuk setiap item dalam daftar, tuple, set, dll. Contoh :

```
fruits = ["apple", "banana", "cherry"]
for   x   in
fruits:
  print(x)
```

Hasil : apple banana cherry

c. The Break Statement

Dengan pernyataan break kita dapat menghentikan loop bahkan jika kondisi while benar. Contoh:

```
 i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

Hasil:

```
1
2
3
```

d. The Continue Statement

Dengan pernyataan lanjutan kita dapat menghentikan iterasi saat ini, dan melanjutkan dengan yang berikutnya: Contoh:

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

Hasil:

```
1
2
4
5
6
```

e. The Else Statement

Dengan pernyataan else kita dapat menjalankan blok kode sekali ketika kondisinya tidak lagi benar. Contoh:

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

Hasil:

```
1
2
```

```
3
4
5
i is no longer less than 6
```

2.3.8 Python For Loops

For Loops digunakan untuk mengulangi urutan (yaitu daftar, tupel, kamus, himpunan, atau string). Contoh:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
print(x)
```

Hasil:

```
apple
banana
cherry
```

a. The break Statement

Dengan pernyataan break kita dapat menghentikan loop sebelum loop melalui item. Contoh:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

Hasil: `apple`

2.3.9 Python Functions

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu. Fungsi hanya akan dieksekusi ketika dipanggil, sehingga memungkinkan penggunaan kembali kode dan pengorganisasian yang lebih baik dalam program.

a. Calling a Function

Untuk memanggil fungsi, gunakan nama fungsi diikuti dengan tanda kurung. Contoh:

```
def my_function():
    print("Hello from a function")
my_function()
```

Hasil: `Hello from a function`

b. Arguments

Informasi dapat diteruskan ke dalam fungsi sebagai argumen. Argumen ditentukan setelah nama fungsi, di dalam tanda kurung. Anda dapat menambahkan argumen sebanyak yang Anda inginkan, cukup pisahkan dengan koma. Contoh berikut memiliki fungsi dengan satu argumen (fname). Ketika fungsi ini dipanggil, kita meneruskan nama depan, yang digunakan di dalam fungsi untuk mencetak nama lengkap. Contoh:

```
def my_function(fname):
    print(fname + " Refsnes")
my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Hasil:

```
Emil Refsnes
Tobias Refsnes
Linus Refsnes
```

c. Number of Arguments

Secara default, fungsi harus dipanggil dengan jumlah argumen yang benar. Artinya jika fungsi Anda mengharapkan 2 argumen, Anda harus memanggil fungsi dengan 2 argumen, tidak lebih, dan tidak kurang. Contoh:

```
def my_function(fname, lname):
    print(fname + " " + lname)
my_function("Emil", "Refsnes")
```

Hasil: `Emil Refsnes`

d. Arbitrary Arguments, *args

Jika Anda tidak tahu berapa banyak argumen yang akan diteruskan ke fungsi Anda, tambahkan * sebelum nama parameter dalam definisi fungsi. Dengan cara ini fungsi akan menerima *banyak* argumen, dan dapat mengakses item yang sesuai. Contoh:

```
def my_function(*kids):
    print("The youngest child is " + kids[2])
my_function("Emil", "Tobias", "Linus")
```

Hasil: `The youngest child is Linusc`

Return Values

Untuk membiarkan fungsi mengembalikan nilai, gunakan pernyataan pengembalian. Contoh:

```
def my_function(x):
    return 5 * x
print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Hasil:

```
15
25
45
```