# Welcome to Python Fundamentals

by M.A. Sustento © 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

Variable serves as a container for data [1]. Everything about programming is data and manipulating those data so variables are extremely important in programming. There are different types of data which has various different uses [2]. These data types can be seen in the image below.

Saved successfully! ✕

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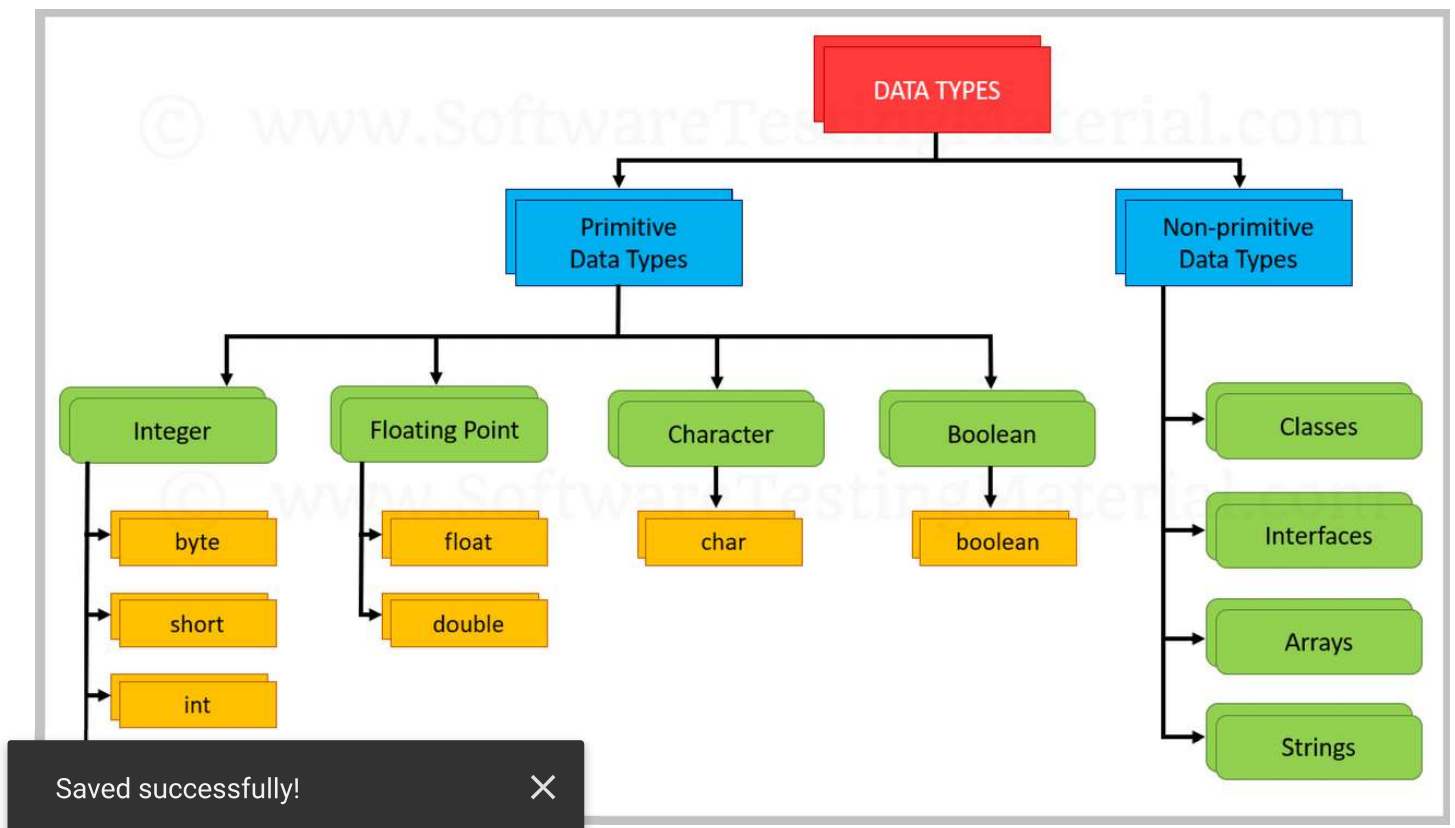
```
a,b = 0, -1
```

```
type(x)
```

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

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

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

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

▾ Operations

Operations is about computing data through the use of mathematics.

Double-click (or enter) to edit

▾ Arithmetic

Arithmetic operators are used to with numbers to do common mathematical operations [3]. The different arithmetic operators can be seen in the image below along with its meanings and examples

Saved successfully! ✕

Arithmetic Operators

Operator	Meaning	Example
+	Addition	4 + 7 ⟶ 11
-	Subtraction	12 - 5 ⟶ 7
*	Multiplication	6 * 6 ⟶ 36
/	Division	30 / 5 ⟶ 6
%	Modulus	10 % 4 ⟶ 2
//	Quotient	18 // 5 ⟶ 3
**	Exponent	3 ** 5 ⟶ 243

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

```
### Addition
S = a+b
S
```

```
    1.5
```

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

```
    31.5
```

Saved successfully!   ✕

```
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

Assignment operators are what is used to assign values to the container of data known as variables [4]. This is to represent values using variables. The operators along with examples can be seen in the image below.

Operator	Example	Equals To
=	a = 10	a = 10
+=	a += 10	a = a+10
-=	a -= 10	a = a-10
*=	a *= 10	a = a*10
/=	a /= 10	a = a / 10
%=	a %= 10	a = a % 10
//=	a //= 10	a = a // 10
**=	a **= 10	a = a ** 10
&=	a &= 10	a = a & 10
|=	a |= 10	a = a |10
^=	a ^= 10	a = a ^10
>>=	a >>= 10	a = a >> 10
<<=	a <<= 10	a = a << 10

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

```
G += a
G
```

```
    2.0
```

```
H -= d
```

```
J *= 2
J
```

```
    4
```

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

```
    4
```

Saved successfully!   ✕

Comparators are exactly what its name says, it is used to compare two values [5]. These


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

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

```
    True
```

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

```
    True
```
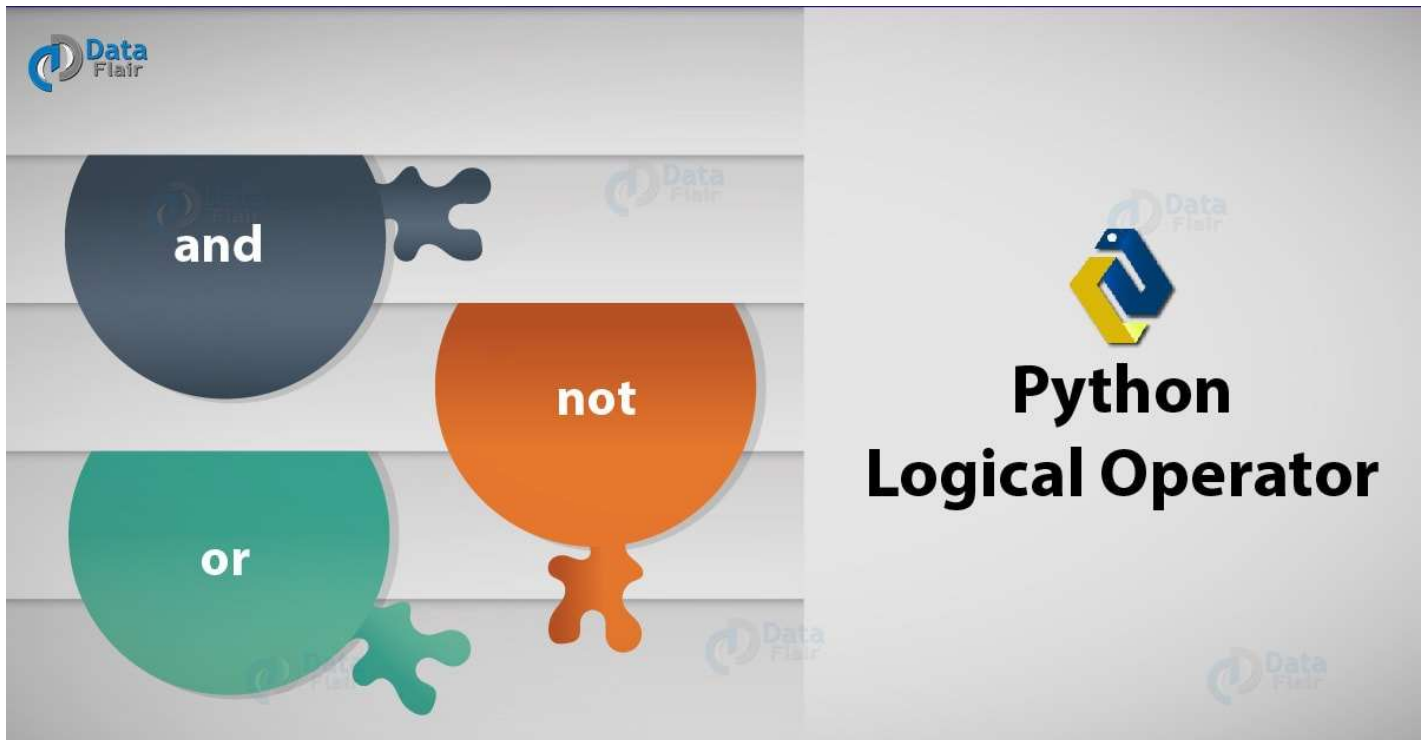
```python
## Inequality
```

```python
t3 = res_1 >= res_2/2
t4 = res_1 <= res_2
t1
```

```
    False
```

▾ Logical

Logical operators are used to combine conditional statements [6]. Logical operators are especially useful in tackling discrete mathematics. The image below provides examples of logical operators.


```
res_1 == true_val
```

```
    True
```

```
res_1 is true_val
```

```
    False
```

```
res_1 is not true_val
```

```
    True
```

```
p, q = True, False
```

```
    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

▾ I/O

I/O stands for input and output. Their names are actually self explanatory. Basically, input is the data that was put in by the user while output is the result that will be displayed by the computer.

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

        Hello World

```
cnt = 1
```

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

        Hello World , Current run count is: 1

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

        Hello World, Current count is: 2

```
sem_grade = 82.243564657461234
name = ""
```

e 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 = float(input("Enter prelim grade: "))
mg = float(input("Enter midterm grade: "))
fg = float(input("Enter finals grade: "))
sem_grade = w_pg*pg + w_mg*mg + w_fg*fg
print("Hello {}, your semestral grade is: {}".format(name, sem_grade))
```

```
        Kimi no nawa: moyk
        Enter prelim grade: 70
        Enter midterm grade: 70
        Enter finals grade: 70
        Hello moyk, your semestral grade is: 70.0
```

Looping Statements

Looping are sequence of instructions that does a specific set of tasks and continues that tasks until the conditions provided is successfully met [7].

While

A while loop can execute a set of statements as long as the condition remains true [8]. This is useful since it is a way to avoid repetition of codes.

```
## while loops
i, j = 0, 10
```

Saved successfully!   ✕

```
        i+=1
```

For

A for loop is used for iterating over sequences such as lists, tuple, set, strings, etc. [9]. This is useful for manipulating sequences while using the least amount of code.

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

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

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

Flow Control

Flow control is about controlling the flow of the program to make it optimized and orderly.

Condition Statements

Conditional statements are used to execute a given task or instruction as long as the given conditions are met

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

```
    Yey
    Hip hip
```

Saved successfully!   ✕

Functions

Function is a block of code that only executes if it was called [10]. Functions are used to organize and easily execute codes.

```
# 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()
```

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

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

▾ Lambda Functions

Lambda functions, also called anonymous functions, can be used to take any number of arguments but can only have one expression [11]. They work the same as regular functions but they can be defined without names. They are especially useful to create one-liners for better optimization of code.

```
x = 4
```

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

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

Saved successfully!   ✕

```
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
'''

def grade_compute(name, course, prelim_grade, midterms_grade, finals_grade): # function
    weight_prelims, weight_midterms, weight_finals = 0.3, 0.3, 0.4 # weights of grades
    happy, lol, sad = "\U0001F600","\U0001F923","\U0001F619" # emojis
    sem_grade = prelim_grade*weight_prelims + midterms_grade*weight_midterms + finals_grade*wei

    ### conditional statements
    if sem_grade > 70: ## if sem_grade > 70,
        print("Hello {} of {}, your semestral grade is: {:.2f} {}".format(name, course, sem_grade
    elif sem_grade == 70: ## if sem_grade == 70,
        print("Hello {} of {}, your semestral grade is: {:.2f} {}".format(name, course, sem_grade
    else: ## if sem_grade is not >= 70,
        print("Hello {} of {}, your semestral grade is: {:.2f} {}".format(name, course, sem_grade

# calling the function with name, course, and grades as parameters
grade_compute(input("Student Name: "), input("Course: "),
              float(input("Prelim: ")), float(input("Midterms: ")),float(input("Finals: ")))
```

```
Student Name: moyk
Course: BS CpE
Prelim: 70
Midterms: 70
Finals: 70
Hello moyk of BS CpE, your semestral grade is: 70.00 🤣
```

About the Code

The code aims to create a semestral grade calculator where users or students could type in their names, course, prelim grade, midterm grade, and final grade. The program should output the semestral grade with 2 decimal points and should display a happy emoji if the grade is greater than 70.00, a laughing emoji if the grade is exactly 70.00, or sad if the grade is below 70.00.

Firstly, the function grade_compute was created with the parameters: name, course, prelims grade, midterms grade, finals grade. These are the parameters since they are the values that will be inputted by the user. The variables for the weight of grades and emojis were set to use for the conditional statements. Afterwards, arithmetic operations were used to compute for the semestral grade. The conditional statements checks the computed semestral grade if it is either above 70.00,

uts a message where the user can see the inputted name, nal grade, semestral grade with a 2 decimal limit, and a corresponding emoji depending on the semestral grade computed. It is important to remember that the function can only run if it was called so the last part of the code is the function being called.

References

[1] W3 Schools (2021) *W3 Schools: Variables*

[2] W3 Schools (2021) *W3 Schools: Data Types*

[3] W3 Schools (2021) *W3 Schools: Arithmetic Operators*

[4] W3 Schools (2021) *W3 Schools: Assignment Operators*

[5] W3 Schools (2021) *W3 Schools: Comparison Operators*

[6] W3 Schools (2021) *W3 Schools: Logical Operators*

[7] W3 Schools (2021) *W3 Schools: Loops*

[8] W3 Schools (2021) *W3 Schools: While Loops*

[9] W3 Schools (2021) *W3 Schools: For Loops*

[10] W3 Schools (2021) *W3 Schools: Python Functions*

[11] W3 Schools (2021) *W3 Schools: Python Lambda*

Saved successfully! ✕