What is PYTHON ???

Python is a computer programming language created initially by Guido Van Rossum

It is a very high level programming language

Many consider it the easiest programming language to learn and to program on

It is very useful as an easy and quick way to program anything from small little scripts to large scale client/server systems

Programmers tend to like Python because of its clean and logical syntax, which shortens development cycles tremendously

How Python works ???

Python programs run with the aid of an “interpreter”

The interpreter can be run directly in what is called the “Interactive Command Line”

Python commands can be typed directly into this session and they are processed immediately showing the results of the command

A program is a sequence of instructions that specifies how to perform a computation.

```
>>>print “Hello World”
Hello World
```

What is a variable?

A variable is a name (a word) that refers to a certain value. We can use almost any word and assign a value to it. For example:

```
>>>Hi = “Hello World”
>>>print Hi
Hello World

>>>Hi = 3.0
>>>print Hi
3.0
```

Expressions

At the interactive command prompt one can type expressions and see their result interactively. “An expression is a combination of variables, operators and values that represents a single result value”i. For example:

```
>>>Num + 2
5.0
>>>((5-1) + Num)*2
14.0
```

We can assign the value of an expression to a variable. Such as:

```
>>>Numb= ((5-1) + Num)*2
>>>Numb
14.0
```

Numbers

Numbers can be Integers or Floating Point

```
>>>7/2
3    (Integer)

>>> 7.0/2.0
3.5 (Floating Point)
```

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder (e.g. 9%2=1; 8%2=0)
**	Power (4**2=16)

```
from math import *
```

Function	Description
acos(x)	Return the arc cosine of x
cos(x)	Return the cosine of x
log(x)	Return the natural logarithm of x
sqrt(x)	Return the square root of x
tanh(x)	Return the hyperbolic tangent of x

Strings

A string is an ordered collection of characters, used to store and represent text-based information".

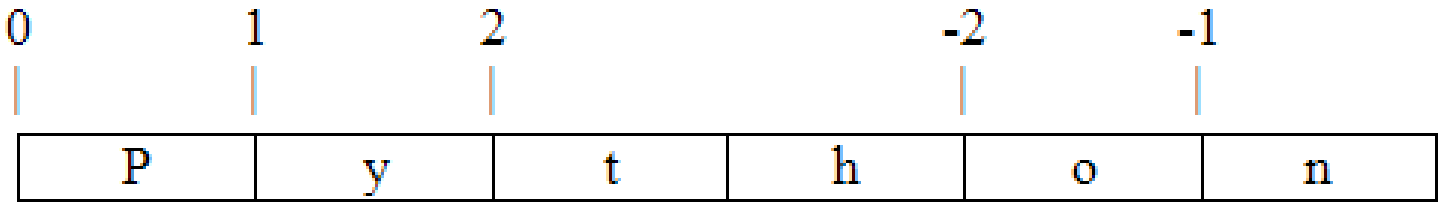
Python strings can be enclosed in double, single, or even triple (for multi-line strings) quotes.

>>>'Hello World'
'Hello World'
>>>"Hello 'Hello World'"
"Hello 'Hello World'"
'''This is a longer string that spans
... several lines, doesn't say anything
... meaningful, and ends here'''

Strings are “ordered”

We can access Strings' components by position

Each character in a string has an index position before it



To get the characters located at certain indexes we can use their index number inside brackets.

To get the character located at the beginning of the string we can use:

```
>>>”Python”[0]
‘P’
```

Or if we are using a variable:

```
>>>Hi = “Hello World”
>>>Hi[0]
‘H’
>>>Hi[1]
‘e’
>>>Hi[-1]
‘d’
```

We can also get “slices” of a string by specifying the first and last index of the slice separated by a colon.

```
>>>Hi[1:7]
‘ello W’
>>>Hi[4:-2]
‘o Wor’
```

The left or right bounds of the string can be specified in a slice by leaving the index blank. For example:

```
>>>Hi[:5]
'Hello'
>>>Hi[6:]
'World'
>>>Hi[:]
'Hello World'

When adding a string to a string
>>>Hi + " Today"
'Hello World Today'
>>>Hi[:5] + " You"
'Hello You'
```

Multiplication

```
>>>Hi[:6] * 5
'Hello Hello Hello Hello Hello '
```

Strings also have built-in methods such as the following:

```
>>>”hello”.capitalize()
‘Hello’
>>>”HELLO”.lower()
‘hello’
>>>’hello’.upper()
‘HELLO’>>>’Hello World’.replace(” “, “-”)

‘Hello-World’
>>>’ Hello World ‘.lstrip()
‘Hello World ‘
>>>’ Hello World ‘.rstrip()
‘ Hello World’
>>>’ Hello World ‘.strip()
```

Lists

Lists are ordered sequences.
Lists are mutable sequences.
Lists can contain any type of object: numbers, strings, even other lists

To define a list, we simply have to enclose it in square brackets using commas as item delimiters:

```
>>>L=[1, 2,'three', 4]
The L variable is now a list object. We can use
indexes and slices to access its members:

>>>L[0]
'1'
>>>L[2:]
['three', 4]
```

```
>>>L[2]=3
>>>L
[1, 2, 3, 4]
>>>L[0]=['one', 'two', 'three']
>>>L
[['one', 'two', 'three'], 2, 3, 4]
```

We can also concatenate lists (add them together):

```
>>>L + [8, 9]
[['one', 'two', 'three'], 2, 3, 4, 8, 9]
```

And repeat or multiply:

```
>>>[1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

We can delete items from lists using the del function:

```
>>>L = [['one', 'two', 'three'], 2, 3, 4]
del L[1]
L
[['one', 'two', 'three'], 3, 4]
```

Lists can be sorted and reverse sorted:

```
>>>L = ['c', 'b', 'a', 1]
>>>L.sort()
>>>L
[1, 'a', 'b', 'c']
>>>L.reverse()
>>>L
['c', 'b', 'a', 1]
```

Tuples

Tuples are very similar to lists. The difference (and the reason for their existence) is that tuples are immutable.

```
>>>T=(1, 2, 3)
>>>T[1]
2
>>>T[1:]
(2, 3)
```

Dictionaries

For each item in a dictionary, there is an accompanying key

```
>>>D={'key1':'dog', 2:'cat'}
>>>D['key1']
'dog'
>>>D[2]
'cat'
```

Changing the value of a key:

```
>>>D['key1'] = ['milk', 'cake']
>>>D
{('key', 'number', 'three'): 'mouse', 2:
'cat', 'key1': ['milk', 'cake']}

>>>D.keys()
[2, 'key1', 'new']
>>>D.values()
['cat', ['milk', 'cake'], 1.5]
```

For Loops

To Iterate over a sequence

```
>>>L = [1, 2, 3, 4, 5]
>>>for i in L:
...   print i
1
2
3
4
5

>>>Students=['John', 'Peter', 'Jeremy', 'Carly']
>>>for student in Students:

'John'
'Peter'
...
```

“while” Loops

A while loop repeats a set of commands until some terminating condition is met. They may never execute the body of the loop if the terminating condition is false to start with. Here is a basic example:

```
>>>X=1
>>>while X < 10:
...    print X,
...    X=X+1
1 2 3 4 5 6 7 8 9
```

“if” Statements

The if statement allows our programs to select a course of action depending on whether or not a certain condition exists.

```
>>>if 5 > 4:
...  print “Life is good”

Age = int(raw_input(”How old are you? “))
    #The int function returns its input as
integer.
Gender = raw_input(”Are you male or female?
“)
if Age >= 13 and Age <= 19:
    if Gender.capitalize() == “Male”:
        print “You are a teenage boy.”
    else:
        print “You are a teenage girl.”
```

* ‘Guess the number’ game