Welcome to Python Fundamentals

Canoza, Cherrylyn S.

BS CpE

Numerical Methods | 58015

click me : https://github.com/cherrylyncanoza/Numerical-Method/tree/main/Coding%20Activity%201%20Revisiting%20Python

---

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

Variables are the primary data of the program and it is the storage of values which is being manipulated all through out the codes. Data Types are the types of variables or the specific type of values that are inside a variable name.

```
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

Operations are used to perform different operators for the variables not only mathematics-related but also with some logics and other functions for the whole code. It consists of different symbols depending on their uses and there are five types of operations which are the Arithmetic, Assignment Operations, Comparators, Logical and, I/O.

Arithmetic

The Arithmetic Operation are basically the mathematical symbols that are usually used which are the Addition (+), Subtraction (-), Multiplication (*), Division (/), Modulo (%), Parenthesised operation ((a+b)/c) and Exponentiation (**) and Floor Division (//).

```
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

Assignment Operators are basically consist of equal sign (=) which is called the assignment. It is used to assign values for the variables, also, arithmetic operators are mixed in the assignment or the equal sign to assign new value for a certain variable.

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

```
G += a
G
```

```
2.0
```

```
H -= d
H
```

```
196
```

```
J *= 2
J
```

```
4
```

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

```
4
```

Comparators

Comparators have the function to compare two or more variables or values. It is more like a boolean since it returns a value of either true or false depends on the condition.

```
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

Logical Operators are used to evaluate most likely two or rarely more expression or variable and gives a boolean result since it is more on True or False result depending on what is being evaluated.

```
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
```

I/O

It is the most essential part of the code for the input is the one that being processed inside the code with different methods and functions that are being executed and the output is the result of overall

execution.

```
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 = ""
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
```

```
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

Looping statements are set of codes that will execute multiple times depending on the condition and depends also on how you want it to be, for it has two types which are the While and For loop for Python language that has the same output but different in method of execution.

While

This statement has the function to create a loop of which this is a control flow statement that will only stop until a certain condition has met. It has initialized specific value and conditions until when the loop ends.

```
## 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

It is another control statement that will loop in a pre-determined time. There is also an initialized value and will only stop until the condition of assigned range of the desired loop has met.

```
# 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

These statements are used for the whole flow of decision-making and process of the codes which controls what will be the output of the executed code when conditions are being satisfied.

Condition Statements

This is one of the most important statement to be kept in mind for it possess different options for decision-making in some part of the code. There are also conditions inside that leads to different path or different possible ways dependent on where the condition will be true or satisfied.

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

```
Yey
Hip hip
```

Functions

Functions contain segmemts of codes with different set of methods with pre-determined parameters or arguments to be accomplished or executed only when it is called.

```
# 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

Lambda is a nameless and usually a one-liner function that evaluates and returns a result immediately without calling the return command.

```
x = 4
```

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

```
 31
```

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

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

```
31
```

```
'''
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"
```

```
# Function compute_sem_grade that computes for the sem_grade
# Accepts the parameters pg(prelim grade), mg(midterm grade) and fg(final grade)
def compute_sem_grade(p_grade, m_grade, f_grade):

    # Fixed weight of prelim, midterm and finals grades
    w_pg, w_mg, w_fg = 0.3, 0.3, 0.4

    # Computes for the equivalent sem grade
    sem_grade = p_grade*w_pg + m_grade*w_mg + f_grade*w_fg

    # Check if the grade is failed, passing or higher than passing
    # Returns the name, course and sem_grade with an emoji
    if (sem_grade > 70.00):
        return "Hello {} of {}, your semestral grade is: {:,.2f} \U0001F600".format(name, cou
    elif (sem_grade < 70.00):
        return "Hello {} of {}, your semestral grade is: {:,.2f} \U0001F61E".format(name, cou
    else:
        return "Hello {} of {}, your semestral grade is: {:,.2f} \U0001F923".format(name, cou

# Asks user to input name and course
name = input('Enter your name: ')
course = input('Enter your course: ')

# Asks user to input prelim, midterm and finals grade
pg = float(input('Enter Prelim grade: '))
mg = float(input('Enter Midterm grade: '))
fg = float(input('Enter Final grade: '))

# Calls the function compute_sem_grade
result = compute_sem_grade(pg, mg, fg)

# Prints the returned result from the function
print(result)
```

```
Enter your name: Cherrylyn Canoza
Enter your course: Computer Engineering
Enter Prelim grade: 92
Enter Midterm grade: 95
Enter Final grade: 98
Hello Cherrylyn Canoza of Computer Engineering, your semestral grade is: 95.30 😀
```

The code works as follows. First is that the program will ask the user to input his/her name and course. Then the user will be prompted to input his/her prelim, midterm and finals grade. The program will then call the function compute_sem_grade and will pass the values of prelim, midterm and finals grade as the parameter. Inside the function is an already fixed weights of each term grade which are 0.3, 0.3 and 0.4 respectively. The function will then compute for the semestral grade. The sem grade would then be checked if it is failed, passing or higher than the passing grade. The function will finally return the correct string to show which contains the name, course and sem grade of the student. The difference between the 3 possible returned strings are the emojis displayed. Sad face for a failing grade, laughing face for a passing grade and a smiley face for a higher than passing grade. The returned value will then be printed for the user to see.