**Assignment-2**

**Python - Variable Names**

**Variable Names**

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

1. A variable name must start with a letter or the underscore character
2. A variable name cannot start with a number
3. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
4. Variable names are case-sensitive (age, Age and AGE are three different variables)

**Example**

***Legal variable names***:

myvar = "John"

my\_var = "John"

\_my\_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

***Illegal variable names:***

2myvar = "John"

my-var = "John"

my var = "John"

**Multi Words Variable Names**

Variable names with more than one word can be difficult to read.

There are several techniques you can use to make them more readable:

***Camel Case***

Each word, except the first, starts with a capital letter:

myVariableName = "John"

***Pascal Case***

Each word starts with a capital letter:

MyVariableName = "John"

***Snake Case***

Each word is separated by an underscore character

my\_variable\_name = "John"

**Some of the important points to keep in mind:-**

* Variables are referred to "envelop" or "buckets" where information can be maintained and referenced. Like any other programming language Python also uses a variable to store the information.
* Variables can be declared by any name or even alphabets like a, aa, abc, etc.
* Variables can be re-declared even after you have declared them for once
* In Python you cannot concatenate string with number directly, you need to declare them as a separate variable, and after that, you can concatenate number with string
* Python constants can be understood as types of variables that hold the value which can not be changed. Usually Python constants are referenced from other files. Python define constant is declared in a new or separate file which contains functions, modules, etc.
* Types of variables in Python or Python variable types : Local & Global
* Declare local variable when you want to use it for current function
* Declare Global variable when you want to use the same variable for rest of the program
* To delete a variable, it uses keyword "del".

Many values to multiple variables:

Python allows you to assign values to multiple variables in one line:

x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)

One value to multiple variables:

you can assign the *same* value to multiple variables in one line:

x = y = z = "Orange"  
print(x)  
print(y)  
print(z)

Unpack a collection:

If you have a collection of values in a list, tuple etc. Python allows you extract the values into variables. This is called *unpacking*.

fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits  
print(x)  
print(y)  
print(z)

Output variables:

The Python print statement is often used to output variables.

To combine both text and a variable, Python uses the + character:

x = "awesome"  
print("Python is " + x)

o/p- python is awesome

For numbers, the + character works as a mathematical operator:

x = 5  
y = 10  
print(x + y)

o/p- 5 + 10 = 15

If you try to combine a string and a number, Python will give you an error:

x = 5  
y = "John"  
print(x + y)

o/p- error: unsupported operand type(S) for +: ‘int’ and ‘str’

Global variables:

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

Create a variable outside of a function, and use it inside the function

x = "awesome"  
  
def myfunc():  
  print("Python is " + x)  
  
myfunc()

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

Create a variable inside a function, with the same name as the global variable

x = "awesome"  
  
def myfunc():  
  x = "fantastic"  
  print("Python is " + x)  
  
myfunc()  
  
print("Python is " + x)

The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

If you use the global keyword, the variable belongs to the global scope:

def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

Also, use the global keyword if you want to change a global variable inside a function.

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = "awesome"  
  
def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

Python Comments

* Comments can be used to explain Python code.
* Comments can be used to make the code more readable.
* Comments can be used to prevent execution when testing code.

Creating a comment:

1. Comments starts with a #, and Python will ignore them:

#This is a comment  
print("Hello, World!")

2) Comments can be placed at the end of a line, and Python will ignore the rest

print("Hello, World!") #This is a comment

3) Comments does not have to be text to explain the code, it can also be used to prevent Python from executing code:

#print("Hello, World!")  
print("Cheers, Mate!")

Multiline comments:

1. Python does not really have a syntax for multi line comments.

To add a multiline comment you could insert a # for each line:

#This is a comment  
#written in  
#more than just one line  
print("Hello, World!")

1. Or, not quite as intended, you can use a multiline string.
2. Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

"""  
This is a comment  
written in  
more than just one line  
"""

print("Hello, World!")

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.