**PYTHON 3**

1. Print
   1. print(“Hello world!”)
   2. print(10//4) #ans. 2 truncated operator
   3. print(10%3) #ans. 1 modulus operator
   4. print(4 \*\* 2) #ans. 1 power
   5. print(2 \*\* 3 \*\* 2) # the right-most \*\* operator gets done first!
   6. Print( type(30.12) ) # <class ‘float’>
2. Type Conversion Functions
   1. int() # it doesn’t round off
      1. print(int(3.99)) # ans. 3
      2. print(int(-3.99)) # ans. -3
   2. float()
   3. str()
3. Length
   1. print( len(“Hello”) ) # ans. 5
4. Input
   1. X= input(“Enter the value: ”) # always return string, use Typecasting
5. For Loop
   1. for \_ in range(10):

doSomething()

1. Random
   1. import random
   2. print( random.random() ) #can give any real numbers b/w 0&1
   3. print ( random.randrange(1,7) ) # int between [1,7)
   4. from random import randrange, random
   5. print( random() )
   6. print ( randrange(1,7) )
   7. import random as rnd
   8. print( rnd.random() )
   9. print ( rnd.randrange(1,7) )
2. Math
   1. math.sqrt()
3. Collection Data Types
   1. Strings (Immutable)
   2. List (Mutable)
   3. Tuples (Immutable)
4. String
   1. S= “Hello World”
   2. S= ‘Hello World’
   3. m = “””

This is a Multi-Line

String.

“””

* 1. m = ‘’’

This is a Multi-Line

String.

‘’’

* 1. print( S[0] ) # index possible
  2. len (s) # no. of items
  3. print ( S[len(S) -1] ) = print ( S[-1] ) # negative indices possible

1. List
   1. myList = [“apple”, 5, “Banana”, 10]
   2. print( myList(0) )
   3. len( myList) )
   4. print (myList[ len(myList) -1] ) = print (myList [-1] )
2. Tuple
   1. myTuple = (“apple”, 5, “Banana”, 10)
   2. myTuple = (500) # int not Tuple
   3. myTuple = (500,) # now it’s a Tuple
   4. myTuple = () # Empty Tuple
   5. print( myTuple(0) )
   6. len( myTuple) )
   7. print (myTuple [ len(myTuple) -1] ) = print (myTuple [-1] )

NOTES

1. Some general errors:
   1. Syntax Error
   2. Logic Error
   3. Compilation Error
   4. Run time Error
   5. Resource Error
   6. Interface Error
2. Python is an Interpreted Language.
3. Workflow:
   1. Compiler
      1. Source Code
      2. Compiler
      3. Object Code or the executable
      4. Executor
      5. Output
   2. Interpreters
      1. Source Code
      2. Interpreter
      3. Output
4. Variables
   1. Can only start with a letter
   2. Can only contain letters and numbers.
   3. Underscore is also allowed.
   4. Variable names can never contain spaces.
   5. It should not be a Python Keyword (Ex: class)
5. The input function returns a string value
6. Functions imported as part of a module live in their own **namespace**. A namespace is simply a space within which all names are distinct from each other. The same name can be reused in different namespaces but two objects can’t have the same name within a single namespace.
7. Python Libraries <https://docs.python.org/3.6/library/index.html>
   1. <https://docs.python.org/3.6/py-modindex.html>
8. Don’t overwrite standard library modules!
9. It is important to note that random number generators are based on a deterministic algorithm — repeatable and predictable. So they’re called **pseudo-random generators** — they are not genuinely random. They start with a seed value. Each time you ask for another random number, you’ll get one based on the current seed attribute, and the state of the seed (which is one of the attributes of the generator) will be updated. The good news is that each time you run your program, the seed value is likely to be different, meaning that even though the random numbers are being created algorithmically, you will likely get random behaviour each time you execute.