# Python Basics

***Python Introduction:***

1) Python was developed by Guido van Rossum in the late 1980s. 2) Python is an interpreted, object oriented , high level programming language. 3) Python has easy to learn language and syntax. 4) There is no compilation stage in python. The code is directly converted to machine understandable. 5) Supported on multiple platforms and has extensive standard libraries like, TensorFlow, Scikit-Learn, Numpy, Keras, Scipy.

6) The current latest version of python is 3.7.2

**Print Statement**

In [24]:
```
# A simple print statement is
print ('hello world') #whatever message you want to print is given in quotes
# if you want to print the varible value, pass the name of the variable with
print_ = 'python demo' #print_ is a varibale having value 'python demo'
print (print_)
```

```
hello world
python demo
```

**Pyhton Variables and Keywords:**

Keywords are python specific words which has specific meaning to python interpreter. for eg (True, False, None, if, else, elif, def , for, while etc)

Variables defines the memory location where any value assigned to the variable can be stored.

Variable name can start with [A-z], [a-z] followed by digits [0-9] and underscore. Digits can't be starting of the name. And no special characters are allowed for naming the varaiable

Basically there are 4 types of literals in python

1) Numeric 2) String 3) Boolean 4) Collections

Numeric Literals can be either integers, float or complex

***Numeric Literals***

In [9]:
```
x = 12 # stores an integer value
type (x) #type(): use to check type of the value that is stored
```

Out[9]:
```
int
```

In [11]:
```
x = 12.5 #stores a decimal value
type(x)
# In above eg we have used x to store 12 and here also we used the same x to
# is now modified to 12.5
```

Out[11]:
```
float
```

In [12]:
```
#In python we store complex values as a+bj
comp_ = 3+4j
type(comp_)
```

Out[12]:
```
complex
```

***Numeric Operators:***

***Arithmatic operators:***

In [14]:
```
#Arithmatic Operators: (+,-,*,/,//,%)
print (3+4) #addition
print (5-2) #Subtraction
print (6*2) #Multiplication
print (8/4) #Division
print (10//3) #Floor division (quotient is rounded off to nearest integer)
print (14%3) #Returns the remainder of the division
```

```
7
3
12
2.0
3
2
```

***Comparison operators:***

In [20]:
```
#comparision operators:
#comparision operators compare the two values and return boolean (True and F
a = 9
b = 3
print (a > b) #greater then
print (a < b) #less then
print (a >= b) #greater then equal to
print (a <= b) #less then equal to
print (a != b) #Not equals to
print (a == b) #equals to
```

```
True
False
True
False
True
False
```

***Assignment operators:***

In [22]:
```
# Assignment Operators:
# it assigns the value to whatever variable on the left side
a = 5 #assigning value to variable a
b = 6 #assigning value to variable b
a += b #assigning value to a after addition of b to it
print (a)
#Similarly we have a-=b , a*=b , a/=b, a//=b and a%=b
```

```
11
```

***Logical operators:***

In [30]:
```
#Logical operators: (and, or, not)
#and returns true when all conditions with and are true for eg:
a = 5
b = 6
c = 'demo'
print (a!=b and c=='demo') #since both conditions are True it returns True
print (a==b and c=='demo') #since one condition is false it returns False
#or return true if any of the conditions given is True for eg:
print (a==b or c=='demo') #since any of the conditions given is true it retu
#not returns the inverse of true or false for eg
print (not(True)) #inverse of true is false
```

```
True
False
True
False
```

**Operator precedence:**

In [ ]:
```
#In a given sequence of Expression operator precedence will be as follows:
** #exponent
*, /, //, % #multiply, divide, floor division, modulous
+, - #Addition and subtraction
<=, <, >, >= #comparison operators
=, %=, /=, //=, -=,+=, *=, **= # Assignment operators
and or not # Logical operators
```

**String Operators:**

In [15]:
```
#A string is simply defined in single quotes or double quotes as follows:
x = '   This is a demo string to show string operations    '
print (x.lower()) #Converts string to lower case
print (x.upper()) #converts string to upper case
x.lstrip() #remove unwanted spaces from the left side
```

```
this is a demo string to show string operations
THIS IS A DEMO STRING TO SHOW STRING OPERATIONS
```

Out[15]:
```
'This is a demo string to show string operations    '
```

In [7]:
```
x.rstrip() #remove unwanted spaces from the right side
```

Out[7]:
```
'   This is a demo string to show string operations'
```

In [9]:
```
x.strip() #rmove unwanted spaces from both the sides
```

Out[9]:
```
'This is a demo string to show string operations'
```

In [16]:
```
print (x.startswith('   ')) #checks whether the string starts with given init
print (x.endswith(' ')) #checks whether string ends with given initials and
```

```
True
True
```

In [17]:
```
x.find('demo') #finds the given string in the whole string and returns its s
```

Out[17]:
```
13
```

In [18]:
```
x.count('string') #counts the number of appeareances of a given string
```

Out[18]:
```
2
```

In [20]:
```
x.replace('demo','actual') #replaces old string with the new string
```

Out[20]:
```
'   This is a actual string to show string operations    '
```

In [24]:
```
p = x.split() #split the give string and returns a list
```

In [25]:
```
'/'.join(p) #joins the splittede  elements by the delimiter given
```

Out[25]:
```
'This/is/a/demo/string/to/show/string/operations'
```

In [34]:
```
#Slicing: (Start:stop:step)
#Gives a part of a string it takes three values, bascially the start is the
#Stop is the ending point of part if not given default is the last of the st
#value of step by default is 1
y = 'This is a demo string to show string operations'
#    0123456789
print (y[0:])
print (y[0:5])
print (y[-10:])
print (y[::-1])
```

```
This is a demo string to show string operations
This
operations
snoitarepo gnirts wohs ot gnirts omed a si sihT
```

**Comments in python**

In [ ]:
```
#In python comments are given by # and ''' '''.
#for single line comment we use (#)
#for multi-line comments we use (''' ''')
#Comments are generally given to add some message with the code. Comments ar
```

***Practise Questions:***

In [ ]:
```
# print the result of 10 + 5
# calculate the percentage 20 / 25

# for string 'I am learning python'
## convert whole string to upper case
## find the index of the start of 'python'
## count the number of occurance of 'python'
## split the string with spaces.
## join the above string (splited) with '/'
## slice only the 'python' part
```

***Answers:***

In [11]:
```
x = 'I am learning python'
x.upper()
x.find('python')
x.count('python')
z = x.split(' ')
'/'.join(z)
x[-6:]
```

Out[11]: 'python'

In [ ]:
```

```