

## ▼ Welcome to Python Fundamentals

*by R.J. Surio ©2021*

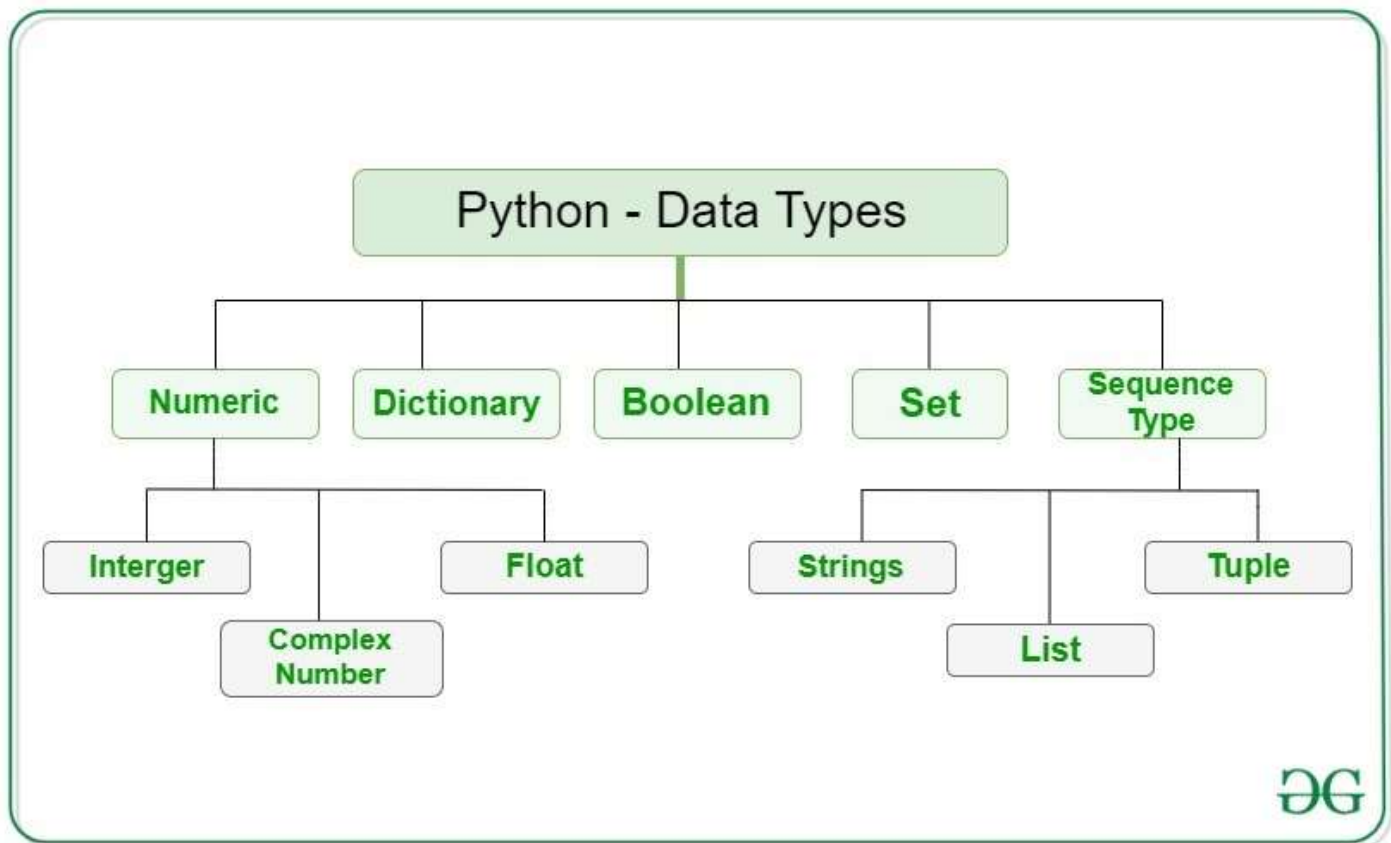
In this module, we are going to establish or review our skills in Python programming. In this notebook we are going to cover:

- Variables and Data Types
- Operations
- Input and Output Operations
- Logic Control
- Iterables
- Functions

## ▼ Variable and Data Types

Discussion :

A variable is something like a container in programming that holds data of different types. It is needed because data needs to have a place to become accessible throughout the process or logic being implemented to solve problems. The different Python built-in data types are shown below [1].



## ▼ Examples

```
x = 1
a,b = 0, -1
```

```
type(x)
```

```
int
```

```
y = 1.0
type(y)
```

```
float
```

```
x = float(x)
type(x)
```

```
float
```

```
s,t,u = "0", '1', 'one'
type(s)
```

str

```
s_int = int(s)
s_int
0
```

## ▼ Operations

### Arithmetic

#### Discussion :

Arithmetic operators are used to perform the basic mathematical operations used with numerical values [2].



## ▼ Examples

```
a,b,c,d = 2.0, -0.5, 0, -32
```

```
### Addition
```

```
S = a+b  
S
```

1.5

```
### Subtraction  
D = b-d  
D
```

31.5

```
### Multiplication  
P = a*d  
P
```

-64.0

```
### Division  
Q = c/a  
Q
```

0.0

```
### Floor Division  
Fq = a//b  
Fq
```

-4.0

```
### Exponentiation  
E = a**b  
E
```

0.7071067811865476

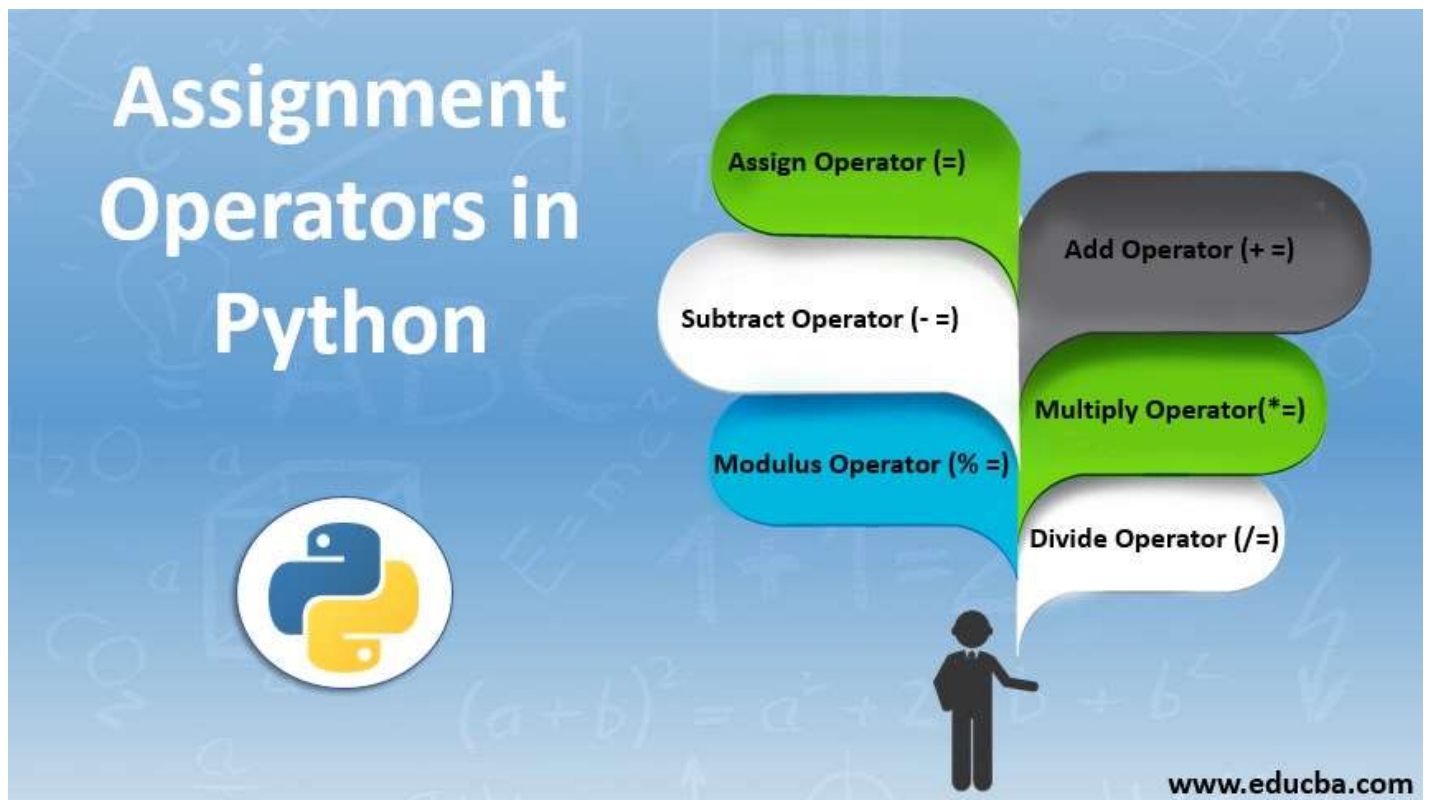
```
### Modulo  
mod = d%a  
mod
```

0.0

## Assignment Operations

Discussion :

These are used to assign values to any variables [2].



## ▼ Examples

```
G, H, J, K = 0, 100, 2, 2
```

```
G += a  
G
```

2.0

```
H -= d
```

```
J *= 2  
J
```

4

```
K **= 2  
K
```

4

## Comparators

## Discussion :

Comparators are used to make comparison of 2 values [2].



# Python Comparison Operators



## ▼ Examples

```
res_1, res_2, res_3 = 1, 2.0, "1"  
true_val = 1.0
```

```
## Equality  
res_1 == true_val
```

True

```
## Non-equality  
res_2 != true_val
```

True

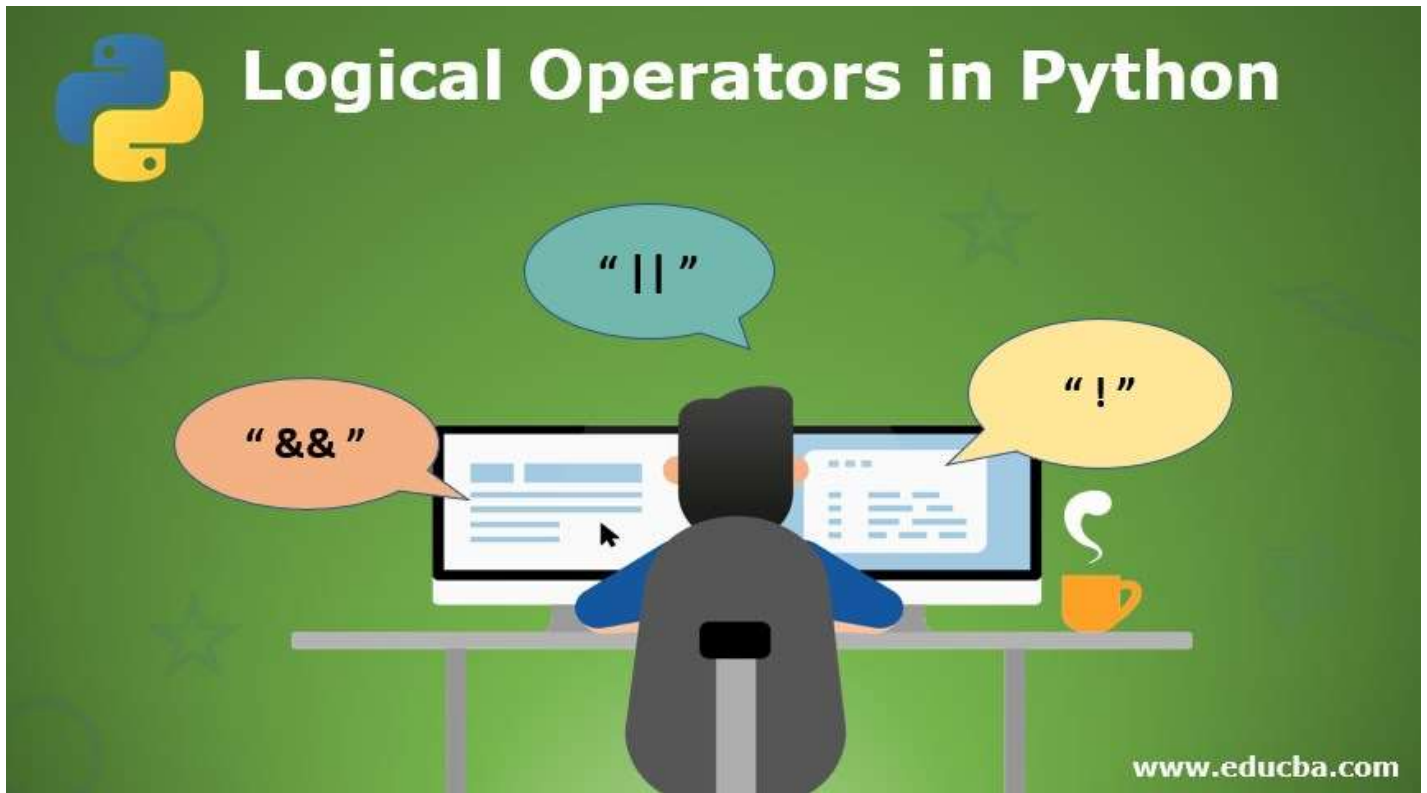
```
## Inequality  
t1 = res_1 > res_2  
t2 = res_1 < res_2/2  
t3 = res_1 >= res_2/2  
t4 = res_1 <= res_2  
+1
```

False

## Logical

### Discussion :

Logical operators help to have better program control flow or logic design [3]. The AND, OR, and, NOT are logical operators.



### ▼ Examples

```
if 3+1 and 2 + 2 == 4:  
    print("yey")
```

yey

```
res_1 == true_val
```

True

```
res_1 is true_val
```

False

```
res_1 is not true_val
```

True

```
p, q = True, False  
conj = p and q  
conj
```

False

```
p, q = True, False  
disj = p or q  
disj
```

True

```
p, q = True, False  
nand = not(p and q)  
nand
```

True

```
p, q = True, False  
xor = (not p and q) or (p and not q)  
xor
```

True

## Identity

### Discussion :

There are two identity operators and the first one is 'Is'. This is used to return a 'true' or 'false' values as result to check if the location of 2 objects in memory are the same. The second one is the 'Is not' which works in opposite way of the first operator. This will return 'true' if the object is not in the same memory location else 'false' [4].



# Identity Operators in Python

Code:

```
m = 70
n = 70
if ( m is n ):
    print("Result: m and n have same identity")
else:
    print("Result: m and n do not have same identity")
```

Output

Result: m and n have same identity



[www.educba.com](http://www.educba.com)

## ▼ Examples

```
megumin = "waifu"
roxy = "also waifu"

if megumin is roxy:
    print("what do you mean")

else: #Represents 'false'
    print("ofcourse megumin is not roxy!")
```

ofcourse megumin is not roxy!

```
if megumin is not roxy: #Represents 'true'
    print("tho they are both waifu!")

else:
    print("ofcourse megumin is not roxy!")
```

tho they are both waifu!

I/O

## Discussion :

Input and output is one of the fundamental operations of a computer. In other words, it is the read and write of data. For the examples below, it shows how Python gets the input of the user and store in a specific memory location and it's being access using the assigned variables of it.



### ▼ Examples

```
print("Hello World")
```

```
cnt = 1
```

```
string = "Hello World"  
print(string, ", Current run count is:", cnt)  
cnt += 1
```

```
print(f"{string}, Current count is: {cnt}")
```

```
    Hello World, Current count is: 2
```

```
sem_grade = 82.243564657461234  
name = ""  
print("Hello {}, your semestral grade is: {}".format(name, sem_grade))
```

```
Hello , your semestral grade is: 82.24356465746123
```

```
w_pg, w_mg, w_fg = 0.3, 0.3, 0.4
print("The weights of your semestral grades are:\n\t{:.2%} for Prelims\n\t{:.2%} for Midterms, and\n\t{:.2%} for Finals.".format(w_pg, w_mg, w_fg))
```

```
The weights of your semestral grades are:
    30.00% for Prelims
    30.00% for Midterms, and
    40.00% for Finals.
```

```
x = input("enter a number: ")
x
```

```
enter a number:
''
```

```
name = input("Kimi no nawa: ")
pg = input("Enter prelim grade: ")
mg = input("Enter midterm grade: ")
fg = input("Enter finals grade: ")
sem_grade = None
print("Hello {}, your semestral grade is: {}".format(name, sem_grade))
```

```
Kimi no nawa:
Enter prelim grade:
```

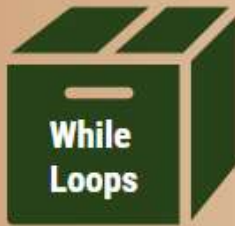
## ▼ Looping Statements

### Discussion :

Looping statements is a concept in Python programming that allows repeating a line of code such as printing of string or execution of function for the number of times specified. The while loops are used to execute something until a certain condition is met and the for loops is used for executing something until it reached the end of the specified range of values [5].



# Loops in Python



[www.educba.com](http://www.educba.com)

## Examples

### ▼ While

```
## while loops
i, j = 0, 10
while(i<=j):
    print(f"{i}\t|\t{j}")
    i+=1
```

0		10
1		10
2		10
3		10
4		10
5		10
6		10
7		10
8		10
9		10
10		10

### ▼ For

```
# for(int i=0; i<10; i++){  
# printf(i)  
# }
```

```
i=0  
for i in range(10):  
    print(i)
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

```
playlist = []  
print('Now Playing:\n')  
for song in playlist:  
    print(song)
```

```
Now Playing:
```

## ▼ Flow Control

## ▼ Conditional Statements

### Discussion :

Conditional statements are used to run a specific line of code if a certain condition met and it is very useful in controlling the flow of the code [6].

# Conditional Statements in Python



```
var1 = 1000
var2 = 1000
if var2 > var1:
    print("var2 is greater than var1")
elif var2 == var1:
    print("var1 is equal to var2")
else:
    print("var1 is greater than var2")
```

Output:  
var1 is equal to var2

[www.educba.com](http://www.educba.com)

## ▼ Examples

```
numeral1, numeral2 = 12, 12
if(numeral1 == numeral2):
    print("Yey")
elif(numeral1>numeral2):
    print("Hoho")
else:
    print("Aww")
print("Hip hip")
```

Yey  
Hip hip

## ▼ Functions

### Discussion :

Functions in Python programming have a lot of use and one of these is to organize your code because a specific line block of code can be inside a function and it will just run if that function was called. For me, I find it very useful especially in object oriented programming with Python.

## ▼ Examples

```
# void DeleteUser(int userid){
#     delete(userid);
# }

def delete_user (userid):
    print("Successfully deleted user: {}".format(userid))

def delete_all_users ():
    print("Successfully deleted all users")
```

```
userid = 0
delete_user(0)
delete_all_users()

    Successfully deleted user: 0
    Successfully deleted all users
```

```
def add(addend1, addend2):
    return addend1 + addend2

def power_of_base2(exponent):
    return 2**exponent
```

## Lambda Functions

### Discussion :

Lambda functions are also known as anonymous functions and this is very useful in optimizing code because this function that contains specific instructions can be used inside a certain code block without needing to declare or create an actual function. The two examples below are the comparison of using the lambda function and a normal function with a declaration.

## ▼ Examples

```
x = 4
```

```
def f(x):
    return 2*(x*x)-1
f(x)
```

```
g = lambda x: 2*(x*x)-1
print(g(x))
```

## ▼ Activity

```
'''
Create a grade calculator that computes for the semestral grade of a course.
Students could type their names, the name of the course, then their prelim,
midterm, and final grade.
The program should print the semestral grade in 2 decimal points and should
display the following emojis depending on the situation:
happy - when grade is greater than 70.00
laughing - when grade is exactly 70.00
sad - when grade is below 70.00
'''

happy, lol, sad = "\U0001F600","\U0001F923","\U0001F619"

class Menu:
    def __init__(self): # Method for main menu
        print("Grade calculator")
        print("Menu")
        print("1. Add your details minna")
        print("2. Compute for your semestral grade")

class Student_details:

    def __init__(self): # Initializer
        """ Instance Variables"""

        self.student_name = "Default text"
        self.course = "Default text"
        self.pg = "Default text"
        self.mg = "Default text"
        self.fg = "Default text"

    def set_student_details(self): # Method for the inputs
        self.student_name = input("Enter your name: ")
        self.course = input("Enter your course: ")
        self.pg = float(input("Enter prelim grade: "))
        self.mg = float(input("Enter midterm grade: "))
        self.fg = float(input("Enter finals grade: "))

class Operations:
```



```

def __init__(self):
    self.student_lists = list()

def add_student(self):
    stud_details = Student_details() # Creation of object/instantiation
    stud_details.set_student_details() # Calling the method "set_student_details()"
    self.student_lists.append(stud_details) # appending the object to the list

def compute(self):
    w_pg, w_mg, w_fg = 0.3, 0.3, 0.4
    for grade in self.student_lists: # Accessing the grade attribute
        sem_grade = grade.pg*w_pg + grade.mg*w_mg + grade.fg*w_fg # Grade computation

    print ("Your semestral grade is:",(round(sem_grade,2))) # rounding off to 2 decimals

    if sem_grade > 70:
        print(happy)

    elif sem_grade == 70:
        print(lol)

    else:
        print(sad)

if __name__ == '__main__':
    operation = Operations()
    user_choice = "1"
    while user_choice == "1":
        menu = Menu()
        user_choice = input("Input selected option: ")
        if user_choice == "1":
            operation.add_student()
        elif user_choice == "2":
            operation.compute()

    user_choice = input("Press 1 to Continue, Press 2 to Exit ")

```

```

Grade calculator
Menu
1. Add your details minna
2. Compute for your semestral grade

```

## ▼ About code

The grade calculator was created with Python operators, functions, and statements. Also, object oriented programming was used to have a better algorithm that allows storing of data into a data structure which is a list. Three classes were created that contain methods that have instructions to do something. The class Menu prints the features of the program while class student\_details

contains the instance variables and the method for getting the inputs of the user about the details of the student. The class Operations holds the functions for the computation of grade and the creation of objects which will be appended into a list afterward. Lastly, when the function for computation was called, it will print the grade of the student with two decimal accuracy and its corresponding emoji.

## References

- [1] W3 Schools (2021). [W3 Schools: Python Data Types](#)
- [2] W3 Schools (2021). [W3 Schools: Python Operators](#)
- [3] EDUCBA (2021). [EDUCBA: Logical Operators in Python](#)
- [4] EDUCBA (2021). [EDUCBA: Identity Operators in Python](#)
- [5] EDUCBA (2021). [EDUCBA: Loops in Python](#)
- [6] EDUCBA (2021). [EDUCBA: Conditional statements in Python](#)