WELCOME TO PYTHON WORKSHOP

Department of Computer Science
University of Delhi

Tutorial_Session1

March 17, 2018

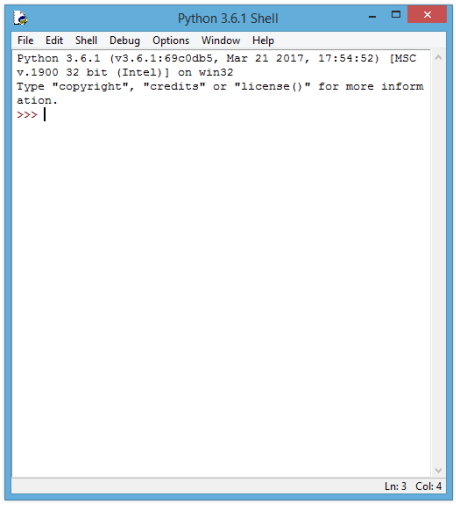
---

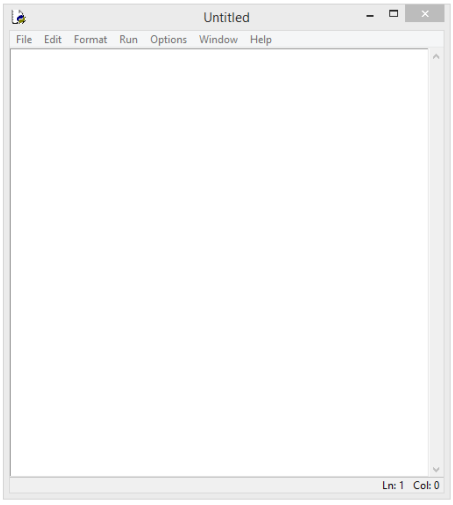
1 PYTHON

- Interactive, interpreted, and object-oriented programming language.
- Simple syntax
- Developed by Guido Van Rossum in 1991 at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Name was inspired by: Monty Python's Flying Circus

1.1 PYTHON PROGRAMMING ENVIRONMENT

- Available on a wide variety of platforms including Windows, Linux and Mac OS X.
- Official Website: **python.org**
- IDLE stands for Integrated Development and Learning Environment. Python IDLE comprises Python Shell and Python Editor. Python Shell Python Editor





1.2 Display on screen

```
In [2]: print('hello world')
```

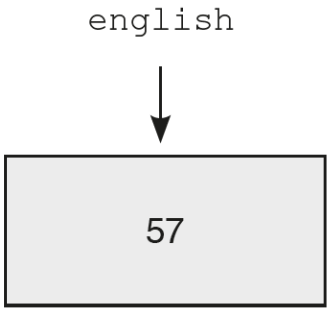
```
hello world
```

1.3 Names (Variables) and Assignment Statements

- Variables provide a means to name values so that they can be used and manipulated later.
- Assignment Statement: Statement that assigns value to a variable.

```
In [ ]: english = 57
        print(english)
```

Python associates the **name** (variable) **english** with value **57** i.e. the name (variable) **english** is assigned the value **57**, or that the name (variable) **english** refers to value **57**. Values are also called **objects**.



1.3.1 Rules for creating a name (variable)

- Must begin with a letter or _ (underscore character)
- May contain any number of letters, digits, or underscore characters. No other character apart from these is allowed.

1.3.2 Shorthand Notation

```
a = a <operator> b            is equivalent to
a <operator>= b
```

```
In [3]: a = 6
        a = a + 5
        print(a)
        a = 6
        a += 5
        print(a)
```

```
11
11
```

1.3.3 Multiple Assignments

- Used to enhance the readability of the program.

```
In [7]: msg, day, time = 'Meeting', 'Mon', '9'
        totalMarks = count = 0
```

1.4 Arithmetic Operators

```
In [11]: print("18 + 5 =", 18 + 5)     #Addition
         print("18 - 5 =", 18 - 5)     #Subtraction
         print("18 * 5 =", 18 * 5)     #Multiplication
         print("27 / 5 =", 27 / 5)     #Division
         print("27 // 5 =", 27 // 5)   #Integer Division
         print("27 % 5 =", 27 % 5)     #Modulus
         print("2 ** 3 =", 2 ** 3)     #Exponentiation
         print("-2 ** 3 =", -2 ** 3)   #Exponentiation
```

```
18 + 5 = 23
18 - 5 = 13
18 * 5 = 90
27 / 5 = 5.4
27 // 5 = 5
27 % 5 = 2
2 ** 3 = 8
-2 ** 3 = -8
```

```
In [9]: print("'how' + ' are' + ' you?':", 'how' + ' are' + ' you?')
        print("'hello' * 5             :", 'hello' * 5)
```

```
'how' + ' are' + ' you?': how are you?
'hello' * 5             : hellohellohellohellohello
```

1.4.1 Precedence of Arithmetic Operators

() (parentheses)	decreasing order
** (exponentiation)	
- (negation)	
/ (division) // (integer division) * (multiplication) % (modulus)	
+ (addition) - (subtraction)	

1.5 Relational Operators

- Used for comparing two expressions and yield True or False.
- The arithmetic operators have higher precedence than the relational operators.

```
In [ ]: print("23 < 25  :", 23 < 25)           #less than
        print("23 > 25  :", 23 > 25)           #greater than
        print("23 <= 23 :", 23 <= 23)          #less than or equal to
        print("23 - 2.5 >= 5 * 4 :", 23 - 2.5 >= 5 * 4) #greater than or equal to
        print("23 == 25 :", 23 == 25)          #equal to
        print("23 != 25 :", 23 != 25)          #not equal to
```

- When the relational operators are applied to strings, strings are compared left to right, character by character, based on their ASCII codes, also called ASCII values.

```
In [12]: print("'hello' < 'Hello' :", 'hello' < 'Hello')
         print("'hi' > 'hello'    :", 'hi' > 'hello')
```

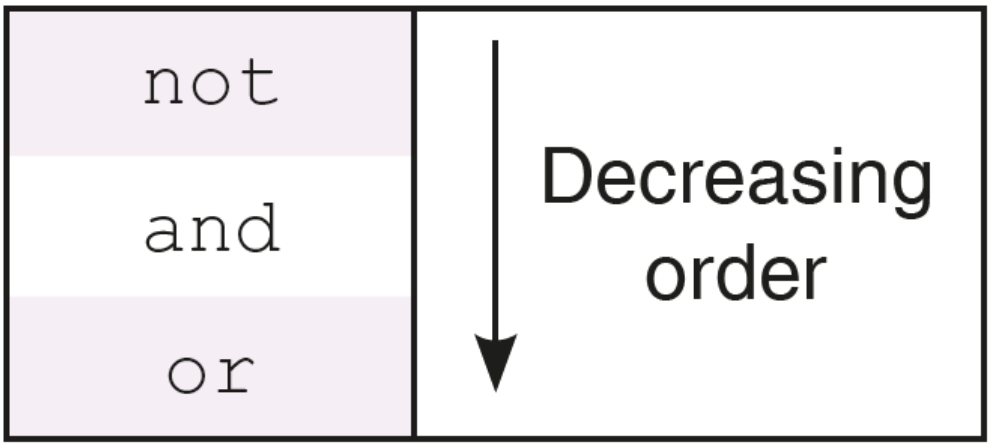
```
'hello' < 'Hello' : False
'hi' > 'hello'    : True
```

1.6 Logical Operators

- The logical operators not, and, and or are applied to logical operands True and False, also called Boolean values, and yield either True or False.
- As compared to relational and arithmetic operators, logical operators have the least precedence level.

```
In [ ]: print("not True < 25      :", not True)              #not operator
        print("10 < 25 and 5 > 6  :", 10 < 25 and 5 > 6) #and operator
        print("10 < 25 or 5 > 6   :", 10 < 25 or 5 > 6)   #or operator
```

1.6.1 Precedence of Logical Operators



1.7 Python Keywords

- Reserved words that are already defined by the Python for specific uses.

```
In [ ]: import keyword
        print(keyword.kwlist)
```

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class','continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda','nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

1.8 Functions

- Functions provide a systematic way of problem solving by dividing the given problem into several sub-problems, finding their individual solutions, and integrating the solutions of individual problems to solve the original problem.
- This approach to problem solving is called stepwise refinement method or modular approach.

1.9 Built-in Functions

- Predefined functions that are already available in Python.

1.9.1 Type Conversion: int, float, str functions

```
In [13]: str(123)

Out[13]: '123'

In [14]: int('234')

Out[14]: 234

In [15]: int(234.8)

Out[15]: 234
```

1.9.2 input function

- Enables us to accept an input string from the user without evaluating its value.
- The function input continues to read input text from the user until it encounters a newline.

```
In [16]: name = input('Enter a name: ')
         print('Welcome', name)

Enter a name: Suresh
Welcome Suresh
```

```
In [21]: costPrice = int(input('Enter cost price: '))
         profit = int(input('Enter profit: '))
         sellingPrice = costPrice + profit
         print('Selling Price: ', sellingPrice)
```

```
Enter cost price: 50
Enter profit: 12
Selling Price:  62
```

1.9.3 eval function

- Used to evaluate the value of a string.

```
In [22]: a = eval('15+10')
         print(a)
```

```
25
```

1.9.4 min and max functions

- Used to find maximum and minimum value respectively out of several values.

```
In [23]: a = max(59, 80, 95.6, 95.2)
         b = min('hello', 'how', 'are', 'you', 'Sir')
         print("Minimum Value", a)
         print("Maximum Value", b)
```

```
Minimum Value 95.6
Maximum Value Sir
```

1.9.5 Functions from math module

- Used to find maximum and minimum value respectively out of several values.

```
In [25]: import math
         print("math.ceil(3.4)  :", math.ceil(3.4))
         print("math.pow(3, 3)  :", math.pow(3, 3))
         print("math.sqrt(65)   :", math.sqrt(65))
         print("math.sqrt(65)   :", round(math.sqrt(65),2))
         print("math.log10(100) :", math.log10(100))
```

```
math.ceil(3.4)  : 4
math.pow(3, 3)  : 27.0
math.sqrt(65)   : 8.06225774829855
math.sqrt(65)   : 8.06
math.log10(100) : 2.0
```

1.9.6 help function

- Used to know the purpose of a function and how it is used.

```
In [26]: import math
         print(help(math.cos))
```

```
Help on built-in function cos in module math:

cos(...)
    cos(x)

    Return the cosine of x (measured in radians).

None
```

1.10 Function Definition and Call

The **syntax** for a function definition is as follows:

```
def function_name ( comma_separated_list_of_parameters):
    statements
```

Note: Statements below **def** begin with four spaces. This is called **indentation**. It is a requirement of Python that the code following a colon must be indented.

```
In [28]: def triangle():
             '''
             Objective: To print a right angled triangle.
             Input Parameter: None
             Return Value: None
             '''
             '''
             Approach: To use a print statement for each line of output
             '''
             print('*')
             print('* *')
             print('* * *')
             print('* * * *')
```

Invoking the function

```
In [29]: triangle()
```

```
*
* *
* * *
* * * *
```

1.10.1 Computing Area of the Rectangle

```
In [30]: def areaRectangle(length, breadth):
             '''
             Objective: To compute the area of rectangle
             Input Parameters: length, breadth  numeric value
             Return Value: area - numeric value
             '''
             area = length * breadth
             return area

In [33]: areaRectangle(7,5)

Out[33]: 35

In [34]: help(areaRectangle)

Help on function areaRectangle in module __main__:

areaRectangle(length, breadth)
    Objective: To compute the area of rectangle
    Input Parameters: length, breadth  numeric value
    Return Value: area - numeric value



In [43]: def areaRectangle(length, breadth=1):
             '''
             Objective: To compute the area of rectangle
             Input Parameters: length, breadth - numeric value
             Return Value: area - numeric value
             '''
             area = length * breadth
             return area

         def main():
             '''
             Objective: To compute the area of rectangle based on user input
             Input Parameter: None
             Return Value: None
             '''
             print('Enter the following values for rectangle:')
             lengthRect = int(input('Length : integer value: '))
             breadthRect = int(input('Breadth : integer value: '))
             areaRect = areaRectangle(lengthRect, breadthRect)
             print('Area of rectangle is', areaRect)

         if __name__ == '__main__':
             main()
```

```
Enter the following values for rectangle:
Length : integer value: 7
Breadth : integer value: 5
Area of rectangle is 35
```

1.11 Control Structures

- Needed for non-sequential and repetitive execution of instructions.

1.12 if Conditional Statement

- Used to execute a certain sequence of statements depending upon fulfilment of a particular condition > The general form of **if-elif-else** statement is as follows:

  if < condition1 >: < Sequence S1 of statements to be executed > elif < condition2 >: < Sequence S2 of statements to be executed > elif < condition3 >: < Sequence S3 of statements to be executed > ...

  else: < Sequence Sn of statements to be executed >

1.12.1 Problem: Grade assignment on the basis of marks obtained

```
In [44]: def assignGrade(marks):
             '''
             Objective: To assign grade on the basis of marks obtained
             Input Parameter: marks  numeric value
             Return Value: grade - string
             '''
             assert marks >= 0 and marks <= 100
             if marks >= 90:
                 grade = 'A'
             elif marks >= 70:
                 grade = 'B'
             elif marks >= 50:
                 grade = 'C'
             elif marks >= 40:
                 grade = 'D'
             else:
                 grade = 'F'
             return grade

         def main():
             '''
             Objective: To assign grade on the basis of input marks
             Input Parameter: None
             Return Value: None
             '''
```

```
            marks = float(input('Enter your marks: '))
            print('Marks:', marks, '\nGrade:', assignGrade(marks))

        if __name__ == '__main__':
            main()
```

```
Enter your marks: 89
Marks: 89.0
Grade: B
```

1.13 for Statement

- It is used when we want to execute a sequence of statements (indented to the right of keyword for) a fixed number of times. > Syntax of **for** statement is as follows:

  for variable in sequence:

```
In [38]: for letter in "hello":
             print(letter)
```

```
h
e
l
l
o
```

1.13.1 Generating sequence of numbers using range function

Syntax:

```
range(start, end, increment)
```

The function call **range(1,n + 1)** produces a sequence of numbers from 1 to n

```
In [ ]: start = 1
        limit = 11
        for num in range(start, limit):
            print(num)
```

```
In [ ]: start = 1
        limit = 11
        step = 2
        for num in range(start, limit, step):
            print(num)
```

```
In [ ]: start = 30
        limit = -4
        step = -3
        for num in range(start, limit, step):
            print(num)
```

```
In [ ]: limit = 5
        for num in range(limit):
            print(num)
```

1.13.2 Problem: Printing a Triangle

```
In [1]: def rightTriangle(rows):
            '''
            Objective: To print a triangle comprising of asterisks
            Input Parameter: rows - numeric
            Return Value: None
            '''
            for i in range(1, rows + 1):
                print('*' * i)

        def main():
            '''
            Objective: To compute factorial of a number provided as an input
            Input Parameter: None
            Return Value: None
            '''
            rows = int(input('Enter number of rows: '))
            rightTriangle(rows)

        if __name__ == '__main__':
            main()
```

```
Enter number of rows: 6
*
**
***
****
*****
******
```

1.13.3 Problem: Factorial of a number

```
In [41]: def factorial(num):
             '''
             Objective: To compute factorial of a number
             Input Parameter: num - numeric
             Return Value: num! - numeric
             '''
             if num <= 0:
                 return 'Factorial Not defined'
             fact = 1
             for i in range(1, num+1):
```

```
            fact = fact * i
        return fact

    def main():
        '''
        Objective: To compute factorial of a number provided as an input
        Input Parameter: None
        Return Value: None
        '''
        num = int(input('Enter the number: '))
        fact = factorial(num)
        print("Result:", fact)

    if __name__ == '__main__':
        main()
```

```
Enter the number: 5
Result: 120
```

1.14 while Statement

- It is used for executing a sequence of statements again and again on the basis of some test condition.
- If the test condition holds True, the body of the loop is executed, otherwise the control moves to the statement immediately following the while loop. > Syntax of **while** statement is as follows:

  while :

```
In [ ]: count, n = 1, 5
        while count < n+1:
            print(count)
            count += 1
```

1.14.1 Sum of digits of a number

```
In [45]: def sumOfDigits(num):
             '''
             Objective: To compute sum of digits of a number
             Input Parameter: num - numeric
             Return Value: numeric
             '''
             '''
             Approach:
                 Ingore the sign of number. Initialize sum to zero.
                 Extract digits one by one beginning unit's place and keeps on
                 adding it to sum.
             '''
```

```
    num = abs(num)
    total = 0
    while num >= 1:
        total += (num % 10)
        num = num // 10
    return total

def main():
    '''
    Objective: To compute sum of digits of a number provided as an input
    Input Parameter: None
    Return Value: None
    '''
    num = int(input('Enter the number: '))
    total = sumOfDigits(num)
    print("Result:", total)

if __name__ == '__main__':
    main()
```

```
Enter the number: 123
Result: 6
```

Tutorial_Session2

March 17, 2018

1 String and List

1.1 Strings

- A string is a sequence of characters.
- A string may be specified by placing the member characters of the sequence within quotes (single, double or triple).
- Triple quotes are typically used for strings that span multiple lines.

```
In [1]: message = 'Hello Gita'
```

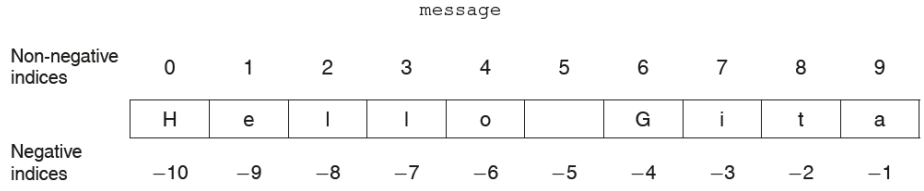
1.1.1 Computing Length using len function

```
In [2]: print(len(message))
```

```
10
```

1.1.2 Indexing

- Individual characters within a string are accessed using a technique known as indexing.




```
In [3]: index = len(message) - 1
        print(message[0], message[6], message[index], message[-1])
```

```
H G a a
```

```
In [4]: print(message[15])
```

```
---------------------------------------------------------------------------
```

```
IndexError                                Traceback (most recent call last)

<ipython-input-4-a801df50d8d1> in <module>()
----> 1 print(message[15])


IndexError: string index out of range
```

1.1.3 Slicing

- In order to extract the substring comprising the character sequence having indices from start to end-1, we specify the range in the form start:end.
- Python also allows us to extract a subsequence of the form *start:end:inc*.

```
In [89]: message = 'Hello Sita'
         print(message[0:5], message[-10:-5])
```

```
Hello Hello
```

```
In [90]: print(message[0:len(message):2])
         print(message[:])
```

```
HloSt
Hello Sita
```

1.1.4 Membership Operator in

- Python also allows us to check for membership of the individual characters or substrings in strings using in operator.

```
In [91]: 'h' in 'hello'
```

```
Out[91]: True
```

```
In [92]: 'ell' in 'hello'
```

```
Out[92]: True
```

```
In [93]: 'h' in 'Hello'
```

```
Out[93]: False
```

1.2 Built-in Functions on Strings

1.2.1 Function: count

- For counting number of occurrences of a substring.

```
In [94]: 'Encyclopedia'.count('c')
```

```
Out[94]: 2
```

```
In [95]: vowels = 'AEIOUaeiou'
         vowelCount = 0
         for ch in vowels:
             vowelCount += 'Encyclopedia'.count(ch)
         print(vowelCount)
```

```
5
```

1.2.2 Functions find and rfind

- Function **find**: Returns the index of the first occurrence of a string.
- Function **rfind**: Returns the index of the last occurrence of a string.

```
In [96]: colors = 'green, red, blue, red, red, green'
         print("colors.find('red'): ", colors.find('red'))
         print("colors.rfind('red'): ", colors.rfind('red'))
         print("colors.find('orange'): ", colors.find('orange'))
```

```
colors.find('red'):  7
colors.rfind('red'):  23
colors.find('orange'):  -1
```

1.2.3 Functions capitalize, title, lower, upper, and swapcase

- Function **capitalize**: converting the first letter of a string to uppercase character and converting the remaining letters in the string to lowercase.
- Function **title**: Capitalize the first letter of each word in a string and change the remaining letters to lowercase.
- Function **lower**: Convert all letters in a string to lowercase.
- Function **upper**: Convert all letters in a string to uppercase.

```
In [97]: 'python IS a Language'.capitalize()
```

```
Out[97]: 'Python is a language'
```

```
In [98]: 'python IS a PROGRAMMING Language'.title()
```

```
Out[98]: 'Python Is A Programming Language'
```

```
In [99]: emailId1 = 'geek@gmail.com'
         emailId2 = 'Geek@gmail.com'
         emailId1.lower() == emailId2.lower()
```

```
Out[99]: True
```

1.2.4 Function swapcase

```
In [4]: 'AnilKumar'.swapcase()
```

```
Out[4]: 'aNILkUMAR'
```

1.2.5 Functions islower, isupper, isalpha, isdigit, and isalnum

```
In [101]: 'python'.islower()
```

```
Out[101]: True
```

```
In [102]: 'Python'.isupper()
```

```
Out[102]: False
```

```
In [103]: '9953799924'.isdigit()
```

```
Out[103]: True
```

```
In [104]: 'Nikhil Kumar'.isalpha()
```

```
Out[104]: False
```

```
In [105]: password = 'Kailash107Ganga'
          password.isalnum()
```

```
Out[105]: True
```

1.2.6 Function replace

- It allows to replace part of a string by another string.
- It takes two arguments as inputs. The first argument is used to specify the substring that is to be replaced. The second argument is used to specify the string that replaces the first string.

```
In [106]: message = 'Amey my friend, Amey my guide'
```

```
In [107]: message.replace('Amey', 'Vihan')
```

```
Out[107]: 'Vihan my friend, Vihan my guide'
```

1.2.7 Functions strip, lstrip, and rstrip

- The functions **lstrip** and **rstrip** remove whitespaces from the beginning and end, respectively.
- The function **strip** removes whitespaces from the beginning as well as the end of a string.

```
In [108]: '       Hello How are you!      '.lstrip()
```

```
Out[108]: 'Hello How are you!      '
```

```
In [109]: '       Hello How are you!      '.rstrip()
```

```
Out[109]: '       Hello How are you!'
```

```
In [110]: '       Hello How are you!      '.strip()
```

```
Out[110]: 'Hello How are you!'
```

1.2.8 Functions split and partition

- The function **split** enables us to split a string into a list of strings based on a delimiter.
- The function **partition** divides a string S into two parts based on a delimiter and returns a tuple comprising string before the delimiter, the delimiter itself, and the string after the delimiter

```
In [111]: colors = 'Red, Green, Blue, Orange, Yellow, Cyan'
          colors.split(',')

Out[111]: ['Red', ' Green', ' Blue', ' Orange', ' Yellow', ' Cyan']

In [112]: 'Hello. How are you?'.partition('.')

Out[112]: ('Hello', '.', ' How are you?')
```

1.2.9 Function join

- The function **join** returns a string comprising elements of a sequence separated by the specified delimiter.

```
In [123]: ' > '.join(['I', 'am', 'ok'])

Out[123]: 'I > am > ok'

In [124]: ' '.join(('I', 'am', 'ok'))

Out[124]: 'I am ok'
```

1.3 Problem: Reversing a string

```
In [1]: def reverse(str1):
            '''
            Objective: To reverse a string
            Input Parameter: str1 - string
            Return Value: reverse of str1 - string
            '''
            reverseStr = ''
            for i in range(len(str1)):
                reverseStr = str1[i] + reverseStr
            return reverseStr

        def main():
            '''
            Objective: To reverse the string provided as input
            Input Parameter: None
            Return Value: None
            '''
            myString=input('Enter a string to be reversed:')
            reverseStr=reverse(myString)
```

```
        print('The reverse is:' + reverseStr)
    if __name__ == '__main__':
        main()
```

```
Enter a string to be reversed:PYTHON
The reverse is:NOHTYP
```

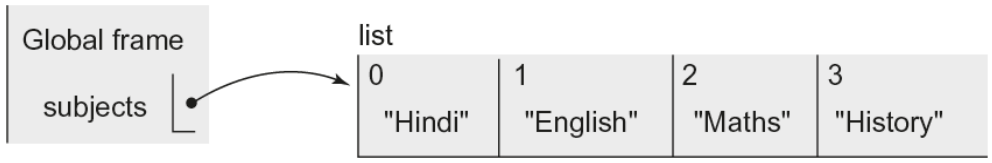
```
In [2]: reverse('Python')
```

```
Out[2]: 'nohtyP'
```

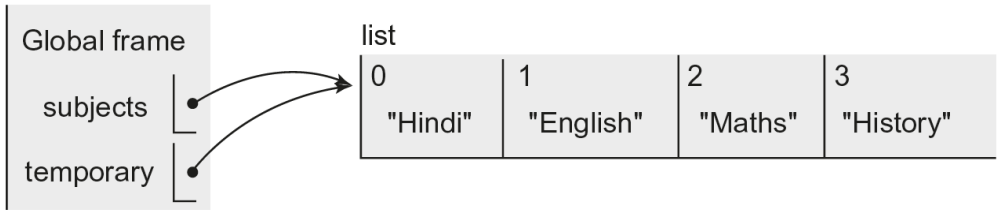
1.4 Lists

- A list is an ordered sequence of values.
- Values stored in a list can be of any type such as string, integer, float, or list.
- Note!! Elements of a list are enclosed in square brackets, separated by commas.
- Unlike strings, lists are mutable, and therefore, one may modify individual elements of a list.

```
In [127]: subjects=['Hindi', 'English', 'Maths', 'History']
```




```
In [128]: temporary = subjects
```




```
In [129]: temporary[0] = 'Sanskrit'
          print(temporary)
          print(subjects)
```

```
['Sanskrit', 'English', 'Maths', 'History']
['Sanskrit', 'English', 'Maths', 'History']
```

```
In [130]: subjectCodes = [['Sanskrit', 43], ['English', 85] , ['Maths', 65], ['History', 36]]
          subjectCodes[1]
```

```
Out[130]: ['English', 85]
```

```
In [131]: print(subjectCodes[1][0],subjectCodes[1][1])
```

```
English 85
```

1.4.1 Heterogeneous List

```
In [132]: details = ['Megha Verma', 'C-55, Raj Nagar, Pitam Pura, Delhi - 110034', 9876543210]
```

1.5 List Operations

```
In [156]: list1 = ['Red', 'Green']
          list2 = [10, 20, 30]
```

1.5.1 Multiple Operator *

```
In [157]: list2 * 2

Out[157]: [10, 20, 30, 10, 20, 30]
```

1.5.2 Concatenation Operator +

```
In [158]: list1 = list1 + ['Blue']
          list1

Out[158]: ['Red', 'Green', 'Blue']
```

1.5.3 Length Operator len

```
In [159]: len(list1)

Out[159]: 3
```

1.5.4 Indexing & Slicing

```
In [160]: list2[-1]

Out[160]: 30

In [161]: list2[0:2]

Out[161]: [10, 20]

In [162]: list2[0:3:2]

Out[162]: [10, 30]
```

1.5.5 Function min & max

```
In [163]: min(list2)

Out[163]: 10

In [164]: max(list1)

Out[164]: 'Red'
```

1.5.6 Function sum

```
In [165]: sum(list2)
```

```
Out[165]: 60
```

1.5.7 Membership Operator: in

```
In [166]: 40 in list2
```

```
Out[166]: False
```

```
In [167]: students = ['Ram', 'Shyam', 'Gita', 'Sita']
          for name in students:
              print(name)
```

```
Ram
Shyam
Gita
Sita
```

1.5.8 Function list

- The function list takes a sequence as an argument and returns a list.

```
In [168]: vowels = 'aeiou'
          list(vowels)
```

```
Out[168]: ['a', 'e', 'i', 'o', 'u']
```

1.6 Built-in Functions on Lists

1.6.1 Function append

- The function append insert the object passed to it at the end of the list.

```
In [170]: a = [10, 20, 30, 40]
          a.append(35)
          print(a)
```

```
[10, 20, 30, 40, 35]
```

1.6.2 Function extend

- The function extend accepts a sequence as an argument and puts the elements of the sequence at the end of the list.

```
In [171]: a = [10, 20, 30]
          a.extend([35,40])
          print(a)
```

```
[10, 20, 30, 35, 40]
```

1.6.3 Function:count

- The function count returns the count of the number of times the object passed as an argument appears in the list.

```
In [172]: a = [10, 20, 30, 10, 50, 20, 60, 20, 30, 55]
          print(a.count(20))
```

```
3
```

1.6.4 Function pop

- The function pop returns the element from the list whose index is passed as an argument, while removing it from the list.

```
In [173]: a = [10, 20, 30, 10, 50, 20, 60, 20, 30, 55]
          a.pop(3)
          print(a)
```

```
[10, 20, 30, 50, 20, 60, 20, 30, 55]
```

1.6.5 Function remove

- The function remove takes the value to be removed from the list as an argument, and removes the first occurrence of that value from the list.

```
In [174]: a.remove(20)
          print(a)
```

```
[10, 30, 50, 20, 60, 20, 30, 55]
```

1.6.6 del Operator

- The del operator is used to remove a subsequence of elements (start:end:increment) from a list.

```
In [175]: a = [10, 20, 30, 20, 60, 20, 30, 55]
          del a[2:6:3]
          print(a)
```

```
[10, 20, 20, 60, 30, 55]
```

1.6.7 Function insert

- The insert function can be used to insert an object at a specified index. This function takes two arguments: the index where an object is to be inserted and the object itself.

```
In [176]: names = ['Ram', 'Sita', 'Gita', 'Sita']
          names.insert(2, 'Shyam')
          print(names)
```

```
['Ram', 'Sita', 'Shyam', 'Gita', 'Sita']
```

1.6.8 Function reverse

- The function reverse reverses the order of the elements in a list.

```
In [178]: names = ['Ram', 'Sita', 'Sita', 'Anya']
          names.reverse()
          print(names)
```

```
['Anya', 'Sita', 'Sita', 'Ram']
```

1.6.9 Function sort

- The function sort can be used to arrange the elements in a list in ascending order.

```
In [179]: names = ['Ram', 'Sita', 'Sita', 'Anya']
          names.sort()
          print(names)
```

```
['Anya', 'Ram', 'Sita', 'Sita']
```

```
In [180]: names = ['Ram', 'Sita', 'Sita', 'Anya']
          names.sort(reverse = True)
          print(names)
```

```
['Sita', 'Sita', 'Ram', 'Anya']
```

1.7 Problem: List of n terms of fibonacci series

```
In [3]: def fib(n):
            '''
            Objective: To return list of n terms of fibonacci series.
            Input Parameter: n - numeric
            Return Value: numeric
            '''
            '''
```

```
        Approach:
            Create a list with 0 and 1 as first two numbers of fibonacci series.
            For each subsequent number, append sum of previous two numbers to
            the list.
        '''
        if n <=0:
            return None
        elif n ==1:
            return [0]
        elif n == 2:
            return [0, 1]
        else:
            resList = []
            a, b = 0, 1
            resList.append(a)
            resList.append(b)
            count = 3
            while  count <= n:
                c = a + b
                resList.append(c)
                a = b
                b = c
                count += 1
            return resList

    def main():
        '''
        Objective: To print n terms of fibonacci series based on user input
        Input Parameter: None
        Return Value: None
        '''
        num = int(input('Enter no. of terms:'))
        result = fib(num)
        print(result)

    if __name__ == '__main__':
        main()
```

```
Enter no. of terms:6
[0, 1, 1, 2, 3, 5]
```

Tutorial_Session3

March 17, 2018

1 Tuples and Dictionaries

1.1 Tuples

- A tuple is an ordered sequence of objects.
- A tuple may be specified by enclosing in the parentheses, the elements of the tuple (possibly of heterogeneous types), separated by commas.

```
In [6]: myTuple = (4, 6, [2, 8], 'abc', {3, 4})
```

```
In [11]: digits = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
In [12]: subjects = ('Physics', 'Chemistry', 'Computer Science')
```

```
In [13]: months = ((1, 'January'), (2, 'February'), (3, 'March'))
```

- If a tuple comprises a single element, the element should be followed by a comma to distinguish a tuple from a parenthesized expression.
- A tuple having a single element is also known as singleton tuple.

- Tuples being **immutable**, an attempt to modify an element of a tuple yields an error.

```
In [14]: digits[1] = 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-14-93313e5806a7> in <module>()
----> 1 digits[1] = 3


TypeError: 'tuple' object does not support item assignment
```

1.2 Tuple Operations

```
In [15]: weekDays = ('Monday', 'Tuesday')
         marks = (78, 99, 34, 45)
         dateOfBirth = (1, 'October', 1990)
```

1.2.1 Multiplication Operator *

```
In [16]: weekDays * 2
```

```
Out[16]: ('Monday', 'Tuesday', 'Monday', 'Tuesday')
```

1.2.2 Concatenation Operator +

```
In [17]: weekDays = weekDays + ('Wednesday',)
         print(weekDays)
```

```
('Monday', 'Tuesday', 'Wednesday')
```

1.2.3 Length Operator len

```
In [18]: len(weekDays)
```

```
Out[18]: 3
```

1.2.4 Indexing & Slicing

```
In [19]: yearOfBirth = dateOfBirth[-1] #Indexing
         print(yearOfBirth)
```

```
1990
```

```
In [20]: weekDays[1:2] #Slicing
```

```
Out[20]: ('Tuesday',)
```

1.2.5 Function min & max

```
In [21]: min(marks)
```

```
Out[21]: 34
```

```
In [22]: max(marks)
```

```
Out[22]: 99
```

1.2.6 Function sum

```
In [23]: sum(marks)
```

```
Out[23]: 256
```

1.2.7 Membership Operator in

```
In [24]: 'Friday' in weekDays
```

```
Out[24]: False
```

1.2.8 Function tuple

- The function tuple can be used to convert a sequence to a tuple.

```
In [25]: vowels = 'aeiou'
         tuple(vowels)
```

```
Out[25]: ('a', 'e', 'i', 'o', 'u')
```

1.3 Built-in Functions on Tuples

1.3.1 Function count

- Returns count of occurrences of an element in the tuple.

```
In [26]: age = (20, 18, 17, 19, 18, 18)
         print(age.count(18))
```

```
3
```

1.3.2 Function index

- Returns index of the first occurrence of an element in the tuple.

```
In [27]: age.index(18)
```

```
Out[27]: 1
```

1.4 Problem: Sort list of tuples

```
In [28]: def sortList(studentList):
             '''
             Objective: To sort a given list of tuples in increasing order on the basis
                        of marks.
             Input Parameter: studentList: a list of tuples such that each tuple
                              contains student names and their marks
             Return Value: studentList: sorted list
             '''
             '''
             Approach:
                 Use bubble sort.
                 Starting from the first element (tuple), compare two consecutive elements
                 of the list and swap the two if marks in second element is smaller than the
                 marks in first element. At the end of first iteration, smallest element in
```

```
            the list will come in the beginning. If there are n elements, the above
            procedure will repeat n-1 times
        '''

        for i in range(0, len(studentList) - 1):
            for j in range(0, len(studentList) - 1 - i):
                if studentList[j+1][1] < studentList[j][1]:
                    studentList[j+1], studentList[j] = studentList[j], studentList[j+1]
        return studentList

    def main():
        '''
        Objective: To call a function sortList to sort a given list of tuples in
        increasing order on the basis of marks provided as input.
        Input Parameter: None
        Return Value: None
        '''
        #studentList = [('Rohit', 50), ('Deepak', 75), ('Sonali', 47)]
        studentList = []
        num = int(input('Enter the number of students:'))
        for i in range(num):
            pair = eval(input('Enter tuple <student name, marks>'))
            studentList.append(pair)
        sortList(studentList)
        print(studentList)

    if __name__ == '__main__':
        main()
```

```
Enter the number of students:4
Enter tuple <student name, marks>('Sheetal',99)
Enter tuple <student name, marks>('Ankit',100)
Enter tuple <student name, marks>('Arun',56)
Enter tuple <student name, marks>('Bhawna',89)
[('Arun', 56), ('Bhawna', 89), ('Sheetal', 99), ('Ankit', 100)]
```

1.5 Dictionaries

- A dictionary is an unordered sequence of key-value pairs.
- Key and value in a key-value pair in a dictionary are separated by a colon. Further, the key:value pairs in a dictionary are separated by commas and are enclosed between curly parentheses.
- Indices in a dictionary can be of any immutable type and are called keys.

```
In [29]: month = {}
         month[1] = 'Jan'
```

```
month[2] = 'Feb'
month[3] = 'Mar'
month[4] = 'Apr'
print(month)
```

```
{1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr'}
```

- The search in a dictionary is based on the key.
- Therefore, in a dictionary, the keys are required to be unique. However, the same value may be associated with multiple keys.

```
In [30]: price = {'tomato':40, 'cucumber':30,
         'potato':20, 'cauliflower':70, 'cabbage':50,
         'lettuce':40, 'raddish':30, 'carrot':20,
         'peas':80}
```

- Values associated with keys can be mutable objects and thus, may be changed at will.

```
In [31]: price['tomato'] = 25
```

- Keys in a dictionary may be of heterogeneous types

```
In [32]: counting = {1:'one', 'one':1, 2:'two', 'two':2}
```

1.6 Dictionary Operations

```
In [33]: digits = {0:'Zero', 1:'One', 2:'Two', 3:'Three', 4:'Four', 5:'Five', 6:'Six', 7:'Seve
```

1.6.1 length Operator len

```
In [34]: len(digits)
```

```
Out[34]: 10
```

1.6.2 Indexing

```
In [35]: digits[1]
```

```
Out[35]: 'One'
```

1.6.3 Functions min and max

```
In [36]: min(digits)
```

```
Out[36]: 0
```

```
In [37]: max(digits)
```

```
Out[37]: 9
```

1.6.4 Function sum

```
In [38]: sum(digits)
```

```
Out[38]: 45
```

1.6.5 Membership Opeartor in

```
In [39]: 5 in digits
```

```
Out[39]: True
```

```
In [40]: 'Five' in digits
```

```
Out[40]: False
```

Note: Membership operation in, and functions min, max and sum apply only to the keys in a dictionary.

1.6.6 Deleting a key-value pair from dictionary

```
In [41]: winter = {11:'November ', 12: 'December', 1:'January', 2:'February'}
         del winter[11]
         print(winter)
```

```
{12: 'December', 1: 'January', 2: 'February'}
```

1.7 Built-in Functions on Dictionaries

1.7.1 Deleting all key-value pairs using clear function

```
In [42]: winter.clear()
         print(winter)
```

```
{}
```

1.7.2 Function get

- The function get is used to extract the value corresponding to a given key
- The first parameter is used to specify the key and the second parameter is used to specify the value to be returned in case the key is not found in the dictionary. In case, the second parameter is not specified, the system returns None

```
In [43]: passwords = {'Ram':'ak@607', 'Shyam':'rou.589'}
```

```
In [44]: passwords.get('Ram',-1)
```

```
Out[44]: 'ak@607'
```

```
In [45]: passwords.get('Raman',-1)
```

```
Out[45]: -1
```

1.7.3 Function update

- The function update is used to insert in a dictionary, all the key–value pairs of another dictionary

```
In [46]: morePasswords = {'Raman':'vi97@4', 'Kishore':'23@0jsk'}
```

```
In [47]: passwords.update(morePasswords)
         print(passwords)
```

```
{'Ram': 'ak@607', 'Shyam': 'rou.589', 'Raman': 'vi97@4', 'Kishore': '23@0jsk'}
```

1.7.4 Function keys

- Return an object comprising of all keys of the dictionary.

```
In [48]: months = {1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr'}
         print(months.keys())
```

```
dict_keys([1, 2, 3, 4])
```

1.7.5 Function values

- Return an object comprising of all values of the dictionary.

```
In [49]: print(months.values())
```

```
dict_values(['Jan', 'Feb', 'Mar', 'Apr'])
```

1.7.6 Function items

- Return an object comprising of tuples of key-value pairs present in the dictionary.

```
In [50]: print(months.items())
```

```
dict_items([(1, 'Jan'), (2, 'Feb'), (3, 'Mar'), (4, 'Apr')])
```

1.8 Dictionary of state and its capitals

```
In [51]: def stateCapitalDict():
             '''
             Purpose: To form a dictionary of state and its capital as specified by user.
             Input Parameter: None
             Return Value: stateCapital - Dictionary containing state as keys and capital
                           as values
             '''
             '''
```

```
        Approach:
        For each state and capital taken as input from the user
            Assign value capital to the key state
        '''
        stateCapital = dict()
        state = input('Enter state:')
        capital = input('Enter capital:')
        while state != '' and capital != '':
            stateCapital[state] = capital
            state = input('Enter state:')
            capital = input('Enter capital:')
        return stateCapital

    def main():
        '''
        Purpose: To form a dictionary of state and its capital as specified by user.
        Input Parameter: None
        Return Value: None
        '''
        dict1 = stateCapitalDict()
        print(dict1)

    if __name__ == '__main__':
        main()
```

```
Enter state:Goa
Enter capital:Panaji
Enter state:Haryana
Enter capital:Chandigarh
Enter state:
Enter capital:
{'Goa': 'Panaji', 'Haryana': 'Chandigarh'}
```

Tutorial_Session4

March 17, 2018

1 FILE HANDLING AND ERROR HANDLING

1.1 File Handling

- Files provide a means of communication between the program and the outside world.
- A file is a stream of bytes, comprising data of interest.
- **File Operations:** > * Reading from a file > * Writing to a file > * Append to a file > * Rename a file > * Delete a file

1.1.1 Opening a File

- Built-in function used: **open()**
- This function takes the name of the file as the first argument. The second argument indicates the mode for accessing the file.
- Modes for opening a file:
  - read(r): to read the file
  - write(w): to write to the file
  - append(a): to write at the end of the file.

Syntax:

```
f = open(file_name, access_mode)
```

- Opening a non-existent file in w or a mode creates a new file with the given name
- However, opening a non-existent file in r mode leads to an error.

** Accessing attribute of an instance of class** * To specify an attribute of a class (or class instance), we write the name of the class (or class instance) followed by a dot, followed by the name of that attribute. The method **lower** defined in class **str** has been invoked for the object **name**.

```
In [8]: f = open('PYTHON','w')
```

1.1.2 Writing to a File

- Built-in function used: **write()**
- To use write function, specify name of the file object, followed by the dot operator (.), followed by the name of the function.
- Note that apart from writing the given data into a file, the function write also returns the number of characters written into the file.

```
In [9]: f.write('''Python:
        Python is Simple.
        Simple syntax.''')

Out[9]: 40
```

1.1.3 Reading a File

- Built-in function used: **read()**
- To use read function, specify name of the file object, followed by the dot operator (.), followed by the name of the function
- The read() function retrieves the contents of the entire file.

```
In [11]: f.read() #attempt to read file without read mode on.


         ---------------------------------------------------------------------------

         UnsupportedOperation                      Traceback (most recent call last)

         <ipython-input-11-99d52ada1abc> in <module>()
        ----> 1 f.read() #attempt to read file without read mode on.


         UnsupportedOperation: not readable


In [16]: f = open('PYTHON','r')
         f.read()

Out[16]: 'Python:\nPython is Simple.\nSimple syntax.'

In [17]: f.read() # reading a file reached EOF yields empty string

Out[17]: ''
```

We can read a fixed number of bytes from the file by specifying the number of bytes as the argument to read function.

```
In [18]: f = open('PYTHON','r')
         f.read(6)

Out[18]: 'Python'
```

Displaying the multi-line string using the print function

```
In [19]: f = open('PYTHON','r')
         print(f.read())
```

```
Python:
Python is Simple.
Simple syntax.
```

1.1.4 Function close

- When a file is no longer required for reading or writing, it should be closed by invoking the function close

```
In [20]: f.close()
```

The function close also saves a file, which was opened for writing. Once a file is closed, it cannot be read or written any further unless it is opened again and an attempt to access the file results in an I/O (input/output) error:

```
In [21]: f.write('Python was developed by Guido Van Rossum in 1991.')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-21-12657766d5ec> in <module>()
----> 1 f.write('Python was developed by Guido Van Rossum in 1991.')


ValueError: I/O operation on closed file.
```

1.1.5 Functions readline and readlines

readline function: reads a stream of bytes beginning the current position until a newline character is encountered

```
In [22]: f = open('PYTHON','r')
         line = f.readline()
         print('line1:', line)
         line = f.readline()
         print('line2:', line)
         line = f.readline()
         print('line3:', line)
         line = f.readline()
         print('line4:', line)
         f.close()
```

```
line1: Python:

line2: Python is Simple.

line3: Simple syntax.
line4:
```

readlines function: returns all the remaining lines of the file in the form of a list

```
In [23]: f = open('PYTHON','r')
         lines = f.readlines()
         print(lines)
         f.close()
```

```
['Python:\n', 'Python is Simple.\n', 'Simple syntax.']
```

1.1.6 Function writelines

writelines function: takes a list of lines to be written in the file as an argument

```
In [24]: f = open('PYTHON', 'w')
         description = ['Python:\n', 'Python is simple.\n']
         f.writelines(description)
         f.close()

         f = open('PYTHON', 'r')
         print(f.read())
         f.close()
```

```
Python:
Python is simple.
```

1.1.7 Functions seek and tell

- **seek()**: to reach desired position in a file Syntax:

  seek(offset) # offset indicates the number of bytes to be moved # Returns the new absolute position

- **tell()**: to find current position of the file object

```
In [25]: f = open('PYTHON','r')
         print(f.readline())
         f.seek(0) # Changes the current file position to the begining of the file
         print('After seeking file pointer in the beginning\n')
         print(f.readline())
```

```
Python:

After seeking file pointer in the beginning

Python:
```

```
In [26]: f.seek(0)
         f.read(4)

Out[26]: 'Pyth'

In [27]: f.tell()

Out[27]: 4
```

1.2 Error Handling

- Error occurs when something goes wrong.
- The errors in Python programming may be categorized as:
  - Syntax Errors
  - Exceptions

1.3 Syntax Error

A syntax error occurs when a rule of Python grammar is violated.

```
In [53]: print('Hello)   # missing quote mark at the end of the string


           File "<ipython-input-53-5741a0378b6f>", line 1
         print('Hello)   # missing quote mark at the end of the string
                                                                      ^
     SyntaxError: EOL while scanning string literal


In [54]: for i in range(0, 10)   # missing colon in the for statement


           File "<ipython-input-54-a1c3f00d6a60>", line 1
         for i in range(0, 10)   # missing colon in the for statement
                                                                     ^
     SyntaxError: invalid syntax
```

1.4 Exceptions

- Errors that occur at execution time.

- These errors disrupt the flow of the program at a run-time by terminating the execution at the point of occurrence of the error.
- We have noticed that whenever an exception occurs, a Traceback object is displayed which includes error name, its description, and the point of occurrence of the error such as line number.

1.4.1 NameError

This exception occurs whenever a name that appears in a statement is not found globally.

```
In [28]: marks = Input('Enter your marks')
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-28-e2d0c49b2ae3> in <module>()
----> 1 marks = Input('Enter your marks')


NameError: name 'Input' is not defined
```

```
In [30]: price=10
         print(Price)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-30-f01eed036878> in <module>()
  1 price=10
----> 2 print(Price)


NameError: name 'Price' is not defined
```

1.4.2 TypeError

This exception occurs when an operation or function is applied to an object of inappropriate type.

```
In [31]: 'sum of 2 and 3 is ' + 5
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-31-135c7253899e> in <module>()
----> 1 'sum of 2 and 3 is ' + 5


TypeError: must be str, not int
```

1.4.3 ValueError

This exception occurs whenever an inappropriate argument value, even though of correct type, is used in a function call.

```
In [32]: int('Hello')
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-32-6765ce49acfe> in <module>()
----> 1 int('Hello')


ValueError: invalid literal for int() with base 10: 'Hello'
```

1.4.4 ZeroDivisionError

This exception occurs when we try to perform numeric division in which the denominator happens to be zero.

```
In [33]: 78/(2+3-5)
```

```
---------------------------------------------------------------------------

ZeroDivisionError                         Traceback (most recent call last)

<ipython-input-33-5c961f5673b5> in <module>()
----> 1 78/(2+3-5)


ZeroDivisionError: division by zero
```

1.4.5 OSError

This exception occurs whenever there is an error related to input/output.

```
In [34]: # opening a non-existent file for reading or reading a file opened in write mode
         f = open('passwordFile.txt')
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

<ipython-input-34-880578fe47b9> in <module>()
  1 # opening a non-existent file for reading or reading a file opened in write mode
----> 2 f = open('passwordFile.txt')


FileNotFoundError: [Errno 2] No such file or directory: 'passwordFile.txt'
```

1.4.6 IndexError

This exception occurs whenever we try to access an index that is out of a valid range.

```
In [35]: colors = ['red', 'green', 'blue']

In [36]: colors[4]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-36-1f89a8f1c888> in <module>()
----> 1 colors[4]


IndexError: list index out of range
```

1.5 Problem: Copying contents of a file to another file

```
In [41]: import os

         def fileCopy(file1,file2):
             '''
             Objective: This method copies the contents in a file to another file
             Input Parameter: file1 - input file, file2 - output file
             Return Value: None
```

```
        '''
        '''
        Approach:
        Read input from file1, line by line and copy to file2 until
        null string is returned on reading
        '''
        f1 = open(file1, 'r')
        f2 = open(file2, 'w')
        line = f1.readline()
        while line != '':
            f2.write(line) #write the line from f1 with additional newline
            line = f1.readline()
        f1.close()
        f2.close()

    def main():
        '''
        Objective: To call function fileCopy to copy contents in a file to another file.
        Input Parameter: None
        Return Value: None
        '''
        fileName1=input('Enter the source file name: ')
        fileName2=input('Enter the destination file name : ')
        fileCopy(fileName1, fileName2)

    if __name__ == '__main__':
        main()
```

```
Enter the source file name: studentMarks
Enter the destination file name : test
```

1.6 Computing moderated marks

- The file studentMarks contains the student data that includes roll number (rollNo), name (name), and marks (marks) for each student.
- The data about each student is stored in a separate line. Sample data in the file is shown below:

```
4001,Nitin Negi,75
4002,Kishalaya Sen,98
4003,Kunal Dua,80
4004,Prashant Sharma,60
4005,Saurav Sharma,88
```

- We define addPerCent as the percentage of maxMarks that should be added to the marks obtained to get the moderated marks, subject to the upper limit of maxMarks.

- The output file moderatedMarks containing moderated marks of the students

    1. Open file studentMarks in read mode.
    2. Open file moderatedMarks in write mode.
    3. Read one line of input (line1) from studentMarks.
    4. while (line1 != ''):

```
> Compute moderated marks and write one line of output in the file
```

moderatedMarks. > Read one line of input (line1) from studentMarks.

1.7 Problem: Compute moderated marks based on user input

```
In [1]: import sys
        def computeModeratedMarks(file1, file2, addPercent):
            '''
            Objective: To compute moderated marks of students
            Input Parameters: file1, file2: file names - string values
                              addPercent  numeric value
            Return Value: None
            Side effect: A new file  file2 of moderated marks is produced
            '''
            try:
                fIn = open(file1, 'r')
                fOut = open(file2,'w')
            except IOError:
                print('Problem in opening the file'); sys.exit()
            line1 = fIn.readline()
            while(line1 != ''):
                sList = line1.split(',')
                try:
                    rollNo = int(sList[0])
                    name = sList[1]
                    marks = int(sList[2])
                except IndexError:
                    print('Undefined Index'); sys.exit()
                except (ValueError):
                    print('Unsuccessful conversion to int'); sys.exit()
                maxMarks= 100
                moderatedMarks = marks+((addPercent*maxMarks) /100)
                if moderatedMarks > 100:
                    moderatedMarks = 100
                fOut.write(str(rollNo) + ',' + name + ',' +\
                str(moderatedMarks) + '\n')
                line1 = fIn.readline()
            fIn.close()
            fOut.close()
        def main():
            '''
```

```
        Objective: To compute moderated marks based on user input
        Input Parameter: None
        Return Value: None
        '''
        import sys
        sys.path.append('F:\PythonCode\Ch09')
        # To compute moderated marks of students
        file1 = input('Enter name of file containing marks:')
        file2 = input('Enter output file for moderated marks:')
        addPercent = int(input('Enter moderation percentage:'))
        computeModeratedMarks(file1, file2, addPercent)
    if __name__=='__main__':
        main()
```

```
Enter name of file containing marks:studentMarks
Enter output file for moderated marks:moderatedMarks
Enter moderation percentage:10
```

Tutorial_Session5

March 17, 2018

---

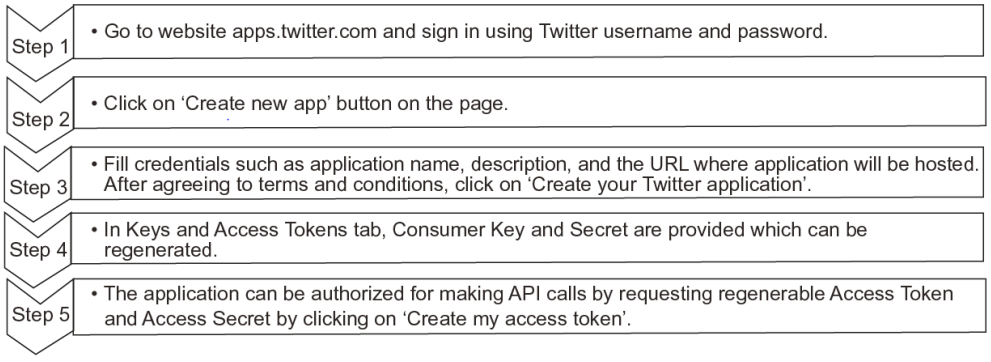
1 Twitter

- Twitter can also be used to retrieve information about the tweets, connections, or followers of a user, or topics trending on Twitter.
- Information is shared on Twitter in the form of tweets. A tweet may contain photos, videos, links, and up to 140 characters of text.
- Twitter provides several APIs (Application Programming Interfaces) to access Twitter data such as user's profile information or tweets
- An API request on Twitter is first authenticated using Open Authentication (OAuth). OAuth allows the users to connect to Twitter and send authorized Twitter API request.
- For performing analysis on Twitter data, we need the package tweepy.

  Commands for installing library tweepy

```
> pip install tweepy
```

Steps for Open Authentication



1.1 Collecting User's Information

```
In [1]: import tweepy

        def OAuthVerifier():
            '''
            Objective: To authorize the application to access Twitter
```

```
        Input Parameter: None
        Return Value: API object
        '''
        consumerKey = 'iym0XRGOSOgyPOjmlPQgB4XrC'
        consumerSecret = 'I6gBj8RpcXJvN6xaKUUGeTkPEDpHziRXBmT9d9yG8k5Ik0H0bF'
        authorization = tweepy.OAuthHandler(consumerKey, consumerSecret)
        accessToken = '727494459118653446-XWf9MmBJmCUOM7Ic9xvoHlZCBQAWWA1'
        accessSecret = 'jl4UJPXYSC8ygV8EsyNz6fMOW2NEs9NFvJc2InyfXNvve'
        authorization.set_access_token(accessToken, accessSecret)
        api = tweepy.API(authorization)
        return api

    def getUserStatistics(user):
        '''
        Objective: To get user statistics using various
                variables of the api
        Input Parameter: user - string
        Return Value: None
        '''
        print('\nName: ', user.name)
        print('Screen Name: ', user.screen_name)
        print('ID: ', user.id)
        print('Account creation date and time: ', user.created_at)
        print('Location: ', user.location)
        print('Description: ', user.description)
        print('No. of followers: ', user.followers_count)
        print('No. of friends: ', user.friends_count)
        print('No. of favourite tweets: ', user.favourites_count)
        print('No. of posted tweets: ', user.statuses_count)
        print('Associated URL: ', user.url)

    def main():
        '''
        Objective: To collect user information
        Input Parameter: None
        Return Value: None
        '''
        # To print user's information
        api = OAuthVerifier()
        # Authenticated User
        user = api.me()
        getUserStatistics(user)

    if __name__ == '__main__':
        main()
```

```
Name:  Suzzane Mathew
```

```
Screen Name:  SuzzaneMathew
ID:  727494459118653446
Account creation date and time:  2016-05-03 13:46:10
Location:  India
Description:  A Python Programmer
No. of followers:  4
No. of friends:  34
No. of favourite tweets:  0
No. of posted tweets:  3
Associated URL:  None
```

1.2 Collecting Tweets Having Specific Words

- **StreamListener class**: used for collecting streaming tweets.
- Class MyStreamListener inherits the class StreamListener of the tweepy module.
- In the class MyStreamListener, we define two methods: on_status and on_error.

  * The method on_status tells what to do when a status (input parameter) known as tweet update is received.
  * The method on_error handles the error and gets automatically invoked on occurrence of an error.

In [ ]:
```
import tweepy
```

```
class MyStreamListener(tweepy.StreamListener):
    # Class inheriting StreamListener of tweepy module

    def on_status(self, status):
        '''
        Objective: To print text stream of tweets
        Input Parameters:
            self (implicit parameter) - object of type
            MyStreamListener
            status - string value representing tweet
        Return Value: None
        '''
        print(status.text)

    def on_error(self, status):
        '''
        Objective: To disconnect the stream by returning False
        if error 420 occurs
        Input Parameters:
            self (implicit parameter) - object of type
```

```
                    MyStreamListener
                    status - int value representing error code
                Return Value: result - int
                '''
                if status==420:
                    return False

        def main():
            '''
            Objective: To print streaming data containing given keywords
            Input Parameter: None
            Return Value: None
            '''
            api = OAuthVerifier()
            # Creates a stream listener object listenerOb
            listenerOb = MyStreamListener()
            # Create a Stream object
            myStream = tweepy.Stream(api.auth, listenerOb)
            # Starts streaming by specifying search keywords
            searchList = eval(input('Enter search keywords list: '))
            myStream.filter(track = searchList)

        if __name__ == '__main__':
            main()
```

```
Enter search keywords list: ['Python','Programming']
RT: Forest Edge ES #ForestEdgeES #Elementary #School for #Robotics #Programming at Royal Cyber
RT: Sunrise Valley ES #SunriseValleyES #Elementary #School for #Robotics #Programming at Royal
RT @LoharPrasanna: Which Programming language should you learn ? https://t.co/WcwhuGg4Q8
2       -   ++

# #Programming #cplusplus
Diet-Microbiota Interactions Mediate Global Epigenetic Programming in Multiple Host Tissues - S
RT: Daniels Run ES #DanielsRunES #Elementary #School for 3D Game Programming (1-6th Grade) @ ST
RT: Clearview ES #ClearviewES #Elementary #School for 3D Game Programming (1-6th Grade) @ STEM
@python NowTime: 2018/03/17 03:09:23.3863
TweetCount: 319
RT: Cub Run ES #CubRunES #Elementary #School for #Robotics #Programming at Royal Cyber Club  ht
RT @sensitiveemmett: i know people don't like daylight savings time because it screws up their
The Top 9 Technology of 2016  #javascript #Python https://t.co/dPgXc3pcHe
RT @DIBIADream: DREAM is proud to present our 2017 Impact &amp;amp; Initiatives Report. Continuing
Zigbee Showcased at CES2017 Path to Unified.. #programming # https://t.co/iGxjc8RHzh
RT @allen_data: Python GUI Programming using Tkinter and Python 3

 https://t.co/UZktnwixdN

#hadoop #spark https://t.co/WYZ4mXuOc5
RT @TheBrando2: Monty Python and the Holy Grail (1975) https://t.co/purMRtV0vJ
```

Global IIoT Technologies, Solutions... #programming # https://t.co/RBuRZjmjrz
Connectivity Foundation and AllSeen Alliance Team up... #javascript #Python https://t.co/sEn4UI
Immediate Availability of ConnectCore for i.MX6UL Starter Kit.. #javascript #Python https://t
RT @enthought: You used to write #python like C code... @LEGOWorldsGame https://t.co/OhVNLd96IS
John Cleese taking a break on the set of Monty Python and the Holy Grail (1975) https://t.co/YI
VMs look to edge out gateways in push.. #javascript #Python https://t.co/oWBOOqapuA
RT: Fairfax Community Church for #Programming cool games #STEMCamps #Kids #Education https://t
RT: Fairfax Presbyterian Church for #STEM #Classes #Designing #Legos #Programming #Modeling #Ki
CVproof will connect with HR groups specifically and additionally through associations with rea
RT: Centreville Presbyterian Church for #STEM #Classes #Designing #Legos #Programming #Modeling
RT: Korean Central Presbyterian for #Robotics #Programming #Modeling #STEMCamps #Kids #Educatio
Public libraries invited to apply for The Great American Read PBS programming grants
https://t.co/oSzqpIxd0r
RT @elpais_cultura: Dice el ex Monty Python que sus integrantes son como la muchedumbre que asa
RT: Centreville Farms Community Association for #Robotics #Programming #Modeling #STEMCamps #Ki
Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state of boxing as a s
RT: Pat White Center at Ben Lomond for #STEM #Classes #Designing #Legos #Programming #Modeling
RT: Church of the Good Shepherd for #Programming cool games #STEMCamps #Kids #Education https
RT @Robzmob: #FridayMotivation ;) They have #Programming . . , but we have #Brains !! #TheStor
RT: Stenwood ES #StenwoodES #Elementary #School for 3D Game Programming (1-6th Grade) @ STEM Su
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
Which are the most popular programming languages at hackathons? #hackathon #javascript #reactjs
https://t.co/Z8R1xAKGAl
How to run #Capybara feature specs with #Selenium and headless #Chrome #ruby #rails #rubyonrail
Do you have Python 2 code that urgently needs to get migrated to Python 3? Just @anthonypjshaw
RT @JRLibrarian: As requested - here are four weeks of programming ideas and activities reflect
RT @Hakin9: A Practical Introduction to Blockchain with Python https://t.co/fwmWjrPH3a #infosec
RT @feminnazty: Yo shut up I remember losing a whole day of programming cause Nickelodeon wante
Ya kids will live while 17
STMicroelectronics Innovative.. #programming # https://t.co/lV66oc9FyD
New World of Intelligent Devices.. #programming # https://t.co/c0xE9qxHqi
RT @CodeWisdom: "Programming is the art of algorithm design and the craft of debugging errant c
RT: Center Ridge Region Home Owner for #Robotics #Programming Educational Classes For Kids htt
RT @DuncanGalbrait1: Monty Python - Life of Brian - PFJ Union meeting https://t.co/jVlH3BdUSW v
Commercial Craft Brewing Appliances for... #programming # https://t.co/9p9GGYiFzS
RT: Chantilly Highlands Community for #STEM #Classes #Designing #Legos #Programming #Modeling #
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
RT @abubakar47i: It mobilizes the smartest minds from >100 countries to apply their business
RT @KirkDBorne: Programming Languages for #DataScience and #MachineLearning (with samples of so
RT: London Towne ES #LondonTowneFCPS #Elementary #School for #Robotics #Programming at Royal Cy
RT @jeremynewberger: .@SeanHannity and @IngrahamAngle scolding @ShepNewsTeam on how newsy their
@python NowTime: 2018/03/17 03:10:33.7043
TweetCount: 321
RT: Stacy C. Sherwood Community Center for #Robotics #Programming Educational Classes For Kids
Practical Python Data Science Techniques

 https://t.co/wRehxtDtxn

#bigdata https://t.co/eC9H5UTA65
RT @lucasoft_co_uk: RT @Hakin9: A Practical Introduction to Blockchain with Python https://t.co
RT: Brookfield ES #BrookfieldES #Elementary for #Programming #MinecraftModding at Royal Cyber C
RT: Herndon ES #Herndon_ES #Elementary #School for 3D Game Programming (1-6th Grade) @ STEM Sum
RT: Herndon Community Center for #Robotics #Programming #Modeling #STEMCamps #Kids #Education
RT: Grace Covenant Church for #Robotics #Programming #Modeling #STEMCamps #Kids #Education htt
RT: Centreville Community Church for #Robotics #Programming Educational Classes For Kids https
Yay for cool graphs and charts in your reports! https://t.co/8Mgyx6yymE
RT: Grace Covenant Church for #Programming cool games #STEMCamps #Kids #Education https://t.co
@IngrahamAngle No he is right. Your programming is illlegit propaganda
RT: Greenbriar Community Center for #Robotics #Programming Educational Classes For Kids https
RT @FoxBusiness: Programming Alert: Peter Thiel sits down with @MariaBartiromo for an exclusive
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
#OpenGL #Programming #Math #Matrix #CPP #Developers #GLSL Reminder: your regular modelview (MV)
RT: Lord of Life Lutheran Church for #Robotics #Programming Educational Classes For Kids https
RT: Centreville Presbyterian Church for #STEM #Classes #Designing #Legos #Programming #Modeling
RT @ReaktorNow: Together with @helsinkiuni we are developing a free online course called the El
RT @SecurityTube: Python for Pentesters: Directory Navigation https://t.co/8yaK0A4TO9 https://t
RT: Daniels Run ES #DanielsRunES #Elementary #School for 3D Game Programming (1-6th Grade) @ ST
RT @Aly_Raisman: Please join the effort to address sexual abuse in sport. I have partnered with
QISKit: quantum computing with python https://t.co/ut0D9Sbrv6 via @IBMCode
RT: Center Ridge Region Home Owner for #Robotics #Programming #Modeling #STEMCamps #Kids #Educa
RT: Aldrin ES #AldrinEagles #Elementary #School for 3D Game Programming (1-6th Grade) @ STEM Su
RT @Mason9780: New technology is now making it possible for viewers to record and store high de
RT @christi3k: I'm looking for my next gig as a software engineer or developer advocate in Port
RT: Centreville Baptist Church for #Robotics #Programming Educational Classes For Kids https:/
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
Comienza #BetaBeersODB con @JAR_omero hablándonos de Django #python https://t.co/FfQgR8R7cA
Over 1600 students in 22 programs... 75 in #engineering... plus 100% of last year's welding gra
I'm excited to attend "Using Scratch Programming to Teach World Language" at CABE 2018. Come wi
RT: Sully Station Community Center for #Robotics #Programming Educational Classes For Kids htt
RT: Powell ES #ColinPowellES #Elementary #School for #Robotics #Programming at Royal Cyber Club
RT @Hakin9: A Practical Introduction to Blockchain with Python https://t.co/m8vK9WUzHU #infosec
RT: Centreville Community Church for #STEM #Classes #Designing #Legos #Programming #Modeling #R
RT @svpersof: Hello everyone! I got accepted into my study abroad program to learn about intern
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
RT @KitPloit: Powershell-RAT - Python Based Backdoor That Uses Gmail To Exfiltrate Data Through
@mia_0327 https://t.co/oopF1g9ThH

RT: Aldrin ES #AldrinEagles #Elementary for #Programming #MinecraftModding at Royal Cyber Club
RT @chris__martin: The "domain object" pattern was one of the most destructive forces at my fir
RT: Floris United Methodist Church for #STEM #Classes #Designing #Legos #Programming #Modeling
RT @mocityschool: Did you see @TheSpec article about our Intro to Construction for Women cours
RT: Clearview ES #ClearviewES #Elementary for #Programming #MinecraftModding at Royal Cyber Clu
RT: Daniels Run ES #DanielsRunES #Elementary #School for 3D Game Programming (1-6th Grade) @ ST
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
RT @BigDataFr_: Offre de Stage HPC Ingénieur/Bac+5 Maths-Info

Cuda, Python, NumPy, Scikit-Learn, Keras, Caffe, https://t.co/z4ex4T1KRA
https://t.co/meDjQfnVam [Programming & Design] Open Question : SQL database in trouble? #
fun time remote pair programming to practice C++ TDD
with a @TKPJava graduate https://t.co/0uzn8r2aOS
@python NowTime: 2018/03/17 03:11:44.773
TweetCount: 323
RT: Sully Station Community Center for #Robotics #Programming Educational Classes For Kids htt
This is going to strangle entrepreneurs and small sellers. A $12 Etsy sale will now not only ha
The new Conditional Types capabilities in @typescriptlang 2.8 move the language even more firml
RT: Virginia Run ES #VirginiaRunES #Elementary for #Programming #MinecraftModding at Royal Cybe
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
We're at a standstill at work because my boss's boss never gives us a programming budget and ju
RT: Crossfield ES #CrossfieldES #Elementary for #Programming #MinecraftModding at Royal Cyber C
RT: Greenbriar Community Center for #Robotics #Programming #Modeling #STEMCamps #Kids #Educatio
Excellent! Thanks for PRIMM-ing!! @KingsECS https://t.co/2QjyawHVU4
Canterbury Woods ES #Canterbury_Wood #Elementary #School for 3D Game Programming (1-6th) @ STEM
RT @gethackteam: Which are the most popular programming languages at hackathons? #hackathon #ja
https://t.co/Z8R1x
RT: Summer Camp registration is open! STEM, Minecraft, Robotics, Game Design, Game Programming
@JDM1114 & @GregHughesNBCSG You guys are better than this, right? If not, we can help. #sum
RT @saeedamenfx: the master of Python @dyjh speaking! Great talk, always learn something new fr
Real Good Community Programming only on 91.1 The Bridge! https://t.co/2mQ6K9uLw2 on #Podbean
Real Good Community Programming only on 91.1 The Bridge! https://t.co/qs54crPbj5 on #Podbean
Real Good Community Programming only on 91.1 The Bridge! https://t.co/R6Qzg2fYaz on #Podbean
RT @lucasoft_co_uk: RT @Hakin9: A Practical Introduction to Blockchain with Python https://t.co
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
Thanks for having us! https://t.co/CJTJfpVhHd
I wish I had more people to work out with. I love seeing different people's programming and goa
RT: #STEM #Robotics #Programming #Modeling #STEMCamps #Kids #Education at Royal Cyber Club htt
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
RT @LorenaABarba: I'm making an open online course with the first module of our computing cours
RT @jcldf: Python based backdoor that uses Gmail to exfiltrate data as an e-mail attachment.

This RAT will help someone during red team en
Por isso que eu digo que os caras do Monty Python eram gênios. https://t.co/vT7uGGUxmh
I know I complain a lot, and freak out a lot, but let me first say that I am blessed to have a
RT: Poplar Tree ES #PoplarTreeES #Elementary #School for #Robotics #Programming at Royal Cyber
Practical Python Data Science Techniques

 https://t.co/ndqjsgz0pp

#bigdata https://t.co/54gkBmNjMD
RT @CodeWisdom: "Everyday life is like programming, I guess. If you love something you can put
RT @SecurityTube: Python for Pentesters: Directory Navigation https://t.co/8yaK0A4TO9 https://t
RT @dustmop: Hi! I'm a full-stack / backend freelance engineer looking for part-time contract w
RT @mlemweb: Our Digital Humanities Workshop is tomorrow! @dustyweb and I are finalizing our tu
RT: Southgate Community Center for #Robotics #Programming #Modeling #STEMCamps #Kids #Educatio
Our Education Program partners with Inwater Research Group to promote quality educational progr

RT @TalkPython: Do you have Python 2 code that urgently needs to get migrated to Python 3? Just
RT: Aldrin ES #AldrinEagles #Elementary #School for #Robotics #Programming at Royal Cyber Club
@briangcronin That's the main ingredient to making great programming videos -- great content an
RT @Onitaset: New working theme: "Today we will discuss how you are failing Black people and wh
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
RT @jdmorenob: ąDiario Lenguajes de programación! https://t.co/RE7n2nBgni #in Gracias a @daniel
Join Jules Damji (@2twitme) @ApacheSpark & Community Advocate at @Databricks to learn how t
RT @mlemweb: Our Digital Humanities Workshop is tomorrow! @dustyweb and I are finalizing our tu
RT: Reston Community Center for #STEM #Classes #Designing #Legos #Programming #Modeling #Kids #
@scuttlingvoid prob a piebald ball python ye !!
i always feel so powerful learning programming but then one mistake happens that wrecks all my
RT @ReaktorNow: Together with @helsinkiuni we are developing a free online course called the El
I just liked Introduce 'JS Weight tracer' Python Effector by @oinon on #Vimeo: https://t.co/20x
@__Zanele Oh no. Imagine being a python just nje.
For even more #Python training, come to my sessions at #FedGIS next week! https://t.co/cV9uUj9U
7 Ridicuous Complaints the FCC Received About WWE's Programming https://t.co/Hm39lzhZNp https:/
Mastering Bitcoin: Programming the Open Blockchain 2nd Edition, (Kindle Edition) https://t.co/i
Deciding between a beardie or a ball python is one of the hardest decisions ive ever made
RT: Centreville Community Church for #STEM #Robotics #Programming #Modeling #STEMCamps #Kids #P
Michael Palin is the only good, nice boy from Monty Python
RT: Forest Edge ES #ForestEdgeES #Elementary #School for #Robotics #Programming at Royal Cyber
RT @CyberToolsBooks: Pro Bash Programming teaches you how to effectively utilize the Ba https:/
@annehelen @shannoncoulter Thats been my opinion for a number of years, sadly. Its really taint
Modern Serverless Python on Coherence API https://t.co/Hgtsd0Y6ne
Keyword Generator https://t.co/Ic71uaiuFV
@john_overholt The least known member of Monty Python finally makes headlines. He shoukd have s
RT: McNair Farms Community Center for #Programming cool games #STEMCamps #Kids #Education http
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
It's the clarity of F#'s type and function signatures that I miss the most when programming in
@BenedictEvans I do all my programming now off a macbook pro with the touchpad, full screen win
RT @RealDebrid: @TeamZT_ @jsergio123 The guy is just a piece of shit providing a shitty fix the
RT @SetonMontessori: Jennifer Nolan contributed a beautiful blog post about our intergeneration
RT: Reston Community Center for #STEM #Robotics #Programming #Modeling #STEMCamps #Kids #Educat
RT @abubakar47i: It mobilizes the smartest minds from >100 countries to apply their business
python
Macromedia Mx Elearning: Advanced Training From The Source https://t.co/3T2VxqJYJ8 #Ruby #prog
Python Networking: FTP Programming https://t.co/oHutevQjI9
RT @lamifunami: #SignalsNetwork
Signals is developing a platform for building, training and monetizing crypto trading strategie
@UKRobotWars Im sure theyll fund some educational programming instead, say more baking or taler
RT: Willow Springs ES #WSESfox #Elementary for #Programming #MinecraftModding at Royal Cyber Cl
RT: Cherry Run ES #CherryRunES1 #Elementary #School for 3D Game Programming (1-6th Grade) @ STE
@1776Stonewall @IngrahamAngle @FoxNews I'm down to @cvpayne & @LouDobbs ...thats it for any
@YYCBoarder @NewWorldHominin @FaithGoldy Invading a country using immigration and neuro linguis
      !! @BCBtigers all players bites like a python! #BANvSL #SLvBAN https://t.co/Fvg0oJ4w88
RT @Hakin9: A Practical Introduction to Blockchain with Python https://t.co/m8vK9WUzHU #infosec
RT @sensitiveemmett: i know people don't like daylight savings time because it screws up their
RT: Greenbriar Community Center for #Programming cool games #STEMCamps #Kids #Education https

RT: Little Run ES #LRroadrunner #Elementary #School for 3D Game Programming (1-6th Grade) @ STE
Most Python fans arent Trump fans. https://t.co/WVnni846Dj
Why is Python so good for AI and Machine learning?

https://t.co/dwYr04TMiC
@PegCoversTheLaw @steveury Thank you for these midwinter meeting tweets! ABALEL Committee midwi
Pandas Cookbook: Recipes For Scientific Computing, Time Series Analysis And Data https://t.co
RT @SecurityTube: Python for #Pentesters: Introduction to Python and Setting Up an Environment
@python NowTime: 2018/03/17 03:14:06.045
TweetCount: 327
Introduction to the Python 3 Programming Language - #python course by @wfbutton https://t.co/XS
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
RT: Deer Park ES #DeerParkFCPS #Elementary #School for #Robotics #Programming at Royal Cyber Cl
High Performance Python: Practical Performant Programming For Humans https://t.co/3oCq4x44p4 #
@rickathy227 @talace @cheechablunt @Nickelodeon @DLoesch (Someone might want to warn her that h
RT @lfcseattle: @JDM1114 & @GregHughesNBCSG You guys are better than this, right? If not, w
RT @fabienLeboeuf: Need to alter Hip joint centre positions of the Conventional Gait Model (#CG
@santome7 @thehill Smith is fact based programming. Hannity is opinion based programming. Its a
RT @jeffgiesea: ESPN infused its programming with "social justice," ratings declined, and now w
RT @CsChapters: Hurry up! Today is the last day to register for @CsharpCorner Gurgaon Chapter M
RT @ITsax: FPGA-Entwickler / VHDL-Entwickler (m/w) in Freital/Dresden in Freital , #FPGA #VHDL
RT: Greenbriar Community Center for #Robotics #Programming #Modeling #STEMCamps #Kids #Educatio
C/C++ Programming Course Near Sector-72 C and C++ Programming are mu..For more info visit... ht
RT: Canterbury Woods ES #Canterbury_Wood #Elementary #School for 3D Game Programming (1-6th) @
RT: McNair ES #FCPSMcNairES #Elementary for #Programming #MinecraftModding at Royal Cyber Club
RT @RichRogersIoT: Programming is about 13% math, 15% theology and the rest is just looking stu
System engineers also do programming.
RT @UPROXX: Netflix has a lovely stockpile of action television programming to binge through. M
RT: Greenbriar Community Center for #Robotics #Programming Educational Classes For Kids https
RT: Korean Central Presbyterian for #STEM #Classes #Designing #Legos #Programming #Modeling #Ki
#redfarmers https://t.co/mOef9bxmbR
RT: Powell ES #ColinPowellES #Elementary #School for 3D Game Programming (1-6th Grade) @ STEM S
RT: McNair Farms Community Center for #Robotics #Programming #Modeling #STEMCamps #Kids #Educat
@navytheplug Yep, had a couple.

In church, business, life coach and programming
There is a need to integrate climate change issues into the development planning process at all
RT @YourSmartyBro: Python GUI Programming Projects using Tkinter and Python 3 https://t.co/ENsM
RT @christi3k: I'm looking for my next gig as a software engineer or developer advocate in Port
RT: Centreville Baptist Church for #Programming cool games #STEMCamps #Kids #Education https:/
RT @FoxBusiness: Programming Alert: Boxing promoter #DonKing joins @cvpayne to discuss he state
RT @bheklilr: I'm a mid-level Python developer with plenty of experience in data analysis with
RT: Centreville Community Church for #STEM #Classes #Designing #Legos #Programming #Modeling #P
Week in Review: Oct 30 Nov 5 #programming # https://t.co/Ef6AwOzOyY
RT @oliverdarcy: Hate to sound like a broken record, but this once again underscores the divide
RT @WWEGraves: Im terribly sorry that your parents didnt love you enough. Here is some of the a
Modify your own #reports in #Microsoft #DynamicsNAV & stay #supported! Find out more about
RT: Powell ES #ColinPowellES #Elementary #School for 3D Game Programming (1-6th Grade) @ STEM S

RT: Grace Covenant Church for #STEM #Robotics #Programming #Modeling #STEMCamps #Kids #Educatio
RT @oliverdarcy: Hate to sound like a broken record, but this once again underscores the divide
If you're looking for a job for next semester, look no further!
Applications for Campus Activities Programming Boar https://t.co/qEj3befIu5
RT: Pat White Center at Ben Lomond for #Robotics #Programming #Modeling #STEMCamps #Kids #Educa
RT: Fairfax Villa ES #FairfaxVillaES #Elementary for #Programming #MinecraftModding at Royal Cy
Emploi Societe Generale SGCIB : Developpeur Fonctionnel Commando Risque et PnL H/F... https://t
Twitter wants to promote more on-demand viewing as it seeks new deals for live shows ahead of t
RT @AML_WorldWide: Join Us In @KCMO #KansasCityMO for Your 50th Anniversary Miss Black America
"BETA PROGRAMMING" SINGLE + VIDEO OUT NOW!

"SAVAGE BEAUTY" DROPS MARCH 23 on @elemental_95

https://t.co/UuwYwFPwS3
Next #post form my series about #Programming #DesignPatterns today about #Prototype #code in #j
C/C++ Programming Course Near Sector-74C and C++ Programming are mus..For more info visit... ht
Understanding gender roles in agriculture, making programming accessible for all. https://t.co/
RT @pvanheus: I finished teaching my Introduction to Python for Bioinformatics course for the y
RT: Centreville Community Church for #Robotics #Programming Educational Classes For Kids https
RT: Centreville Baptist Church for #STEM #Classes #Designing #Legos #Programming #Modeling #Kid
PythonLINE (1) - https://t.co/60QxKSEsIx
RT @qiita_python: PythonLINE (1) - https://t.co/60QxKSEsIx
RT: Compton Village Homeowners for #Robotics #Programming Educational Classes For Kids https:/
RT: Sully Station Community Center for #Robotics #Programming #Modeling #STEMCamps #Kids #Educa
RT: #Robotics #Programming Educational Classes For Kids at Royal Cyber Club https://t.co/iD6Zu
#fortysevenbank, #gofortyseven https://t.co/vC2r3uKZl8
RT @CodeWisdom: "Programming is the art of algorithm design and the craft of debugging errant c
RT @wimlds: Plotcon Boston is happening April 14-15. There are 2-day workshops for:
1) Dash
2) R, Shiny and DashR
3) Getting Started wi
@psykogrl @Michael_Fisher_ @gdlittledorf @Mikel_Jollett It was when contracts with CNN ran out
RT: Centreville Farms Community Association for #STEM #Robotics #Programming #Modeling #STEMCam
RT: Herndon ES #Herndon_ES #Elementary #School for #Robotics #Programming at Royal Cyber Club
RT @byLilyV: TRENDING #UDEMY #COURSES

#Python & #PostgreSQL Developer Course

Build 9 projects master 2 essential and modern technologies
Using yield is an easy way to make collections lazy #csharp ã#devops #webops #aspnet #coding #p
Another informative and empowering @WCPS72 #PreK Family Oriented Programming session at the bea
RT: Stenwood ES #StenwoodES #Elementary for #Programming #MinecraftModding at Royal Cyber Club
RT: Grace Covenant Church for #Robotics #Programming #Modeling #STEMCamps #Kids #Education htt
RT @EndorseEm: RT: Pat White Center at Ben Lomond for #Robotics #Programming #Modeling #STEMCam
RT: Fairfax Presbyterian Church for #STEM #Classes #Designing #Legos #Programming #Modeling #Ki
RT @dustmop: Hi! I'm a full-stack / backend freelance engineer looking for part-time contract w
Thank you @HybargerMarsha for coming to talk to us about programming offered at the GTC!   #HIT
C/C++ Programming Course Near Sector-75 C and C++ Programming are mu..For more info visit... ht

```
RT @mlemweb: Our Digital Humanities Workshop is tomorrow! @dustyweb and I are finalizing our tu
hey that's my mom https://t.co/wPqYCjutFl
RT: Sully Station Community Center for #Programming cool games #STEMCamps #Kids #Education  htt
RT @EqualToken: Congratulations to @contestsinkie for completing the programming challenge, rev
RT: Centreville Farms Community Association for #Programming cool games #STEMCamps #Kids #Educa
RT @MSFTImagine: A new definition in #programming. #DevHumor  https://t.co/OqmRdIPvzX
though, OK, lets be real: monty python as a group *all* hated women, aggressively, so none of t
@HuffPost NO WAY a Trump support could understand Python!
#Boston fans: #BernsteininBoston Don't miss 3/16 and 3/17 8pm as Giancarlo Guerrero conducts @N
```

Tutorial_Session6

March 17, 2018

1 CLASSES

- A class is a template that provides a logical grouping of data and methods that operate on them.
- Instances of a class are called objects.
- Data and methods associated with a class are collectively known as class attributes.

1.1 Classes and Objects

- Variables used so far took values of types (also called classes) string (str), integer (int), floating point (float), Boolean (bool), list, tuple, or dictionary (dict).

```
In [3]: print(type(12), type(12.5), type('hello'))
```

```
<class 'int'> <class 'float'> <class 'str'>
```

** Accessing attribute of an instance of class** * To specify an attribute of a class (or class instance), we write the name of the class (or class instance) followed by a dot, followed by the name of that attribute. The method **lower** defined in class **str** has been invoked for the object **name**.

```
In [4]: name = 'Raman'
        lname = name.lower()
        print(lname)
```

```
raman
```

Alternative way of invoking the method associated with an instance of class: * Specify the name of the class (str), followed by the dot operator (.), followed by the name of the method (lower), followed by an object (name). The object name being an argument is enclosed in parentheses.

```
In [5]: lname = str.lower(name)
        print(lname)
```

```
raman
```

1.2 PERSON class

1.2.1 Syntax of Class Definiton

A class definition begins with the keyword class followed by the name of the class, and a colon. By convention, the first letter of the class name is capitalized. The syntax for class definition is as follows:

```
class ClassName:
    classBody
```

1.2.2 Operations supported by classes:

1. **Instantiation**: It refers to the creation of an object, i.e. an instance of the class.
2. **Attribute references**: Methods and data members of an object of a class are accessed using the notation: name of the object, followed by dot operator, followed by the member name.

```
In [5]: class Person:
            ''' The class Person describes a person'''
            count = 0
            def __init__(self, name, DOB, address):
                '''
                Objective: To initialize object of class Person
                Input Parameters:
                    self (implicit parameter) - object of type Person
                    name - string
                    DOB - string (Date of Birth)
                    address - string
                Return Value: None
                '''
                self.name = name
                self.DOB = DOB
                self.address = address
                Person.count += 1

            def getName(self):
                '''
                Objective: To retrieve name of the person
                Input Parameter: self (implicit parameter) - object of type Person
                Return Value: name - string
                '''
                return self.name

            def getDOB(self):
                '''
                Objective: To retrieve the date of birth of a person
                Input Parameter: self (implicit parameter) - object of type Person
```

```
        Return Value: DOB - string
        '''
        return self.DOB

    def getAddress(self):
        '''
        Objective: To retrieve address of person
        Input Parameter: self (implicit parameter) - object of type Person
        Return Value: address - string
        '''
        return self.address

    def getCount(self):
        '''
        Objective: To get count of objects of type Person
        Input Parameter: self (implicit parameter) - object of type Person
        Return Value: count: numeric
        '''
        return Person.count

    def __str__(self):
        '''
        Objective: To return string representation of object of type Person
        Input Parameter: self (implicit parameter)- object of type
        Person
        Return Value: string
        '''
        return 'Name:'+self.name+'\nDOB:'+str(self.DOB)\
        +'\nAddress:'+self.address
```

1.2.3 Creating an instance of class Person

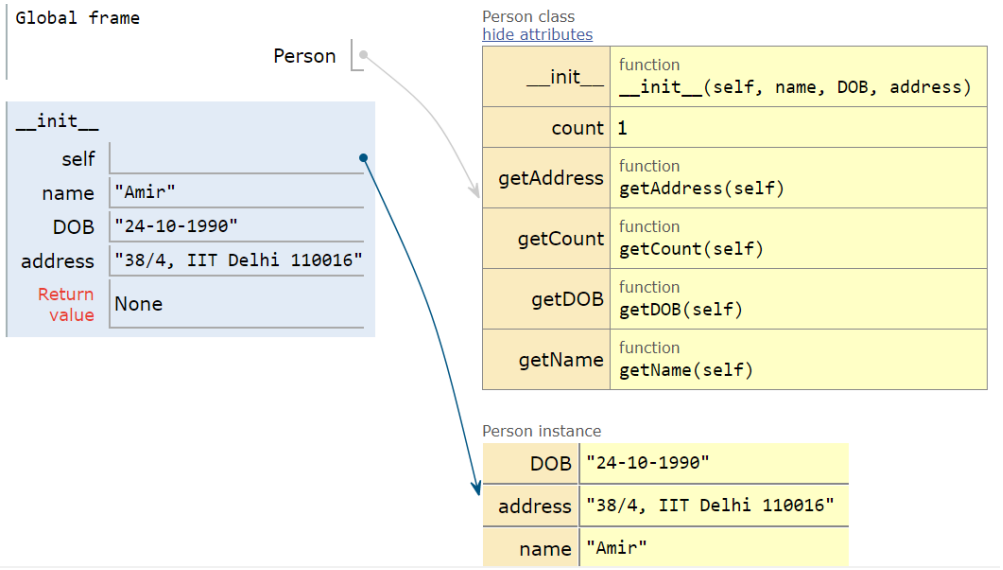
```
In [6]: p1 = Person('Amir','24-10-1990','38/4, IIT Delhi 110016')
        print(Person.count)
```

```
1
```

The execution of the above statement does three things: 1. Creates an instance of class Person 2. Initializes it by invoking the method **__init__** defined in lines 3. Returns a reference to it, so the name p1 now refers to the instance of the class Person that has just been created

By default, Python passes object itself (such as p1) as the first argument to the method **__init__**.

1.2.4 Instance p1 of class Person



1.2.5 Printing an instance of class

Python invokes **__str__** method of the corresponding class to obtain a string representation of the object

```
In [61]: print(7)
         # OR
         print(int.__str__(7))
         print(str(7))
```

```
7
7
7
```

```
In [151]: print(p1)
          print('******************')
          print(p1.__str__())
          print('******************')
          print(Person.__str__(p1))
          print('******************')
          print(str(p1))
```

```
Name:Amir
DOB:24-10-1990
Address:38/4, IIT Delhi 110016
******************
Name:Amir
DOB:24-10-1990
Address:38/4, IIT Delhi 110016
```

```
*****************
Name:Amir
DOB:24-10-1990
Address:38/4, IIT Delhi 110016
*****************
Name:Amir
DOB:24-10-1990
Address:38/4, IIT Delhi 110016
```

```
In [63]: p1.getDOB()
```

```
Out[63]: '24-10-1990'
```

1.2.6 List of attributes of the object

```
In [64]: dir(p1)
```

```
Out[64]: ['DOB',
          '__class__',
          '__delattr__',
          '__dict__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__le__',
          '__lt__',
          '__module__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__setattr__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          '__weakref__',
          'address',
          'count',
          'getAddress',
          'getCount',
```

```
          'getDOB',
          'getName',
          'name']
```

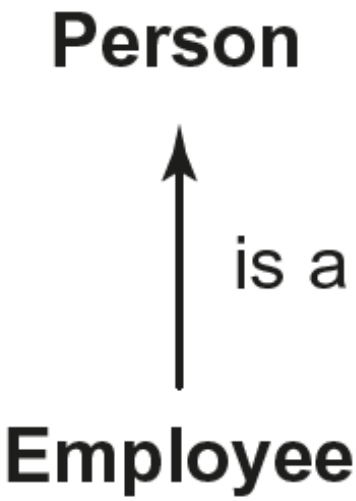
1.2.7 Deleting an object of class Person

```
In [7]: def __del__(self):
            '''
            Objective: To be invoked on deletion of an instance of the
            class Person
            Input Parameter:
            self (implicit parameter)  object of type Person
            Return Value: None
            '''
            print('Deleted !!')
            Person.count -= 1

In [8]: p1.__del__ = __del__

In [9]: del p1

In [10]: print(p1)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-10-71a0f0e933fe> in <module>()
    ----> 1 print(p1)


        NameError: name 'p1' is not defined
```

1.3 Inheritance

- Inheritance is an important feature of object oriented programming that imparts ability to a class to inherit properties and behavior of another class



- In the language of Object-oriented Programming (OOP), we say that Employee class inherits or derives the data and method attributes from the Person class.
- Here, Person class is called base, super, or parent class, and Employee class is called derived, sub, or child class.

1.3.1 Single Inheritance

- When inheritance involves a derived class that derives its properties from a single base class, it is called **single inheritance**

```
In [36]: class Employee(Person):
             nextId = 1001
             empCount = 0

             def __init__(self, name, DOB, address, basicSalary, dateOfJoining):
                 '''
                 Objective: To initialize an object of class Employee
                 Input Parameters:
                     self (implicit parameter)  object of type Employee
                     name - string, address  string
                     DOB - Date of Birth  object of type MyDate
                     basicSalary - numeric value
                     dateOfJoining  object of type MyDate
                 Return Value: None
                 '''
                 Person.__init__(self, name, DOB, address)
                 self.idNum = Employee.nextId
                 self.basicSalary = basicSalary
                 self.dateOfJoining = dateOfJoining
                 Employee.nextId += 1
                 Employee.empCount += 1
```

```
    def getId(self):
        '''
        Objective: To retrieve id of the Employee
        Input Parameter: self (implicit parameter) object of type Employee
        Return Value: id - numeric value
        '''
        return self.idNum

    def getSalary(self):
        '''
        Objective: To retrieve salary of the Employee
        Input Parameter: self (implicit parameter) - object of type Employee
        Return Value: basicSalary - numeric value
        '''
        return self.basicSalary

    def reviseSalary(self, newSalary):
        '''
        Objective: To update salary of the Employee
        Input Parameters: self (implicit parameter) - object of type Employee
        newSalary - numeric value
        Return Value: None
        '''
        self.basicSalary = newSalary

    def getJoiningDate(self):
        '''
        Objective: To retrieve joining date of the Employee
        Input Parameter: self (implicit parameter) - object of type Employee
        Return Value: dateOfJoining - object of type MyDate
        '''
        return self.dateOfJoining

    def __str__(self):
        '''
        Objective: To return string representation of object of type
        Employee.
        Input Parameter: self (implicit parameter) - object of type Employee
        Return Value: string
        '''
        return Person.__str__(self)+'\nId:'+str(self.getId())+\
            '\nSalary:'+str(self.getSalary())+\
            '\nDate of Joining:'+str(self.getJoiningDate())
```

```
In [37]: emp1 = Employee('Rehman', '5 June 1990', ' D-9, Vivek Vihar, Delhi', 50000, '2 August
```

```
In [38]: print(Employee.empCount)
```

```
1
```

- Call to the method **init** is made using a superclass name and the object instance is explicitly passed as an argument to the superclass method.
- Alternatively, we may use the **super** function to access a method of the superclass.

```
super(Employee, self).__init__(name, DOB, address)
super().__init__(name, DOB, address)
```

1.4 5. Built-in Functions for Classes

1.4.1 Function issubclass

- The function issubclass returns True if sub is the subclass of class super, and False otherwise.

```
issubclass(sub, super)
```

```
In [146]: issubclass(Employee, Person)
```

```
Out[146]: True
```

1.4.2 Function isinstance

- The function isinstance returns True if either obj is an instance of class class1 or it is an instance of a subclass of class class1.

```
isinstance(obj, class1)
```

```
In [147]: isinstance(emp1, Person)
```

```
Out[147]: True
```

1.4.3 Function hasattr

- The function hasattr returns True if instance obj contains an attribute attr, and False otherwise.

```
hasattr(obj, attr)
```

```
In [149]: hasattr(emp1, 'dateOfJoining')
```

```
Out[149]: True
```