Welcome to Python Fundamentals

by I.J. Timbungco ©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:

- In programming is like a storage that holds values/data types within a program. Unlike Python, some programming language needs to explicitly states the data types of the variables; char, int, float, boolean, string.

The variables must start in a **character**, not in numerical, or in a special characters (@, ., ;, -)

```
x
Adamson
```

Data Types:

- The classification of data, which contains the specific type or range of values. [1] At the point when programs store information in factors, every factor should be assigned particular data type. Some common data types include integers, floating point numbers, characters, strings, and arrays.

```
int x
str Adamson
```

- To know the data type in Python, used a type() method.

```
1
```

```
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

- Is a stepping stone to programming is doing operations. [2] Computers comprehend commands that are written in a specific syntactical form called a "**programming language**".
- Instructing the program to perform various task; **Arithmetic, Assignment Operations, Comparators, Logical, and Input/ Output operations.**
- Most of the time, showing the output is the introductory in doing coding.

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

Arithmetic

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

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

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

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

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

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

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

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

Assignment Operations

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

```
G += a
G
```

```
H -= d
```

```
J *= 2
J
```

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

Comparators

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

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

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

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

Logical

```
res_1 == true_val
```

```
res_1 is true_val
```

```
res_1 is not true_val
```

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

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

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

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

I/O

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

```
cnt = 1
```

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

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

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

```
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))
```

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

```
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))
```

Looping Statements

- Is a programming syntax that do repetition of instructions until a certain condition has met. [3] Loops are accessible at all advanced programming language, however their executions and syntax may contrast. Two of the most common types of loops are the **while loop** and **for loop**.
- Loops can run infinitely, and it's a bad practice. [3] it is important to make sure the **loop will break at some point**. If the condition is never met, the loop will continue indefinitely creating an infinite loop

While

- This type of loop have an **initial condition; As the condition is valid, it keep looping.**

```
while test expression:
  Body expression
```

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

For

- Similar to while loop, this type of loop have a start and end condition. It will **iterate over and over until the end condition has met.**

```
for test expression:
  Body expression
```

```
# 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

Condition Statements

- If statements, is a logical condition that tells the computer what to do. [4] If statements essentially mean: **If something is true, then do something, otherwise do something else.**

```
if <condition>:
  body expression
else:
  body expression
```

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

Functions

- Can be used again in any part of the program. [5] Functions makes the program lesser. As created more programs, functions make it more structured.

```
def name_of_function(parameters):
  Body expression
statements()
```

```
# 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

- It is like the Function but, a more simpler one. It is a one liner code of Function. [6] A lambda function can have numerous of parameters, but the function can only contain one expression. In addition, a lambda is written in a single line of code and can likewise be used right away.

```
lambda p1, p2: expression
```

```
x = 4
```

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

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

```
'''
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"
```

```
def student_name(s_name):
  s_name = str(input("Please enter your name(Surname, First name, Middle Initial): "))
  print(f'Hello, {s_name.upper()}')

def course_name(c_name):
  c_name = str(input("\nPlease enter your course: "))
  print(f'Your course is: {c_name.upper()}')
```

```
def grade_calc(pg,mg,fg):
  w_pg, w_mg, w_fg = 0.3,0.3,0.4
  pg = float(input("\nEnter your Prelim Grade: "))
  mg = float(input("Enter your Midterm Grade: "))
  fg = float(input("Enter your Final Grade: "))
  sg = w_pg*pg + w_mg*mg + w_fg*fg
  sg = round(sg,2)
  if sg > 70.00:
     print(f"Your semsetral grade is: {sg}")
     print("\U0001F600")
  elif sg == 70.00:
     print(f"Your semsetral grade is: {sg}")
     print("\U0001F923")
  else:
     print(f"Your semsetral grade is: {sg}")
     print("\U0001F619")
```

```
student_name("")
course_name(" ")
grade_calc(" "," "," ")
```

```
Please enter your name(Surname, First name, Middle Initial): timbungco, ian jude j.
Hello, TIMBUNGCO, IAN JUDE J.

Please enter your course: bs cpe
Your course is: BS CPE

Enter your Prelim Grade: 70
Enter your Midterm Grade: 70
Enter your Final Grade: 70
Your semsetral grade is: 70.0
```

References:

[1] techterms (2007). ***techterms : Data Type***

[2] hackr.io (2020). ***hacker.io: what is programming***

[3] techterms (2016). ***techterms : Loop***

[4] ThinkAutomation (2019). ***ThinkAutomation : A Beginner's guide to IF's statements***

[5] Programiz (2021). ***Programiz : Python Functions***

[6] guru99(2021). ***guru99 : Python Lambda Functions***