Welcome to Python Fundamentals

by J.Parco ©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 is the place holders or a way to store data values; to use the variable in python; the user must declare a variable name followed by the equal sign "=" and the data, the data can have different forms such as integers for whole numbers, float for fractional numbers, a string for word or sentence and boolean for declaring True or False.

The code snippets below shows how to use variable, showing the type of data, and changing data type using casting.

```
1 ## Declaring Variables
2 x = 1
3 a,b = 0, -1
```

```
1 ## Showing the type of data
2 type(x)
```

```
    int
```

```
1 ## Showing the type of data ##
2 y = 1.0
3 type(y)
```

```
    float
```

```
1 ## Casting or changing data types
2 x = float(x)
```

```
3 type(x)
```

```
    float
```

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

```
    str
```

```
1 ## Limitation of Casting
2 u_int = int(u)
3 u_int
```

```
    ---------------------------------------------------------------------------
    ValueError                                Traceback (most recent call last)
    <ipython-input-6-971f9fdf88ba> in <module>()
          1 ## Limitation of Casting
    ----> 2 u_int = int(u)
          3 u_int

    ValueError: invalid literal for int() with base 10: 'one'
```

SEARCH STACK OVERFLOW

Operations

Arithmetic

Python can perform arithmetic such as addition, subtraction, multiplication, division, floor division, exponential, and modulo by the way, floor division disregards the decimal while modulo gets the remainder of two values.

Below is the code snippets for python arithmetic.

```
1 ## Declaring Variables
2 a,b,c,d = 2.0, -0.5, 0, -32
```

```
1 ## Addition
2 S = a+b
3 S
```

```
    1.5
```

```
1 ## Subtraction
2 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

Assignment operators are the same as arithmetic but with a shorter code, but use it with caution because a simple assignment operator can change the variable value if the code runs a second time or more. To mitigate this, the user must declare the value again at the start of the code so that the variable will not override the value again.

Below is a sample of Assignment operators and it drawbacks and solution.

```
## Declaring Variables
G, H, J, K = 0, 100, 2, 2
```

```
## When its run a second time it will add the vaule again
G += a
G
```

```
2.0
```

```
## Solution for the problem
H = 100
H -= d
H
```

```
132
```

```
## Using Multiplication
J *= 2
J
```

```
4
```

```
## Using Exponentiation
K **= 2
K
```

```
4
```

Comparators

Comparators or Comparison Operator is a build-in function for Pyhton that compares two values and returns a boolean or True/False.

Below is a sample code snippet for comparison operator.

```
## Declaring Variables
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
t1
```

```
False
```

Logical

Using Logical Operators uses a conditional statement to output a boolean or True/False.

Below is a code snippet of logical operators.

```
res_1 == true_val
```

```
True
```

```
## Using "is" output's "False" because the file type is not he same
res_1 is true_val
```

```
False
```

```
## Using the "not" oprator
res_1 is not true_val
```

```
True
```

```
## Using "and" operator will only outputs "True" id both values are "True"
p, q = True, False
conj = p and q
conj
```

```
False
```

```
## Using "or" operator only outputs "True" if one of the values is "True"
p, q = True, False
disj = p or q
disj
```

```
True
```

```
## Using the negation of "and" or "nand" operator which is the opposite of "and" operator
```

```
## Using the negation of "and" or "nand" operator which is the opposite of "and" operator
p, q = True, False
nand = not(p and q)
nand
```

```
True
```

```
## Using "exclusive or" operator outputs "True" only when inputs differ
p, q = True, False
xor = (not p and q) or (p and not q)
xor
```

```
True
```

I/O

Inputs and Outputs for Python is the most basic function that will always be used, using inputs to declare a variable and print function to output the data. To integrate inputs and outputs, 3 ways are using the typical way, f function, and format function.

bellow is a code snippet on how to use inputs and outputs for Python.

```
## Outputing a String
print("Hello World")
```

```
Hello World
```

```
## Declaring a Variable
cnt = 1
```

```
## Typical way of integrating Variables and Outputs
string = "Hello World"
print(string, ", Current run count is:", cnt)
cnt += 1
```

```
Hello World , Current run count is: 1
```

```
## Using f Function to integrate Variables and Outputs
print(f"{string}, Current count is: {cnt}")
```

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

```
## Using format Function to integrate Variables and Outputs
sem_grade = 82.243564657461234
name = ""
print("Hello {}, your semestral grade is: {}".format(name, sem_grade))
```

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

```
## Using format Function with decorators to integrate Variables and Outputs
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.
```

```
## Inputing an integeger for a Variable
x = input("enter a number: ")
x
```

```
    enter a number: 69
    '69'
```

```
## Combining Outputs and Inputs
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: SPongebob
    Enter prelim grade: 12
    Enter midterm grade: 99
    Enter finals grade: 50
    Hello SPongebob, your semestral grade is: 53.3
```

Looping Statements

A looping statement is a group of function where a task or a statement runs 2 or more times.

While

While loop is a function that will only run if the given expression is True.

```
## while loops
```

```
## 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 loop will run until the number of the iterating sequence is finished; the sequence can be a number or a list.

```
# for(int i=0; i<10; i++){
# printf(i)
# }
## Using Integer in a for loop
i=0
for i in range(10):
    print(i)
```

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

```
## Using list in a for loop
playlist = []
print('Now Playing:\n')
for song in playlist:
    print(song)
```

```
Now Playing:
```

Flow Control

Condition Statements

Condition statement has three parts "if", "elif", and "else"; "if" statement run if the condition is True, "elif" statement will run if the "if" statement is False and "elif" statement is True while "else" is a catch-all statement where all expression didn't meet.

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

```
Yey
Hip hip
```

Functions

A function is a block of code that will only run if the function is called

```
## Setting up a Function
def delete_user (userid):
    print("Successfully deleted user: {}".format(userid))

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

```
## Declaring Variables and Running the functions with arguments
userid = 0
delete_user(0)
delete_all_users()
```

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

```
## Having A function with parameters
```

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

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

Lambda Functions

Lambda function is used to have a function inside a statement and can only have one expression or operation.

```
## Declaring Variables
x = 4
```

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

```
31
```

```
## Using Lambda to shorten the code
g = lambda x: 2*(x*x)-1
print(g(x))
```

```
31
```

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"
```

```
def compute_sem_grade():
    name = input("Enter Name: ")
    subject = input("Enter Subject: ")
```

```
    prelim = float(input("Enter Prelim Grade: "))
    midterms = float(input("Enter Midterm Grade: "))
    finals = float(input("Enter Finals Grade: "))
    semestral_grade = 0.3*prelim + 0.3*midterms + 0.4*finals
    if semestral_grade > 70: emotion = "\U0001F600"
    elif semestral_grade == 70 : emotion = "\U0001F923"
    else: emotion = "\U0001F61E"
    print("Hello {}, your semestral grade is: {}{}".format(name, semestral_grade, emotion)

compute_sem_grade()
```

```
Enter Name: Jovian
Enter Subject: Universe
Enter Prelim Grade: 5
Enter Midterm Grade: 78
Enter Finals Grade: 90
Hello Jovian, your semestral grade is: 60.9😞
```