Class – XI

Subject: Computer Science

UNIT- 1 CBSE

Boolean Algebra

cbseacademic.nic.in/web_material/doc/cs/2_Computer_Science_Python_ClassXII.pdf

Unit – 4 Introduction to Boolean Algebra

Chapter -1 and 3

The notion of variable and concept of L-value and R-value

Reference :

- NCERT, Computer Science Textbook for Class XI, Chapter 5: Getting Started with Python
- CBSE Text Book, Computer Science, Class XI, Unit -3, Chapter-1: Getting Started

Variable

Variable, in Python, is an object that is used to store values viz. numeric (e.g., 345), string (e.g., 'Python') or any combination of alphanumeric characters (e.g., CD67). In Python, we can use an assignment statement to assign specific value to a variable.

Example :

```
>>> count = 100
>>> count
100
>>>
```

Here, 100 is assigned to newly created variable count.

Write a Python Program to illustrate the use of variables in calculating simple interest.

```
principal = 1000 #Variable named principal to store principal amount
rate      = 10   #Variable named rate to store rate of interest (in % per annum)
time      = 2    #Variable named time to store time duration in years

simple_interest = (principal*rate*time)/100

print("The value for Principal is : ",principal)
print("The value for rate is : ",rate)
print("The value for time is : ",time)

print("The value of Simple Interest is : ",simple_interest)
```

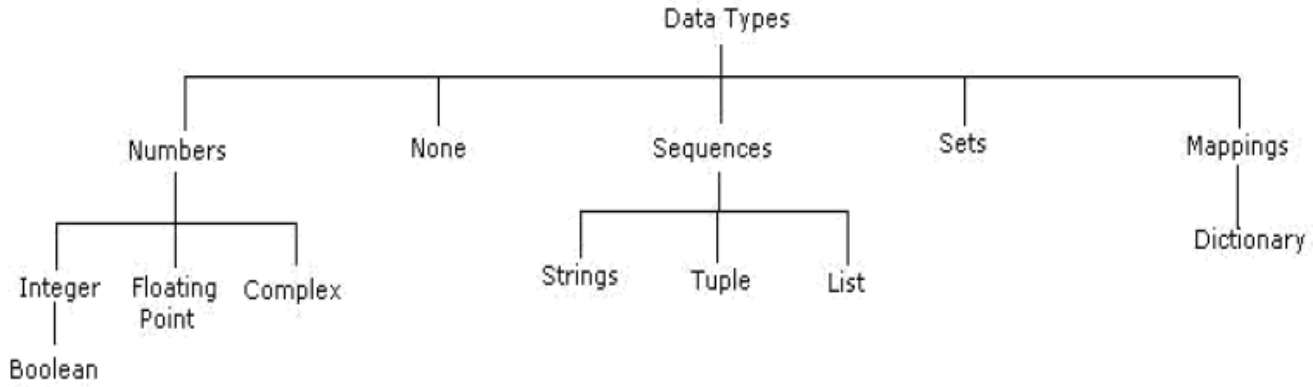
```
========== RESTART: C:\Users\p88si\Desktop\CLASS_11\Variable_Demo.py ==========
The value for Principal is :  1000
The value for rate is :  10
The value for time is :  2
The value of Simple Interest is :  200.0
>>>
```

In Python, every object has

- An Identity - can be known using id(object)
- A Type - can be known using type(object)
- A Value

Identity of the object:
It is the object's address in memory and does not change once it has been created.

Type:
It tells the data type of the object.



Value:
The value assigned to the object. To bind a value to a variable, we use the assignment operator (=).

A variable name:

- Allowed characters : a-z, A-Z, 0-9 and underscore (_)
- should begin with an alphabet or underscore.
- should not be a keyword.

It is a good practice to follow the following naming conventions:

- A variable name should be meaningful and short.
- Generally, they are written in lower case letters.

Write a Python Program to illustrate the use of variables in calculating simple interest and print Identity, Type and Value of each variable used.

```
principal = 1000 #Variable named principal to store principal amount
rate      = 10   #Variable named rate to store rate of interest (in % per annum)
time      = 2    #Variable named time to store time duration in years

s_int = (principal*rate*time)/100

print("Name of Variable : principal  id : {}, type : {}, value : {}".format(id(principal), type(principal), principal))
print("Name of Variable : rate  id : {}, type : {}, value : {}".format(id(rate), type(rate), rate))
print("Name of Variable : time  id : {}, type : {}, value : {}".format(id(time), type(time), time))
print("Name of Variable : s_int  id : {}, type : {}, value : {}".format(id(s_int), type(s_int), s_int))
```

```
======================== RESTART: C:\Users\p88si\Desktop\CLASS_11\Variable_Demo1.py ========================
Name of Variable : principal  id : 69279888, type : <class 'int'>, value : 1000
Name of Variable : rate  id : 1347745856, type : <class 'int'>, value : 10
Name of Variable : time  id : 1347745728, type : <class 'int'>, value : 2
Name of Variable : s_int  id : 66633312, type : <class 'float'>, value : 200.0
>>>
```

L-value and R-value concept

In any assignment statement, the assignable object (e.g. variable) must appear on the left side of the assignment operator else python will report an error. In this case, the assignable object is called L-value.

```
a = 10
b = 20
c = 30

b+c = a     #invalid L-value object, error : cannot assign to operator
print("The sum of b and c is ",b+c)

50 = b       #invalid L-value object error : cannot assign to literal
print("The Value of b is ",b)
```

The value to be assigned (or any expression producing value) must appear on the right side of the assignment operator. In this case, the value (or expression resulting in value) is called R-value.

```
a = 10
b = 20
c = 30

a = b+c      #a is l-value and (b+c) is r-value
print("The Value of a is ",a)

b = 40       #b is l-value and 40 is r-value
print("The Value of b is ",b)
```

CBSE – Sorting

cbseacademic.nic.in/web_material/doc/cs/2_Computer_Science_Python_ClassXII.pdf

Unit-2: Advanced Programing with Python

Page number 99 t0 102

Class - XII

Subject: Computer Science

Changed portion from NCERT

Functions in continuation to NCERT chapter – 7 (Class – 11)

Passing List, Tuple & Dictionary to a Function, Positional Parameter

Passing a List to a Function

Like we pass numbers and strings to a function, similarly, we can pass List to a function. The following example illustrates the same.

Write a program to input a word having Upper Case Alphabets only. Store each letter in a list and pass this list to a function that replaces each letter of the word by a letter three places down the alphabet i.e. each 'A' will be replaced by 'D', each 'B' will be replaced by 'E' and similarly, each 'Z' will be replaced by 'C'.

```
def change_Word(char_list):
    newword = ''
    shift = 3
    for letter in char_list:
        newword += chr(ord('A') + (ord(letter)+shift-ord('A'))%26)
    return newword

word = input("Enter a word in Uppercase Alphabets : ")
check = word.isupper() and word.isalpha()
if(check):
    characters = list(word)
    new_word = change_Word(characters)

    print("Entered Word", word)
    print("Changed Word", new_word)
else:
    print("Each character in the word must be an Uppercase Alphabet")
```

```
================== RESTART: C:\Users\p88si\Desktop\Py_Test\1.py ==================
Enter a word in Uppercase Alphabets : PYTHONPROGRAMMING
Entered Word PYTHONPROGRAMMING
Changed Word SBWKRQSURJUDPPLQJ
>>>
```

Passing a Tuple to a Function

The following program illustrates how to pass a tuple to a function.

Write a program to input a word having Upper Case Alphabet characters only. Store each letter in a tuple and pass this tuple to a function that replaces each letter of the word by a letter three places down the alphabet i.e. each 'A' will be replaced by 'D', each 'B' will be replaced by 'E' and similarly, each 'Z' will be replaced by 'C'.

```
def change_Word(char_tuple):
    newword = ''
    shift = 3
    for letter in char_tuple:
        newword += chr(ord('A') + (ord(letter)+shift-ord('A'))%26)
    return newword

word = input("Enter a word in Uppercase Alphabets : ")
check = word.isupper() and word.isalpha()
if(check):
    character_tuple = tuple(word)
    new_word = change_Word(character_tuple)

    print("Entered Word", word)
    print("Changed Word", new_word)
else:
    print("Each character in the word must be an Uppercase Alphabet")
```

```
================= RESTART: C:\Users\p88si\Desktop\Py_Test\2.py =================
Enter a word in Uppercase Alphabets : COMPUTERSCIENCE
Entered Word COMPUTERSCIENCE
Changed Word FRPSXWHUVFLHQFH
>>>
```

Passing a Dictionary to a Function

The following program illustrates how to pass a dictionary to a function.

Write a program to input a word. Count the occurrence of each vowel in the word and store it in a dictionary as key-value pair. Pass this dictionary to a function and print the occurrence of each vowel.

```
def print_Frequency(freq_dict):
    print("The Frequencies are")
    for key in freq_dict:
        print(key, ":", freq_dict[key])

word = input("Enter a word : ")
characters_list = list(word.upper())
vowel_count_dict = {}
for character in characters_list:
    vowel_list = ['A','E','I','O','U']
    if(character in vowel_list):
        unique_vowel_key = character
        if(unique_vowel_key not in vowel_count_dict) :
            frequency = characters_list.count(unique_vowel_key)
            vowel_count_dict[unique_vowel_key]=frequency

print_Frequency(vowel_count_dict)
```

```
================= RESTART: C:\Users\p88si\Desktop\Py_Test\3.py =================
Enter a word : COMPUTERSCIENCEUSINGPYTHON
The Frequencies are
O : 2
U : 2
E : 3
I : 2
>>>
```

Argument (Reference: CBSE Text Book, Computer Science, Class XI, Unit-3, Chapter - 2 : Functions)

Arguments are the value(s) provided in function call/invoke statement.

Default Parameter

When, in a function definition, one or more parameters are provided with a default value. The default value assigned to the parameter should be constant only. Only those parameters which are at the end of the list can be given a default value.

```
import math
def area_Triangle (a,b=10,c=5):
    semi_peri = (a+b+c)/2
    area = math.sqrt(semi_peri*(semi_peri-a)*(semi_peri-b)*(semi_peri-c))
    print("Sides a=",a,"b=",b,"c=",c, "  Area = ",area)

print ("Case-1 : Values Passed are a=10, b=9 and c=8")
area_Triangle(10,9,8) #calling function with all three argument values
print ("Case-2 : Values Passed are a=10, b=9")
area_Triangle(10,9)    #calling function with two argument values
print ("Case-3 : Values Passed are a=10")
area_Triangle(10)       #calling function with one argument value
```

```
========== RESTART: C:\Users\p88si\Desktop\CLASS_11\12TH\default.py ==========
Case-1 : Values Passed are a=10, b=9 and c=8
Sides a= 10 b= 9 c= 8   Area =  34.197039345533994
Case-2 : Values Passed are a=10, b=9
Sides a= 10 b= 9 c= 5   Area =  22.44994432064365
Case-3 : Values Passed are a=10
Sides a= 10 b= 10 c= 5   Area =  24.206145913796355
>>>
```

We cannot have a parameter on left with a default value, without assigning default values to parameters lying on its right side.

Positional Argument

The arguments which get assigned to parameter according to their position. An argument list must have any positional arguments followed by any keyword's arguments.

Note: Refer Keyword argument (CBSE TextBook, Computer Science, Class XI, Unit-3, Chapter - 2: Functions) for better understanding.

Example:

10, 9 and 8 are positional arguments in the following calls:

area_Triangle(10,9,8)

Recursion in Python

What is Recursion?

Recursion actually refers to the process of a function calling itself. The function which calls itself is called a recursive function.

Recursion works by breaking a larger problem into smaller ones. It is useful when we need to perform the same operation on a variable a number of times.

Let us take an example to explain Recursion.

Problem : To find the factorial of a given number . say 4, we shall use recursion.

4!= 4 x 3 x 2 x 1=24

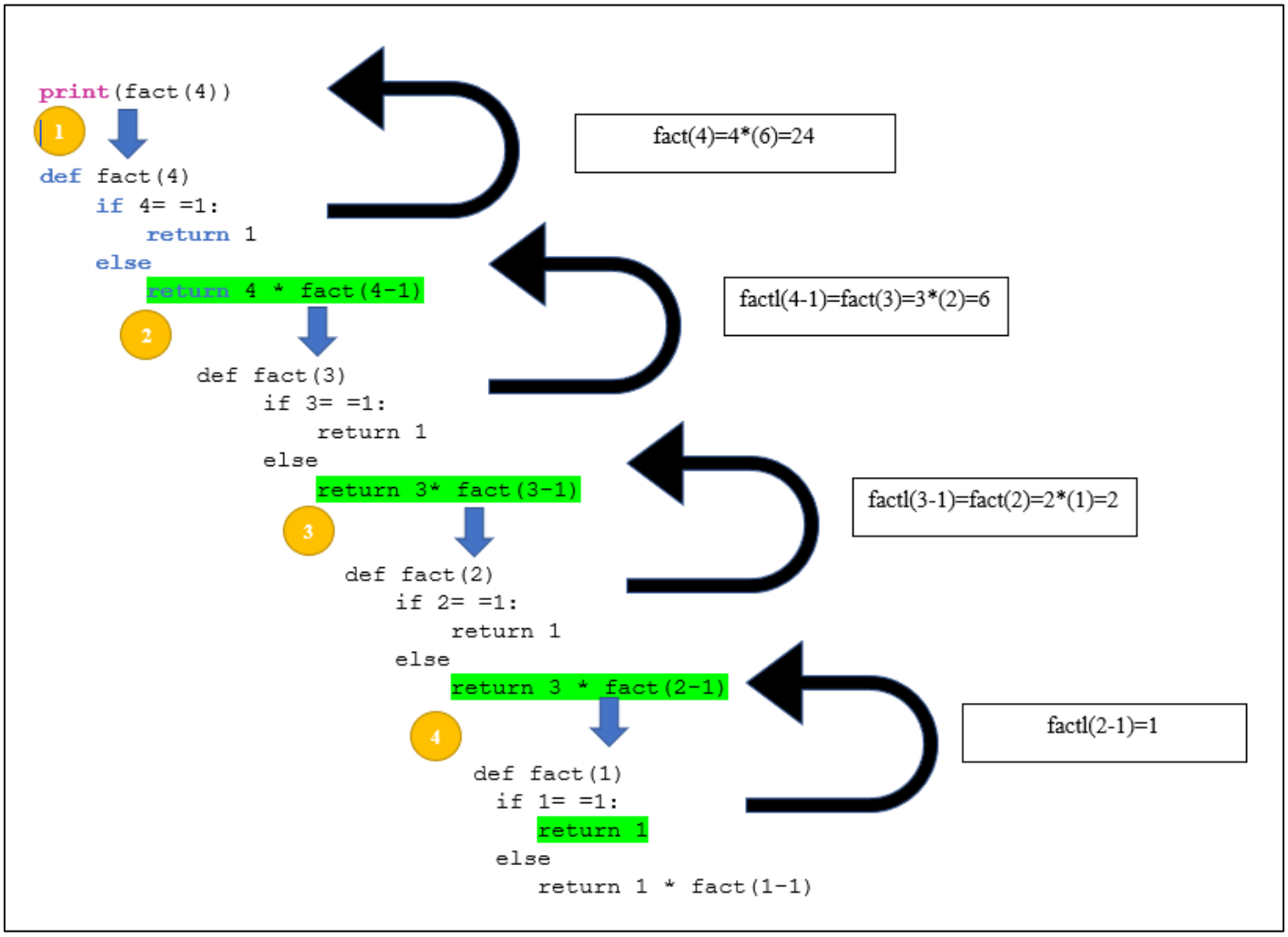
Code using a recursive function:

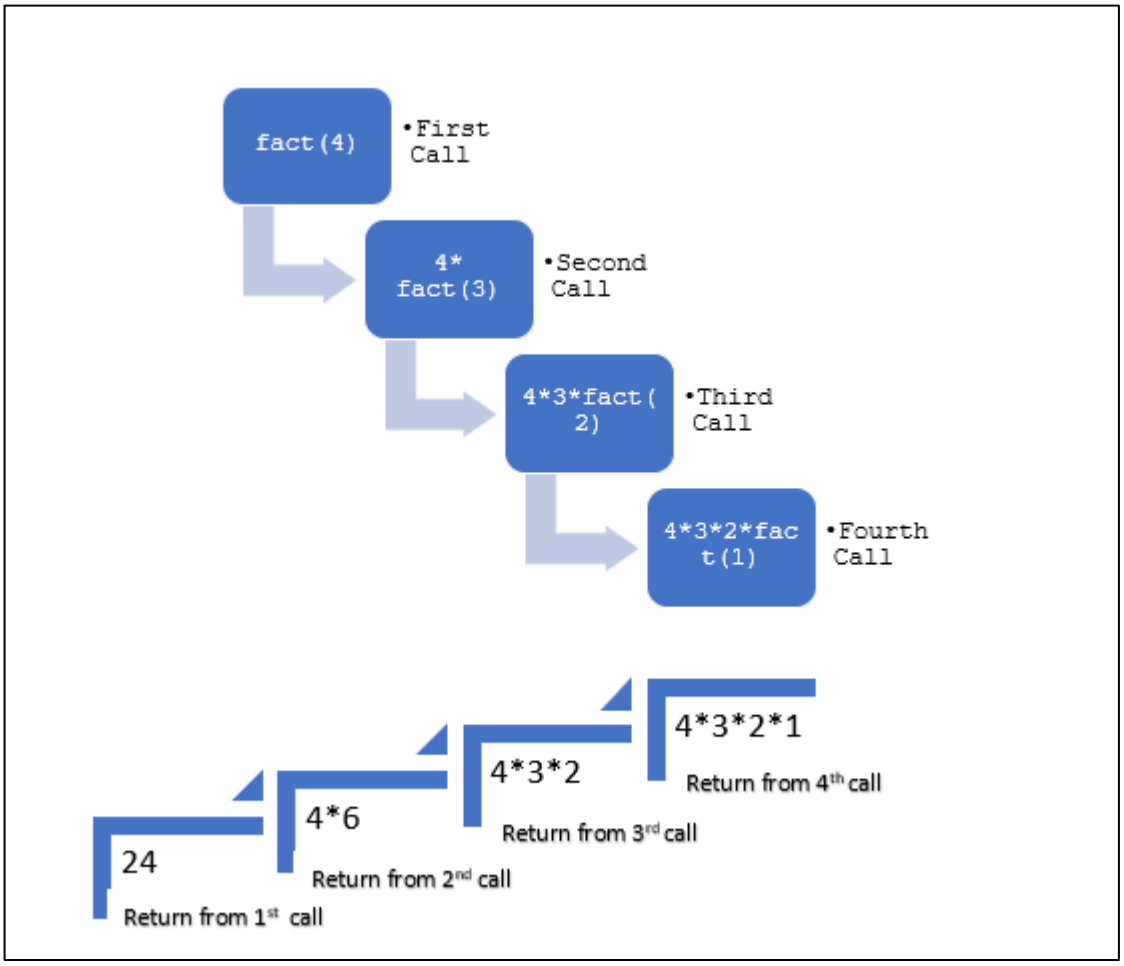
```
def fact(n)
    if n = = 1:
        return 1
    else
        return n * fact(n-1)
print (fact(4))
```

This is the recursive function call. Here the function fact calls itself with n-1 as the argument, which again calls itself till the function returns 1.

Now let us visualize this process:

The first call transfers control to the function fact() with value 4. Then the function is called again with the value 3 and so on till we reach a base case or a call that returns 1 in this case.

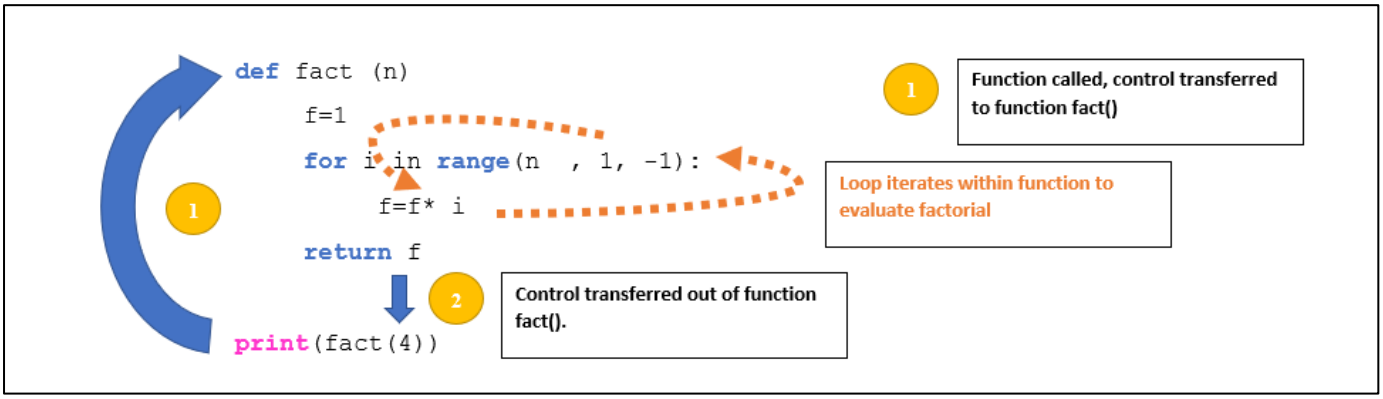




So here, the function fact() is called 4 times and each time the call is nested within the previous call. The evaluation of last call is returned to the second last and so on. Recursion makes the code look neat and is simpler to write in terms of logic. But it requires more memory due to the nested function calls.

Looping vs Recursion

We can also use a function with a loop to find the factorial of a number. The code is a bit more complex and the control remains within the function as the loop iterates.



Here control is transferred to the function when it is called with the value 4. Within the function the orange dotted lines indicate iteration. Once the loop breaks the return statement is executed thus control comes out to the call and the factorial is printed.

Advantages of recursion

1. The code requires less lines.
2. Breaking large complex programs becomes easier.
3. Makes code look more modular.

Disadvantages

1. The program may never terminate if not used properly or too much nesting is there.
2. More difficult to understand than loops.

Programs based on recursion:

1. Find the sum of a Fibonacci series upto n terms.

```
def fibonacci_sum(n):
  if n == 1 or n == 2:
    return 1
  else:
    return (fibonacci_sum(n - 1) + (fibonacci_sum(n - 2)))

print(fibonacci_sum(6))
```

The function fibonacci () is invoked with n as 6 here. The return statement nests two recursive function calls . Let us see how this works through visualizing the calls

fibonacci(6)
= fibonacci(5) + fibonacci(4)
= fibonacci(4) + fibonacci(3) + fibonacci(3) +fibonacci(2)
= fibonacci(3) + fibonacci(2) + fibonacci(2) +fibonacci(1)+fibonacci(2)+fibonacci(1)+1
=fibonacci(2)+fibonacci(1)+1+1+1+1+1+1
=1+1+6
=8

2. Finding sum of a list of numbers

```
def sum(Num):
    if len(Num) == 1:
        return Num[0]
    else:
        return Num[0] + sum(Num[1:])

print(sum([2, 6, 7, 18, 20]))
```

In the above program, sum() is a recursive function with the recursive call within the else part of the if construct.

3. Calculate a number raised to a power using recursion.

```
def power(x,y):
        if y==0:
                return 1
   elif y==1:
                return x
   elif x==0:
                return 0

   else:
                return x*power(x,y-1)

print(power(4,2))
```

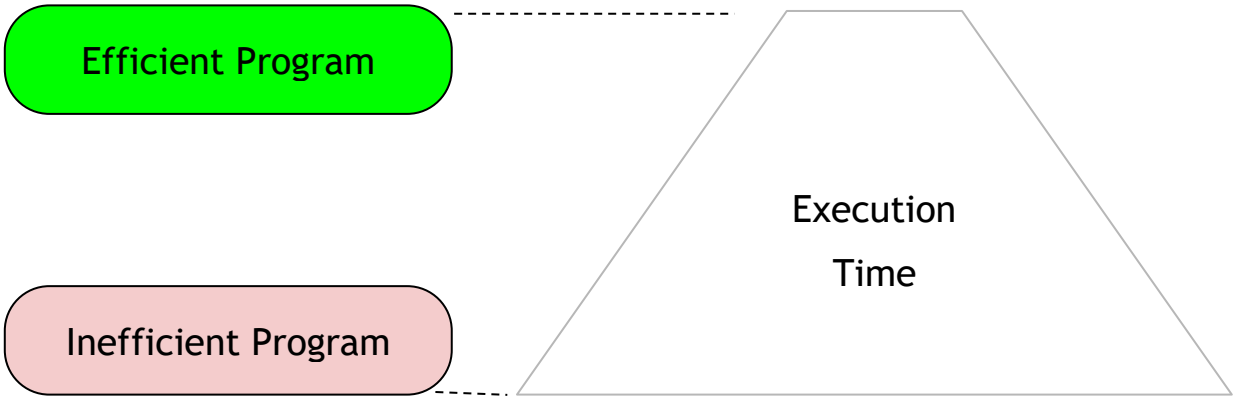
In this program the function power is called recursively when the value of x is more than 0.

4. Binary Search using recursion.

```
def binary_search(a, first, last, no):
mid = int((first+last)/2)
if no>a[mid]:
   binary_search(a, mid, last, no)
elif no<a[mid]:
   binary_search(a, first, mid, no)
elif no==a[mid]:
   print("Number found at", mid+1)
else:
   print("Number is not there in the array")
list=[2,4,5,7]
first=0
last=len(list)
no=7
binary_search(list,first,last,no)
```

Idea of Efficiency

In computer science, the efficiency of a program is measured in terms of the resources it consumes. Two of the most critical resources, which a program consumes are time and memory. For better efficiency, a program execution should consume less resources. In the modern times, we consider the idea of efficiency in terms of time only. So, in simple terms, we can say that an efficient program should take less time to execute.



In 1843, Ada Lovelace (The first computer programmer) emphasised the importance of efficiency concerning time when applying Charles Babbage's mechanical analytical engine.

We have to calculate its execution time to measure the efficiency of code. Execution time of code may vary from computer to computer depending upon the configuration of the system (i.e., Processor speed, RAM, Cache, OS, Compiler/Interpreter version etc.). We can calculate the number of fundamental operations in a code's execution. Assuming that each major operation takes one unit of time to execute, if a code performs 't' number of fundamental operations, then its execution time becomes 't' units.

In the world of computing, we express these complexities with capital O, also called Big O notation. Note that over here "O" means order, i.e. "order of complexity".

So, we can say that Big O notation is used to measure the complexity of any algorithm or a code. Let us consider the following N number of statements expected to be executed in code:

```
Statement 1
Statement 2
...
Statement N
```

Then the total time can be calculated by adding the time consumed by each of the statement:

```
Total Time = Time(Statement1)+Time(Statement2)+ … +Time (StatementN)
```

Big O	Runtime	Examples

O(1)	Constant Runtime	`A=1 #Statement 1`
		`B=2 #Statement 2`
		`C=A+B #Statement 3`
		`print(C) #Statement 4`
		`#Complexity will be 1 + 1 + 1 + 1 i.e.O(1)`

O(N)	Linear Runtime	`for i in range(N):`
		`  print(i)`
		`#Complexity will be N*1 i.e. O(N)`

$O(N^2)$	Quadratic Runtime	`for i in range(N):`
		`  for j in range(N):`
		`  print(j)`
		`#Complexity will be N * (N*1) i.e. O(N²)`

In case, we have the following codes:

Example 1:

```
N=int(input('N'))
for i in range(N):
    print(i)
for i in range(N):
    for j in range(N):
        print(j)
```

Complexity = 1 + N*1 + N*(1*N)

= 1+ N +N^2

So, the complexity of the above code in terms of Big O notation will be **O(N^2)**

Example 2:

```
N=int(input("N:"))
for i in range(1, N+1):
  p=p*i
print(p)
```

```
Complexity = 1 + N*1 + 1
           = N
```

So, the complexity of the above code in terms of Big O notation will be **O(N)**

[Considering only the dominant term]

Example 3:

```
N=int(input("N:"))
if(N<0):
 return
for i in range(1, N+1):
  p1=p1*i
M=int(input("M:"))
for i in range(1, M+1):
  p2=p2*i;
print(p1 + p2)
```

```
Complexity = 1+ 1 + N*1 + M*1 +1
           = N+M
```

So, the complexity of the above code in terms of Big O notation will be **O(N+M)**

[Considering only the dominant terms]

Example 4:

```
N=int(input("N:"))
sum=0;
for j in range(1, N+1):
  for i in range(1, N+1):
     p=p*i
  sum+=p
print(sum)
```

Complexity = 1 + 1 + N* (N*1) + 1

= 1 + 1+ N^2 + 1

So, the complexity of the above code in terms of Big O notation will be **O(N^2)**

In simple ways, we can also find the execution time required by a code

(Using time module to check start and end time of programs for executing the code)

Example 1 a: **Code to check if a given number is prime or not**

```
import time
Start = time.time()
def PrimeCheck(N):
  if N > 1: # Prime numbers are always greater than 1.
     for i in range(2, N//2+1):
       if (N % i) == 0:
         print(N," is not Prime Number")
         break
     else:
       print (N," Prime Number")
  else:
    print(num," is neither prime nor composite")
Num = int(input("Enter a Natural Number: "))
PrimeCheck(Num)
End = time.time()
print("Total Execution Time:",End-Start)
```

Output :

```
Enter a number : 627
627  is not Prime Number
Total Execution Time: 2.990596055984497
```

(Note: The time can vary on different systems)

Example 1 b: Alternative code for Prime Number

```
import math
import time
Start = time.time()
```

```
def PrimeCheck(N):
  if N > 1:
    for i in range(2,int(math.sqrt(N))+1):
      if (N % i) == 0:
        print(N," is not a prime number")
        break
    else:
      print(N," is a prime number")
  else:
    print(N," is neither Prime nor Composite")
Num = int(input("Enter a Natural Number: "))
PrimeCheck(Num)
End = time.time()
print("Total Execution Time:",End-Start)
```

Output:

```
Enter a Natural Number: 627
627  is not a prime number
Total Execution Time: 1.9446911811828613
```

(Note: The time can vary on different systems)

Example 2 a: To find the sum of N Natural Numbers.

```
import time
Start = time.time()
def Sum_N(N):
  S=0
  for i in range(1, N+1):
     S+=i
  print(S)
Num = int(input("Enter a Natural Number: "))
Sum_N(Num)
End = time.time()
print("Total Execution Time:",End-Start)
```

Output:

```
Enter a Natural Number: 100000
50005000
Total Execution Time: 78.48986077308655
```

(Note: The time can vary on different systems)

Example 2 b: Alternative code for finding the sum of N natural number

```
import time
Start = time.time()
def Sum_N(N):
  print(N*(N+1)/2)
Num = int(input("Enter a Natural Number: "))
Sum_N(Num)
End = time.time()
print("Total Execution Time:",End-Start)
```

Output :

```
Enter a Natural Number: 100000
5000050000.0
Total Execution Time: 4.043231248855591
```

(Note: The time can vary on different systems)

Binary File Handling

1. Adding a Record at the end of the file.

```
import pickle
'''
Function 1: A function AddRecord() to add new records at the end of the binary file "student" using list. The list should consist of Student No, Student Name and Marks of the student.
'''
      def AddRecords():
        Student=[]
        while True:
          Stno =int(input("Student No:"))
          Sname=input("Name")
          Marks=int(input("Marks"))
          S=[Stno,Sname,Marks]
          Student.append(S)
          More=input("More(Y/N)?")
          if More=='N':
            break
        F=open("students","ab")
        pickle.dump(Student,F)
        F.close()
```

2. Updating the record in in Binary file.

```
      '''
A function Update () to update the record in the binary file "students", which consists of student number, student name and marks after searching for a student on the basis of student number entered by the user.
      '''
def Update():
        F   =open("students","rb+")
        Recs=pickle.load(F)
        Found=0
        SN=int(input("Student No:"))
        for Rec in Recs:
          sn=Rec[0]
```

```
            if SN==sn:
              print("Record found...")
              print("Existing Name:",Rec[1])
              Rec[1]=input("New Name")
              Rec[2]=int(input("New Marks"))
              Found=1
              break
          if Found:
            F.seek(0)
            pickle.dump(Rec,F)
            print("Record updated")
          F.close()
```

3. Displaying all the records in Binary File.

```
      '''
```

Function 3: A function Display() to display the content of records from a binary file "students", which consists of student number, student name and marks.

```
      '''
      def Display():
        F   =open("students","rb+")
        Recs=pickle.load(F)
        for Rec in Recs:
          print(Rec[0],Rec[1],Rec[2])
          F.close()


      while True:
        CH=input("A:Add U:Update D:Display Q:Quit ")
        if CH=='A':
           AddRecords()
        elif CH=='U':
          Update()
        elif CH=='D':
          Display()
        else:
          Break
```

Sample Output:

```
A:Add U:Update D:Display Q:Quit A
Student No:12
NameArun Jha
Marks98
More(Y/N)?Y
Student No:15
NameTaran Taran
Marks76
More(Y/N)?Y
Student No:19
NameRiya Sen
Marks87
More(Y/N)?N
A:Add U:Update D:Display Q:Quit D
12 Arun Jha 98
15 Taran Taran 76
19 Riya Sen 87
A:Add U:Update D:Display Q:Quit U
Student No:15
Record found...
Existing Name: Taran Taran
New NameTaran Singh
New Marks92
Record updated
A:Add U:Update D:Display Q:Quit Q
```

CSV File handling

CSV (Comma Separated Values) file format is the most common format for tabulated data to be used in any spreadsheet tool (Calc, Google Sheets, Excel) and any database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The name CSV comes from the use of commas as a field separator for this file format.

The following are some of the examples of CSV files without header:
CSV File content with 3 Fields/Columns
1,Simran,80
2,Rishabh,99
CSV File content with 4 Fields/Columns with comma as separator:

```
12,"Simran",3400,"34,ABC Colony, DL"
14,"Rishabh",4300,"R-90,XYZ Nagar, DL"
```

The following is an example of CSV file with header:
CSV File content with 3 Fields/Columns

```
Rno,Name,Percent
```

1,Simran,80
2,Rishabh,99
3,Ajit,88

In Python, we use (import) the csv module to perform read and write operations on csv files. The csv module in Python's standard library presents classes and methods to perform read/write operations on CSV files.

We will make use of the writer () in a csv module which will return a writer object of the csv module to write the content of a sequence onto the file and the reader object to read the content from a csv file into a sequence.

As we have already seen in case of text and binary file handling, open() function is required to connect to the external file and close() function to disassociate with the external file. We will make use of the same functions for csv files too. Here also with the open function, we will use "w" (write) as the second parameter to write the content in a new file, "r" (read) as the second parameter to read the content from an existing file and "a" (append) as the second parameter to add new content below the existing content.

```
# Open a CSV File for writing content into a new file
# It will Overwrite the content, if content already exists
  Csvfile=open('student.csv','w', newline='')
```

```
# Open a CSV File for writing content at the bottom of an existing file
# It will create a new file, if file does not exist
  Csvfile=open('student.csv','a', newline='')
```

```
# Open a CSV File for reading content from an existing file
# It will raise an exception FileNotFoundError, if file does not exist
  Csvfile=open('student.csv','r')
```

The writer class of the module csv provides two methods for writing to the CSV file. They are writerow() and writerows(). This class returns a writer object which is responsible for converting the user's data into a delimited string. A csvfile object should be opened with newline = '' otherwise newline characters inside the quoted fields will not be interpreted correctly.

writer()

This function in the csv module returns a writer object that converts data into a delimited string and stores it in a file object. The function needs a file object with write permission as a parameter. Every row written in the file issues a newline character. To prevent additional space between lines, the newline parameter is set to ''.

CSV.writer class provides two methods for writing to the CSV file. They are writerow() and writerows() function.

writerow()

This function writes items in an iterable (list, tuple or string), separating them by comma character.

Syntax:
writerow(fields)

writerows()

This function takes a list of iterables as a parameter and writes each item as a comma separated line of items in the file.

Syntax:
writerows(rows)

```
import csv
#-----------------------------------------------------------------------#
# To create a CSV File by writing individual lines
#-----------------------------------------------------------------------#
def CreateCSV1():
      Csvfile=open('student.csv','w', newline='')  # Open CSV File
    Csvobj =csv.writer(Csvfile)                    # CSV Object for writing
    while True:
        Rno =int(input("Rollno:"))
        Name =input("Name:")
        Marks=float(input("Marks:"))
        Line=[Rno,Name,Marks]
        Csvobj.writerow(Line)                # Writing a line in CSV file
        Ch=input("More(Y/N)?")
        if Ch=='N':
            break
    Csvfile.close()                          # Closing a CSV File
```

The writer function returns a writer object that converts the data into a delimited string and stores it in a file object. Every row written in the file issues a new line character. To avoid additional lines between rows, the newline is set to ''.

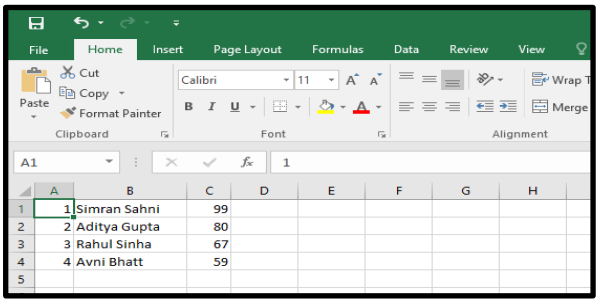
```
#-----------------------------------------------------------------------#
# To create a CSV File by writing all lines in one go
#-----------------------------------------------------------------------#
def CreateCSV2():
```

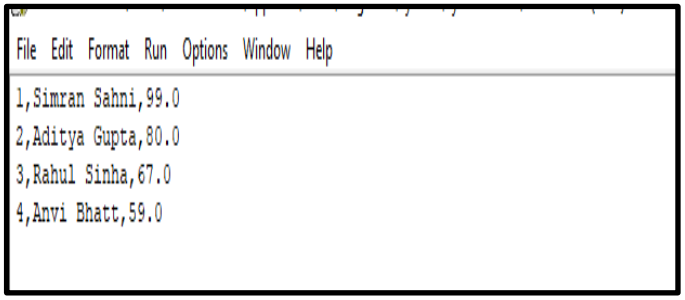
```
    Csvfile=open('student.csv','w', newline='')# Open CSV File
    Csvobj =csv.writer(Csvfile)                # CSV Object for writing
    Lines=[]
    while True:
        Rno=int(input("Rollno:"))
        Name=input("Name:")
        Marks=float(input("Marks:"))
        Lines.append([Rno,Name,Marks])
        Ch=input("More(Y/N)?")
        if Ch=='N':
            break
    Csvobj.writerows(Lines)          # Writing all lines in CSV file
    Csvfile.close()                  # Closing a CSV File
#-------------------------------------------------------------------------#
# To read and show the content of a CSV File
#-------------------------------------------------------------------------#
def ShowAll():
    Csvfile=open('student.csv','r', newline='')# Opening CSV File for reading
    Csvobj=csv.reader(Csvfile)      # Reading the CSV content in object
    for Line in Csvobj:             # Extracting line by line content
        print(Line)
    Csvfile.close()                            # Closing a CSV File
#-------------------------------------------------------------------------#


while True:
    Option=input("1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit ")
    if Option=="1":
        CreateCSV1()
    elif Option=="2":
        CreateCSV2()
    elif Option=="3":
        ShowAll()
    else:
        break
```

```
1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit 1
Rollno:1
Name:Simran Sahni
Marks:99
More(Y/N)?Y
Rollno:2
Name:Aditya Gupta
Marks:80
More(Y/N)?Y
Rollno:3
Name:Rahul Sinha
Marks:67
More(Y/N)?Y
Rollno:4
Name:Avni Bhatt
Marks:59
More(Y/N)?N
1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit 3
['1', 'Simran Sahni', '99.0']
['2', 'Aditya Gupta', '80.0']
['3', 'Rahul Sinha', '67.0']
['4', 'Avni Bhatt', '59.0']
1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit
```

Content of student.csv when viewed in notepad and Spreadsheet tool:





The open function (in any of the modes, namely 'w','r',or 'a') in absence of the newline parameter, by default appends '\r\n' (return key followed by newline character.

In case the newline is not present in the open function, the newline character is taken as '\r\n'. This is indicated by the presence of blank lines in the CSV file or the output which is displayed.

Here is the program where open function is used without the newline argument:

```
#-------------------------------------------------------------------------#
# To create a CSV File by writing individual lines
#-------------------------------------------------------------------------#
def CreateCSV1():
      Csvfile=open('student.csv','w')          # Open CSV File
    Csvobj =csv.writer(Csvfile)              # CSV Object for writing
    while True:
        Rno =int(input("Rollno:"))
        Name =input("Name:")
        Marks=float(input("Marks:"))
        Line=[Rno,Name,Marks]
        Csvobj.writerow(Line)                # Writing a line in CSV file
        Ch=input("More(Y/N)?")
        if Ch=='N':
            break
    Csvfile.close()                          # Closing a CSV File

#-------------------------------------------------------------------------#
# To create a CSV File by writing all lines in one go
#-------------------------------------------------------------------------#
def CreateCSV2():
    Csvfile=open('student.csv','w')              # Open CSV File
    Csvobj =csv.writer(Csvfile)                  # CSV Object for writing
    Lines=[]
    while True:
        Rno=int(input("Rollno:"))
        Name=input("Name:")
        Marks=float(input("Marks:"))
        Lines.append([Rno,Name,Marks])
        Ch=input("More(Y/N)?")
        if Ch=='N':
            break
    Csvobj.writerows(Lines)          # Writing all lines in CSV file
    Csvfile.close()                  # Closing a CSV File
#-------------------------------------------------------------------------#
```

```
# To read and show the content of a CSV File
#-------------------------------------------------------------------------#
def ShowAll():
    Csvfile=open('student.csv','r')# Opening CSV File for reading
    Csvobj=csv.reader(Csvfile)     # Reading the CSV content in object
    for Line in Csvobj:            # Extracting line by line content
        print(Line)
    Csvfile.close()                # Closing a CSV File
#-------------------------------------------------------------------------#

while True:
    Option=input("1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit ")
    if Option=="1":
        CreateCSV1()
    elif Option=="2":
        CreateCSV2()
    elif Option=="3":
        ShowAll()
    else:
        break
```

```
1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit 1
Rollno:1
Name:Simran Sahni
Marks:70
More(Y/N)?Y
Rollno:2
Name:Pooja Sahni
Marks:89
More(Y/N)?Y
Rollno:3
Name:Lalita Sharma
Marks:80
More(Y/N)?Y
Rollno:4
Name:Yashas Gupta
Marks:78
More(Y/N)?N
1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit 3
['1', 'Simran Sahni', '70.0']
[]
['2', 'Pooja Sahni', '89.0']
[]
['3', 'Lalita Sharma', '80.0']
[]
['4', 'Yashas Gupta', '78.0']
[]
1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit
```

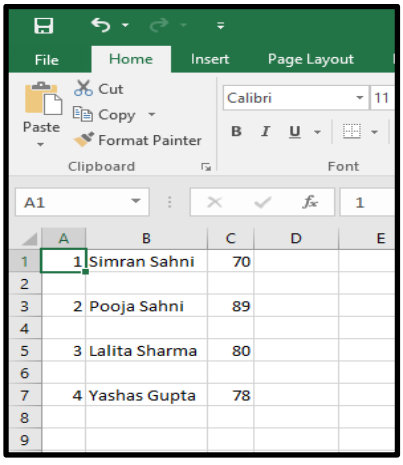
Content of student.csv when viewed in notepad and Spreadsheet tool:

```
File  Edit  Format  Run  Options  Window  Help
1,Simran Sahni,70.0

2,Pooja Sahni,89.0

3,Lalita Sharma,80.0

4,Yashas Gupta,78.0
```



Another alternative using with open as:

```
import csv
#-------------------------------------------------------------------------#
# To create a CSV File by writing individual lines
#-------------------------------------------------------------------------#
def CreateCSV():
  with open('emp.csv','w', newline='') as f:  # Open CSV File
      Csvobj =csv.writer(f)                    # CSV Object for writing
      while True:
      Eno  =int(input("Eno:"))
      Name =input("Name:")
      Pay=float(input("Pay:"))
      Line=[Eno,Name,Pay]
      Csvobj.writerow(Line)                    # Writing a line in CSV file
      Ch=input("More(Y/N)?")
      if Ch=='N':
            break
    # f.close()is not required
#-------------------------------------------------------------------------#
# To add new content in a CSV File
#-------------------------------------------------------------------------#
def AddAtEndCSV():
  with open('emp.csv','a', newline='') as f: # Open CSV File to append
    Csvobj =csv.writer(f)                   # CSV Object for writing
    while True:
      Eno  =int(input("Eno:"))
      Name =input("Name:")
      Pay=float(input("Pay:"))
      Line=[Eno,Name,Pay]
      Csvobj.writerow(Line)                 # Writing a line in CSV file
      Ch=input("More(Y/N)?")
      if Ch=='N':
        break.                              # f.close()is not required
#-------------------------------------------------------------------------#
# To read and show the content of a CSV File
#-------------------------------------------------------------------------#
def ShowAll():
 try:
    with open("student.csv", newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
 except FileNotFoundError:
     print("File not found!")
#-------------------------------------------------------------------------#
while True:
    Option=input("1:CreateCSV 2:AddAtEnd 3:ShowCSV 4:Quit ")
    if Option=="1":
        CreateCSV()
    elif Option=="2":
        AddAtEndCSV()
    elif Option=="3":
       ShowAll()
    else:
break
```

Creating and importing python library

Important: To create a python Library make sure you are using no other versions of python and pip is only installed on a default path.

Module: A Module is a file containing python definitions, functions, variables, classes and statements with .py extension.
[Creation module is covered in Class XI Chapter - 7 Functions in NCERT]

Library: A Library or a Package is a collection of various Modules.

Steps to create a package:

Step No	Step Description
1	Create a new folder with a name you want to give to a package along with a sub folder in it.
2	Create modules and save within the sub folder. [Note that the modules are the executed files]
3	Create an empty file with name as **`__init__.py`** within the same folder. **Note:** The filename init is preceded and succeeded by two underscores
4	Store the package content within the file **__init__.py** via importing all the modules created. **Note:** The Python interpreter recognizes a folder as a package, if it contains **`__init__.py`** file
5	Create a file setup.py in the main folder.
6	Install the Package via using PIP command.

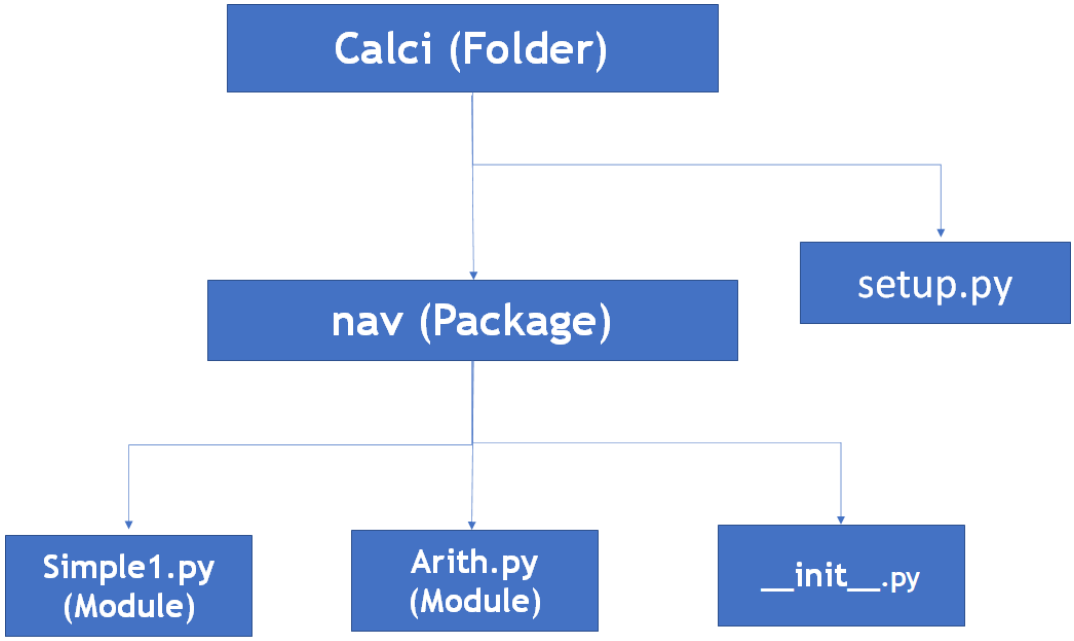
Example:

To create a package Calci containing two modules Simple.py and Arith.py

We have created the Library named "Calci". To create the Library follow the steps and see the structure of the folders and files.

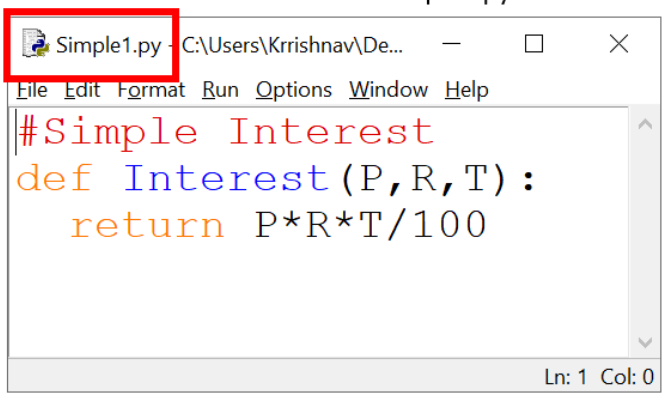
(Where Calci is a folder containing three files: Simple.py, Arith.py and __init__.py. The file __init__.py contains the contents of the files Simple.py and Arith.py, which can be done by simply importing these files.)

Step 1: We have created the folder calci and under that the folder "nav" which is the actual name of the library.

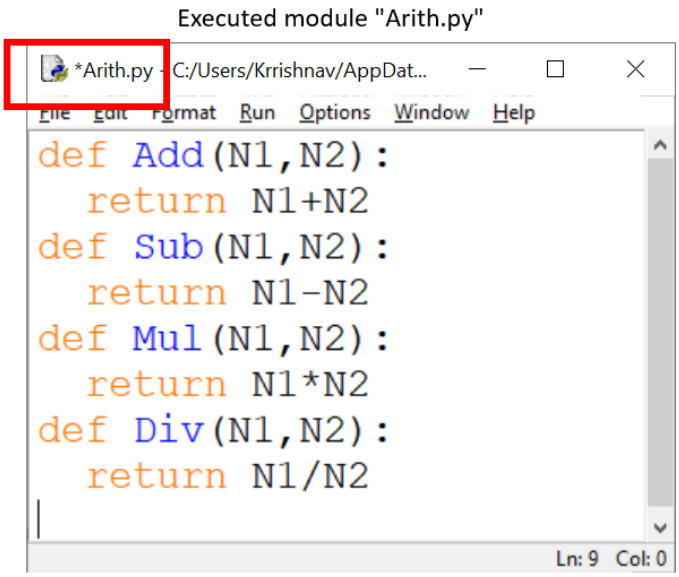


Step -2 Create the file .py name “Simple1.py” in the folder nav.

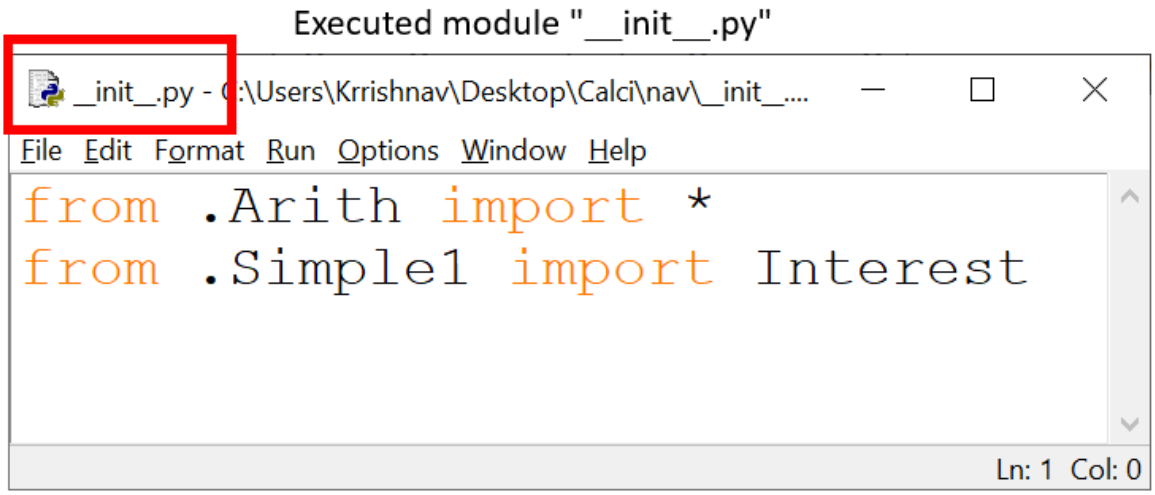
Executed module “Simple1.py"



Step -3 Create the file Arith.py in the folder nav.

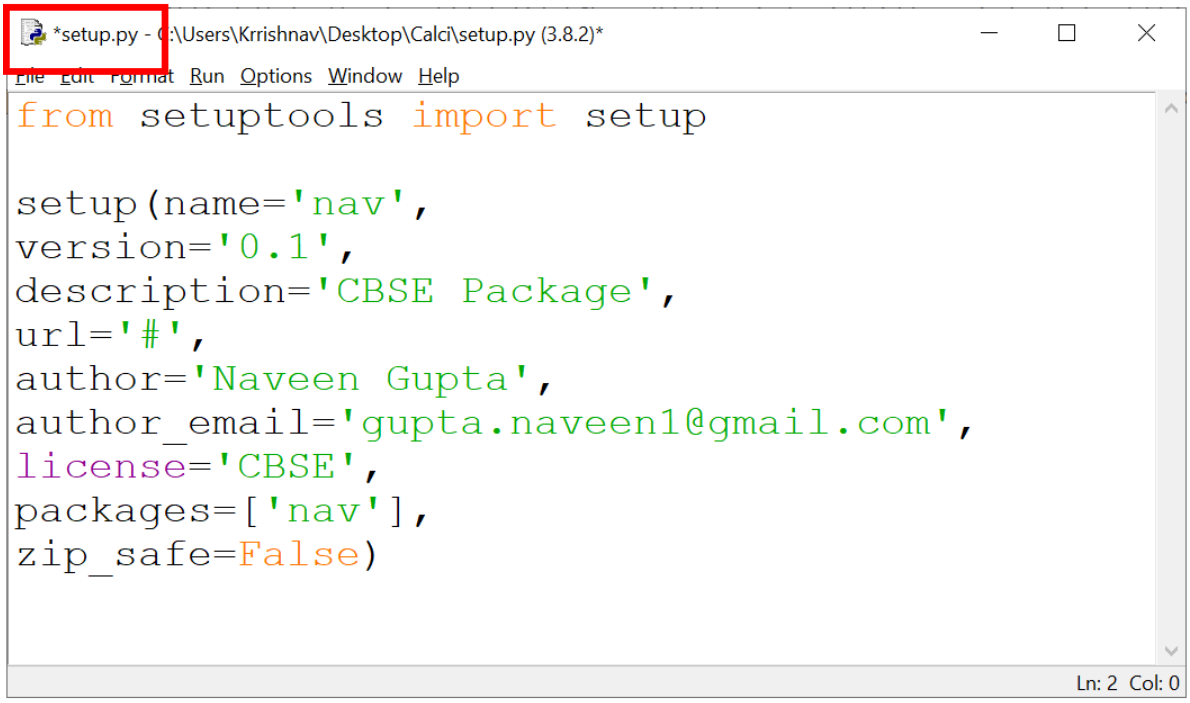


Step -4 Create the file __init__.py file again under the folder nav.



Step -5 Create the file Setup.py under the folder calci.

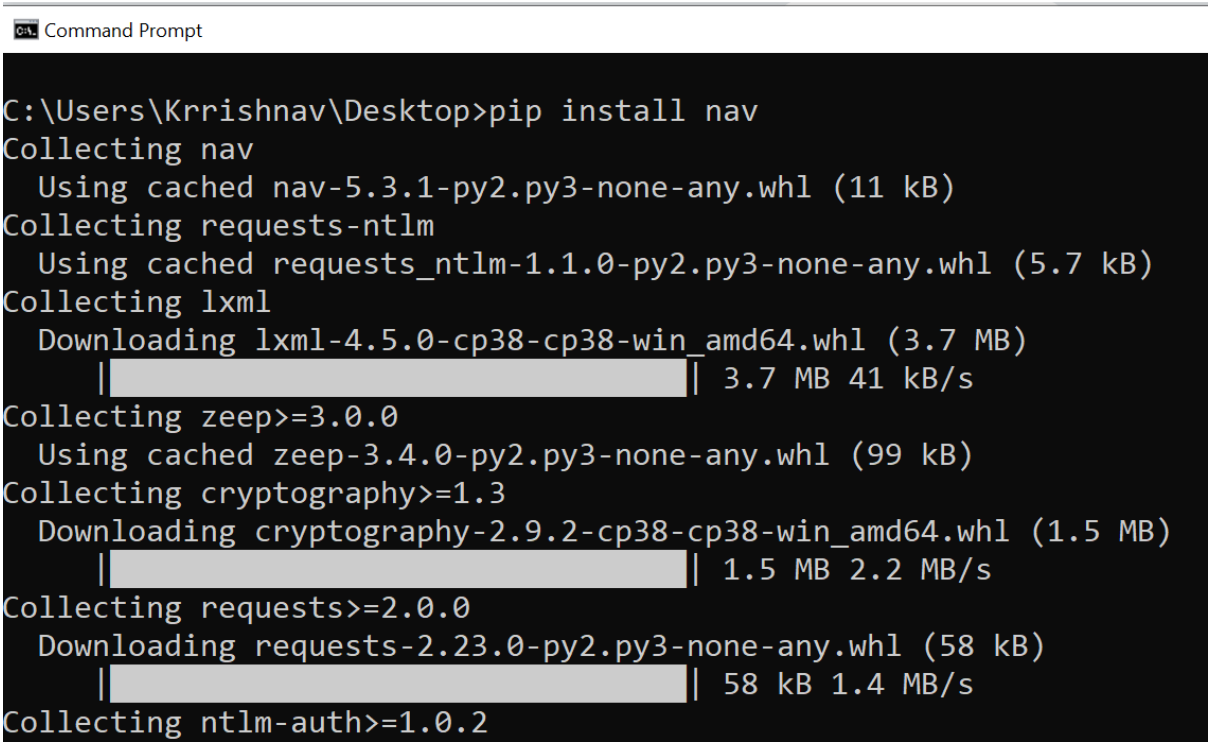
```
from setuptools import setup        #Predefined library to create Package
setup(name='nav',                   #name="nav" is the name of the package
version='0.1',                      #Version of the package
description='CBSE Package',         #Description name of organisation
url='#',                            #URL can be empty for local package
author='Naveen Gupta',              #Name of the person who has created the
                                    package
author_email='gupta.naveen1@gmail.com',
license='CBSE',
packages=['nav'],                   #Name of the package
zip_safe=False)
```



Step 6. To install the package open the command prompt, write **"Pip install nav".**

If you have already set the path of the python or pip then only this command will work. To run this command make sure that you are in the same folder where your pip is installed.

Pip is a package management system used to install and manage python software packages, such as those found in the Python Package Index or in simple words we can say it is a small installer file which helps us to install the python libraries.



```
Collecting certifi>=2017.4.17
  Downloading certifi-2020.4.5.1-py2.py3-none-any.whl (157 kB)
     |████████████████████████████████| 157 kB 3.2 MB/s
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Installing collected packages: pycparser, cffi, cryptography, idna, urllib3, chardet, certifi, requests, ntlm
auth, requests-ntlm, lxml, appdirs, isodate, requests-toolbelt, cached-property, zeep, nav
Successfully installed appdirs-1.4.3 cached-property-1.5.1 certifi-2020.4.5.1 cffi-1.14.0 chardet-3.0.4 crypt
graphy-2.9.2 idna-2.9 isodate-0.6.0 lxml-4.5.0 nav-5.3.1 ntlm-auth-1.4.0 pycparser-2.20 requests-2.23.0 reque
ts-ntlm-1.1.0 requests-toolbelt-0.9.1 urllib3-1.25.9 zeep-3.4.0
```

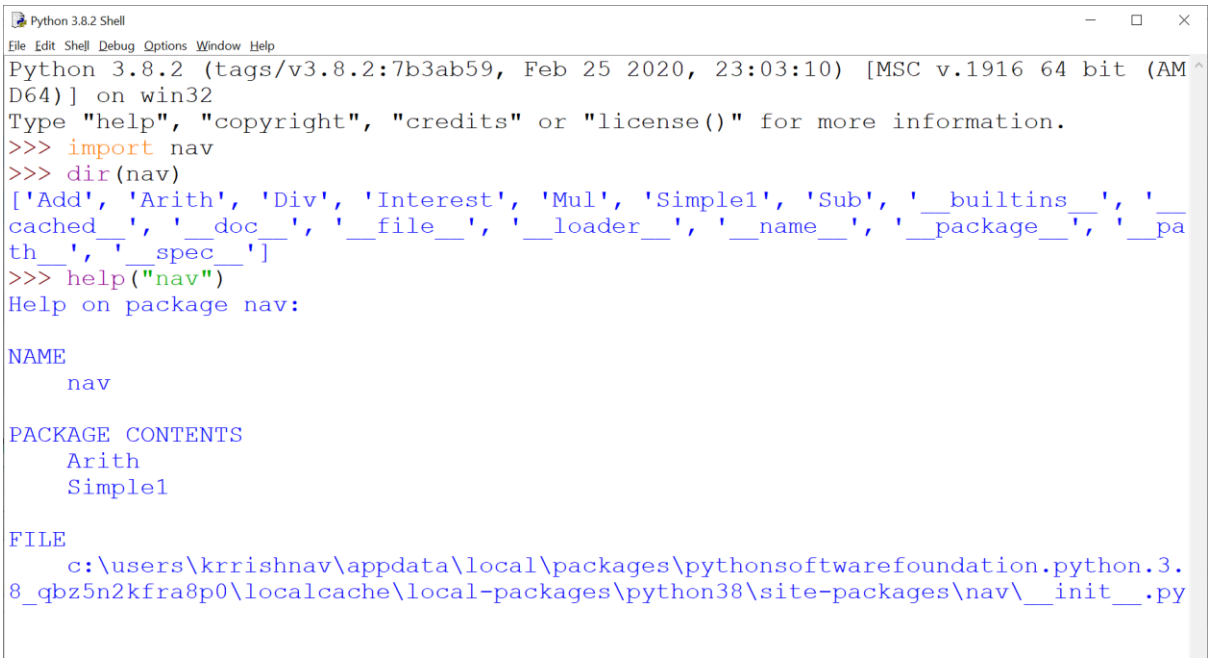
Your package is installed and ready for the use.

To check whether your Library is properly installed or not. Write the following commands:

import nav

dir(nav) #It will display the name of all the user-defined and system modules along with functions

help("nav") #It will display the content of the package.



How to use the package which you have created:

import nav

nav..Interest(990,5,3) #Calling the Inte

```
>>> import nav
>>> nav.Interest(900,5,3)
135.0
>>> nav.Add(2,3)
5
>>> from nav import Add
>>> from nav import Simple1
>>>
```

Difference between module and package.

The main difference between a module and a package is that package is a collection of modules and has an __init__.py file

Importing library

We can import the libraries by the following ways:

import <package name> #importing the complete package

from <Package name> import <Module Name> /<Function Name>

#importing the particular module or function from the package.

```
>>> import random                #imported the complete library
>>> random.random()
0.4651077285903422
>>> from random import randint   #only randint function() will be imported
>>> randint(10,50)
28
```

Interface python with SQL Database

A database is an organized collection of data, stored and accessed electronically from a computer system. The database management system (DBMS) is the software that interacts with end-users, applications, and the database itself to capture and analyze the data.

Data is organized into rows, columns and stored in tables. The rows (tuples) are indexed to make it easier to find relevant information.

Front End in database refers to the user interface or application that enables accessing tabular, structured or raw data stored within it. The front end holds the entire application programming utility for data, requests input and sends it to the database back-end. Form or any user interface designed in any programming language is Front End. (Python is used as front end). Data given by the database as a response is known as the Back-End database. (We will be using Python as Front-End and MySQL as Back-End)

The Python standard for database interfaces is the Python DB-API.

Following are the reason to choose python programming

- Programming more efficient and faster compared to other languages.
- Portability of python programs.
- Support platform independent program development.
- Python supports SQL cursors.
- Python itself take care of open and close of connections.
- Python supports relational database systems.
- Porting of data from one dbms to others is easily possible as it support large range of APIs for various databases.

Steps for creating Database Connectivity Application

1. Start Python
2. Import Packages required for establishing connectivity
3. Create Database
4. Open and establish a connection to the database
5. Create a cursor object or instance
6. Execute a query
7. Extract data from the result set
8. Clean up the environment

Establish a connection

For database interface/database programming, a connection must be established. Before establishing connection there must be MySQL installed on the system (Link: https://downloads.mysql.com/archives/community/) and a database and table are already created.

Installation of mysql.connector

We can install mysql-connector using the following command

```
C:\Users\p88si\Desktop\CLASS_12\MySQL>python -m pip install mysql-connector
Collecting mysql-connector
  Using cached mysql-connector-2.2.9.tar.gz (11.9 MB)
Installing collected packages: mysql-connector
    Running setup.py install for mysql-connector ... done
Successfully installed mysql-connector-2.2.9
```

We can also install pymysql, to work with MySQL.

pin install pymysql

We can use any connector to just one word needs to change in program rest of the code always same.

In the following way, we can establish a connection with the MySQL database through mysql.connector.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="school")
print(mydb)
...
========= RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Connection.py =========
<mysql.connector.connection.MySQLConnection object at 0x03FB9268>
>>>
```

Alternatively we can write the following statement if we are using mysqldb
importMySQLdb
mydb = MySQLdb.connect("localhost",“root",“root",“school")
print(mydb)

Another way to connect using mymysql
import pymysql
mydb = pymysql.connect("localhost",“root",“root",“school")
print(mydb)

In all the ways, we are specifying host, user, password and database name as arguments.
the database is an optional argument if we want to create a database through programming later on.

Cursor object :
The MySQLCursor class instantiates objects that can execute operations such as MySQL statements. Cursor objects interact with the MySQL server using a MySQLConnection object.

To create a cursor object and use it:

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost", user="root", passwd="root")
mycursor=mydb.cursor()
mycursor.execute("CREATE DATABASE school")    #create a database called school
mycursor.execute("SHOW DATABASES")
for x in mycursor:
    print(x)

=========== RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Cursor.py ===========
('information_schema',)
('mysql',)
('school',)
('test',)
>>>
```

The above code will create a database school and display all the databases stored on your computer.

To create a table at run time

To create a table, we have used the query
CREATE TABLE table_name(attributes type……)

Now, we have to pass the CREATE TABLE query in execute() method of cursor object. But before table creation, we must open the database. Here we are opening the database school(through connect() method) before student table creation.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="school")
mycursor=mydb.cursor()
mycursor.execute("CREATE TABLE  student (rollno int(3) primary key, name char(20),age int(2))")
```

On successful execution of the above program, a table named student with three attributes roll no, name, age will be created in the database school.

To insert data into the table:
To insert data into a table, we have used the query

INSERT INTO table_name [field_list] VALUES (respective data)

The following code will demonstrate how to insert data into a table during run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root", passwd="root", database="school")
mycursor=mydb.cursor()
myquery="INSERT INTO student(rollno,name,age)VALUES(%s,%s,%s)"
## storing data in a variable mydada
mydata=(1, 'Arjun', 17)
##executing the query with values stored in mydata
mycursor.execute(myquery , mydata)
mydb.commit()
print("Data Inserted")
```

```
========= RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Data_Insert.py ========
Data Inserted
>>>
```

To change the structure of the table :

To add a new attribute, we have used the query

ALTER TABLE table_name ADD (attribute type……)

The following code will demonstrate adding a new attribute to the table at run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="school")
mycursor=mydb.cursor()
mycursor.execute ("ALTER TABLE student ADD (marks int(3))")
mycursor.execute("DESCRIBE STUDENT")
for x in mycursor:
    print(x)
```

```
========= RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Alter_Table.py ========
('rollno', 'int(3)', 'NO', 'PRI', None, '')
('name', 'char(20)', 'YES', '', None, '')
('age', 'int(2)', 'YES', '', None, '')
('marks', 'int(3)', 'YES', '', None, '')
>>>
```

Above program will add a new attribute marks in the table student and will display the structure of the table

To fetch all records of a table:
To display the values stored in a table we have used the following query:

SELECT <field_list> FROM table_name [WHERE condition]

The following code will display data from a table during runtime.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root", passwd="root", database="school")
```

```
========== RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Select.py ==========
(1, 'Arjun', 17, None)
>>>
```

MySQLCursor.fetchall() Method

This method fetches all rows of a query result set and returns a list of tuples. If no more rows are available, it returns an empty list.

To fetch one record of a table at run time:

```
import mysql.connector
Mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="school")
Mycursor=Mydb.cursor()
Mycursor.execute("SELECT * FROM STUDENT")
Row=Mycursor.fetchone()
while Row is not None:
        print(Row)
        Row = Mycursor.fetchone()
```

```
========== RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Fetch_One.py =========
(1, 'Arjun', 17, None)
>>>
```

MySQLCursor.fetchone() method

This method retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available. By default, the returned tuple consists of data returned by the MySQL server, converted to Python objects.

MySQLCursor.fetchmany() method

rows = cursor.fetchmany(size=1)

This method fetches the next set of rows of a query result and returns a list of tuples. If no more rows are available, it returns an empty list.

To delete a record of a table at run time:

To delete tuples from a table, we have used the query

DELETE FROM table_name [WHERE condition]

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root", passwd="root",database="school")
mycursor=mydb.cursor()
mycursor.execute("DELETE FROM STUDENT WHERE ROLLNO=1")
mydb.commit()
```

In the above program delete query will delete a record with rollno=1. commit()method is necessary to call for a database transaction.

To update the record of a table at run time:

We have used the following query to update a record or more records.
UPDATE table_name SET attribute = value [WHERE condition]

The following code will edit the marks scored by roll no 2 to 99.

Table Data (Before)

```
mysql> Select * from Student;
+--------+-------+------+-------+
| rollno | name  | age  | marks |
+--------+-------+------+-------+
|      2 | Arjun |   17 |    75 |
+--------+-------+------+-------+
```

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root", passwd="root",database="school")
mycursor=mydb.cursor()
mycursor.execute("UPDATE STUDENT SET MARKS=99 WHERE ROLLNO=2")
mydb.commit()
```

Table Data (After)

```
mysql> Select * from Student;
+--------+-------+------+-------+
| rollno | name  | age  | marks |
+--------+-------+------+-------+
|      2 | Arjun |   17 |    99 |
+--------+-------+------+-------+
```

(Students are advised to develop a menu-driven program using the above concepts for better understating of python MySQL database interface. This concept can be used in Project work to be completed by the students.)

To Manage Database Transaction:

Python MySQL Connector provides the following method to manage database transactions.

commit	MySQLConnection.commit() methodsends aCOMMIT statement to the MySQL server, committing the current transaction.
rollback	MySQLConnection.rollback undoes the changes made by the current transaction.
autoCommit	MySQLConnection.autocommit value can be assigned as True or False to enable or disable the auto-commit feature of MySQL. By default its value is False.

To search records of a table at run time:

Table Data

```
mysql> Select * from Student;
+--------+--------+------+-------+
| rollno | name   | age  | marks |
+--------+--------+------+-------+
|      2 | Arjun  |   17 |    99 |
|      3 | Ramesh |   18 |    74 |
+--------+--------+------+-------+
```

The following code explains the use of a select query for searching a specific record from a table.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost", user="root",passwd="root",database="school")
mycursor=mydb.cursor()
name1 =input("Enter Name : ")
mycursor.execute(" select * from student where name='"+name1+"' ")
for x in mycursor:
    print (x)
```

```
========== RESTART: C:\Users\p88si\Desktop\CLASS_12\MySQL\Search.py ==========
Enter Name : Ramesh
(3, 'Ramesh', 18, 74)
>>>
```

The above code will take a name from the user and that name is searched in the table student using the SELECT query, the result will be shown with the help of my cursor collection.

Practice Exercise:

Consider scenario of a school where each student is supposed to enroll for two lab Courses. The structure of the Student and Lab table is as shown below.

Note: Lab1_Code & Lab2_Code (Student Table) are Foreign Key referencing Lab_code (Lab table).

STUDENT							
Roll_No	**Name**	**Class**	**Session**	**Gender**	**Address**	**Lab1_Code**	**Lab2_Code**
1	Ajay	12	2020-21	M	Delhi	PHY	CHE
2	Ramesh	12	2020-21	M	Delhi	PYT	BIO
3	Sumedha	12	2020-21	F	Delhi	CHE	PYT
4	Suresh	12	2020-21	M	Ghaziabad	BIO	PHY
5	Vijay	12	2020-21	M	Noida	PYT	MAT

LAB		
Lab_Code	**Lab_Name**	**Instructor_First_Name**
BIO	BIOLOGY	SANTOSH
CHE	CHEMISTRY	RAJAT
MAT	MATHS	SUDHIR
PHY	PHYSICS	LALIT
PYT	PYTHON	SUNIL

Perform the following operations using Python Program

1. Create a database namely 'School '.
2. Create the 'Student ' and 'Lab' Table.
3. Insert data, as shown above, in both the tables.
4. Select Roll Number, Student Name, Lab Name and Instructor Name where Lab Code is 'PYT'.
5. Select Name, Gender, Instructor Name for Students residing in Delhi where Lab Code is 'PYT'.
6. Update Instructor Name to ‘Deepak’ for Lab Code 'Mat'.

UNIT- II : COMPUTER NETWORK :

REFERENCE:

CBSE : A TEXT BOOK OF INFORMATICS PRACTICES- CLASS XII

Chapter 1: Computer Networking:

The link below provides the material for Unit –II (Evolution of Networking, Data communication technologies, Network devices, Network topologies and types, Network Protocol, Mobile Telecommunication Technologies, Network Security Concepts

cbseacademic.nic.in/supportmaterial.html

informaticspracticesbookclassxiiforweb_2011 (1).zip - ZIP archive, unpacked size 16,929,917 bytes

Introduction to Web Services:

Chapter 7 : Web Applications: Page no. 192, 193, and 225 , 230 and 231