1.4 Playing with Python

Playing with PYTHON!

Congratulations! You have successfully installed python. Let’s explore python a little bit. Go to command prompt and type python, this should take you to the python command prompt. Now, what?

(You can also type and execute below codes in IPython)

Let’s try to execute following instructions (press enter after typing an instruction to execute it) and fill in the output.

S.No	Instruction	Explanation	Output
1	1 +2	Simple addition of two numbers	
2	2+2/2	Python Interpreter follows BODMAS rule .	
3	2*10	Multiply two with 10	
4	Hi	Hi is not a key word in python and is not declared as place holder(variable). Expect an error.	
5	"hi" 'hi'	Hi inside double/single quote is string literal i.e. string constant and is acceptable in python. String is array of characters.	
6	"hi"*2	Telling python to print “hi” two times. Observe, in instruction 3, * acts as multiplier but in instruction 6, it acts as repeater i.e. repeats hi twice. This property of is known as operator overloading i.e. operator behaves differently when different types of arguments are passed to them.	
7	"hi"+2	Trying to add a number to a string. Expect an error .	
8	"hi"+"2"	+ operator concatenates two string.	
9	2**4 Pow(2,4)	Power operator	
10	7%3	Modulus operator which returns reminder	

11	5/2	Dividing two integers will give another integer	
12	5.0/2	Dividing a number having decimal (Floating point number) with an integer	
13	5.0//2	**Floor (https://en.wikipedia.org/wiki/Floor_and_ceiling_functions)** division	
14	[1,2,3]*2	A list is replicated twice.	

What is Happening? when you type “python” in command prompt, python interpreter gets invoked and current command prompt changes to python command prompt i.e. C:\Users\username> to >>> in windows 10. This indicates that python interpreter is running and ready to take instruction. When you type an instruction like 2+2, python interpreter executes it and prompt the result of evaluation. If an instruction is not correct like 4th and 7th then interpreter returns an error message.

It is fun to work with the command line but things get messy when you have to type multiple lines of instruction. So, it is advisable to type your code in text pad and execute via python command line.

How to execute Python code written in text pad?

1. Open a text pad.
2. Copy and paste following code in the text pad.

```
varA=float(raw_input("Please enter Number A: "))
varB=float(raw_input("Please enter Number B: "))
```

```

if varA>varB:
    print "A is greater than B"
elif varB>varA:
    print "B is greater than A"
else:
    print "A and B are equal"

print "A + B =",varA+varB
```

3. Save as “myFirstCode.py”. Note: the extension of the file it is not .txt it is **.py**. All python files should have a .py extension.
4. Come out of python command prompt by pressing ctrl+z. If you are not in python command prompt this step should be skipped.

5. Type following command in the command prompt: python “C:\myCodes \myFirstCode.py”

Note: syntax of the command is python “full path of the file to be executed\fileName.py”

Indentation:

1. In python, there is no begin or end and no curly braces to mark where the function or code block starts and ends.
2. Colon (:) and indentation of code act as a delimiter.
3. A Code block is referred to collection of instructions which are executed together eg. Functions, if statements, while and for loops etc.

Pay attention at the indentation of the code given above. Indention is very important in python, they represent block and scope of a code segment.

If you try to execute following code, without proper indentation, then python interpreter will throw an error. Let’s try it!

```
varA=float(raw_input("Please Inter Number A: "))
varB=float(raw_input("Please Inter Number B: "))
```

```


if varA>varB:
print "A is greater than B"
elif varB>varA:
print "B is greater than A"
else:
print "A and B are equal"

print "A + B =",varA+varB
```

You will get following error

```
print "A is greater than B"

^

IndentationError: expected an indented block
```

Code	Output

`print "Hello World"`
`print "indentation messes with my code"`	

```
if 1>0:
   #start of if block
   print "Hello World"
   print "indentation messes with my code"
   print "if creates a new scope and demands indentation"
   #end of if block
else:
   #start of else block
   print "same is the case with else"
   #end of else block
```

```
if 1>0:
    #start of if block
    print "Hello World"
    print "indentation messes with my code"
    if 2>1: #there is no block of code belonging to this if
    print "if creates a new scope and demands indentation"
    #end of if block
else:
    #start of else block
    print "same is the case with else"
    #end of else block
```

```
if 1>0:
    #start of if block
    print "Hello World"
    print "indentation messes with my code"
    if 2>1:
        print "if creates a new scope and demands indentation"
    print "This print statement belongs to outer if 1>0 "
    #end of if block
else:
    #start of else block
    print "same is the case with else"
    #end of else block
```

Don’t worry if you do not get nuances of indentation at this time, when you practice and write code by yourself you will develop intuition and logic about where to put indentation. Just keep one thing in mind “code belonging to same code block should be at same level of indentation”.

Code Block:

They are a collection of instructions executed together like, all the statements followed by a declaration of if.

No	

```
1   input=[3,4,6,1,9,2,8] #variable declaration and initialization : input list
2
3   for noOfPass in range(1,input.__len__()-1):# outer for loop
4      #print temp
5      for pos in range(0,input.__len__()-noOfPass):#inner for loop
6         if input[pos]>input[pos+1]:
7            temp=input[pos]
8            input[pos]=input[pos+1]
9            input[pos+1]=temp
10
11  print input
12  print temp, noOfPass
13
```

Above code is for bubble sort, which sorts the given input array in ascending order. Don't worry about logic as of now, pay attention to the indentation and blocks of code.

Block 1: belongs to outer for loop and its scope is till line input[pos+1] =temp

Block 2: Belongs to inner for loop and its scope is till input[pos+1] =temp

Block 3: Belongs to if condition and its scope is till line input[pos+1] =temp

Block 4: only contains two line of the print statement.

Block 2 and Block 3 are under the scope of Block 1. While Block 4 is not in the scope of Block 1 or Block 2 Or Block 3.

What is a Scope?

The concept of scope is pivotal to almost all programming language. Without understand this concept, one cannot write a functional and executable code. It tells us, where to place which variable and from where can these variables be accessed.

Well, a scope can be referred to as the scope of a variable or scope of a statement or a function, like for loop, if block etc**. In broad term scope means visibility or influence.**

In above code of bubble sort, the scope of variable "input" is until the end of the entire code i.e. we can access input anywhere in the code after it's declaration.

2. The Scope of variable "temp" is from line 7 to line 13 i.e. after its declaration at line 6 we can access it anywhere between line 8 to 13 . But if we try to access it at line 4 or anywhere before line 7, python will throw an error. Uncomment line 4 by removing # and run above code.

3. Similarly, the scope of variable “noOfPass” is from line 3 to 13.

What are Variables?

Variables are placeholder or name given to any entity. Well, what do they hold? They can hold an integer, character, String, List etc. A variable can hold different values or even different types of values in the same code. In Python, the type of variable is decided by the value they hold.

We assign value to a variable via “=” equals operator

```
myVariable=211 #assign 211 to the variable

myVariable=[1,2,3] #assign a list containing three integers to the variable
```

Rules for variable name:

1. A variable name cannot start with number or any special symbol except underscore. It can start with any of the 26 alphabets of English e.g. “_myFirstVariable” or “myFirstVariable”
2. A variable name can contain numbers at any place except at the starting place. e.g. “my2Variable” ,”myVariable3”
3. Special symbol, except underscore “_”, cannot appear in the variable name at any place.
4. Variable names are case sensitive i.e. “myVariable” and “MYvariable” represents two different variables.

Variable Name	Correct or not	Comment
Input=2		Variable type is integer
Input=2.0		Variable type is Float
_input=”my Name”		Variable type is String
1input =true		Variable type is Boolean
Input1=false		Variable type is Boolean
crazy@variable=[]		Variable type is list
Crazy1variable={}		Variable type is dictionary
myChar=’a’		Variable type is String

What is a constant?

Constants are entities value of which cannot be changed i.e. they cannot be on the left-hand side of “=” equals operator.

```
2=myVariable # here 2 is a constant and we cannot assign any value to it.
```

```
"I am Who I am."=myVariable # Left hand side is a string literal, a constant, and we
cannot assign any value to it.
```