

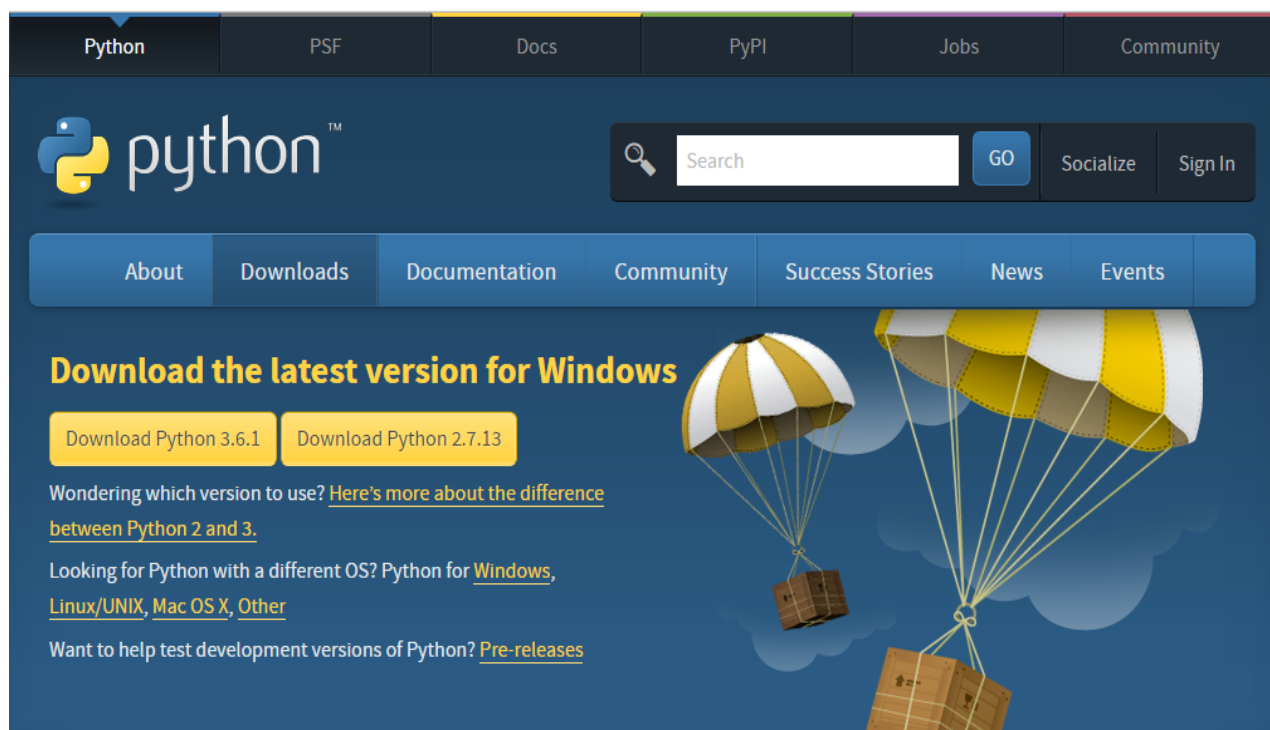
EXERCISE - 1(a)

Aim:

Running instructions in Interactive interpreter and a Python Script.

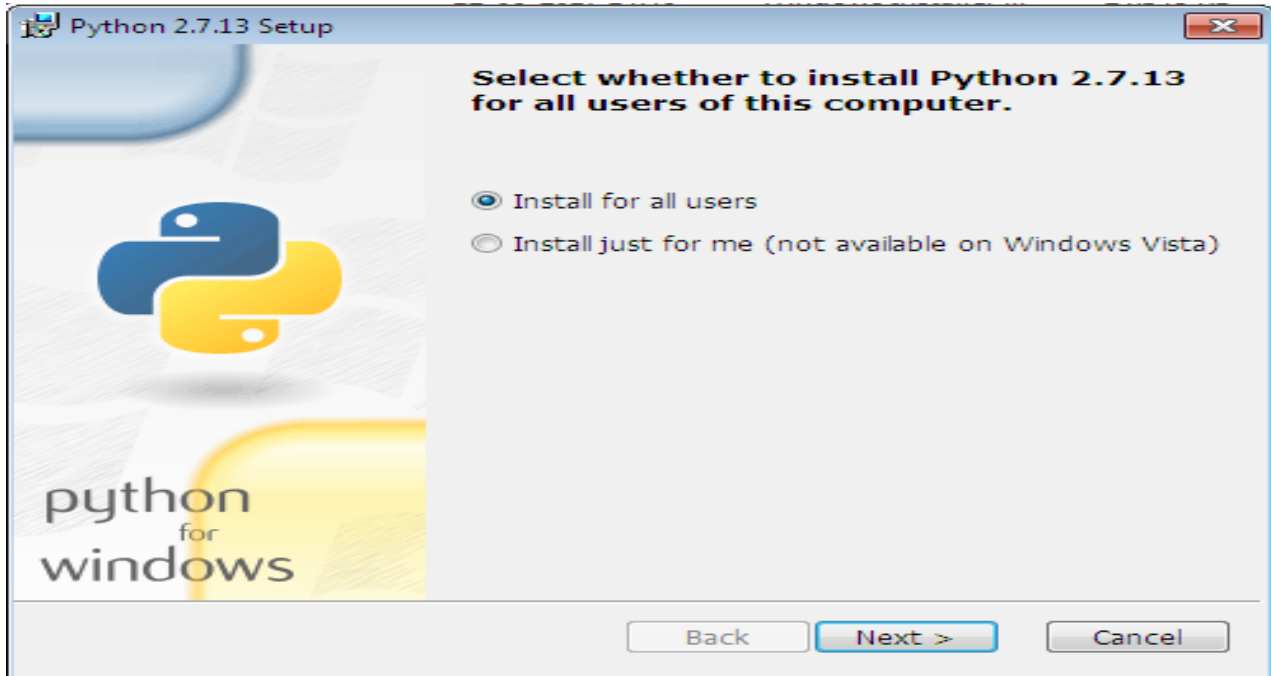
Procedure to Install and Run programs in Python:

In order to install python, Visit <https://www.python.org>. When we visit the Python for Windows download page, we'll immediately see the division. Right at the top, square and center, the repository asks if you want the latest release of Python 2 or Python 3 (2.7.13 and 3.6.1, respectively) as shown in below Figure.

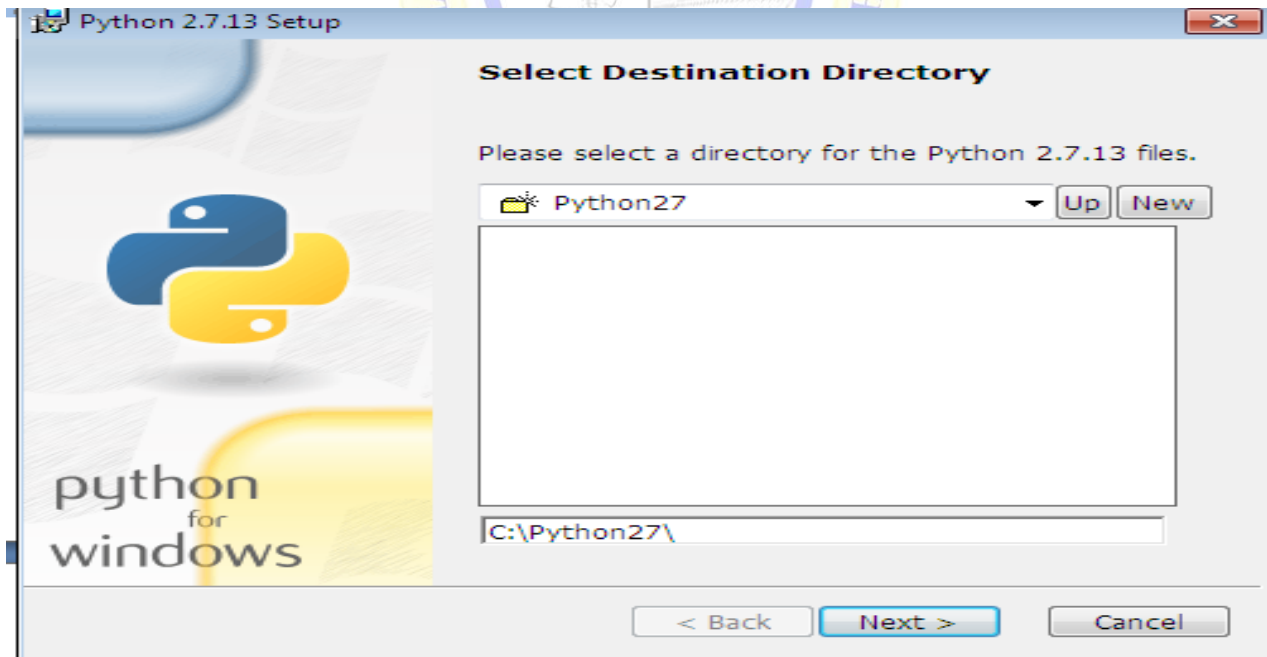


The version we want depends on our end goal. Here we will install Python 2.7.13. Click on Download Python 2.7.13 then python-2.7.13.msi file will be downloaded.

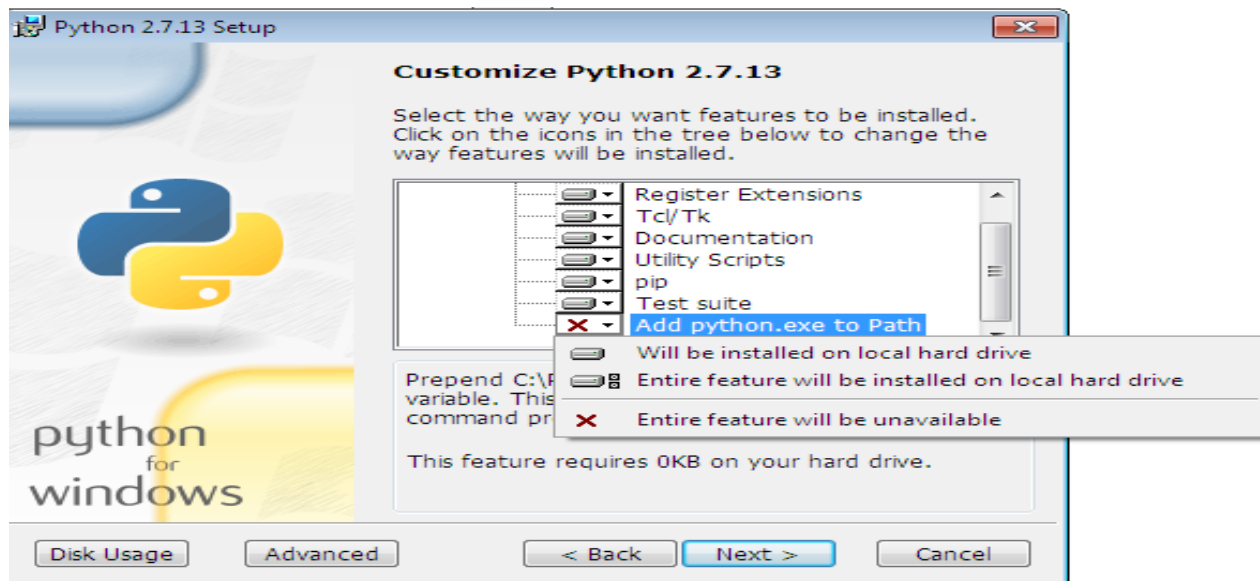
Run the installer, then a window will be opened as shown below. Select “Install for all users,” and then click “Next”.



After Clicking on “Next”, a window will be opened as shown below. On the directory selection screen, leave the directory as “Python27” and click “Next”.



After Clicking on “Next”, a window will be opened as shown below. On the customization screen, scroll down, click “Add python.exe to Path,” and then select “Will be installed on local hard drive.” then click “Next.”



We don't have to make any more decisions after this point. Just click through the wizard to complete the installation. When the installation is finished, set the variable path. After setting up the path, we can confirm the installation by opening up Command Prompt and type the following command as shown below.

Now, we can say that Python 2.7.13 is installed on our machine.

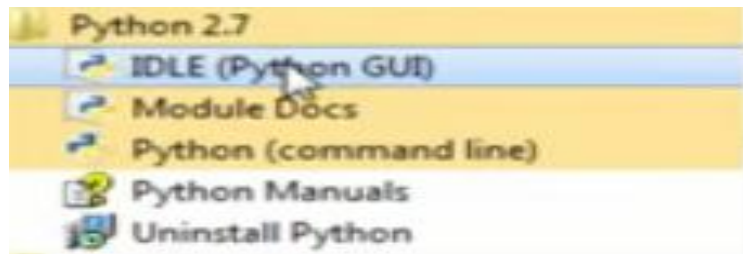
Different Ways of Invoking Python:

- Python GUI
- Python command line
- Command prompt from windows

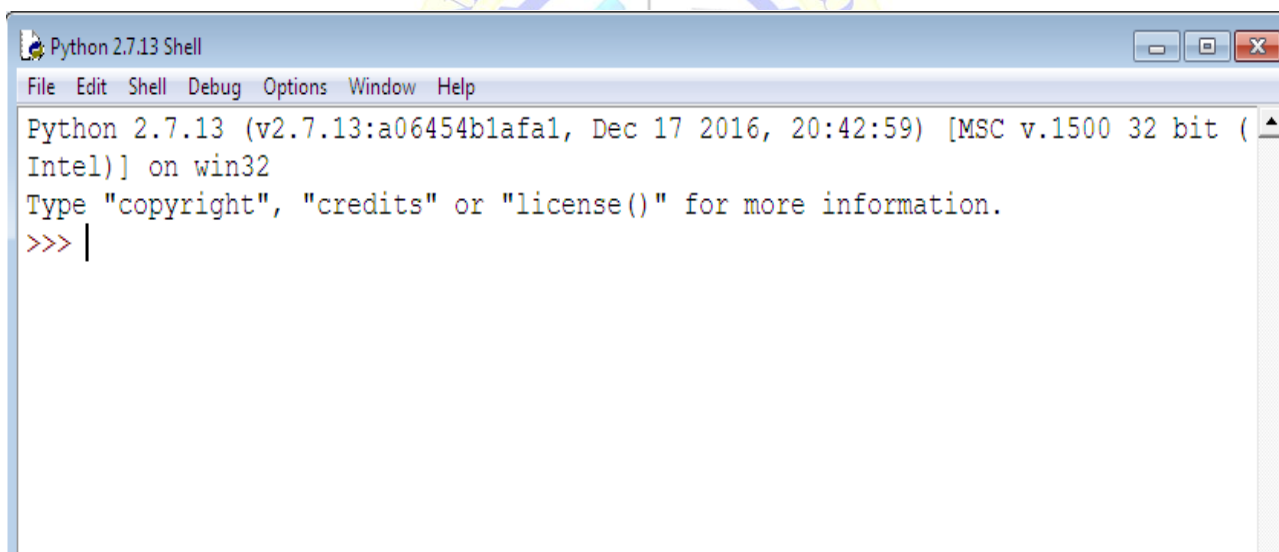
Python Programing Lab Manual

Python GUI:

Click on start -> all programs -> python 2.7 -> IDLE(Python GUI).

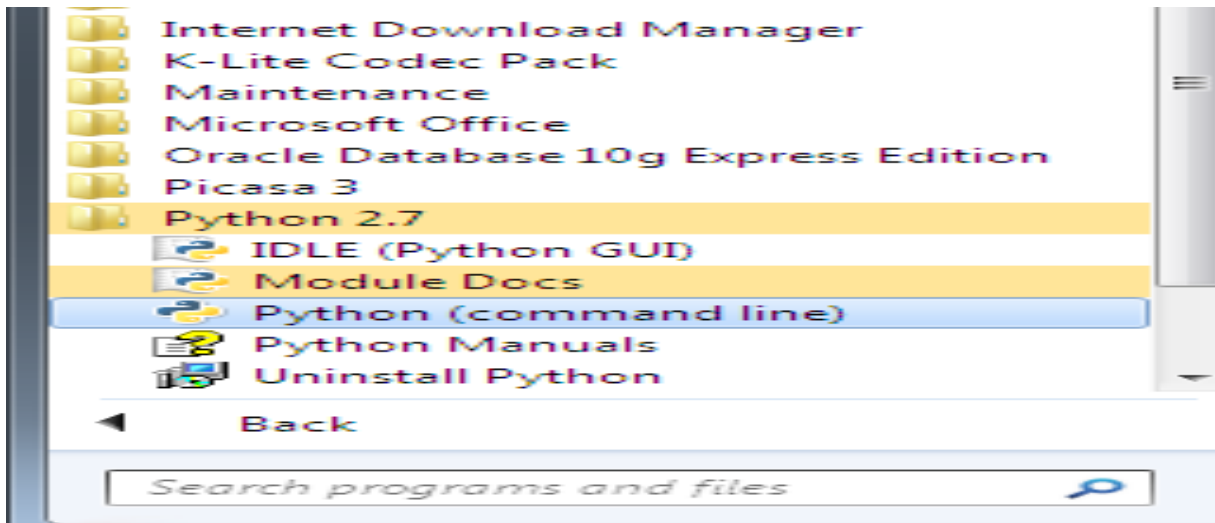


After Clicking on IDLE(Python GUI), a window will be opened as shown below.

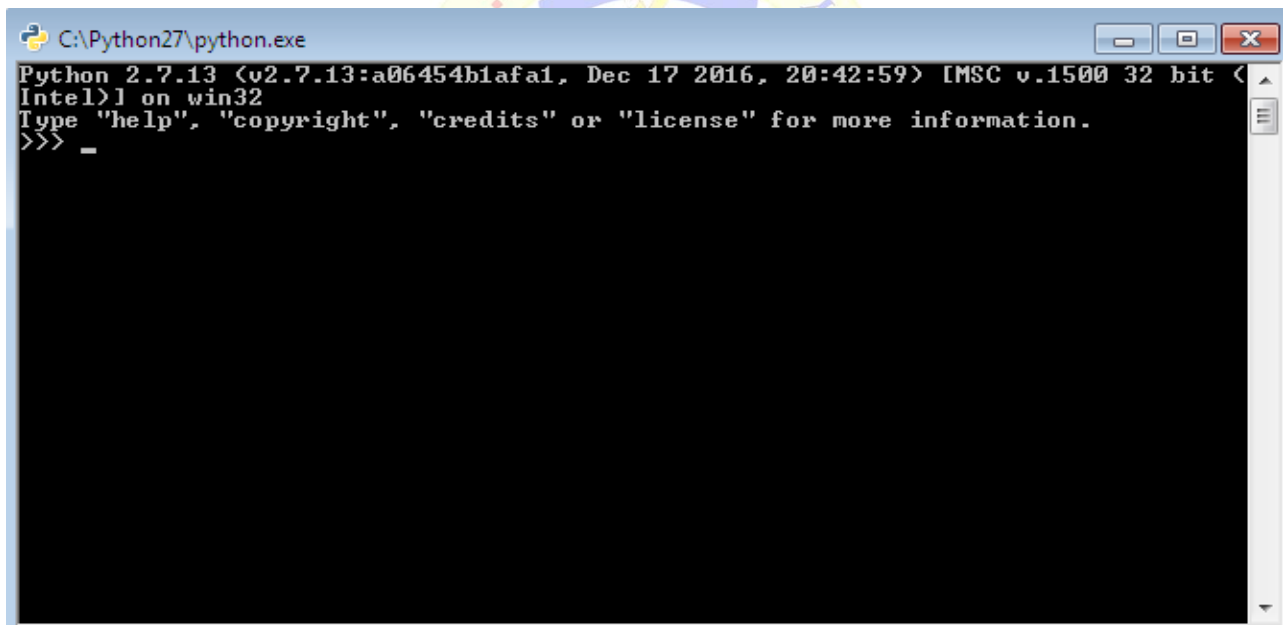


Python command line:

Click on start -> all programs -> python 2.7 -> Python (Command line).



After Clicking on Python (command line), a window will be opened as shown below:



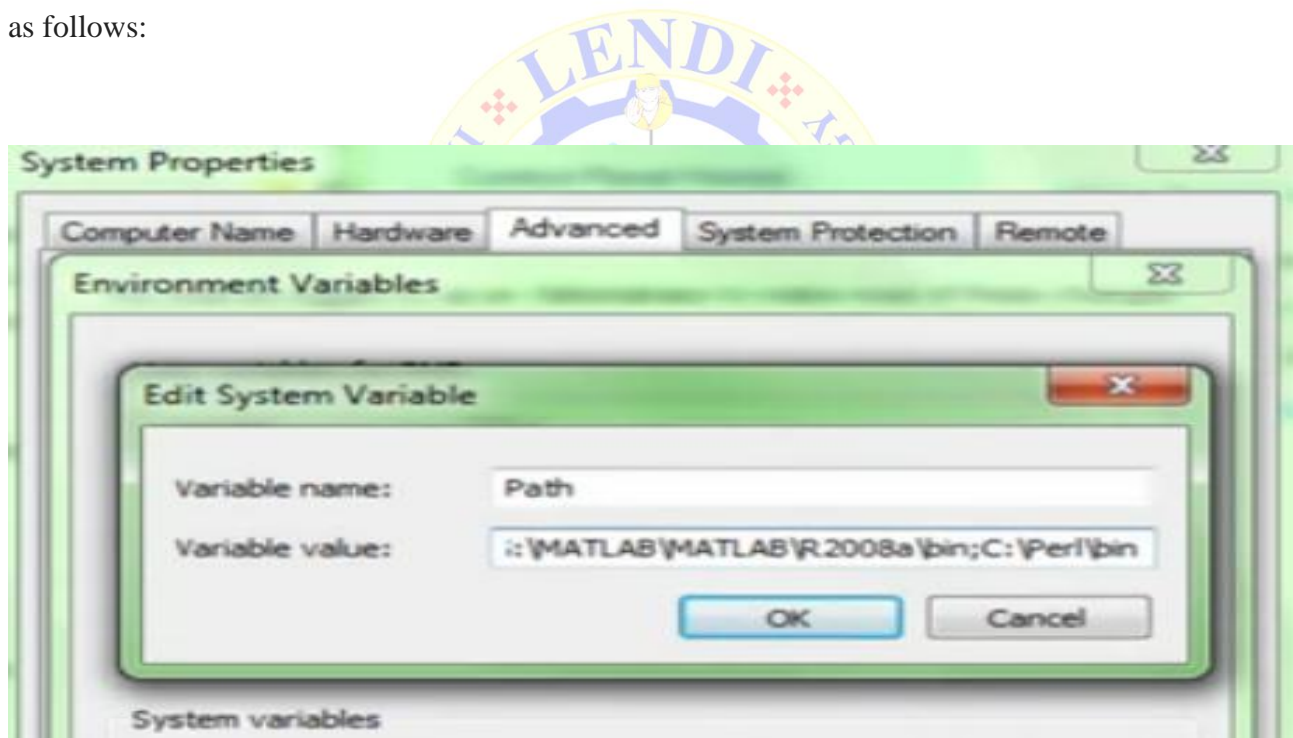
Command prompt from windows:

To open Python from Windows command prompt, We need to set **path**. The procedure to set the path is as follows:

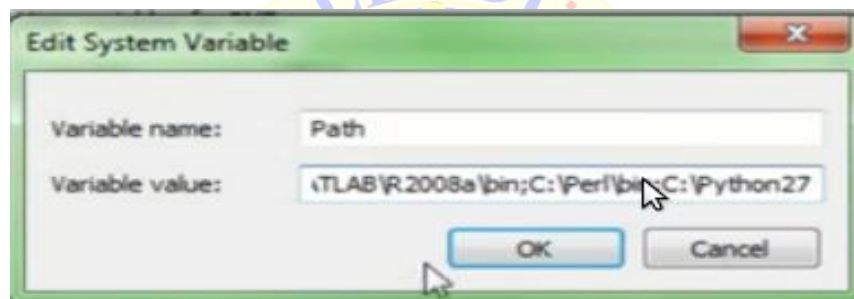
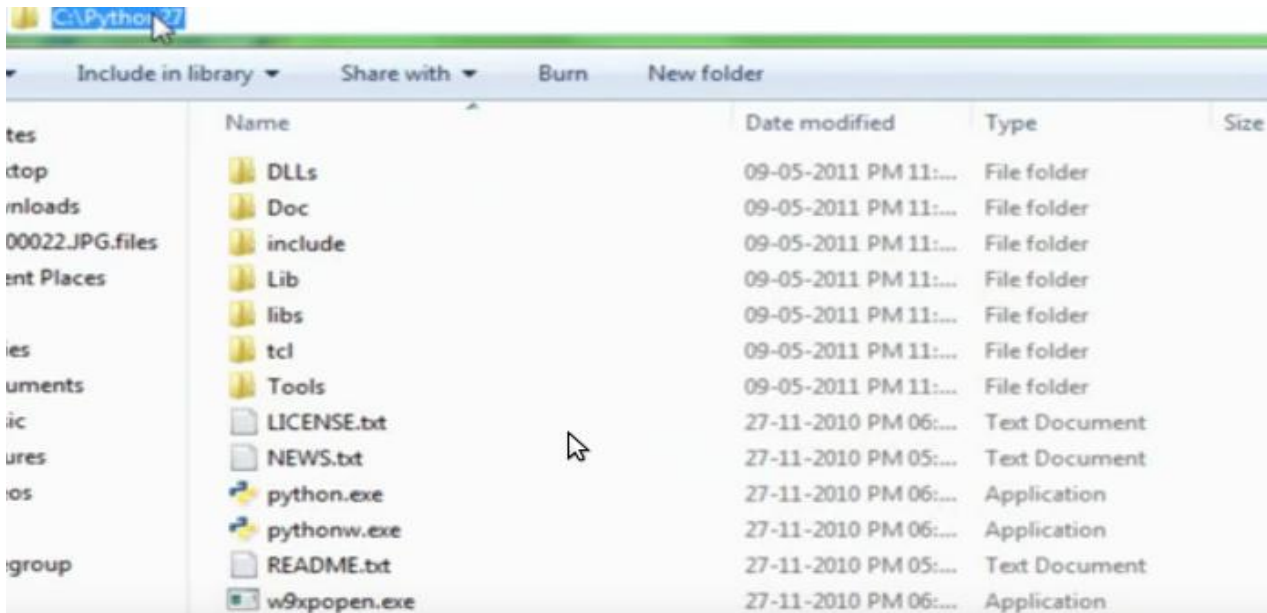
Go to My Computer -> right click and open properties, then a window will be opened as shown below:



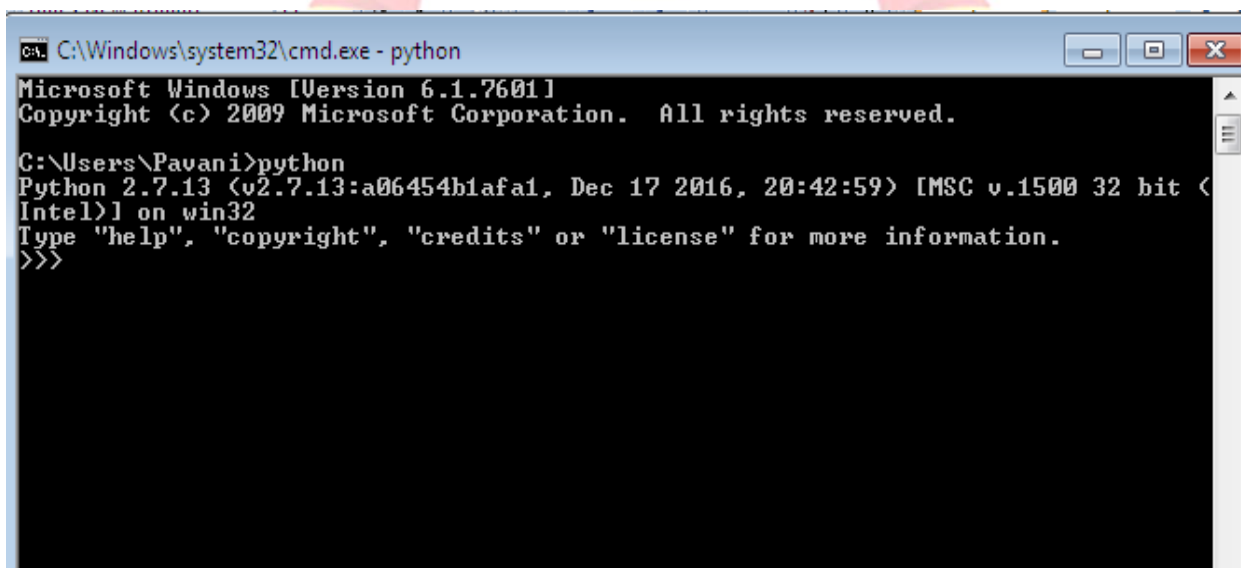
Now, Click on Advanced system settings -> Environmental Variables -> system variables and under system variable, click on Path variable and click on Edit. Then, a window will be opened as follows:



Add python path in variable value and click on OK as follows:



Now Open Command prompt from windows (cmd), and type the command “python” as follows:



EXERCISE - 1(b)

Aim:

Write a program to purposefully raise Indentation Error and Correct it.

Description:

Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.

A code block (body of a function, loop etc.) starts with indentation and ends with the first unindented line. The amount of indentation is depends on our choice, but it must be consistent throughout that block.

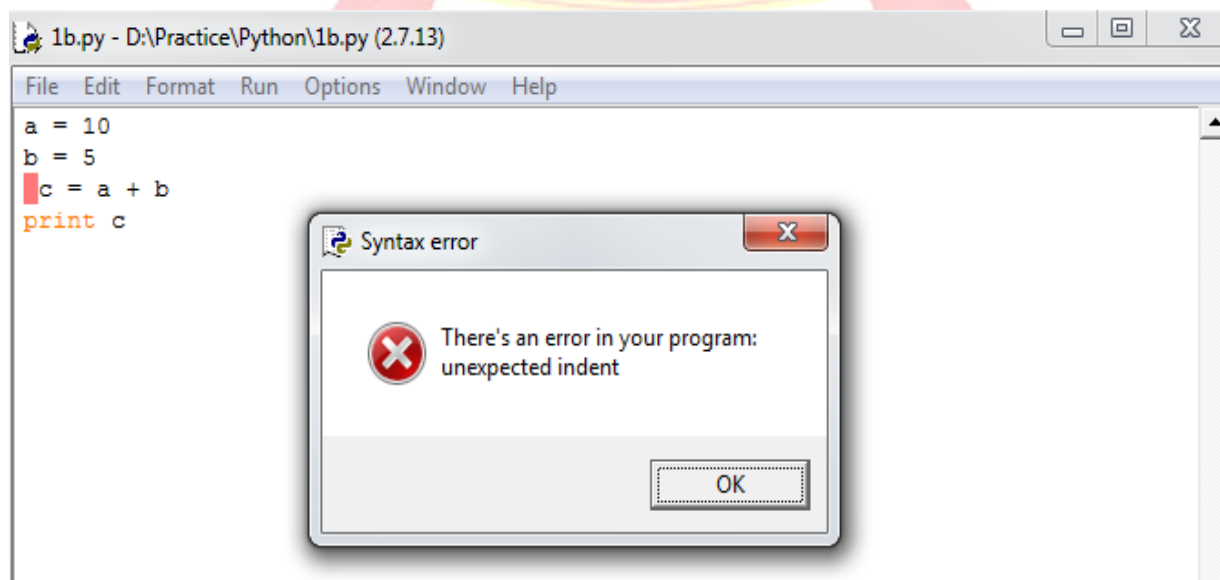
Generally, Four whitespaces are used for indentation and is preferred over tabs. The enforcement of indentation in Python makes the code look neat and clean. This results into Python programs that look similar and consistent.

Incorrect indentation will result into **Indentation Error**.

Program that shows Indentation Error:

```
a = 10
b = 5
  c = a + b
print c
```

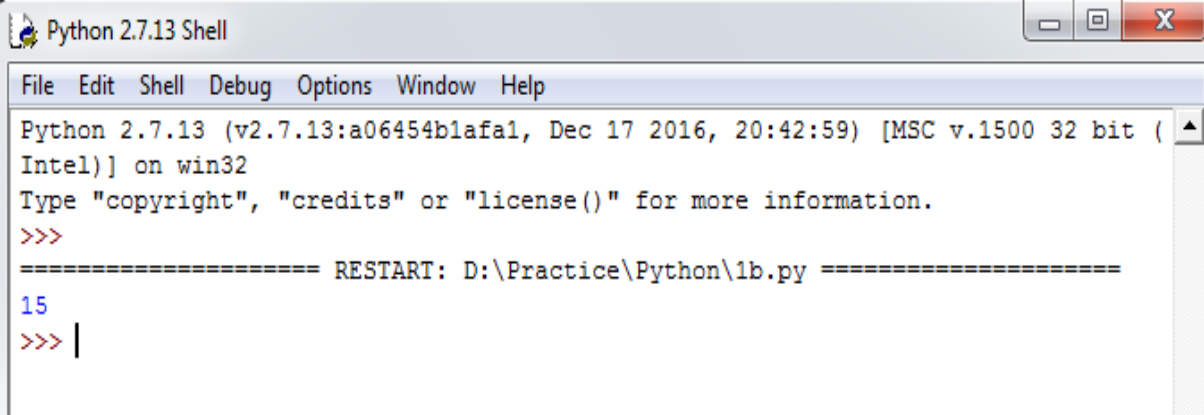
Output:



Program without Indentation Error:

```
a = 10
b = 5
c = a + b
print c
```

Output:

A screenshot of a Python 2.7.13 Shell window. The window title is "Python 2.7.13 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\1b.py =====
15
>>> |
```

Conclusion:

Student gets the knowledge on installing and executing programs in Python and also how Indentation plays a vital role in Python. This experiment maps with CO1 and students can attain PO1, PO2, PO3, PO4, PO5, PO9, PO10, PO11, PO12 and PSO1, PSO2, PSO3.

Viva questions:

1. Define Bytecode?
2. List the different modes of running python scripts?
3. What is the difference between 'end' and 'sep' attributes in print statement?
4. List the differences between python and R?
5. Is python programming language or scripting language? justify your statement.
6. List the differences between compiler and interpreter?
7. How to assign multiple values to variables in single line in python?
8. Write a print statement to display the output Hello"World"Everyone?
9. How to write multiline statements in python?
10. What is indentation error with an example?

EXERCISE - 2(a)

Aim:

Write a program to compute distance between two points taking input from the user (Pythagorean Theorem).

Description:

The Pythagorean theorem is the basis for computing distance between two points. Let (x_1, y_1) and (x_2, y_2) be the co-ordinates of points on xy-plane. From Pythagorean theorem, the distance between two points is calculated using the formulae:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

To find the distance, we need to use the method **sqrt()**. This method is not accessible directly, so we need to import **math** module and then we need to call this method using math static object.

To find the power of a number, we need to use ****** operator.

Algorithm:

Input: x_1, y_1, x_2 and y_2

Output: Distance between two points.

Step1: Start

Step2: Import math module

Step3: Read the values of x_1, y_1, x_2 and y_2

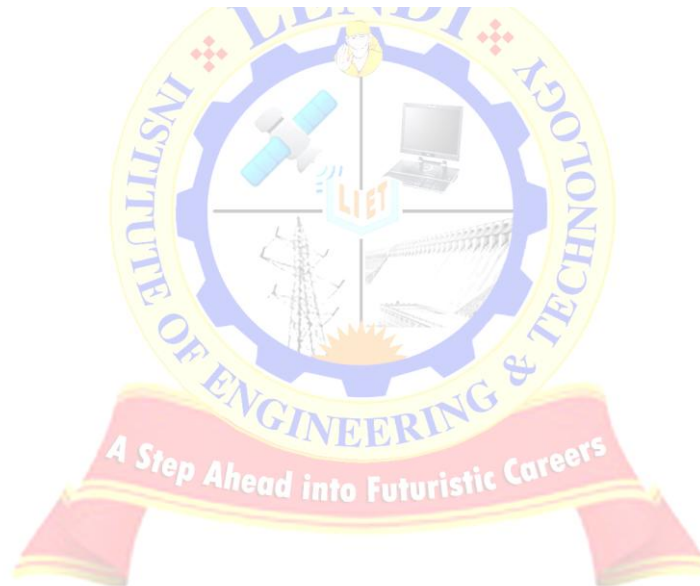
Step4: Calculate the distance using the formulae **math.sqrt((x2 - x1)**2 + (y2 - y1)**2)** and store the result in distance

Step5: Print distance

Step6: Stop

Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\2a.py =====
Enter x1 value:5
Enter y1 value:18
Enter x2 value:7
Enter y2 value:9
Distance between two points is: 9.21954445729
>>>
```



EXERCISE - 2(b)

Aim:

Write a program add.py that takes 2 numbers as command line arguments and prints its sum.

Description:

Command line arguments are the arguments that are passed to the program when the program is invoked for execution.

Python provides a **getopt** module that helps us to pass command line arguments and options. The Python **sys** module provides access to any command line arguments via **sys.argv**. This serves two purposes:

- **sys.argv** is the list of command line arguments.
- **len(sys.argv)** is the number of command line arguments.

The first argument is always script name and it is also being counted in number of arguments. As the command line arguments are strings, here we need to type-cast those arguments to the suitable type.

Algorithm:

Input: Two numbers from command line

Output: Sum of two numbers

Step1: Start

Step2: Import sys module

Step3: Read the arguments using commandline and store the values in a and b

Step4: Type cast a to integer and b to integer then calculate the addition of a and b and store the result in sum

Step5: Print sum

Step6: Stop

Output:

```
D:\Practice\Python>python add.py 75 81
The addition of two numbers is: 156

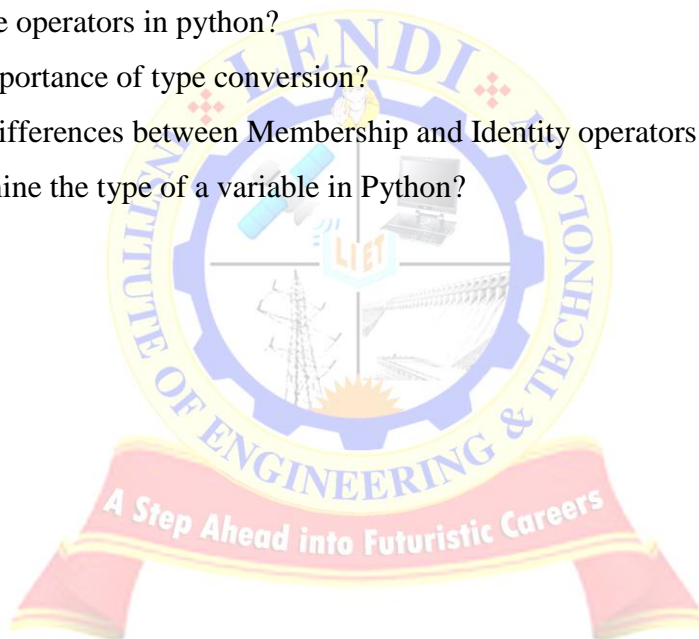
D:\Practice\Python>
```

Conclusion:

Student gets the knowledge on math functions and how the values are passed from the command line. This experiment maps with CO1 and the students can attain PO1,PO2,PO3,PO4, PO5,PO9,PO10,PO11,PO12 and PSO1,PSO2,PSO3.

Viva questions:

1. What are the supported data types in Python?
2. How to represent complex numbers in python?
3. Write a print statement to truncate the number 12.34567 to 2 decimal places?
4. What is command line argument?
5. In which module command line arguments are present in Python?
6. How many reserved keywords are there in python 2.7 and python 3.6?
7. List the bitwise operators in python?
8. What is the importance of type conversion?
9. What are the differences between Membership and Identity operators?
10. How to determine the type of a variable in Python?



EXERCISE - 3(a)

Aim:

Write a Program for checking whether the given number is a even number or not.

Description:

If a number is exactly divisible by 2(Remainder is 0) then it is said to be an Even number otherwise, the number is said to be an Odd number. In Python, We use modulus(%) operator to find the remainder.

Decision making is used to specify the order in which the statements are executed. We can use if...else statement to check whether the number is even or odd.

if...else statement:

This is a two-way decision making statement that decides what to do when the condition is true and what to do when the condition is false. The general form of if...else statement is as follows:

```
if condition:
    intended statement block for true condition
else:
    intended statement block for false condition
```

Algorithm:

Input: A Number

Output: A Message

Step1: Start

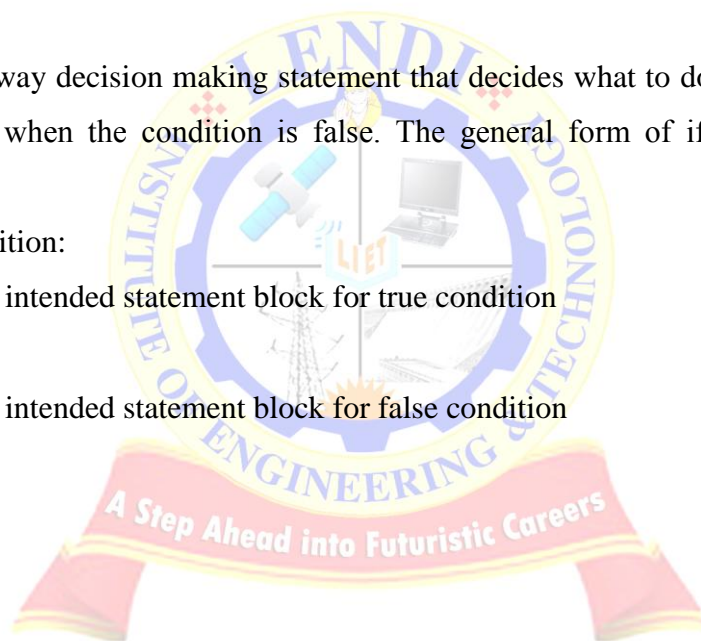
Step2: Read num

Step3: Check whether the num is divisible by 2 or not. If yes, goto Step4. else, goto Step5

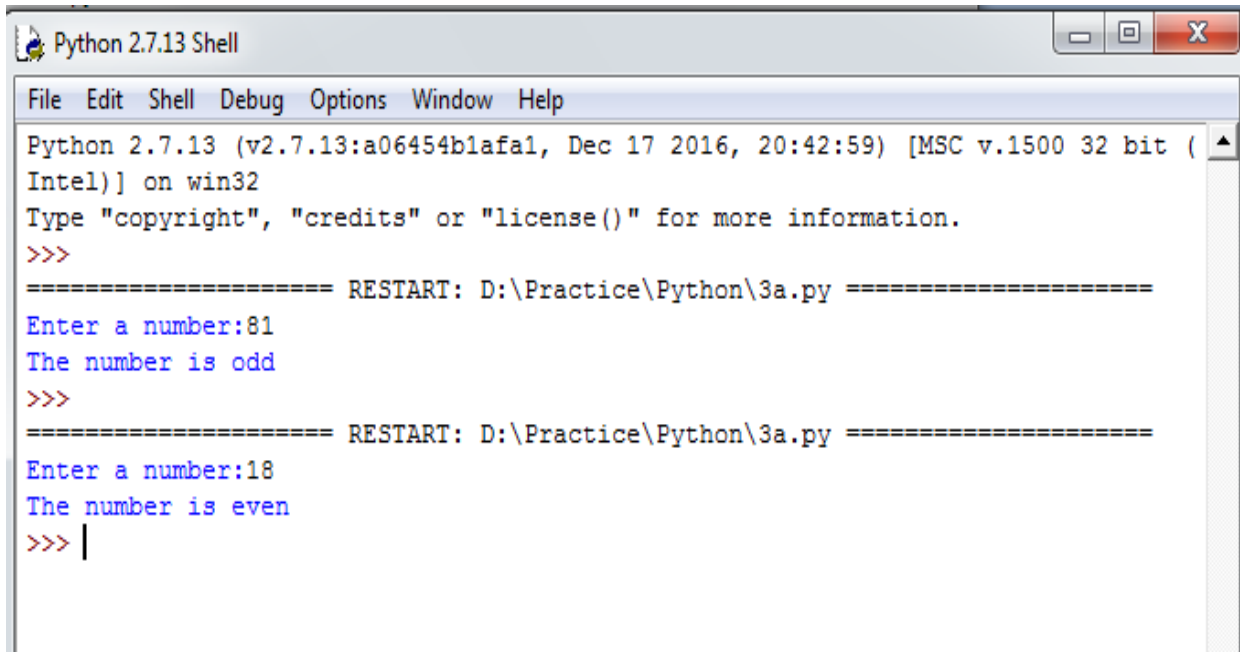
Step4: Display "The number is even" and goto step6

Step5: Display "The number is odd"

Step6: Stop



Output:



The screenshot shows a Windows-style application window titled "Python 2.7.13 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following content:

```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\3a.py =====
Enter a number:81
The number is odd
>>>
===== RESTART: D:\Practice\Python\3a.py =====
Enter a number:18
The number is even
>>> |
```



EXERCISE - 3(b)

Aim:

Using a for loop, write a program that prints out the decimal equivalents of $1/2$, $1/3$, $1/4$, .
 $1/10$.

Description:

A loop statement allows us to execute a statement or group of statements multiple times. We can use for loop to calculate the decimal equivalents of given set of numbers.

for loop statement:

It has the ability to iterate over the items of any sequence, such as a list or a string. Iterating over a sequence is called Traversal. The general form of for loop statement is as follows:

for iterating_var in sequence:

Statement(s)

To find the decimal equivalents, we need to use the method **pow()**. This method is not accessible directly, so we need to import **math** module and then we need to call this method using math static object.

Algorithm:

Output: Decimal equivalents of $1/2$, $1/3$, .., $1/10$

Step1: Start

Step2: Import math module

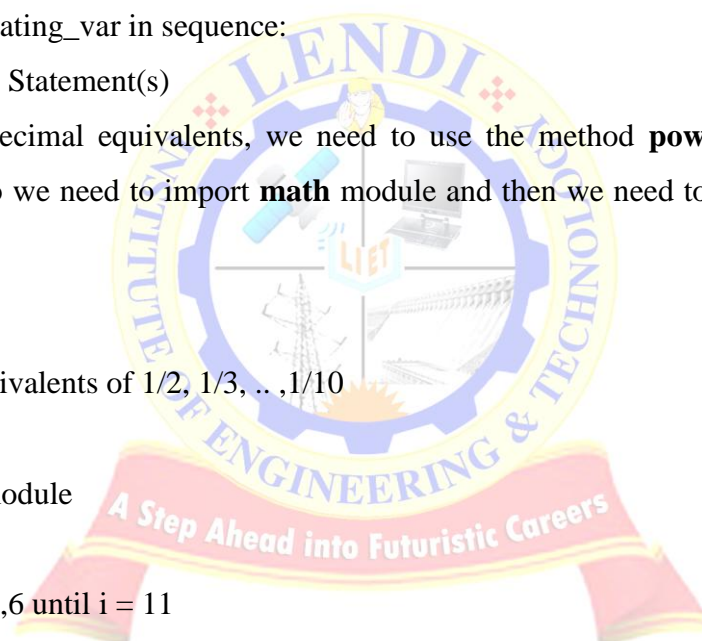
Step3: Initialize $i = 2$

Step4: Repeat Steps 5,6 until $i = 11$

Step5: Print `math.pow(i,-1)`

Step6: Increment i by 1

Step7: Stop



Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\3b.py =====
1/ 2 = 0.5
1/ 3 = 0.333333333333
1/ 4 = 0.25
1/ 5 = 0.2
1/ 6 = 0.166666666667
1/ 7 = 0.142857142857
1/ 8 = 0.125
1/ 9 = 0.111111111111
1/ 10 = 0.1
>>>
```



EXERCISE - 3(c)

Aim:

Write a program using a for loop that loops over a sequence. What is sequence?

Description:

A Sequence is the generic form for an ordered set. There are several types of sequences in Python. The following 3 are most important.

Lists: There are the most versatile sequence type. The elements of a list can be any object and lists are mutable.

Tuples: These are like lists, But Tuples are immutable.

Strings: These are a special type of sequence that can store only characters and having special notations.

Algorithm:

Output: Elements of sequence(List).

Step1: Start

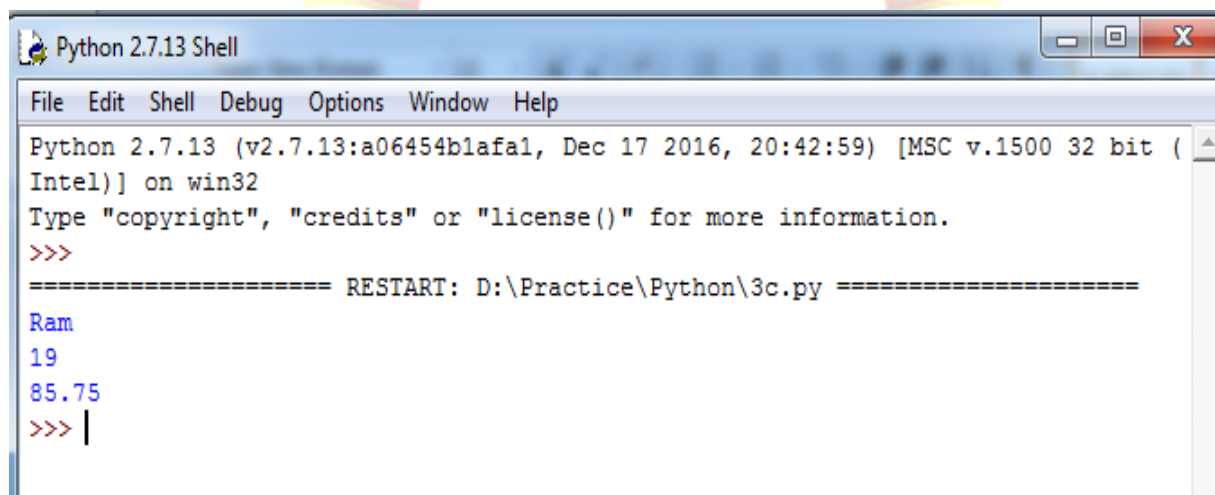
Step2: Initialize the list named a as a = ["Ram",19,85.75]

Step3: Repeat Step4 until the last element in the list is reached

Step4: Print ith element

Step5: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\3c.py =====
Ram
19
85.75
>>> |
```

EXERCISE - 3(d)

Aim:

Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero.

Description:

A loop statement allows us to execute a statement or group of statements multiple times. Here, We are using while loop.

while loop statement:

It repeatedly executes a target statement as long as the given condition is true. The general form while statement is as follows:

while expression:

statements(s)

Here, the statement(s) may be a single a statement or a block of statements. The condition may be any expression, and is true for any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

Algorithm:

Input: A Number

Output: Print the values from given number to zero

Step1: Start

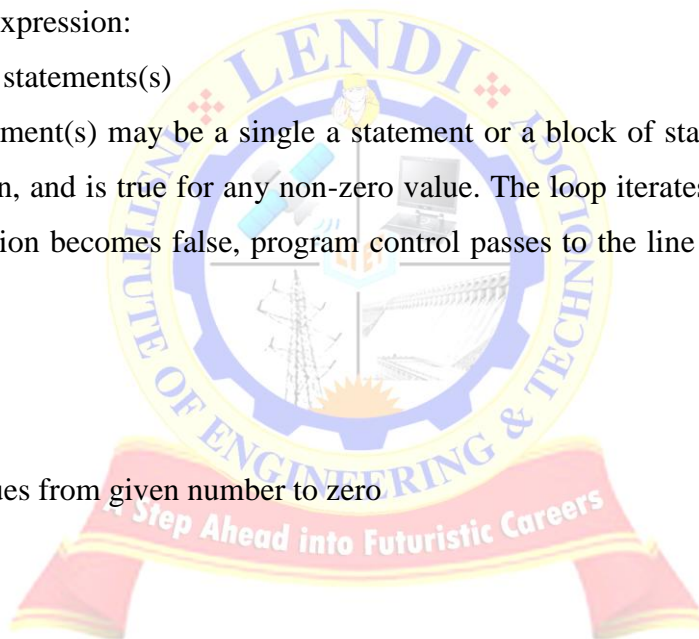
Step2: Read num

Step3: Repeat Steps 4 and 5 while num>=0

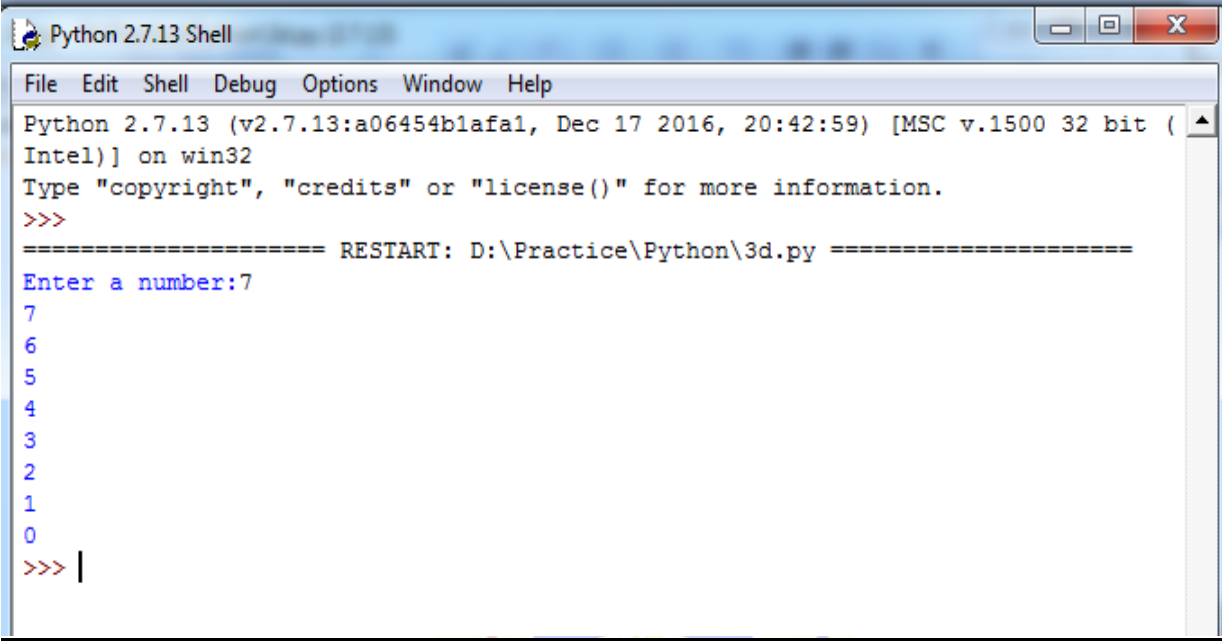
Step4: Display num

Step5: Decrement num by 1

Step6: Stop



Output:

A screenshot of a Python 2.7.13 Shell window. The window title is "Python 2.7.13 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The text area shows the following content:

```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\3d.py =====
Enter a number:7
7
6
5
4
3
2
1
0
>>> |
```

Conclusion:

Student gets the knowledge on conditional statements and looping statements. This experiment maps with CO1 and the students can attain PO1,PO2,PO3,PO4,PO5,PO9,PO10,PO11, PO12 and PSO1,PSO2,PSO3.

Viva questions:

1. What is the difference between a for loop and a while loop?
2. What is the purpose of range() function with an example?
3. How to find ASCII value for a character in Python?
4. What is the difference between sentinel control loop and counter control loop with an example?
5. Does python supports goto statement or not? If yes explain with an example.
6. Does python supports nested while loop or not? If yes explain with an example?
7. What is dangling else problem?
8. How to implement infinite loop?
9. What is pre-test and post-test?
10. How to find the memory location of a variable in Python?

EXERCISE - 4(a)

Aim:

Write a program to find the sum of all primes below two million.

Description:

A Prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 and that is not a prime number is called a composite number.

Here, We are using a **for** loop and a **break** statement.

break:

break statement terminates the loop and resumes execution at the next statement, just like the traditional break statement in C. The break statement can be used in both while and for loops.

If we are using break statement inside a nested loop, it stops the execution of the innermost loop and start executing the next line of code after the block.

Algorithm:

Output: Sum of primes below 2 million

Step1: Start

Step2: Initialize sum = 0 and i = 2

Step3: Repeat Steps 4 to 13 while i < 2000000

Step4: Initialize c = 0

Step5: Check whether i is greater than 2 and i is divisible by 2 or not. If yes, goto Step6 else, goto Step7

Step6: Set c = 1 and goto Step12

Step7: Initialize j = 3

Step8: Repeat Steps 9 to 11 while j <= int(i**0.5)

Step9: Check whether i is divisible by j or not. If yes, goto Step10. Otherwise, goto Step11

Step10: Set c = 1 and goto Step12

Step11: Increment j by 2

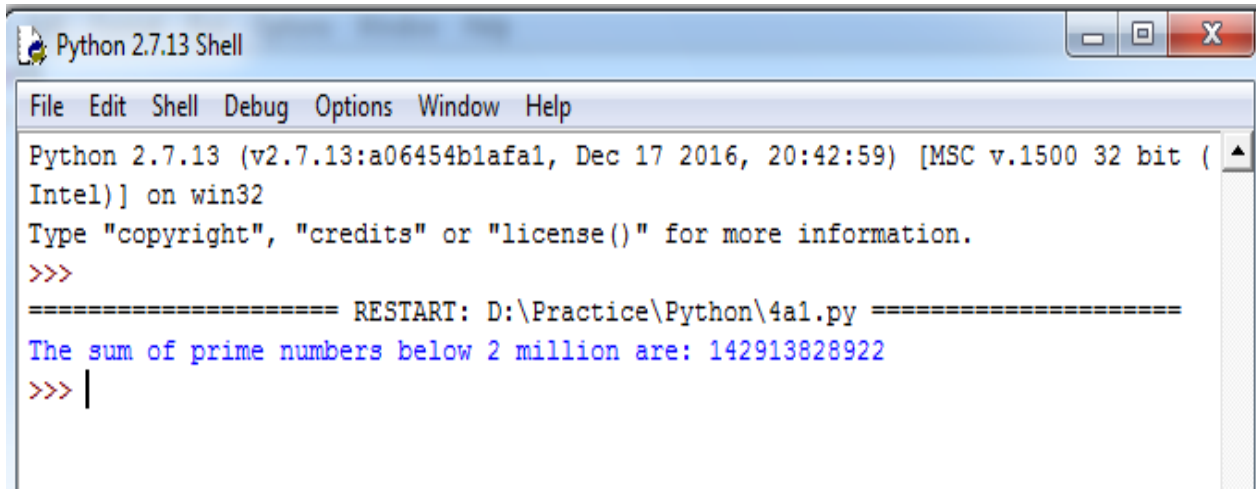
Step12: Check whether c is 0 or not. If yes, goto Step13

Step13: Add i to the sum and store the result in sum

Step14: Display sum

Step15: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\4a1.py =====
The sum of prime numbers below 2 million are: 142913828922
>>> |
```



EXERCISE - 4(b)

Aim:

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

Description:

The Fibonacci sequence is a series of numbers where a number is found by adding up the two numbers before it. Starting with 0 and 1, the series will be 0,1,1,2,3,5,8,13 and so forth.

In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the Recurrence relation:

$$F_n = F_{n-1} + F_{n-2}$$

With initial values $F_0 = 0$ and $F_1 = 1$

Algorithm:

Output: Sum of even Fibonacci numbers below 4 million

Step1: Start

Step2: Initialize fib1 to 1, fib2 to 2 and sum to 2

Step3: Add fib1 and fib2 and store the result in fib3

Step4: Repeat Steps 5 to 9 until $\text{fib3} < 4000000$

Step5: Check whether fib3 is divisible by 2 or not. If yes, goto Step6. Otherwise, goto step7

Step6: Add fib3 to sum and store the result in sum

Step7: Set fib1 to fib2

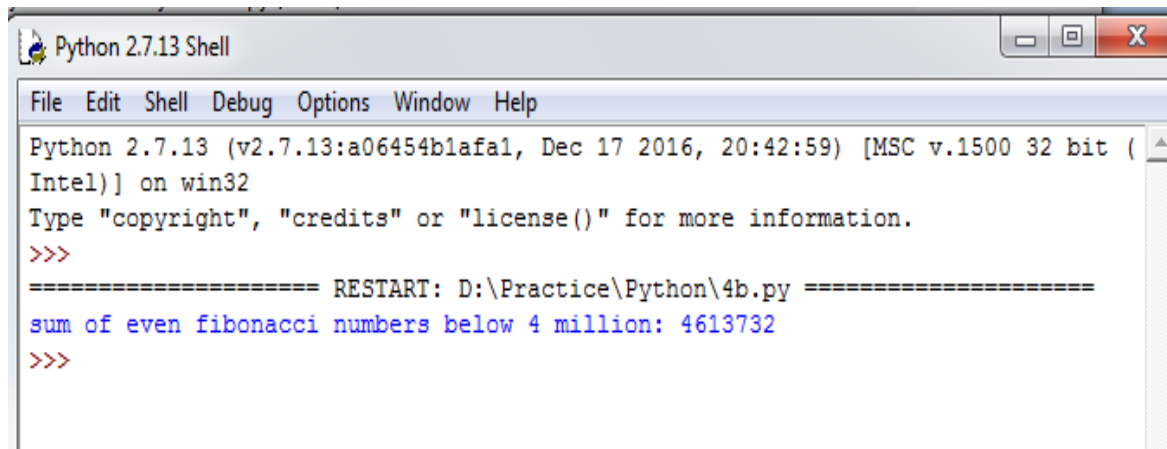
Step8: Set fib2 to fib3

Step9: Add fib1 and fib2 and store the result in fib3

Step10: Display sum

Step11: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\4b.py =====
sum of even fibonacci numbers below 4 million: 4613732
>>>
```

Conclusion:

Student gets the knowledge on conditional statements and looping statements. This experiment maps with CO1 and students can attain PO1,PO2,PO3,PO4,PO5,PO9,PO10,PO11, PO12 and PSO1,PSO2,PSO3.

Viva questions:

1. What is the return type of range() function?
2. What is the output of range(20) ?
3. What is the output of range(1,8,2)?
4. What is the output of range(100,30,-20)?
5. What is the purpose of break statement in python?
6. What is the purpose of continue statement in python?
7. What is the purpose of pass statement in python?
8. Explain the execution process of for...in ?
9. Does continue statement works without loops?
10. Which statement is used for writing empty loops? Explain with an example.

EXERCISE - 5(a)

Aim:

Write a program to count the numbers of characters in the string and store them in a dictionary data structure.

Description:

Traverse each character in the string and its occurrence is stored in the dictionary. Here, We are using a String and a Dictionary.

String:

A String is a sequence of characters. Strings can be created by enclosing characters inside a single quote or double quotes. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

For reading strings in Python2.7, we are using `raw_input()`. This function reads a line from input(i.e., the user) and returns a string by stripping a trailing new line.

Dictionary:

The dictionary is Python's built-in mapping type. Dictionaries map keys to values and these <key,value> pairs provides a useful way to store data in python. The only way to access the value part of the dictionary is by using the key.

We can specify the dictionary <key,value> pairs between '{' and '}'. <key,value> pairs are specified as a list(separated by commas).

Algorithm:

Input: A String

Output: A Dictionary with keys (characters in the string) and values (frequency of corresponding characters)

Step1: Start

Step2: Read the string `input_str`

Step2: Create an empty dictionary `dict_str`

Step3: Repeat Steps 4 to 6 until the end of `input_str` is reached

Step4: Check whether the i^{th} character is present in the dictionary `dict_str` as a key element or not. If yes, goto Step5. Otherwise, goto Step6

Step5: Increment the value of i^{th} key element by 1 and continue with the next iteration

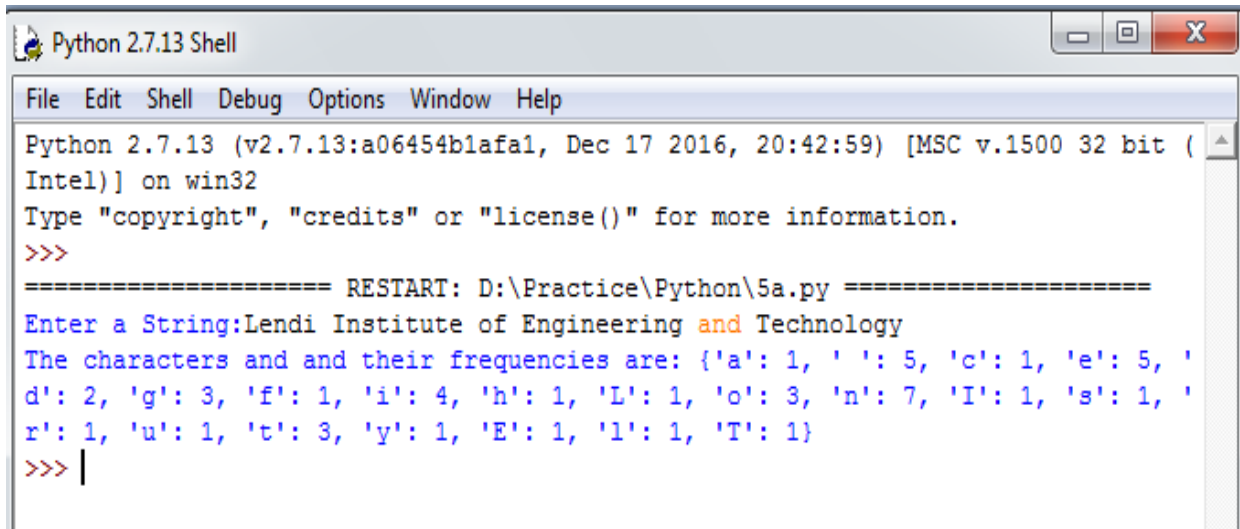
Python Programing Lab Manual

Step6: Assign the value of i^{th} key element to 1

Step7: Display the dictionary dict_str

Step8: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\5a.py =====
Enter a String:Lendi Institute of Engineering and Technology
The characters and and their frequencies are: {'a': 1, ' ': 5, 'c': 1, 'e': 5, 'd': 2, 'g': 3, 'f': 1, 'i': 4, 'h': 1, 'L': 1, 'o': 3, 'n': 7, 'I': 1, 's': 1, 'r': 1, 'u': 1, 't': 3, 'y': 1, 'E': 1, 'l': 1, 'T': 1}
>>> |
```



EXERCISE - 5(b)

Aim:

Write a program to use split and join methods in the string and trace a birthday with a dictionary data structure.

Description:

Compare the entered birthdate with each and every person's date of birth in the dictionary and check whether the entered birthdate is present in the dictionary or not. Here, We are using two built-in methods of the string: split() and join().

split():

This method breaks up a string using the specified separator and returns a list of strings. The general form of split() is as follows:

`str.split(separator)`

join():

This method provides a flexible way to concatenate a string. This method returns a string in which the string elements of a sequence have been joined by 'sep' separator. The general form of join() is as follows:

`sep.join(sequence)`

Here, sequence is the sequence of elements to be joined.

Algorithm:

Input: A Dictionary and a Date Of Birth

Output: A Message

Step1: Start

Step2: Initialize the dictionary birthday to {"Ram":"15/07/1989", "Krishna":"09/12/1988", "Venkat":"11/02/2016"}

Step3: Read date in the format dd-mm-yyyy.

Step4: Split the date using the separator '-' and store the resultant list in date_list

Step5: Join the date_list elements using the separator '/' and store the resultant string in dob

Step6: Repeat Steps 7 and 8 until end of the dictionary is reached

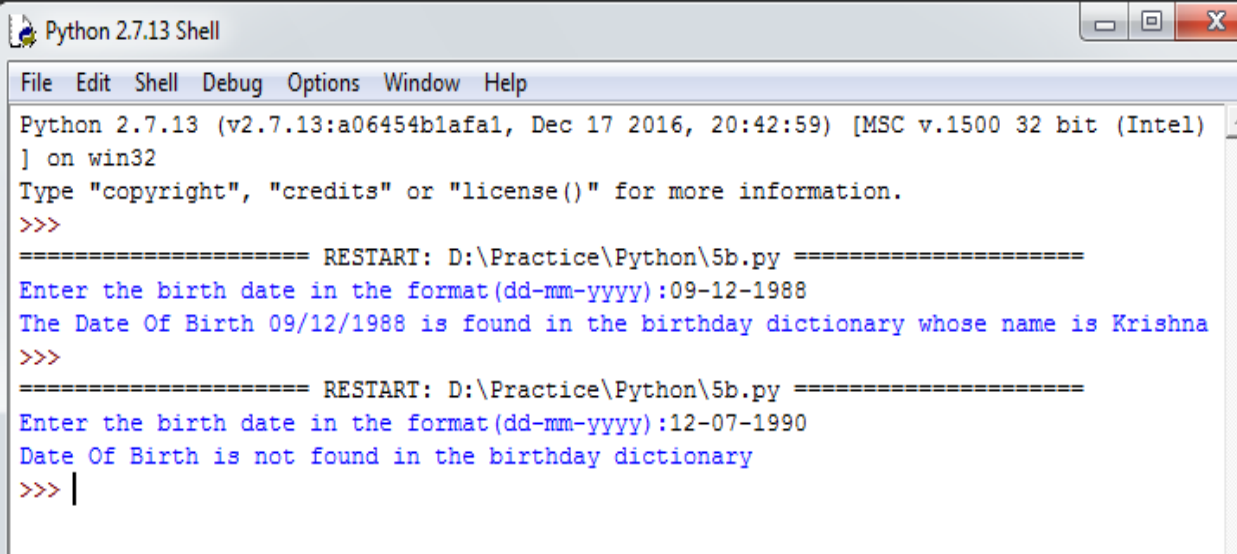
Step7: Check whether dob and corresponding value(Date Of Birth) for the ith key(Person Name) element are equal or not. If yes, goto Step8

Step8: Display 'Date Of Birth is found in the birthday dictionary' and goto Step10

Step9: Display 'Date Of Birth is not found'

Step10: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\5b.py =====
Enter the birth date in the format(dd-mm-yyyy):09-12-1988
The Date Of Birth 09/12/1988 is found in the birthday dictionary whose name is Krishna
>>>
===== RESTART: D:\Practice\Python\5b.py =====
Enter the birth date in the format(dd-mm-yyyy):12-07-1990
Date Of Birth is not found in the birthday dictionary
>>> |
```

Conclusion:

Student gets the knowledge on strings and dictionaries. This experiment maps with CO1,CO2 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10,PO11,PO12 and PSO1,PSO2,PSO3.

Viva questions:

1. What is indexing?
2. What is the purpose of split and join method in python?
3. In Python, What is slicing?
4. What do you mean string immutable?
5. What is the use of id() function?
6. What is negative index in Python?
7. How to convert a string to all lowercase?
8. How to replaces all occurrences of old substring in string with new string?
9. How will you change case for all letters in string?
10. How to check whether the string consisting of alphanumeric characters?

EXERCISE - 6(a)

Aim:

Write a program combine_lists that combines these lists into a dictionary.

Description:

Python offers a range of compound data types often referred to as Sequences. **List** is one of the most frequently used and very versatile data type used in Python.

In Python Programming, a List is created by placing all the elements inside a square bracket [], separated by commas. It can have any number of items and they may be of different types.

Examples:

```
list1 = [10,20,30,40]
```

```
list2 = ["Ram",19,87.81]
```

Similar to string indices, list indices starts at 0 and lists can be sliced, concatenated and so on..

Here, The task is to combine 2 lists into a dictionary. That means, One list elements will become the keys and another list elements will become the values in the resultant dictionary.

Algorithm:

Input: Two Lists

Output: A Dictionary

Step1: Start

Step2: Create two empty lists names and salaries, an empty dictionary person_dict and Initialize j=0

Step3: Read n value

Step4: Read n names into the list names

Step5: Read n Salaries into the list salaries

Step6: Display the lists names and salaries

Step7: Repeat Steps 8 and 9 until end of the list names is reached

Step8: Store the jth item in salaries list as value in the person_dict with key ith item in names list

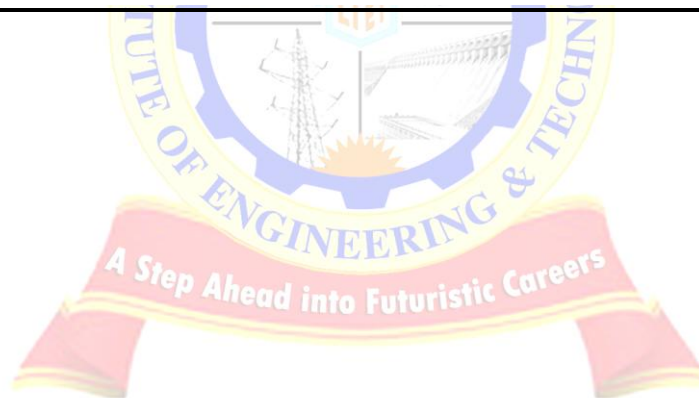
Step9: Increment j by 1

Step10: Display the dictionary person_dict

Step11: Stop

Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\6a.py =====
Enter the n value:3
Enter 3 Names:
Enter the Person1 Name:Ram
Enter the Person2 Name:Krishna
Enter the Person3 Name:Venkat
Enter 3 Salaries:
Enter the Person1 Salary:10000
Enter the Person2 Salary:15000
Enter the Person3 Salary:20000
Names list is: ['Ram', 'Krishna', 'Venkat']
Salaries list is: [10000, 15000, 20000]
After combining two lists, The dictionary is: {'Krishna': 15000, 'Ram': 10000, 'Venkat': 20000}
>>>
```



EXERCISE - 6(b)

Aim:

Write a program to count frequency of characters in a given file. Can you use character frequency to tell whether the given file is a Python program file, C program file or a text file?

Description:

Traverse each character in the file and its occurrence is stored in the dictionary. Here, We are using the concept of files and a Dictionary.

Files:

Python provides basic functions and methods necessary to manipulate files by default. Some of the useful functions in this program are:

open():

Before we can read or write a file, we have to open it using Python's built-in open(). This function creates a file object, which would be utilized to call other support methods associated with it.

The **Syntax** for opening a file object in Python is:

```
file_object = open("Name of the file","Mode of the file")
```

read():

If we need to extract string that contains all characters in the file, We can use read(). The **Syntax** for read() in Python is:

```
str = file_object.read()
```

Dictionary:

The dictionary is Python's built-in mapping type. Dictionaries map keys to values and these <key,value> pairs provides a useful way to store data in python. The only way to access the value part of the dictionary is by using the key.

We can specify the dictionary <key,value> pairs between '{' and '}'. <key,value> pairs are specified as a list(separated by commas).

Algorithm:

Input: A File

Output: A Dictionary and A Message

Step1: Start

Python Programing Lab Manual

Step2: Read file_name

Step3: Open the specified file_name in read mode

Step4: Read all characters of the file into the string file_contents

Step5: Create an empty dictionary Dict

Step6: Find the name of the file and store it in file_name

Step7: Repeat Steps 8 to 10 until the end of file_contents is reached

Step8: Check whether the i^{th} character is present in the dictionary Dict as a key element or not. If yes, goto Step9. Otherwise, goto Step10

Step9: Increment the value of i^{th} key element by 1 and continue with the next iteration

Step10: Assign the value of i^{th} key element to 1

Step11: Display keys and corresponding values in the dictionary Dict

Step12: Check whether the file_name ends with .c or not. If yes, goto Step13. Otherwise, goto Step14

Step13: Display 'Input file is C program file' and goto Step18

Step14: Check whether the file_name ends with .py or not. If yes, goto Step15. Otherwise, goto Step16

Step15: Display 'Input file is Python program file' and goto Step18

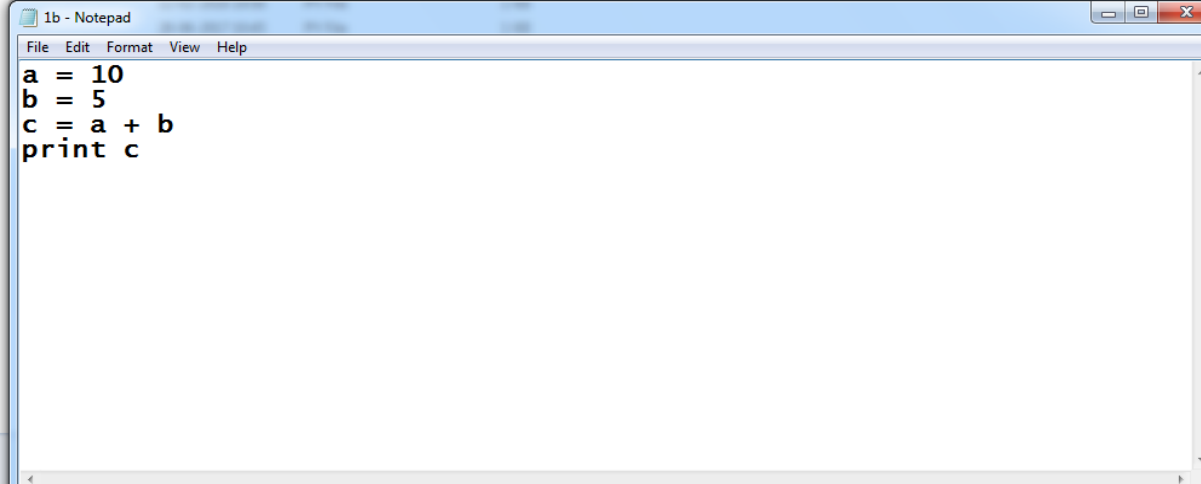
Step16: Check whether the file_name ends with .txt or not. If yes, goto Step17. Otherwise, goto Step18

Step17: Display 'Input file is Text file' and goto Step18

Step18: Close the file

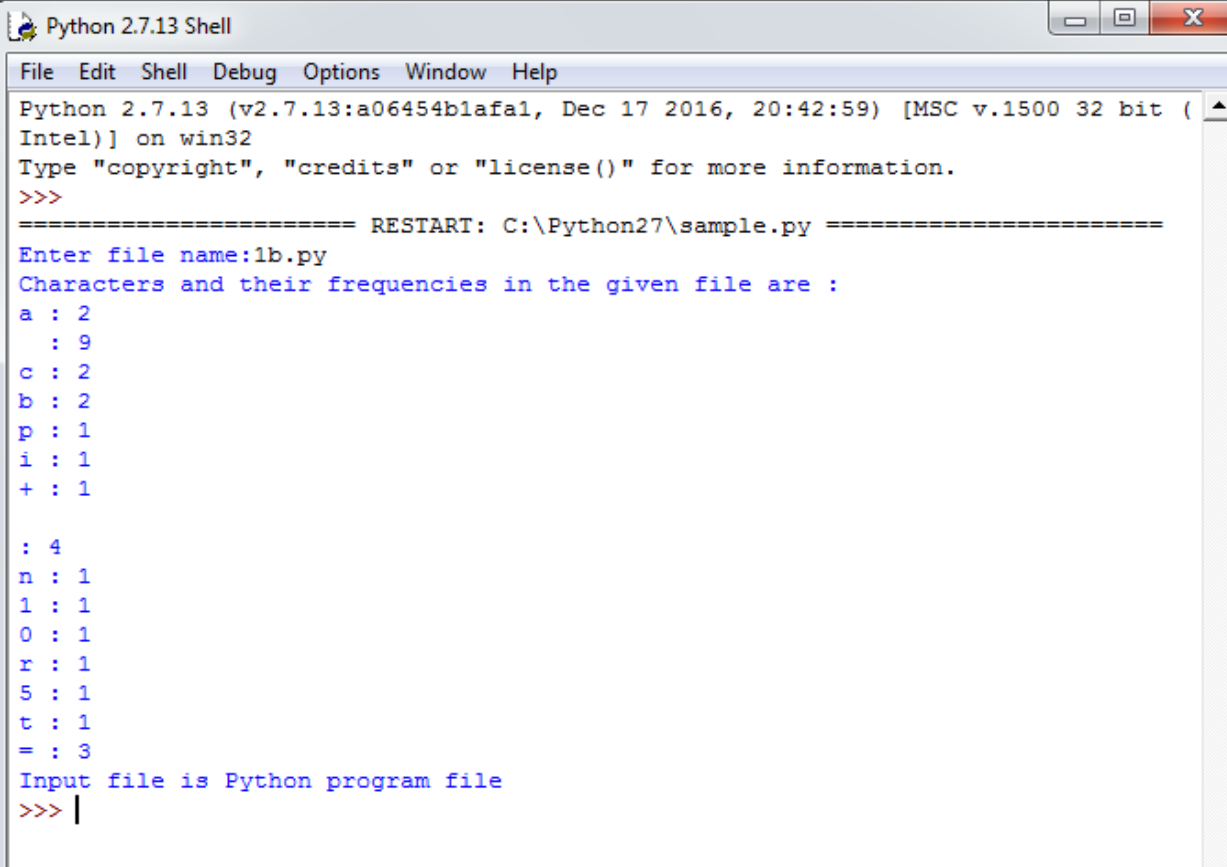
Step19: Stop

Sample Input:

A screenshot of a Notepad window titled '1b - Notepad'. The window has a menu bar with 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains the following code:

```
a = 10
b = 5
c = a + b
print c
```

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\sample.py =====
Enter file name:1b.py
Characters and their frequencies in the given file are :
a : 2
  : 9
c : 2
b : 2
p : 1
i : 1
+ : 1

: 4
n : 1
l : 1
o : 1
r : 1
s : 1
t : 1
= : 3
Input file is Python program file
>>> |
```

Conclusion:

Student gets the knowledge on basic operations on files, dictionaries and lists. This experiment maps with CO2,CO3 and students can attain PO1,PO2,PO3,PO4, PO5,PO6,PO9,PO10, PO11,PO12 and PSO1,PSO2,PSO3.

Viva Questions:

1. How to initialize a dictionary in python?
2. How to get all the keys from the dictionary?
3. When to choose dictionary instead of a list?
4. Write a function to sort a list?
5. Explain Python's zip() function?
6. Write a function to compare two lists?
7. What is the difference between r+ and w+ modes?

8. What is the purpose of tell()?
9. Which function is used to read single line from file?
10. What is the output of the following snippet of code?

```
total={ }  
def insert(items):  
    if items in total:  
        total[items] += 1  
    else:  
        total[items] = 1  
insert('Apple')  
insert('Ball')  
insert('Apple')  
print (len(total))
```



EXERCISE - 7(a)

Aim:

Write a program to print each line of a file in reverse order.

Description:

Traverse each line of the file and we have to display every line contents in reverse order. Here, We are using the some of the basic methods of files and a string method strip().

Python provides basic functions and methods necessary to manipulate files by default. We can do most of the file manipulations using a file object.

open():

Before we can read or write a file, we have to open it using Python's built-in open(). This function creates a file object, which would be utilized to call other support methods associated with it.

The **Syntax** for opening a file object in Python is:

```
file_object = open("Name of the file","Mode of the file")
```

close():

This method of a file object flushes any unwritten information and closes the file object. Python automatically closes a file when the reference object of a file is reassigned to another file.

The **Syntax** for closing a file object in Python is:

```
file_object.close()
```

readlines():

This method will returns every line of the file as a list. The **Syntax** for readlines() in Python is:

```
file_lines = file_object.readlines()
```

strip():

This method returns a copy of the string in which all chars have been stripped from the beginning and end of the string (default whitespace characters).

The **Syntax** for strip() is Python is:

```
str.strip([chars])
```

Algorithm:

Input: A file

Output: Each line of the file in reverse order

Step1: Start

Step2: Read file_name

Step3: Open the specified file_name in read mode

Step4: Read all lines of the file into the list fileContents

Step5: Repeat Steps 6 to 8 until end of the list fileContents is reached

Step6: Create an empty string line1

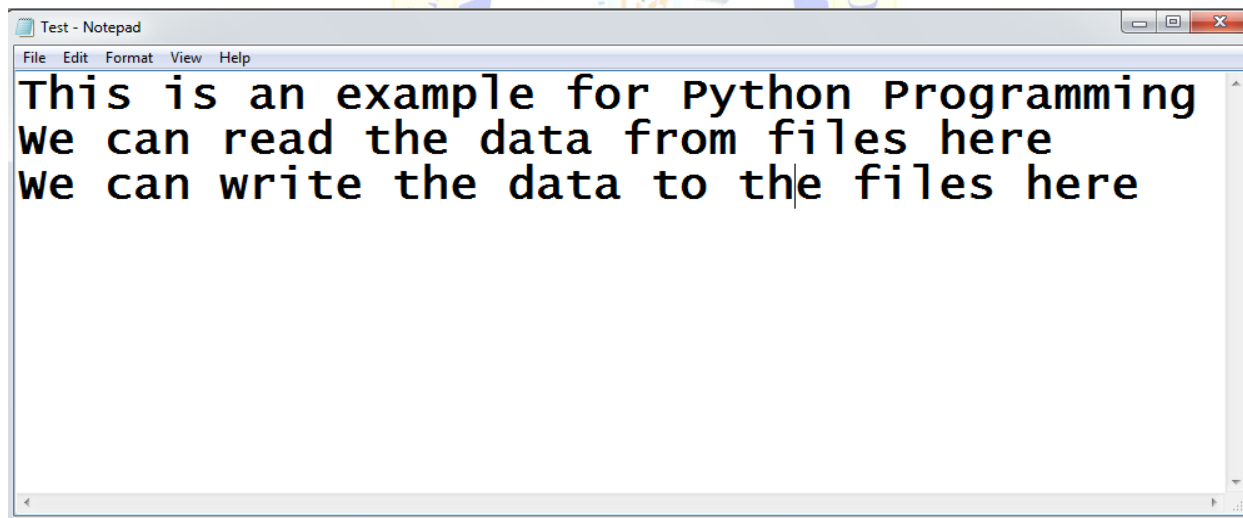
Step7: Remove '\n' from end of the ith element in the list

Step8: Reverse the ith element in the list and display it on the screen

Step9: Close the file

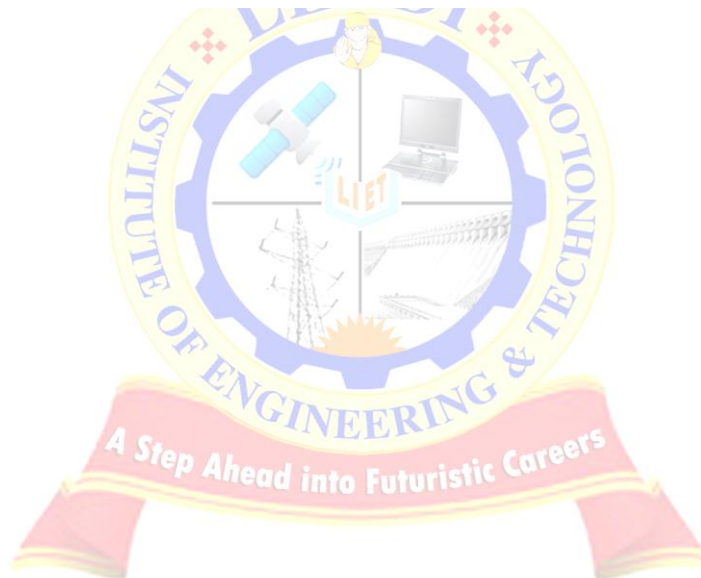
Step10: Stop

Sample Input:



Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454blafa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\7a.py =====
Enter file name:Test.txt
gnimmargorP nohtyP rof elpmaxe na si sihT
ereh selif morf atad eht daer nac eW
ereh selif eht ot atad eht etirw nac eW
>>>
```



EXERCISE - 7(b)

Aim:

Write a program to compute the number of characters, words and lines in a file.

Description:

Traverse each character in the file so that we can find number of characters, words and lines in a text file. Here, We are using the some of the basic methods of files.

We already discussed about open() and close() in the previous programs.

read():

If we need to extract string that contains all characters in the file, We can use read(). The

Syntax for read() in Python is:

```
str = file_object.read()
```

Algorithm:

Input: A file

Output: Number of characters, words and lines in an input file

Step1: Start

Step2: Read file_name

Step3: Read all characters into the string fileContents

Step4: Initialize noc to 0, nol to 1 and now to 1

Step5: Repeat Steps 6 to 8 until end of the string fileContents is reached

Step6: Increment noc by 1

Step7: Check whether ith character in fileContents is '\n' or not. If yes, Increment nol by 1

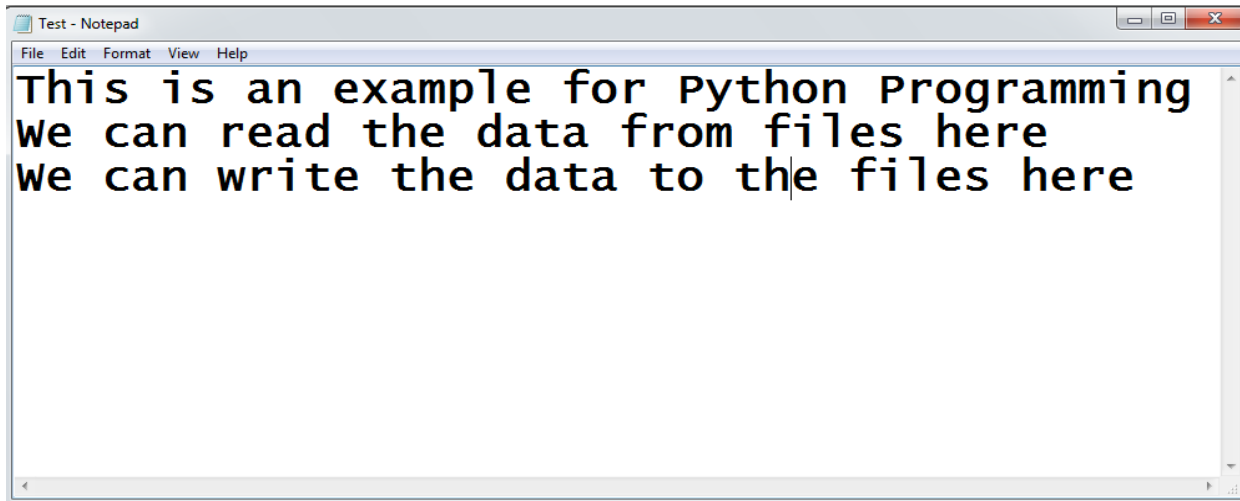
Step8: Check whether ith character is (' ' or '\n') or not. If yes, Increment now by 1

Step9: Display nol, now and noc

Step10: Close the file

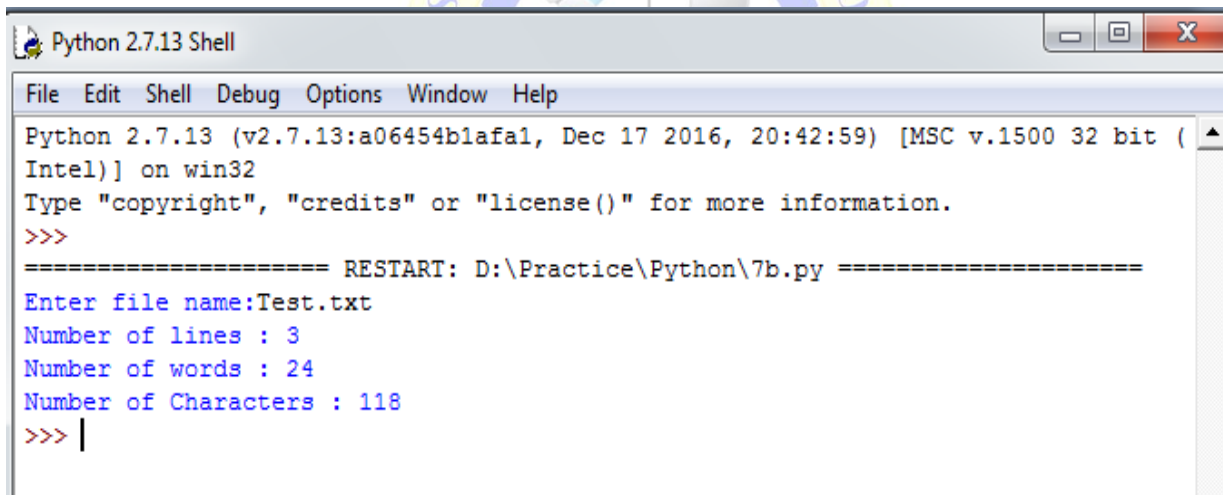
Step11: Stop

Sample Input:



```
Test - Notepad
File Edit Format View Help
This is an example for Python Programming
We can read the data from files here
We can write the data to the files here
```

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\7b.py =====
Enter file name:Test.txt
Number of lines : 3
Number of words : 24
Number of Characters : 118
>>> |
```

Conclusion:

Student gets the knowledge on basic concepts of files and lists. This experiment maps with CO2,CO3 and students can attain PO1,PO2,PO3,PO4, PO5,PO6,PO9,PO10,PO11,PO12 and PSO1, PSO2,PSO3.

Viva Questions:

1. What is file?
2. Explain all the file processing modes supported by Python?
3. What is text file and binary file?
4. What are different file object attributes in python?
5. How to write text in to a file?
6. How to read text from the file?
7. How to get the position of the file pointer?
8. How to rename existing file?
9. How to delete file from directory?
10. Explain how can you make a Python Script executable on Unix?



EXERCISE - 8(a)

Aim:

Write a function ball_collide that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding. Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius. If (distance between two balls centers) <= (sum of their radii) then (they are colliding).

Description:

We need to write a function to find out whether two balls are colliding or not by using the distance formulae i.e., If (distance between two balls centers) <= (sum of their radii) then we must return True. Otherwise, we must return False. Distance between two ball centers can be calculated by using the formulae:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Here, We are using the concept **functions**.

Functions:

A function is a group of related statements that performs a specific task. Functions help break our program into smaller and modular chunks. As our program grows longer, functions make it more organized and manageable.

Furthermore, It avoids repetition and code reusable. The general form of a function is as follows:

```
def function_name(parameter_list):  
    """doc-string"""  
    statement(s)
```

Function Call:

Once we define a function, We can call it from another function, program or even from the Python prompt. To call a function, We can simply type the function name with appropriate parameters.

return statement:

This statement is used to exit a function and go back to place from where it was called. The general form of return statement is:

```
return [expression]
```

Algorithm:

Function ball_collide((x1,y1,r1),(x2,y2,r2))

Input: 2 tuples (containing co-ordinates and radii of two balls)

Output: Boolean Value

Step1: Start

Step2: Calculate $\text{math.sqrt}((x2-x1)**2 + (y2-y1)**2)$ and store the result in distance

Step3: Add the radii r1 and r2 and store the result in sum_radius

Step4: Check whether if distance is less than or equal to sum_radius or not. If yes, goto step5.

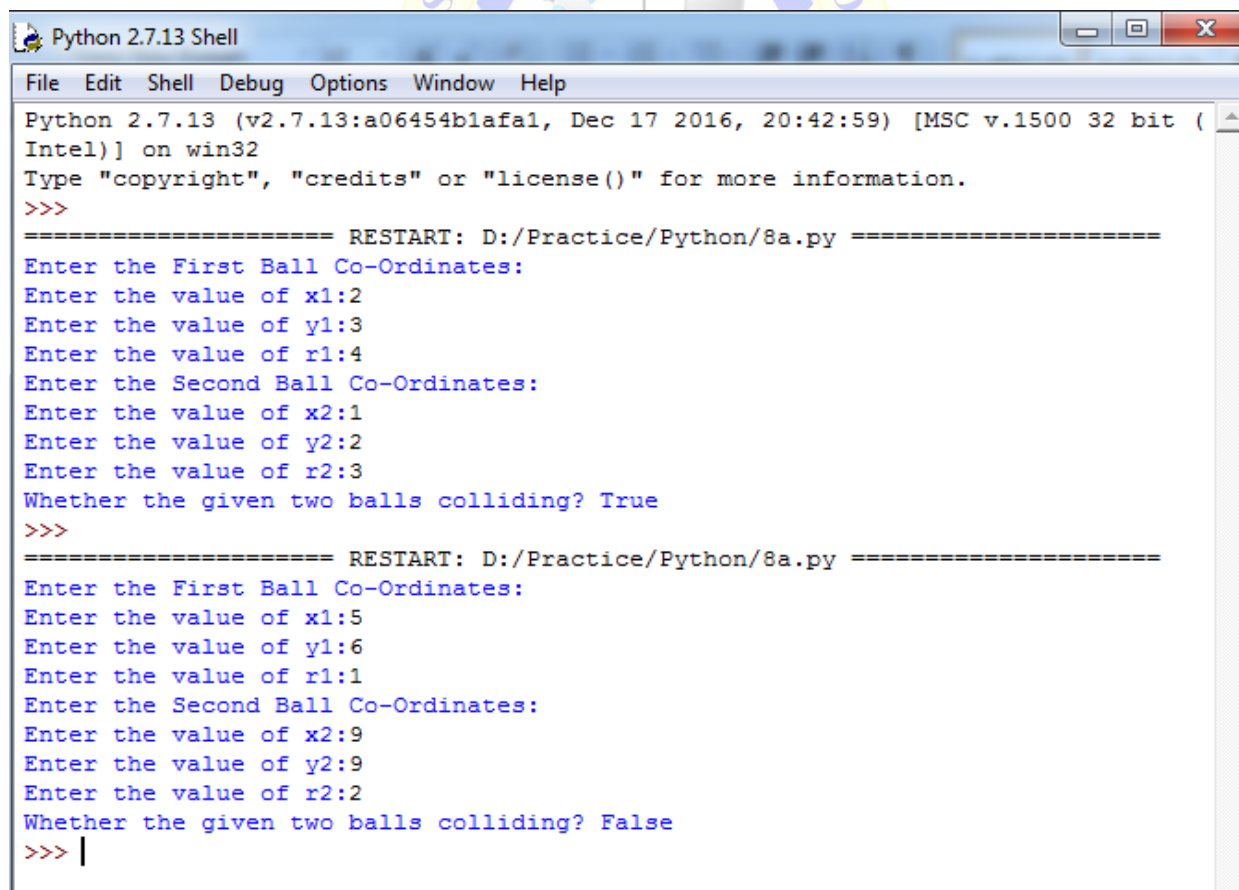
Otherwise, goto Step6

Step5: return True and goto Step7

Step6: return False

Step7: Stop

Output:



```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/8a.py =====
Enter the First Ball Co-Ordinates:
Enter the value of x1:2
Enter the value of y1:3
Enter the value of r1:4
Enter the Second Ball Co-Ordinates:
Enter the value of x2:1
Enter the value of y2:2
Enter the value of r2:3
Whether the given two balls colliding? True
>>>
===== RESTART: D:/Practice/Python/8a.py =====
Enter the First Ball Co-Ordinates:
Enter the value of x1:5
Enter the value of y1:6
Enter the value of r1:1
Enter the Second Ball Co-Ordinates:
Enter the value of x2:9
Enter the value of y2:9
Enter the value of r2:2
Whether the given two balls colliding? False
>>> |
```


EXERCISE - 8(b)

Aim:

Find mean, median, mode for the given set of numbers in a list.

Description:

We need to write a function to find out mean, median and mode for a given set of numbers in a list. Mean, Median and Mode are 3 kinds of averages.

Mean:

It is the average we are used to, where we add up all the numbers and then divide by the total number of numbers.

Median:

It is the middle value in the list of numbers. To find median, all numbers have to be listed in numerical order from smallest to largest.

If total numbers are odd, then median is the middle value. If total numbers are even, then median is the average of 2 middle values.

Mode:

It is the value that occurs most often. If no number is repeated among a list of numbers then there is no mode for the list.

Algorithm:

Function mean_median_mode(marks)

Input: A List

Output: Mean, Median and Mode

Step1: Start

Step2: Initialize sum to 0

Step3: Calculate the length of the list marks and store the result in length

Step4: Calculate the addition of all numbers in the list and store the result in sum

Step5: Divide sum and length and store the result in mean

Step6: Sort all elements in the list marks and store the result in the list marks1

Step7: Check whether the total number of elements in the list marks1 is odd or not. If yes, goto

Step8. Otherwise, goto Step9

Step8: Assign middle element of the list marks1 to median and goto Step10

Python Programing Lab Manual

Step9: Calculate the average of middle two elements of the list marks1 and store the result in median

Step10: Create an empty dictionary dict_mode and an empty list mode

Step11: Repeat Steps 12 to 14 until end of the list marks is reached

Step12: Check whether the i^{th} element of the list marks is present in the dictionary dict_mode or not.

If yes, goto Step13. Otherwise, goto Step14

Step13: Increment value of the key element (i^{th} element in the list marks) in the dictionary dict_mode by 1 and continue with the next element in the list marks

Step14: Assign value of the key element (i^{th} element in the list marks) in the dictionary dict_mode to 1

Step15: Retrieve all values in the dictionary dict_mode and store retrieved values in the list max_frequency_list

Step16: Find out maximum value in the list max_frequency_list and store the result value in max_frequency

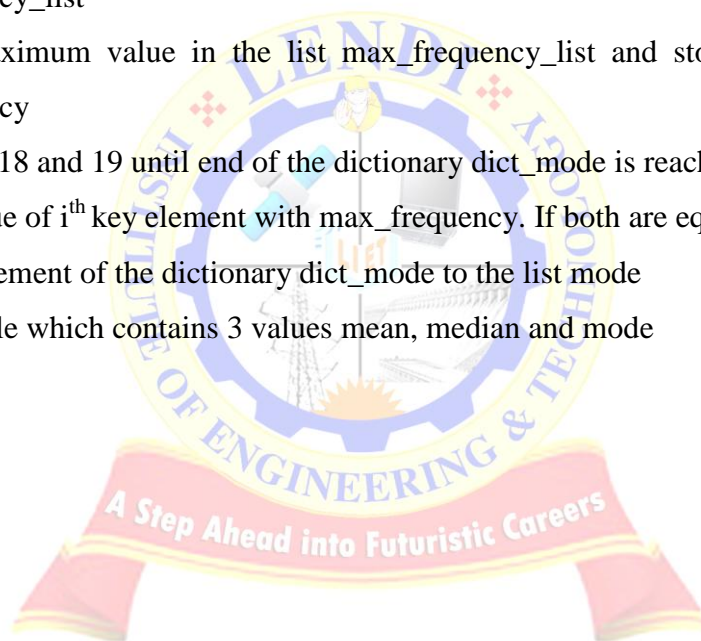
Step17: Repeat Steps 18 and 19 until end of the dictionary dict_mode is reached

Step18: Compare value of i^{th} key element with max_frequency. If both are equal, then goto Step19

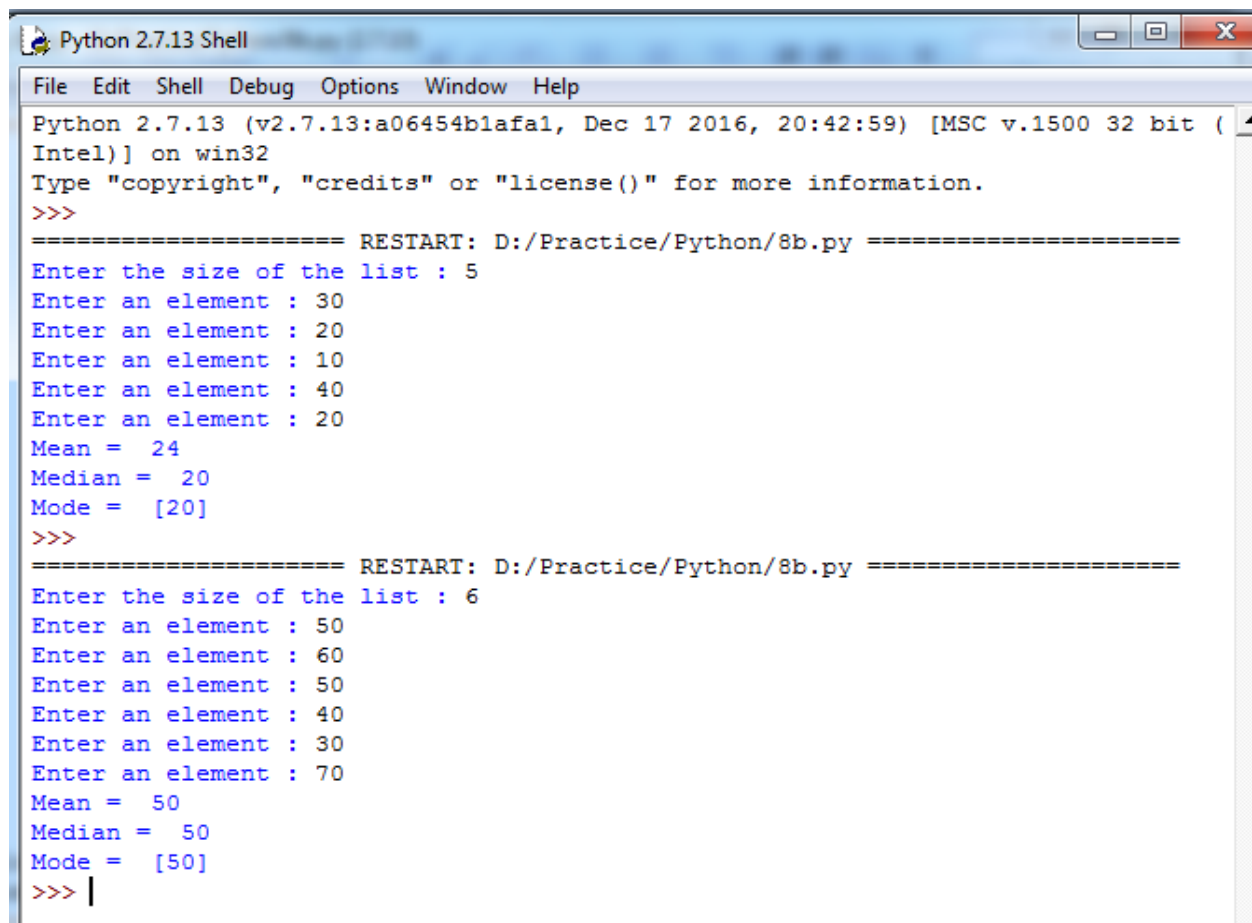
Step19: Add i^{th} key element of the dictionary dict_mode to the list mode

Step20: return the tuple which contains 3 values mean, median and mode

Step21: Stop



Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/8b.py =====
Enter the size of the list : 5
Enter an element : 30
Enter an element : 20
Enter an element : 10
Enter an element : 40
Enter an element : 20
Mean = 24
Median = 20
Mode = [20]
>>>
===== RESTART: D:/Practice/Python/8b.py =====
Enter the size of the list : 6
Enter an element : 50
Enter an element : 60
Enter an element : 50
Enter an element : 40
Enter an element : 30
Enter an element : 70
Mean = 50
Median = 50
Mode = [50]
>>> |
```

Conclusion:

Student gets the knowledge on functions, lists and tuples. This experiment maps with CO2, CO4 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10,PO11,PO12 and PSO1,PSO2, PSO3.

Viva Questions:

1. What is the importance of functions in python?
2. Why programmers need to choose a function programming in python?
3. What is function call and function definition?
4. What is parameter/argument?
5. What is pass by reference and pass by value in python?

6. What is the output of the below program?

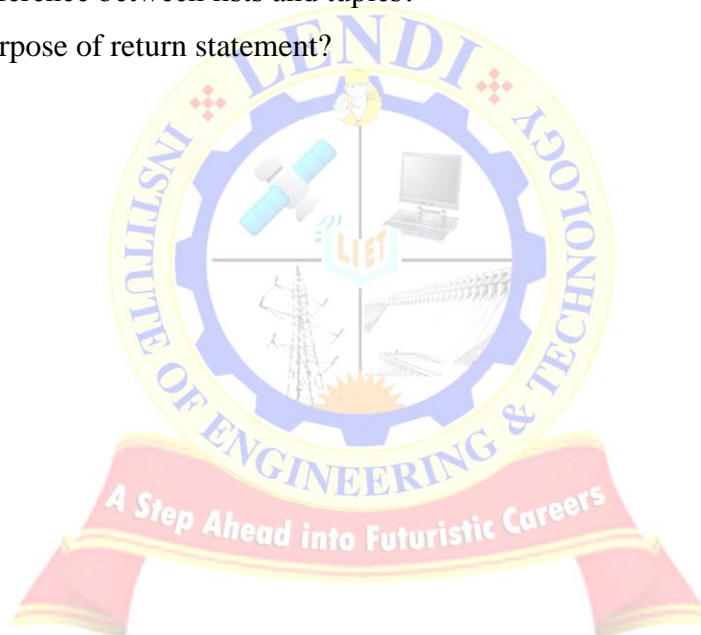
```
def printMax(a, b):  
    if a > b:  
        print(a, ' is maximum')  
    elif a == b:  
        print(a, ' is equal to', b)  
    else:  
        print(b, ' is maximum')  
  
printMax(3, 4)
```

7. How to retrieve all the values in a dictionary?

8. How to sort all the elements of a list in python?

9. What's the difference between lists and tuples?

10. What is the purpose of return statement?



EXERCISE NO : 9(a)

Aim:

Write a function nearly_equal to test whether two strings are nearly equal.

Description:

Two strings are said to be nearly equal iff a single mutation applied to one string will result in another string. That means, Given two strings s1 and s2, find if s1 can be converted to s2 with exactly one edit(mutation). If yes, then the function should return a True value as the result. Otherwise, it must return a False value as the result.

Algorithm:

Function nearly_equal (s1, s2)

Input: Two Strings

Output: A Boolean value

Step1: Start

Step2: Calculate the lengths of s1 and s2 and store the results in m, n

Step3: Check whether the absolute difference between m and n is greater than 1 or not. If yes, goto

Step4. Otherwise, goto Step5

Step4: return the boolean value False and goto Step22

Step5: Initialize count to i and j to 0

Step6: Repeat Steps 7 to 16 until $i < m$ and $j < n$

Step7: Check whether i^{th} character in s1 and j^{th} character in s2 are equal or not. If yes, goto Step16.

Otherwise, goto Step8

Step8: Check whether the value of count is 1 or not. If yes, goto Step9. Otherwise, goto Step10

Step9: return the boolean value False and goto Step22

Step10: Check whether m is greater than n or not. If yes, goto Step11. Otherwise, goto Step12

Step11: Increment the value of i by 1 and goto Step15

Step12: Check whether m is less than n or not. If yes, goto Step13. Otherwise, goto Step14

Step13: Increment the value of j by 1 and goto Step15

Step14: Increment the values of i and j by 1

Step15: Increment the value of count by 1 and continue with the next iteration

Step16: Increment the values of i and j by 1

Python Programing Lab Manual

Step17: Check whether the condition ($i < m$ or $j < n$) is True or not. If yes, goto Step18. Otherwise, goto Step19

Step18: Increment the value of count by 1

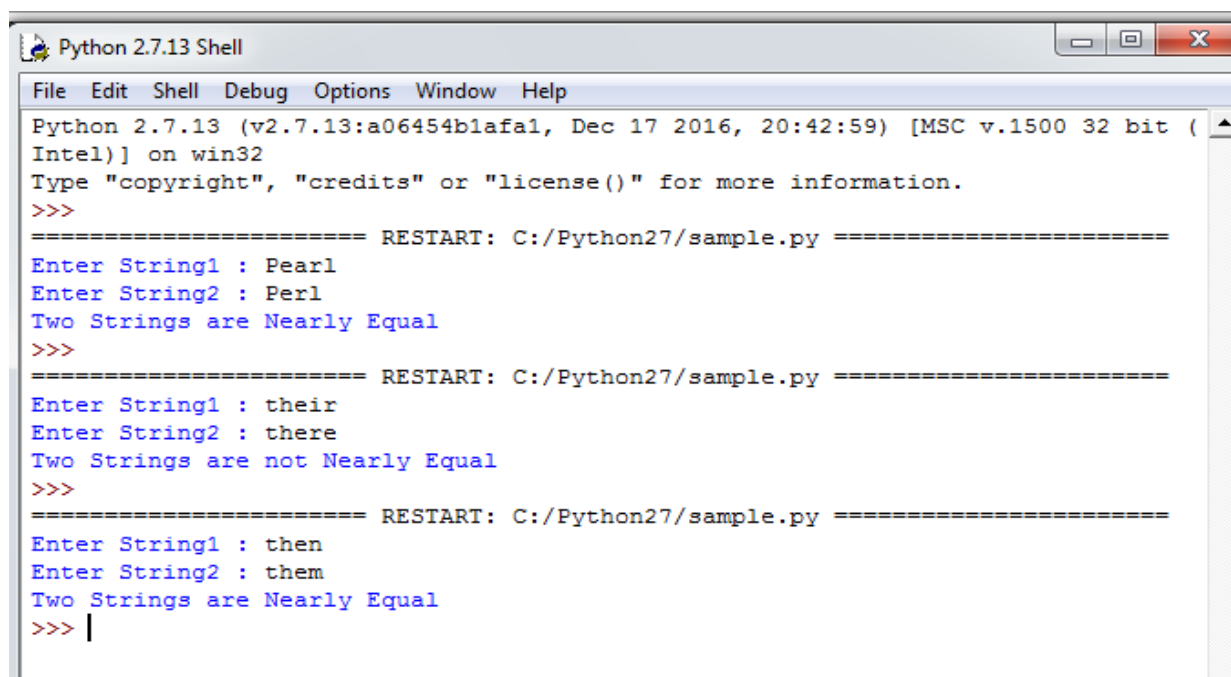
Step19: Check whether the value of count is 1 or not. If yes, goto Step20. Otherwise, goto Step21

Step20: return True and goto Step22

Step21: return False

Step22: Stop

Output:



```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/sample.py =====
Enter String1 : Pearl
Enter String2 : Perl
Two Strings are Nearly Equal
>>>
===== RESTART: C:/Python27/sample.py =====
Enter String1 : their
Enter String2 : there
Two Strings are not Nearly Equal
>>>
===== RESTART: C:/Python27/sample.py =====
Enter String1 : then
Enter String2 : them
Two Strings are Nearly Equal
>>> |
```


EXERCISE - 9(b)

Aim:

Write a function dups to find all duplicates in the list.

Description:

We need to write a function dups to find all duplicate elements in the list. If an element is repeated more than once in the list, then add that repeated element to the resultant list.

Algorithm:

Function dups(elements)

Input: A List

Output: A List containing only duplicate elements

Step1: Start

Step2: Create an empty dictionary dict_dups and an empty list list_dups

Step3: Repeat Steps 4 to 6 until end of the list elements is reached

Step4: Check whether i^{th} element in the list elements is present in the dictionary dict_dups or not. If yes, goto Step5. Otherwise, goto Step6

Step5: Increment the value of key element (i^{th} element of the list elements) in the dictionary dict_dups by 1 and continue with next element of the list elements

Step6: Assign the value of key element (i^{th} element of the list elements) in the dictionary dict_dups to 1

Step7: Repeat Steps 8 and 9 until end of the dictionary dict_dups is reached

Step8: Check whether the value of i^{th} key element in the dictionary dict_dups is more than 1 or not. If yes, goto Step9. Otherwise, Continue with the next element of the dictionary dict_dups

Step9: Add i^{th} key element of the dictionary dict_dups to the list list_dups

Step10: return the list list_dups

Step11: Stop

Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/9b.py =====
Enter the size of the list : 10
Enter an element : 10
Enter an element : 30
Enter an element : 50
Enter an element : 20
Enter an element : 30
Enter an element : 50
Enter an element : 50
Enter an element : 60
Enter an element : 10
Enter an element : 90
The duplicate elements in the list are :
10
50
30
>>> |
```



EXERCISE - 9(c)

Aim:

Write a function unique to find all the unique elements of a list.

Description:

We need to write a function unique to find all unique elements in the list. If an element is found only once in the list, then add that element to the resultant list.

Algorithm:

Function unique(elements)

Input: A List

Output: A List containing only unique elements

Step1: Start

Step2: Create an empty dictionary dict_unique and an empty list list_unique

Step3: Repeat Steps 4 to 6 until end of the list elements is reached

Step4: Check whether i^{th} element in the list elements is present in the dictionary dict_unique or not.

If yes, goto Step5. Otherwise, goto Step6

Step5: Increment the value of key element (i^{th} element of the list elements) in the dictionary dict_unique by 1 and continue with next element of the list elements

Step6: Assign the value of key element (i^{th} element of the list elements) in the dictionary dict_unique to 1

Step7: Repeat Steps 8 and 9 until end of the dictionary dict_unique is reached

Step8: Check whether the value of i^{th} key element in the dictionary dict_unique is equal to 1 or not.

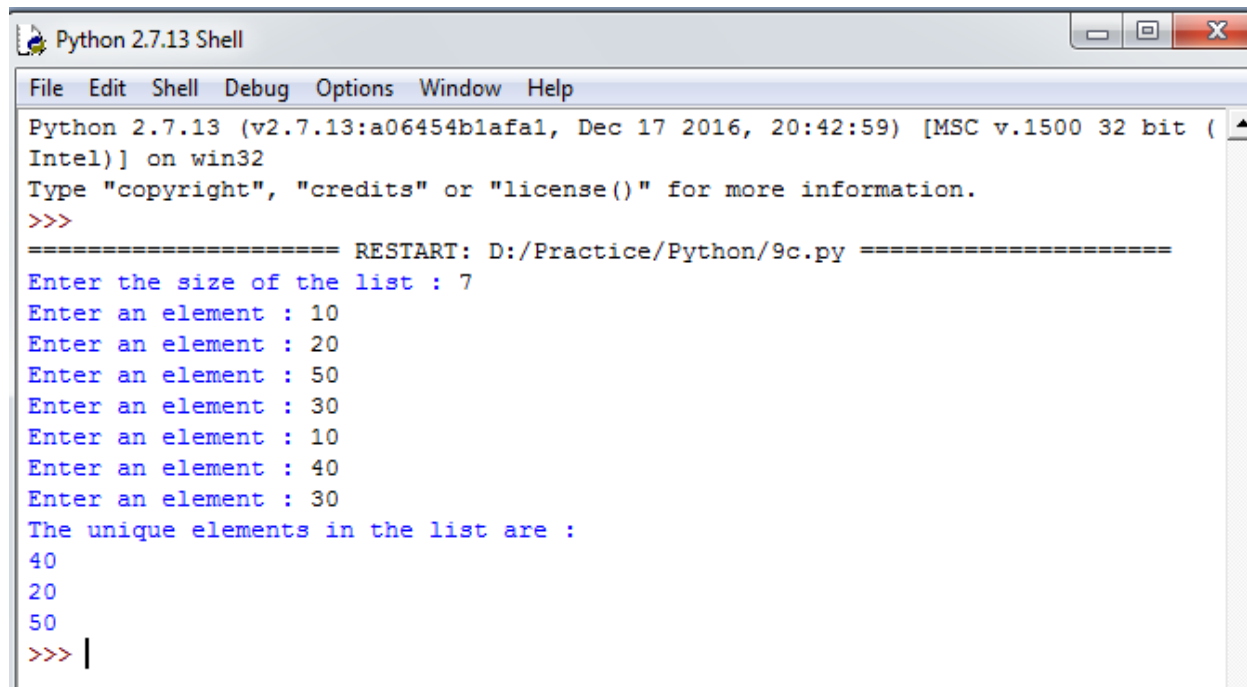
If yes, goto Step9. Otherwise, Continue with the next element of the dictionary dict_unique

Step9: Add i^{th} key element of the dictionary dict_unique to the list list_unique

Step10: return the list list_unique

Step11: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/9c.py =====
Enter the size of the list : 7
Enter an element : 10
Enter an element : 20
Enter an element : 50
Enter an element : 30
Enter an element : 10
Enter an element : 40
Enter an element : 30
The unique elements in the list are :
40
20
50
>>> |
```

Conclusion:

Student gets the knowledge on functions, lists and dictionaries. This experiment maps with CO2,CO4 and the students can attain PO1,PO2,PO3,PO4, PO5,PO6,PO9,PO10, PO11, PO12 and PSO1,PSO2,PSO3.

Viva Questions:

1. What is recursive function?
2. What is scope?
3. What is default argument in python?
4. What is the output of the below program?

```
x = 50
```

```
def func(x):
```

```
    x = 2
```

```
    func(x)
```

```
    print 'x is now', x
```

5. What is the output of the below program?

```
def power(x, y=2):  
    r = 1  
    for i in range(y):  
        r = r * x  
    return r  
print power(3)  
print power(3, 3)
```

6. What is the output of the below program?

```
def a(b):  
    b = b + [5]
```

```
c = [1, 2, 3, 4]  
a(c)  
print(len(c))
```

7. What is the output of the below program?

```
a=10  
b=20  
def change():  
    global b  
    a=45  
    b=56
```

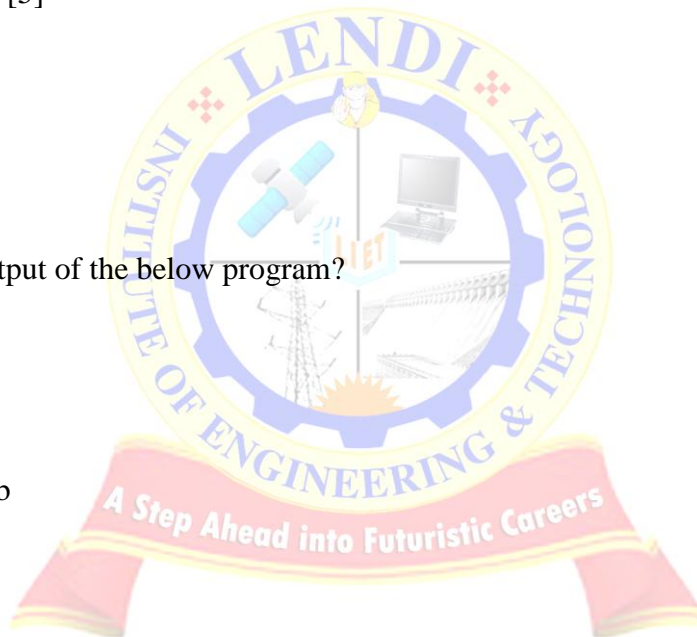
```
change()  
print(a)  
print(b)
```

8. What is the output of the below program?

```
def change(one, *two):  
    print(type(two))  
change(1,2,3,4)
```

9. If a function doesn't have a return statement, which of the following does the function return?

10. What is the type of each element in sys.argv?



EXERCISE - 10(a)

Aim:

Write a function cumulative_product to compute cumulative product of a list of numbers.

Description:

We need to write a function cumulative_product to find cumulative product of numbers in the list.

A Cumulative product is a sequence of partial products of a given sequence. For example, The cumulative product of sequence [a,b,c,.....] are a,ab,abc,.....

Algorithm:

Function cumulative_product(numbers)

Input: A List

Output: Cumulative products of a numbers in a list

Step1: Start

Step2: Create an empty list cum_prod_list and Initialize prod to 1

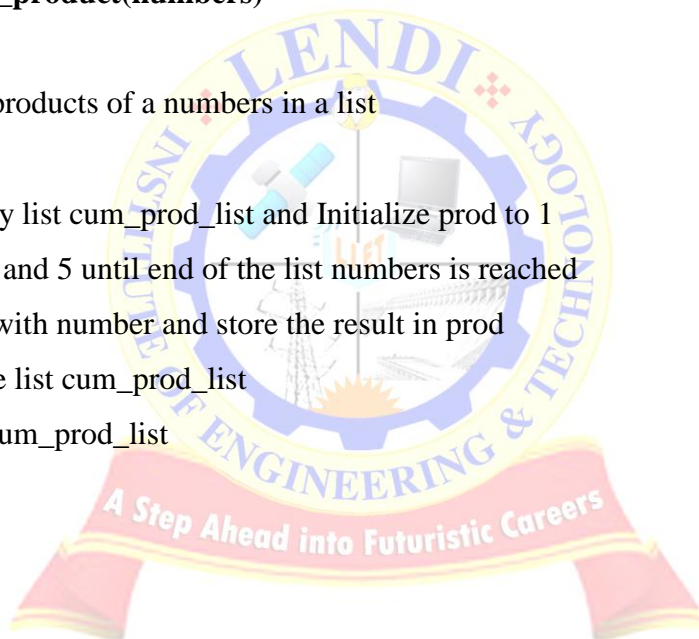
Step3: Repeat Steps 4 and 5 until end of the list numbers is reached

Step4: Multiply prod with number and store the result in prod

Step5: Add prod to the list cum_prod_list

Step6: return the list cum_prod_list

Step7: Stop



Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/9c.py =====
Enter the size of the list : 5
Enter an element : 2
Enter an element : 4
Enter an element : 6
Enter an element : 8
Enter an element : 10
The cumulative product of list of elements are :
2
8
48
384
3840
>>>
```



EXERCISE - 10(b)

Aim:

Write a function reverse to reverse a list. Without using the reverse function.

Description:

We need to write a function reverse to reverse the given elements in a list. Reversing of a list is done by using the feature called **slicing**.

Python's list objects have an interesting feature called slicing. We can view it as an extension of the square brackets indexing syntax. It includes a special case where slicing a list with "[::-1]" produces a reversed copy.

Example:

```
My_list = [10,20,30,40]
```

```
Rev_list = My_list[::-1]
```

Then, Rev_list contains the list [40,30,20,10]

Algorithm:

Function reverse(elements)

Input: A List

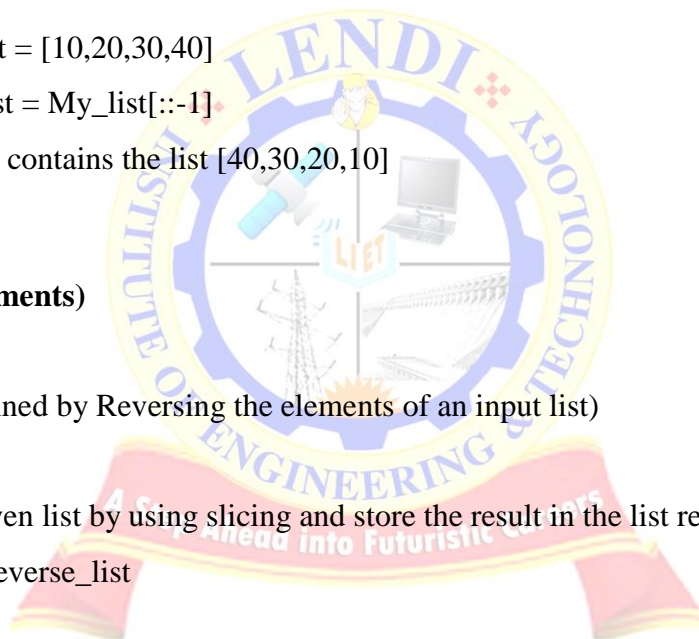
Output: A List (Obtained by Reversing the elements of an input list)

Step1: Start

Step2: Reverse the given list by using slicing and store the result in the list reverse_list

Step3: return the list reverse_list

Step4: Stop



Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/10b.py =====
Enter the size of the list : 5
Enter an element : 20
Enter an element : 30
Enter an element : 40
Enter an element : 10
Enter an element : 50
The Original list is :
[20, 30, 40, 10, 50]
After reversing, all the elements the list becomes :
[50, 10, 40, 30, 20]
>>>
```



EXERCISE - 10(c)

Aim:

Write function to compute gcd, lcm of two numbers. Each function shouldn't exceed one line.

Description:

We need to write a functions to compute gcd, lcm of numbers. But here the constraint is each function should not exceed one line. For this, We are using **lambda function**.

lambda function:

We can use lambda keyword to create small anonymous functions. lambda functions can have any number of arguments but contains only one expression. The expression is evaluated and returned. They cannot contains commands or multiple expressions.

The general form of lambda function is as follows:

lambda arguments : expression

Example:

square = lambda x : x*2

In the above statement, lambda x : x*2 is the lambda function. Here, x is an argument and x*2 is the expression that gets evaluated and returned.

Algorithm:

Function gcd(a,b)

Input: Two Numbers

Output: GCD of two numbers

Step1: Start

Step2: Check whether the value of b is 0 or not. If yes, goto Step3. Otherwise, goto Step4

Step3: return the value of a and goto Step5

Step4: Call gcd(a,b) with a = b and b = a % b

Step5: Stop

Function lcm(a,b)

Input: Two Numbers

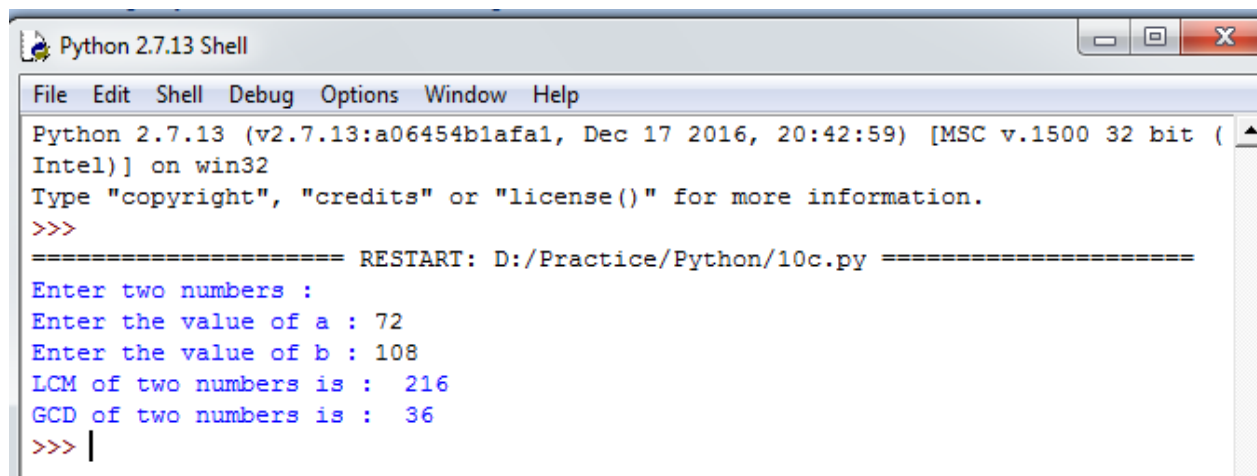
Output: LCM of two numbers

Step1: Start

Step2: return (a * b)//gcd(a,b)

Step3: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/Practice/Python/10c.py =====
Enter two numbers :
Enter the value of a : 72
Enter the value of b : 108
LCM of two numbers is : 216
GCD of two numbers is : 36
>>> |
```

Conclusion:

Student gets the knowledge on functions and lists. This experiment maps with CO2,CO4 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10,PO11,PO12 and PSO1,PSO2, PSO3.

Viva Questions:

1. What are lambda functions in Python?
2. What are fruitful functions in Python?
3. What is the data structure used to perform recursion?
4. What is tail recursion?
5. What actions are performed when a function is called?
6. What is the output of the code shown below?

```
x = 5
```

```
def f1():
```

```
    global x
```

```
    x = 4
```

```
def f2(a,b):
```

```
    global x
```

```
    return a+b+x
```

```
f1()
total = f2(1,2)
print(total)
```

7. What is the output of the code shown below?

```
x=100
def f1():
    global x
    x=90
def f2():
    global x
    x=80
print(x)
```

8. Read the code shown below carefully and point out the global variables:

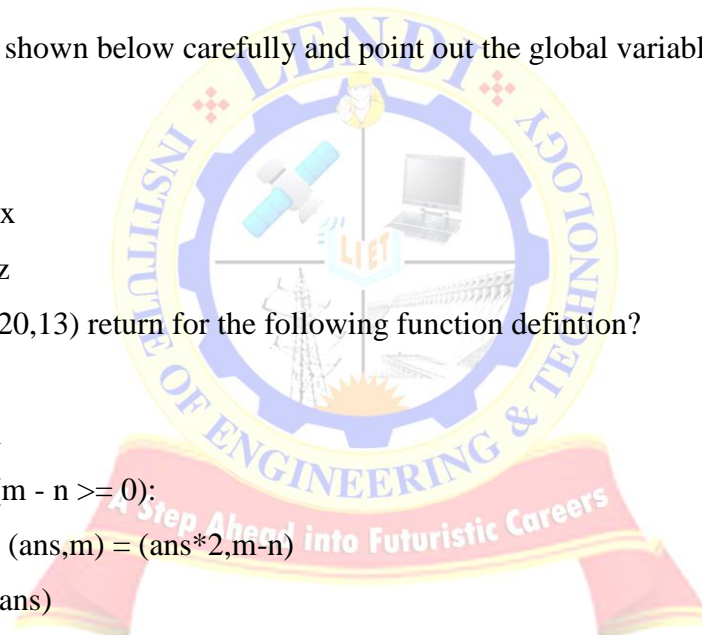
```
y, z = 1, 2
def f():
    global x
    x = y+z
```

9. What does f(120,13) return for the following function defintion?

```
def f(m,n):
    ans = 1
    while (m - n >= 0):
        (ans,m) = (ans*2,m-n)
    return(ans)
```

10. What does g(9000,3) return for the following function definition?

```
def g(x,y):
    val = 0
    while (x > y):
        (val,x) = (val+1,x/y)
    return(val)
```



EXERCISE - 11(a)

Aim:

Write a program that defines a matrix and prints.

Description:

A matrix is a two-dimensional data structure. In python, matrix is a nested list. A list is created by placing all the items (elements) inside a square bracket [], separated by commas.

Example: A = [[1, 2, 3], [3, 4, 5]]

In the above example A represents a 2 * 3 matrix.

Accessing the elements of a matrix:

Similar to list we can access elements of a matrix by using square brackets [] after the variable like a[row-number][col_number].

Example: A = [[1, 2, 3], [3, 4, 5]]

In the above Matrix,

A[0] contains [1, 2, 3]

A[0][0] contains 1.

Algorithm:

Input: A Matrix

Output: A Matrix

Step1: Start

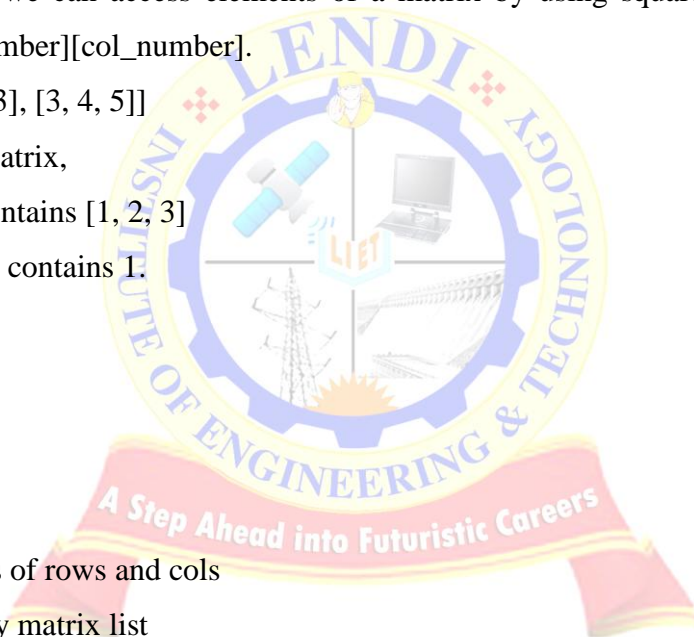
Step2: Read the values of rows and cols

Step3: Create an empty matrix list

Step4: Read the matrix list of size rows*cols

Step5: Display the matrix list

Step6: Stop



Output:

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\11a.py =====
Enter the number of Rows : 2
Enter the number of Columns : 3
Enter an element [0][0] : 1
Enter an element [0][1] : 2
Enter an element [0][2] : 3
Enter an element [1][0] : 4
Enter an element [1][1] : 5
Enter an element [1][2] : 6
Matrix is :
1 2 3
4 5 6
>>> |
```



EXERCISE - 11(b)

Aim:

Write a program to perform addition of two square matrices.

Description:

In order to perform addition of two matrices, the two matrices must have same number of rows and same number of columns.

We should use nested for loops to iterate through each row and each column. At each point, We add the corresponding elements in the two matrices and store it in the result.

Example :

Consider, Two 2*2 Matrices,

A = [[2, 2], [2, 2]]

B = [[1, 1], [1, 1]]

If we perform Addition, then we will get the resultant matrix: [[3, 3], [3, 3]].

Algorithm:

Input: 2 Square Matrices

Output: Addition of 2 Square Matrices or a Message(If Addition is not possible)

Step1: Start

Step2: Create empty Matrices a, b and c

Step3: Read the value of size_a(Size of the first matrix)

Step4: Read the matrix a of size size_a* size_a

Step5: Read the value of size_b(Size of the second matrix)

Step6: Read the matrix b of size size_b* size_b

Step7: Display the matrices a and b

Step8: Check whether sizes of two matrices(size_a and size_b) are equal or not. If yes, goto Step9.

Otherwise, goto Step11

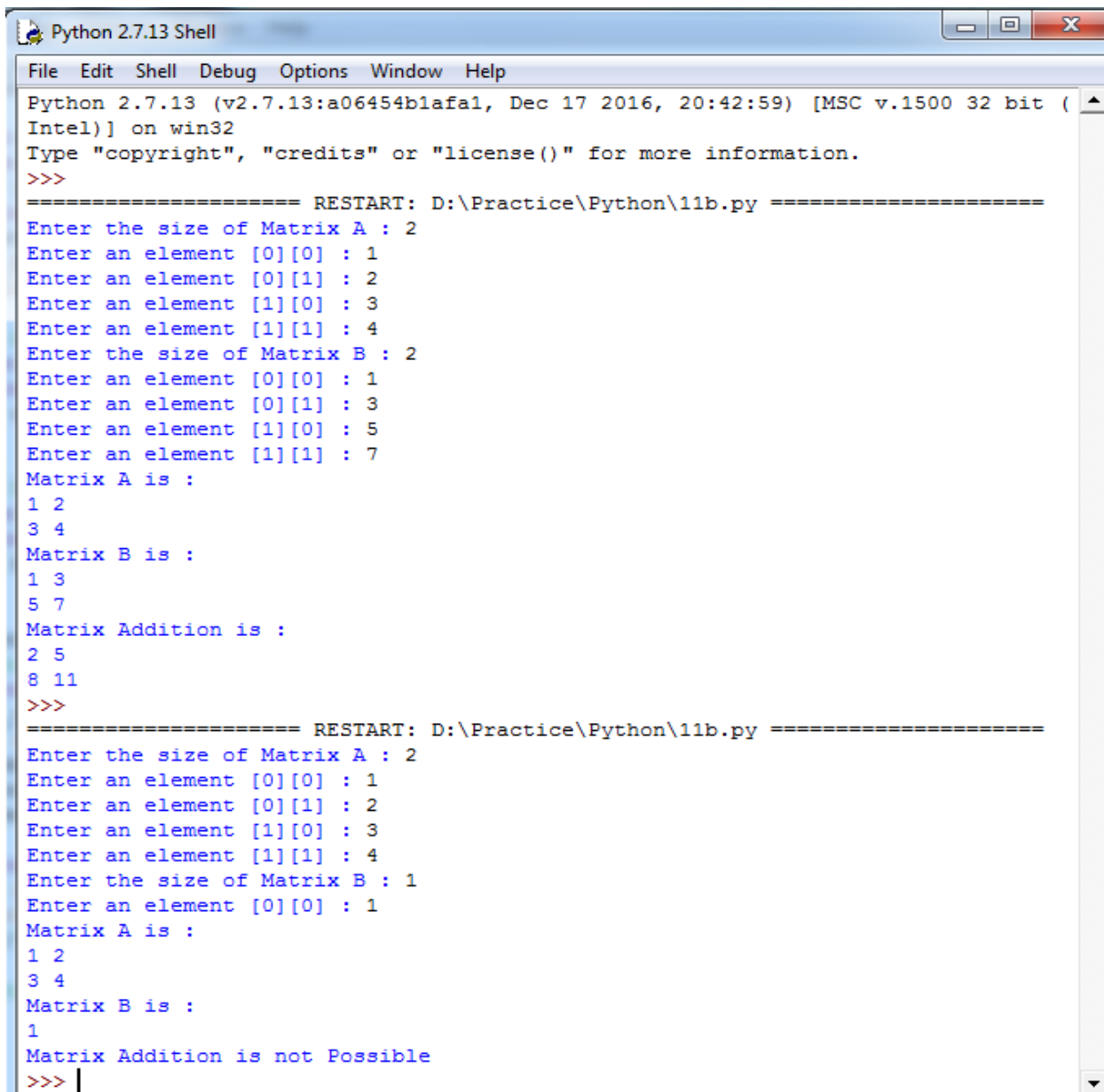
Step9: Add two matrices a and b and store the result in matrix c

Step10: Display the matrix c and goto Step12

Step11: Display the message “Matrix Addition is not Possible”

Step12: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\11b.py =====
Enter the size of Matrix A : 2
Enter an element [0][0] : 1
Enter an element [0][1] : 2
Enter an element [1][0] : 3
Enter an element [1][1] : 4
Enter the size of Matrix B : 2
Enter an element [0][0] : 1
Enter an element [0][1] : 3
Enter an element [1][0] : 5
Enter an element [1][1] : 7
Matrix A is :
1 2
3 4
Matrix B is :
1 3
5 7
Matrix Addition is :
2 5
8 11
>>>
===== RESTART: D:\Practice\Python\11b.py =====
Enter the size of Matrix A : 2
Enter an element [0][0] : 1
Enter an element [0][1] : 2
Enter an element [1][0] : 3
Enter an element [1][1] : 4
Enter the size of Matrix B : 1
Enter an element [0][0] : 1
Matrix A is :
1 2
3 4
Matrix B is :
1
Matrix Addition is not Possible
>>> |
```

EXERCISE - 11(c)

Aim:

Write a program to perform multiplication of two square matrices.

Description:

In order to perform multiplication of two matrices, then the number of columns in first matrix should be equal to the number of rows in second matrix.

When we multiply two matrices by each other, the resulting matrix will have as many columns as the biggest matrix in the equation.

For example, If we are multiplying a 3*3 by a 3*4 matrix, the resulting matrix will be a 3*4.

Example:

Consider two matrices,

$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix}$ Which is a 2 * 3 matrix.

$B = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$ Which is a 3 * 2 matrix.

If we perform multiplication of above matrices, then we will get $\begin{bmatrix} 14 & 6 \\ 6 & 3 \end{bmatrix}$, Which is a 2*2 matrix.

Algorithm:

Input: 2 Square Matrices

Output: Multiplication of 2 Square Matrices or a Message(If Mutiplication is not possible)

Step1: Start

Step2: Create empty Matrices a, b and c

Step3: Read the value of size_a(Size of the first matrix)

Step4: Read the matrix a of size size_a* size_a

Step5: Read the value of size_b(Size of the second matrix)

Step6: Read the matrix b of size size_b* size_b

Step7: Display the matrices a and b

Step8: Check whether sizes of two matrices(size_a and size_b) are equal or not. If yes, goto Step9.

Otherwise, goto Step11

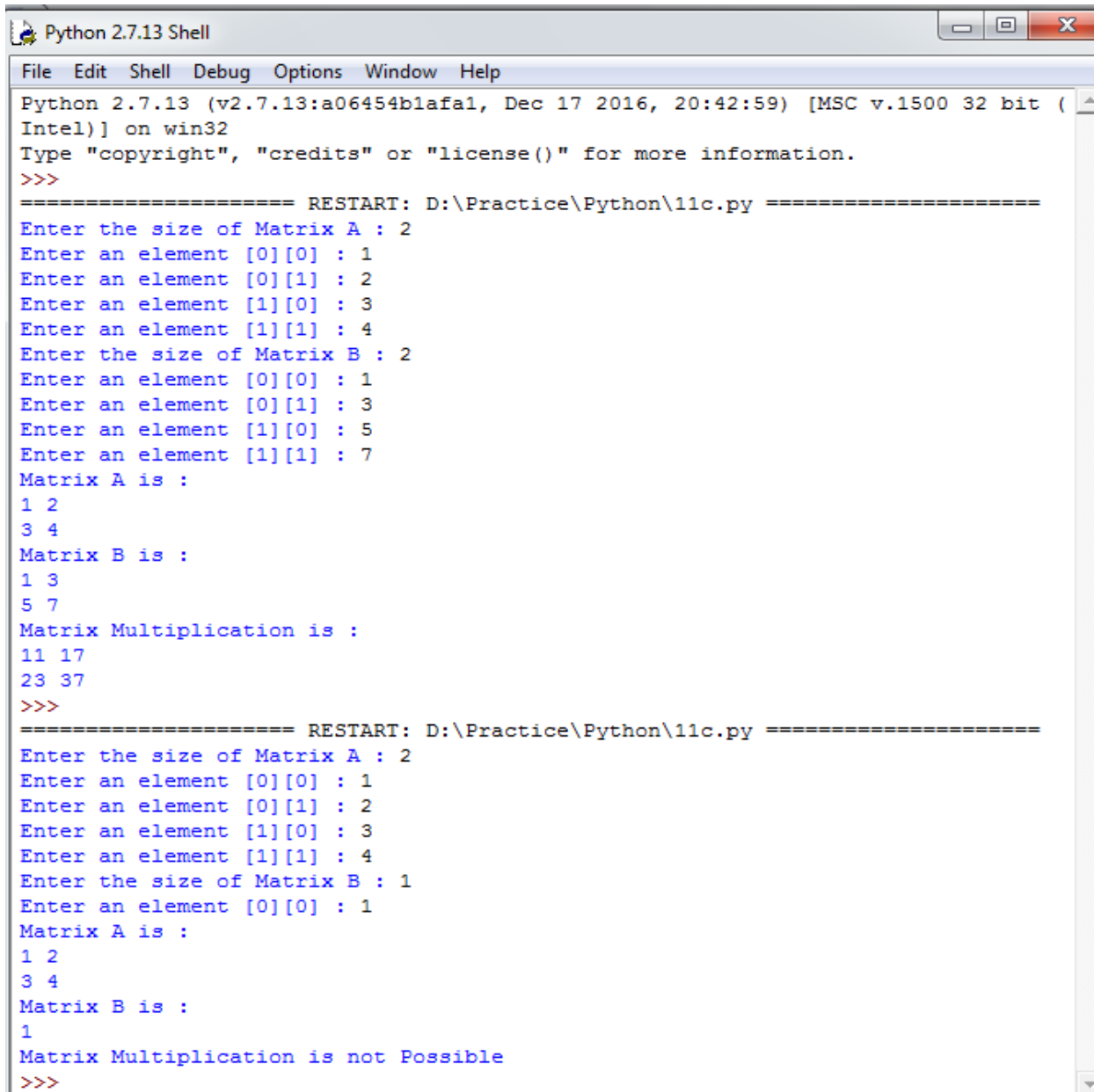
Step9: Multiply two matrices a and b and store the result in matrix c

Step10: Display the matrix c and goto Step12

Step11: Display the message “Matrix Multiplication is not Possible”

Step12: Stop

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\11c.py =====
Enter the size of Matrix A : 2
Enter an element [0][0] : 1
Enter an element [0][1] : 2
Enter an element [1][0] : 3
Enter an element [1][1] : 4
Enter the size of Matrix B : 2
Enter an element [0][0] : 1
Enter an element [0][1] : 3
Enter an element [1][0] : 5
Enter an element [1][1] : 7
Matrix A is :
1 2
3 4
Matrix B is :
1 3
5 7
Matrix Multiplication is :
11 17
23 37
>>>
===== RESTART: D:\Practice\Python\11c.py =====
Enter the size of Matrix A : 2
Enter an element [0][0] : 1
Enter an element [0][1] : 2
Enter an element [1][0] : 3
Enter an element [1][1] : 4
Enter the size of Matrix B : 1
Enter an element [0][0] : 1
Matrix A is :
1 2
3 4
Matrix B is :
1
Matrix Multiplication is not Possible
>>>
```

Conclusion:

Student gets the knowledge on Multi-Dimensional lists. This experiment maps with CO2 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10,PO11,PO12 and PSO1,PSO2, PSO3.

Viva Questions:

1. What is nested list?
2. How to initialize two dimensional list in python?
3. What is cloning list?
4. What is the difference between del() and remove() methods of list?
5. Differentiate between append () and extend () methods?
6. What is enumerate() and range() in python?
7. To insert 5 to the third position in list1, which command can we use?
8. What is the output of the code shown below?

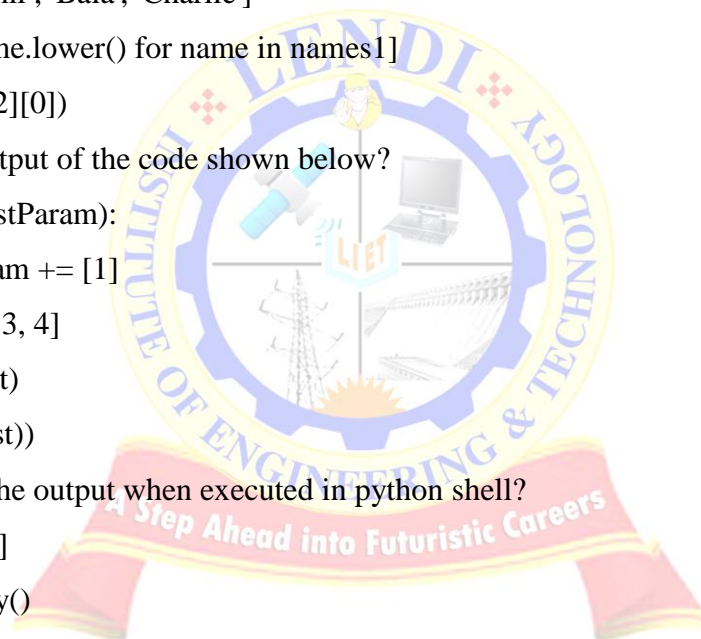
```
names1 = ['Amir', 'Bala', 'Charlie']  
names2 = [name.lower() for name in names1]  
print(names2[2][0])
```

9. What is the output of the code shown below?

```
def addItem(listParam):  
    listParam += [1]  
mylist = [1, 2, 3, 4]  
addItem(mylist)  
print(len(mylist))
```

10. What will be the output when executed in python shell?

```
>>> a = [3,7,1]  
>>> b = a.copy()  
>>> b is a
```



EXERCISE – 12(a)

Aim:

Write a procedure to Install packages requests, flask and explore them using pip.

Description:

pip is a package management system used to install and manage software packages written in Python. Many packages can be found in the Python Package Index(PyPI). Python 2.7.9 and later (on the python2 series), and Python 3.4 and later include pip (pip3 for Python 3) by default.

Pip is a recursive acronym that can stand for either "Pip Installs Packages" or "Pip Installs Python". One major advantage of pip is the ease of its command-line interface, which makes installing Python software packages as easy as issuing one command:

pip install package-name

Example:

pip install requests

pip install flask

Users can also easily remove the package.

Example:

pip uninstall package-name

requests is an HTTP library, written in Python, for human beings.

Example for Basic GET usage:

```
>>> import requests
>>> r = requests.get('https://www.python.org')
>>> r.status_code
200
>>> 'Python is a programming language' in r.content
True
```

EXERCISE – 12(b)

Aim:

Write a script that imports requests and fetch content from the page. Eg. (Wiki).

Description:

The different steps involved in making request to the server is summarized below:

1. Make Request:

Request is made by importing the requests module.

Example:

```
import requests
```

2. Get the webpage using get():

```
r = requests.get('url')(r stands for Response object)
```

HTTP POST request can be made using post()

```
r = requests.post('url', data = {'key':'value'})
```

3. Response Content:

Content of the server's response can be read using text property.

Example:

```
import requests
```

```
r = requests.get('http://www.google.com')
```

```
r.text
```

4. Response status code:

It can be retrieved using r.status_code. Where, r is response object.

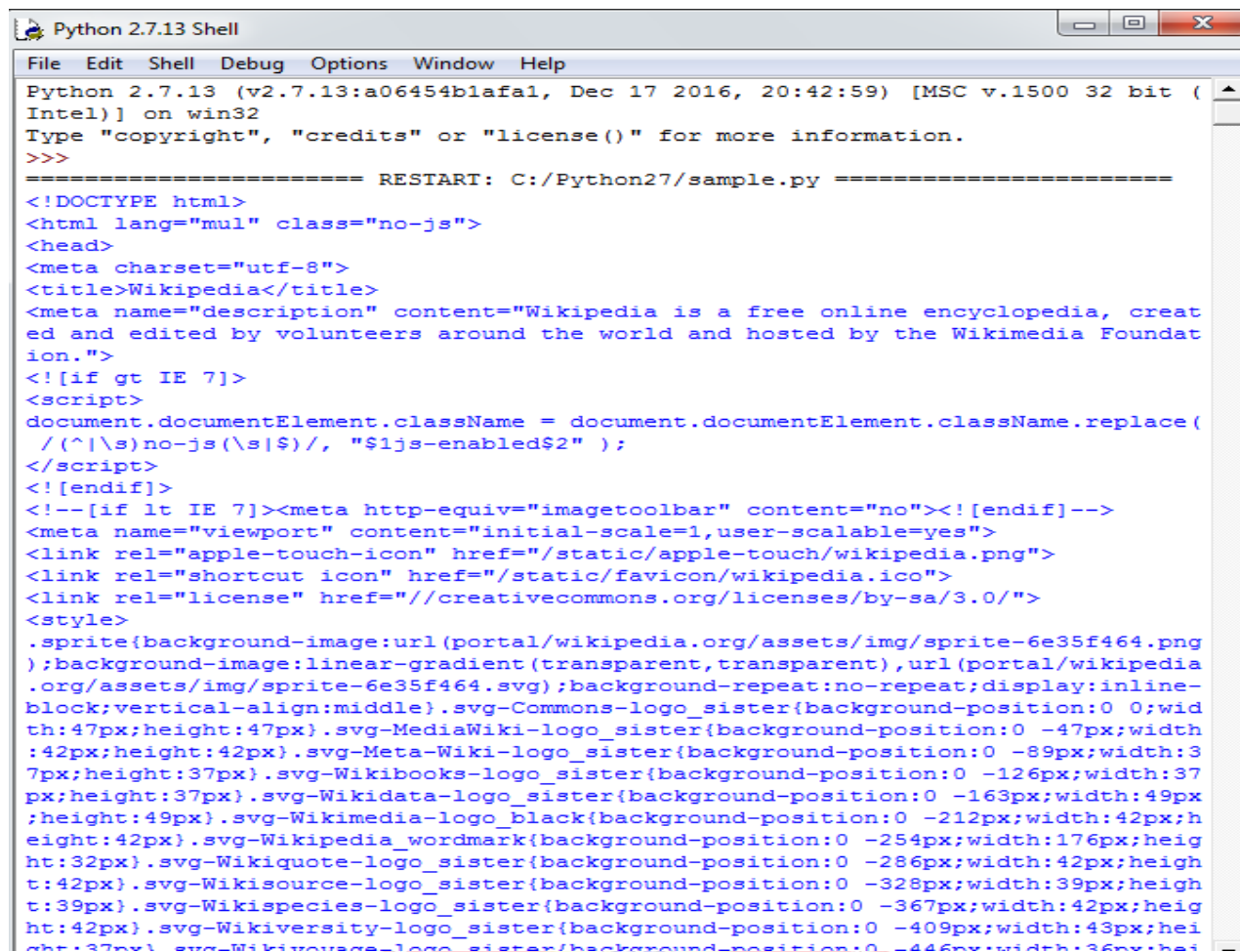
5. Response headers:

Servers response headers can be viewed using r.headers.

6. Cookies:

These can be obtained using r.cookies['cookie name']

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/sample.py =====
<!DOCTYPE html>
<html lang="mul" class="no-js">
<head>
<meta charset="utf-8">
<title>Wikipedia</title>
<meta name="description" content="Wikipedia is a free online encyclopedia, created and edited by volunteers around the world and hosted by the Wikimedia Foundation.">
<![if gt IE 7]>
<script>
document.documentElement.className = document.documentElement.className.replace(
/^(\\s)no-js(\\s|$)/, "$1js-enabled$2" );
</script>
<![endif]>
<!--[if lt IE 7]><meta http-equiv="imagetoolbar" content="no"><![endif]>-->
<meta name="viewport" content="initial-scale=1,user-scalable=yes">
<link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png">
<link rel="shortcut icon" href="/static/favicon/wikipedia.ico">
<link rel="license" href="//creativecommons.org/licenses/by-sa/3.0/">
<style>
.sprite{background-image:url(portal/wikipedia.org/assets/img/sprite-6e35f464.png);background-image:linear-gradient(transparent,transparent),url(portal/wikipedia.org/assets/img/sprite-6e35f464.svg);background-repeat:no-repeat;display:inline-block;vertical-align:middle}.svg-Commons-logo_sister{background-position:0 0;width:47px;height:47px}.svg-MediaWiki-logo_sister{background-position:0 -47px;width:42px;height:42px}.svg-Meta-Wiki-logo_sister{background-position:0 -89px;width:37px;height:37px}.svg-Wikibooks-logo_sister{background-position:0 -126px;width:37px;height:37px}.svg-Wikidata-logo_sister{background-position:0 -163px;width:49px;height:49px}.svg-Wikimedia-logo_black{background-position:0 -212px;width:42px;height:42px}.svg-Wikipedia_wordmark{background-position:0 -254px;width:176px;height:32px}.svg-Wikiquote-logo_sister{background-position:0 -286px;width:42px;height:42px}.svg-Wikisource-logo_sister{background-position:0 -328px;width:39px;height:39px}.svg-Wikispecies-logo_sister{background-position:0 -367px;width:42px;height:42px}.svg-Wikiiversity-logo_sister{background-position:0 -409px;width:43px;height:37px}.svg-Wikiquote-logo_sister{background-position:0 -446px;width:36px;hei
```



EXERCISE – 12(c)

Aim:

Write a simple script that serves a simple HTTPResponse and a simple HTML Page.

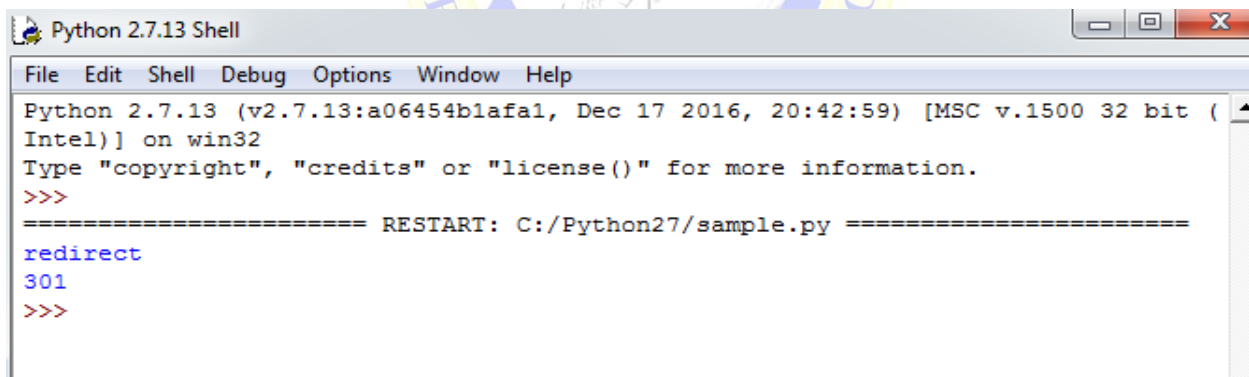
Description:

It defines classes which implement the client side of the HTTP and HTTPS protocols.

The different steps involving in establishing connection to the server is:

- Get HTTP Connection instance using `httplib.HTTPConnection()`.
- Request a web page using `request()` belonging to Connection instance.
- Once the request is sent to the server, response is retrieved from the server using `getresponse()` returning a `HTTPResponse` instance.
- Read the response using response instance `read()`.
- Read status of response.
- Close the connection to the server.

Output:



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/sample.py =====
redirect
301
>>>
```

Conclusion:

Student gets the knowledge on modules, packages and how to install, import modules. This experiment maps with CO4 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10, PO11, PO12 and PSO1,PSO2,PSO3.

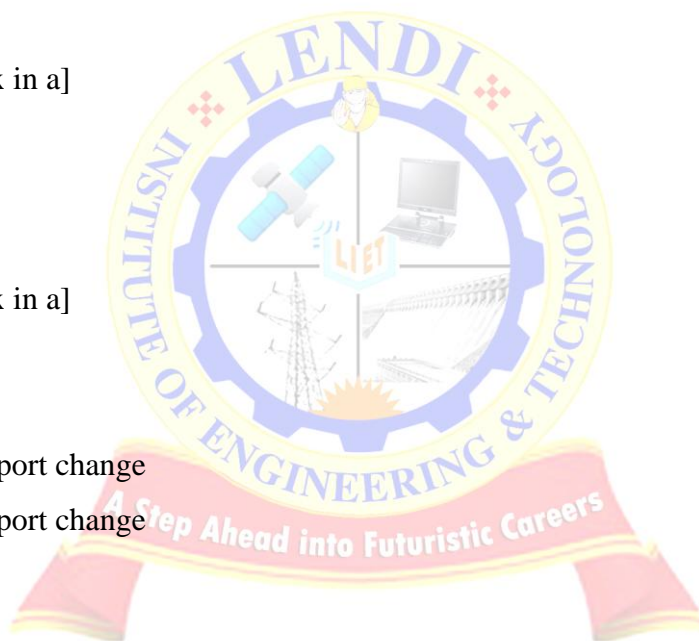
Viva Questions:

1. Define module?
2. What are the advantages of using modules?
3. What is PYTHONPATH Variable?
4. What does dir() returns?
5. What are globals() and locals()?
6. What is the purpose of reload()?
7. What are packages in Python?
8. What is the output of the following piece of code?

```
#mod1
def change(a):
    b=[x*2 for x in a]
    print(b)
#mod2
def change(a):
    b=[x*x for x in a]
    print(b)
```

```
from mod1 import change
from mod2 import change
#main
s=[1,2,3]
change(s)
```

9. What is the output of the following piece of code?
from math import factorial
print(math.factorial(5))
10. What is the return type of globals() and locals()?



EXERCISE - 13

Aim:

Class variables and instance variable and illustration of the self variable for implementing

- (a) Robot
- (b) ATM Machine

Description:

Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

self variable: The first argument to every method inside a class is a special variable **self**. Every time a class refers to one of its variables or methods, it must precede them by itself. The purpose of self is to distinguish class's variables and methods from other variables and functions in the program.

For implementing Class Robot, a class variable called **count**, instance variable called **name** and self variable **self** is used. To distinguish their usage, class variables are accessed using classname with dot(.) operator. Where as, instance variables are accessed using self with dot(.) operator.

For implementing Class ATM Machine, an instance variable called **balance** and self variable **self** is used to call the instance variable within the method of a class.

Output:(13a)

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\13a.py =====
initializing Ram
Hi! this is Ram ,the robot
No of robots present : 1
initializing Krishna
Hi! this is Krishna ,the robot
No of robots present : 2
here the robots are saying hi
Now the robots are going to be destroyed
Ram is being destroyed
No of robots still present : 1
Krishna is being destroyed
Krishna was the last robot
>>>
```

Output:(13b)

```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\13b.py =====
Welcome to ATM Services
Account is created
Your Pin number is : 2745
Enter pin no: 2745
Pin no is verified
1.Deposit
2.Withdraw
3.Display Balance
4.Exit
Enter your choice : 1
Enter amount to be deposited : 8000
Amount 8000 deposited sucessfully
Current Balance : 8000
Press Y to continue
Press any other key to discontinue
Enter your choice : Y
1.Deposit
2.Withdraw
3.Display Balance
4.Exit
Enter your choice : 2
Enter amount to withdraw : 4000
Amount 4000 withdrawn sucessfully
Current Balance : 4000
Press Y to continue
Press any other key to discontinue
Enter your choice : Y
1.Deposit
2.Withdraw
3.Display Balance
4.Exit
Enter your choice : 4
>>>
```

Conclusion:

Student gets the knowledge on Object Oriented Programming concepts and functions. This experiment maps with CO4 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10, PO11, PO12 and PSO1,PSO2,PSO3.

Viva Questions:

1. What is the purpose of setattr() and getattr()?
2. What is instantiation in terms of OOP terminology?
3. What is the output of the following code?

```
class Demo:
```

```
    def __init__(self):
```

```
        pass
```

```
    def test(self):
```

```
        print(__name__)
```

```
obj = Demo()
```

```
obj.test()
```

4. What are special methods in Python?
5. What is the output of the following code?

```
class test:
```

```
    def __init__(self,a="Hello World"):
```

```
        self.a=a
```

```
    def display(self):
```

```
        print(self.a)
```

```
obj = test()
```

```
obj.display()
```

6. What is the output of the following code?

```
class test:
```

```
    def __init__(self,a):
```

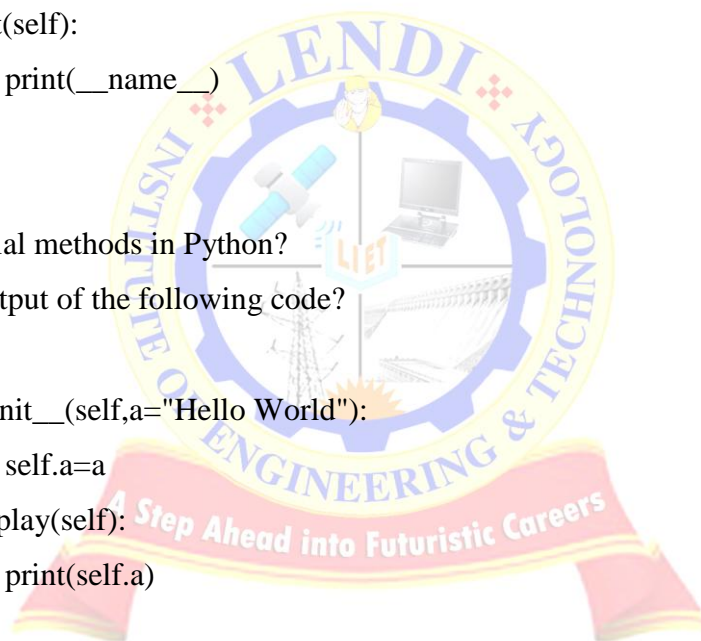
```
        self.a=a
```

```
    def display(self):
```

```
        print(self.a)
```

```
obj = test()
```

```
obj.display()
```



7. What is function overloading?
8. What is operator overloading?
9. What is multiple inheritance? Does python supports multiple inheritance?
10. What is the purpose of issubclass() and isinstance()?



EXERCISE - 14(a)

Aim:

Write a GUI for an Expression Calculator using tk

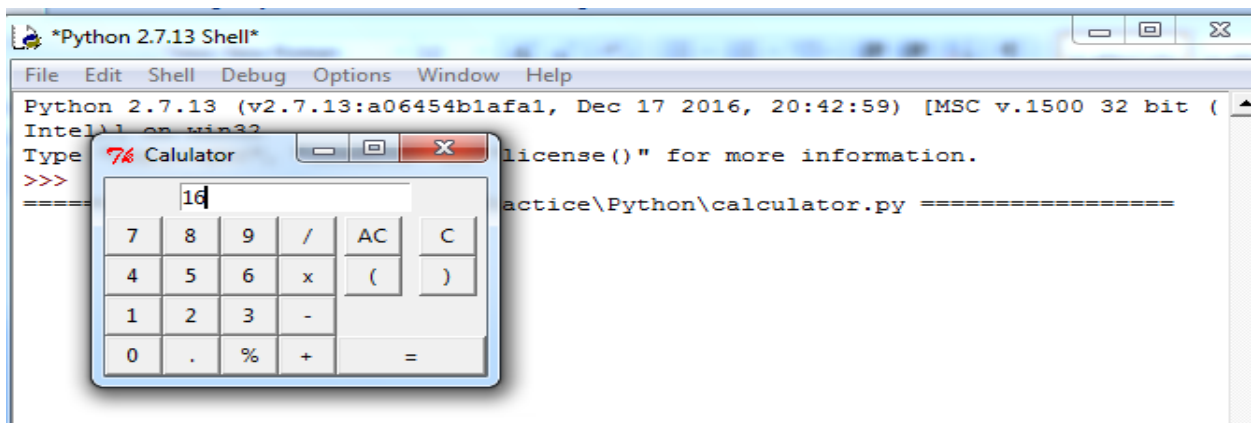
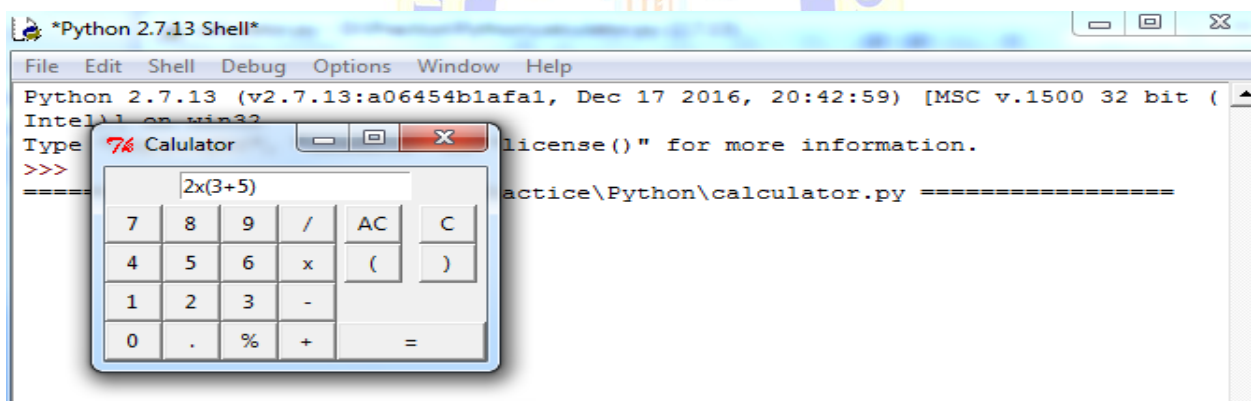
Description:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All we need to do is perform the following steps:

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

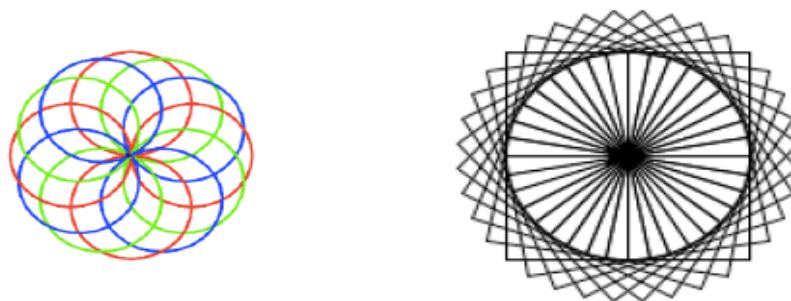
Output:



EXERCISE NO - 14(b)

Aim:

Write a program to implement the following figures using turtle



Description:

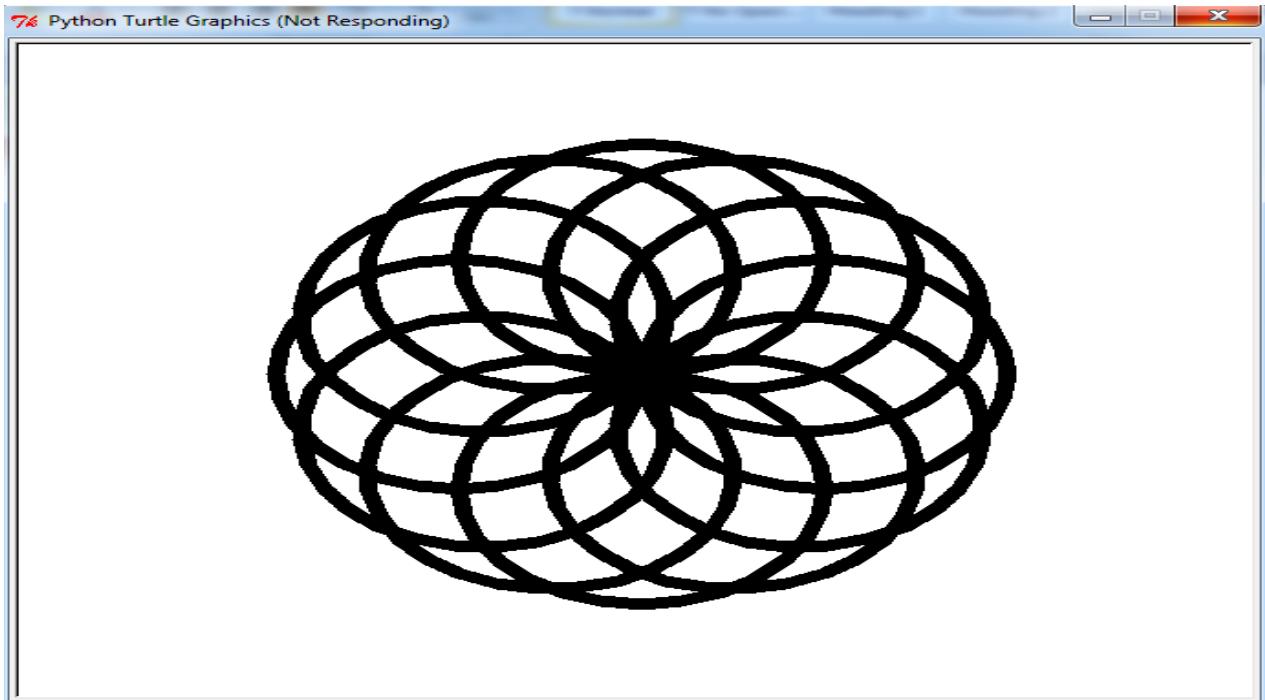
Turtle graphics is a popular way for introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966.

The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. Because it uses Tkinter for the underlying graphics, it needs a version of Python installed with Tk support.

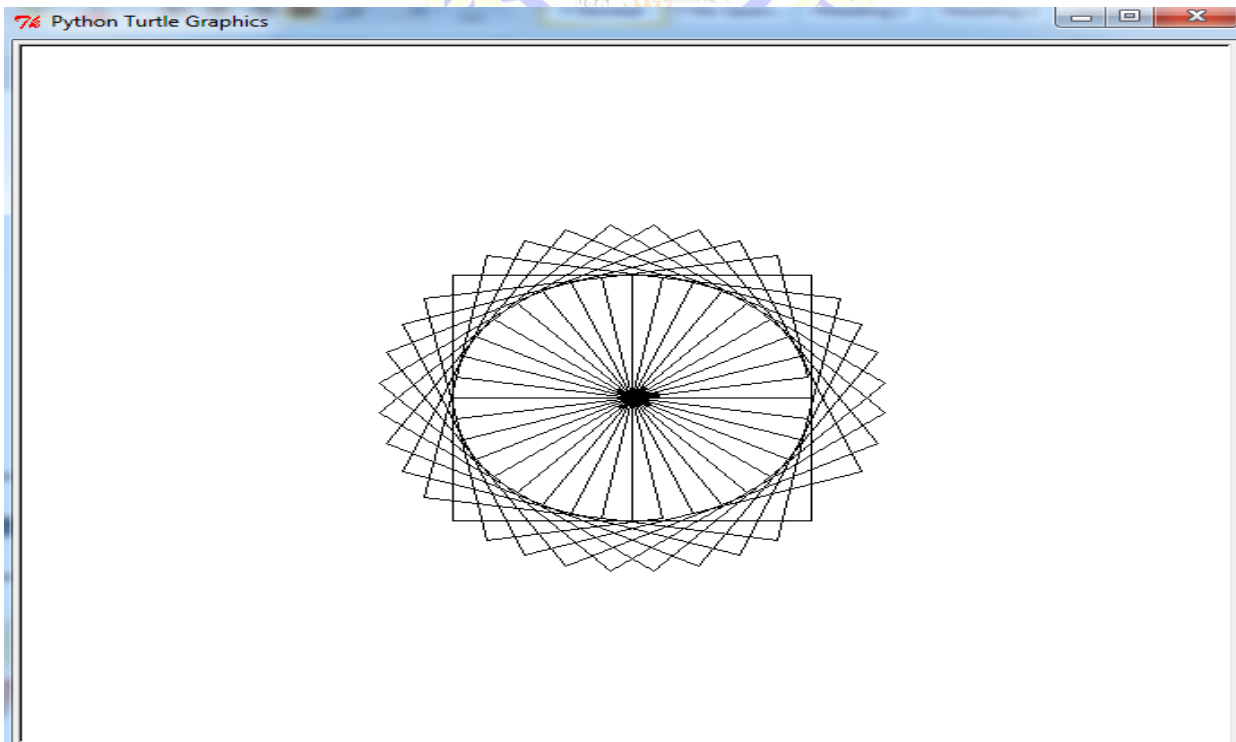
To make use of the turtle methods and functionalities, we need to import turtle. "turtle" comes packed with the standard Python package and need not be installed externally. The roadmap for executing a turtle program follows 4 steps:

- Import the turtle module
- Create a turtle to control.
- Draw around using the turtle methods.
- Run turtle.done().

Output: (14b1)



Output: (14b2)



Conclusion:

Student gets the knowledge on development of basic GUI and turtle module. This experiment maps with CO5 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10, PO11, PO12 and PSO1,PSO2,PSO3.

Viva Questions:

1. What is the purpose of turtle.reset()?
2. What is the output shape of the code shown?

```
import turtle
```

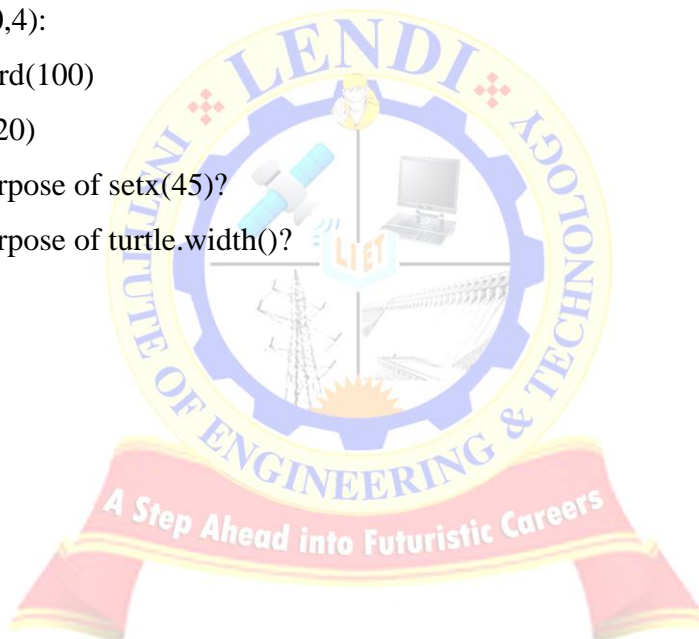
```
t=turtle.Pen()
```

```
for i in range(0,4):
```

```
    t.forward(100)
```

```
    t.left(120)
```

3. What is the purpose of setx(45)?
4. What is the purpose of turtle.width()?



EXERCISE - 15(a)

Aim:

Write a test-case to check the function `even_numbers` which return True on passing a list of all even numbers.

Description:

The Python unit testing framework, sometimes referred to as “PyUnit,” is a Python language version of JUnit, by Kent Beck and Erich Gamma.

unittest supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The unittest module provides classes that make it easy to support these qualities for a set of tests.

The unittest module provides a rich set of tools for constructing and running tests.

test case:

A test case is the smallest unit of testing. It checks for a specific response to a particular set of inputs. unittest provides a base class, `TestCase`, which may be used to create new test cases.

assertTrue:

`assertTrue` is an assertion method that creates a bool value from the received value and then evaluating it.

In this program, We will create a module called `even.py` which consists of a function `even_number`. We will import this module in another file along with another module `unittest`.

Here, we will define a function `test_fun_even` in a class `Test_even_numbers`. In this function, we are passing a list of all even numbers. If all the numbers are even, then all test cases are satisfied and the output will be OK.

Output:

```
C:\Windows\system32\cmd.exe

C:\Users\Pavani>d:
D:\>cd practice
D:\Practice>cd python
D:\Practice\Python>cd 15
D:\Practice\Python\15>python 15a.py
F
=====
FAIL: test_fun_even (__main__.Test_even_numbers)
-----
Traceback (most recent call last):
  File "15a.py", line 7, in test_fun_even
    self.assertTrue(even_number(255))
AssertionError: False is not true
-----
Ran 1 test in 0.000s
FAILED (failures=1)
D:\Practice\Python\15>
```



EXERCISE - 15(b)

Aim:

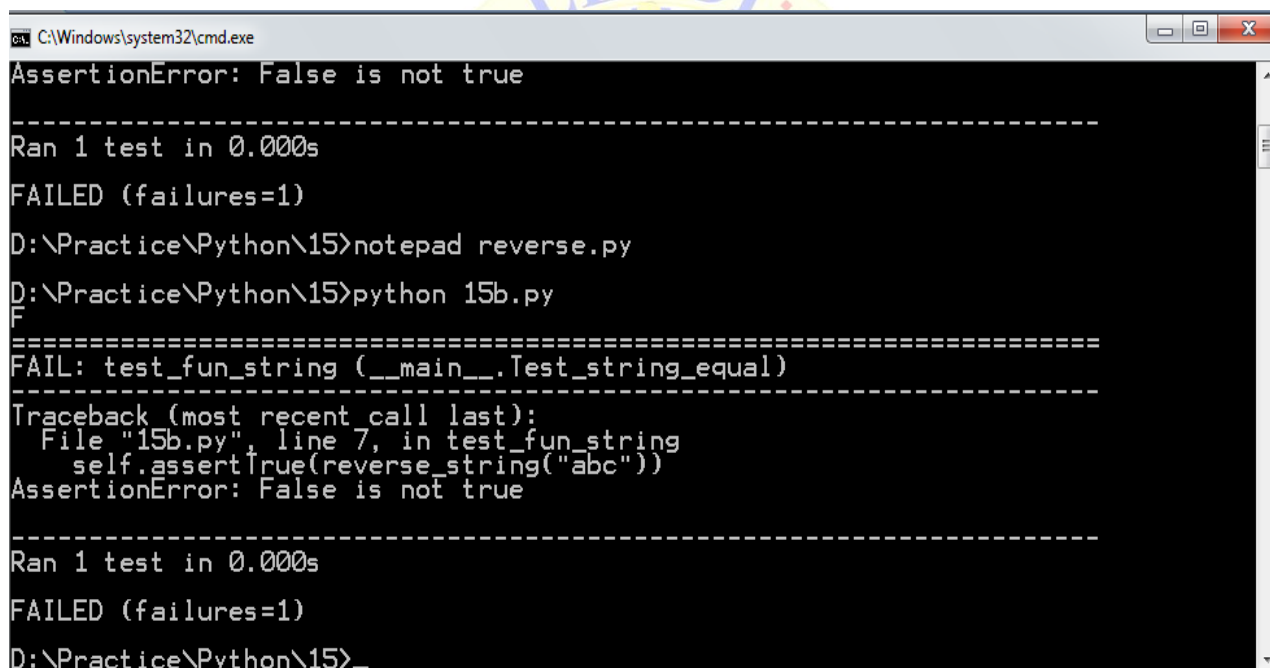
Write a test-case to check the function reverse_string which returns the reversed string.

Description:

In this program, We will create a module called reverse.py which consists of a function reverse_string. We will import this module in another file along with another module unittest.

Here, we will define a function test_fun_string in a class Test_string_equal. In this function, we are passing a list of all strings. If all the strings are palindromes, then all test cases are satisfied then the output will be OK.

Output:



```
C:\Windows\system32\cmd.exe
AssertionError: False is not true
-----
Ran 1 test in 0.000s
FAILED (failures=1)
D:\Practice\Python\15>notepad reverse.py
D:\Practice\Python\15>python 15b.py
F
=====
FAIL: test_fun_string (__main__.Test_string_equal)
=====
Traceback (most recent call last):
  File "15b.py", line 7, in test_fun_string
    self.assertTrue(reverse_string("abc"))
AssertionError: False is not true
-----
Ran 1 test in 0.000s
FAILED (failures=1)
D:\Practice\Python\15>
```

Conclusion:

Student gets the knowledge on writing test cases using unit testing. This experiment maps with CO4,CO6 and students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10,PO11,PO12 and PSO1,PSO2,PSO3.

Viva Questions:

1. Differentiate between Testing and Debugging?
2. What is unittest in Python?
3. What are the conditions for effective debugging?
4. What are different types of Testing?
5. Which module in Python supports regular expressions?
6. What is the difference between re.match() and re.search()?
7. How many except statements can a try-except block have?
8. When will the else part of try-except-else be executed?
9. Is the following code valid?

try:

 # Do something

except:

 # Do something

finally:

 # Do something

10. What is the output of the following code?

```
def foo():
```

```
    try:
```

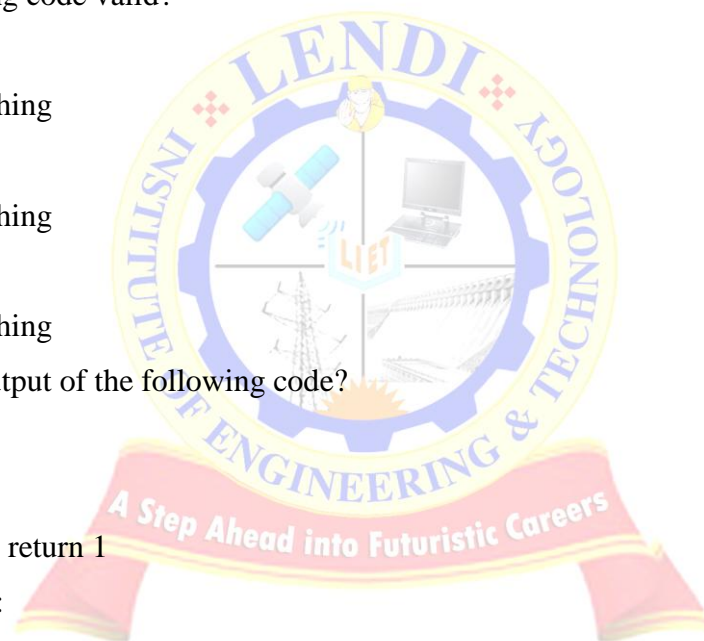
```
        return 1
```

```
    finally:
```

```
        return 2
```

```
k = foo()
```

```
print(k)
```



EXERCISE - 16(a)

Aim:

Build any one classical data structure.

Description:

Here, we will implement a classical data structure known as a stack. Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Mainly the following three basic operations are performed in the stack:

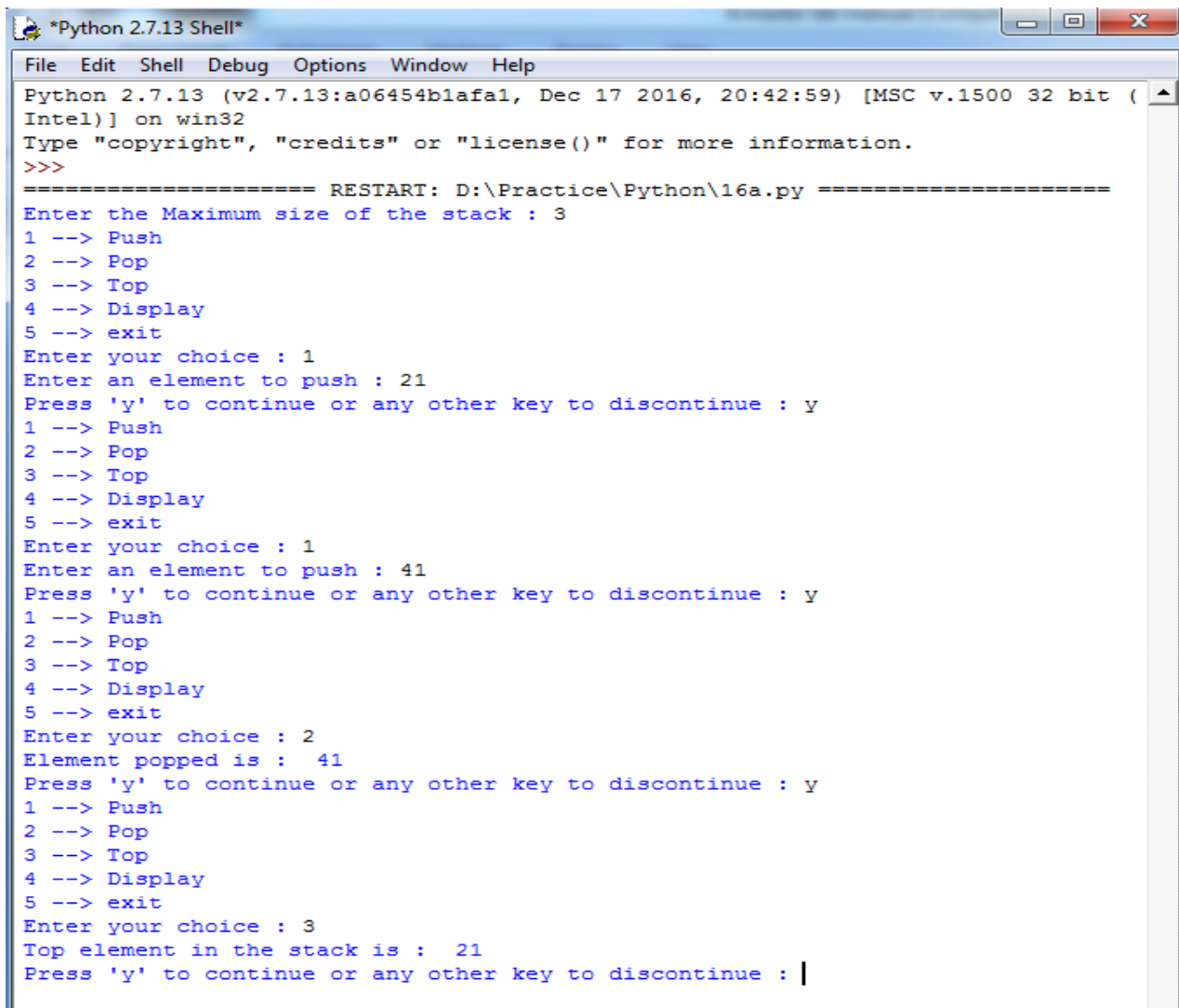
- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns top element of stack.



Applications of stack:

- Balancing of symbols.
- Infix to Postfix /Prefix conversion.
- Redo-undo features at many places like editors, photoshop.
- Forward and backward feature in web browsers.
- Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, histogram problem.
- Other applications can be Backtracking, Knight tour problem, rat in a maze, N queen problem and sudoku solver.

Output:



```
*Python 2.7.13 Shell*
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\16a.py =====
Enter the Maximum size of the stack : 3
1 --> Push
2 --> Pop
3 --> Top
4 --> Display
5 --> exit
Enter your choice : 1
Enter an element to push : 21
Press 'y' to continue or any other key to discontinue : y
1 --> Push
2 --> Pop
3 --> Top
4 --> Display
5 --> exit
Enter your choice : 1
Enter an element to push : 41
Press 'y' to continue or any other key to discontinue : y
1 --> Push
2 --> Pop
3 --> Top
4 --> Display
5 --> exit
Enter your choice : 2
Element popped is : 41
Press 'y' to continue or any other key to discontinue : y
1 --> Push
2 --> Pop
3 --> Top
4 --> Display
5 --> exit
Enter your choice : 3
Top element in the stack is : 21
Press 'y' to continue or any other key to discontinue : |
```

EXERCISE - 16(b)

Aim:

Write a program to solve knapsack problem.

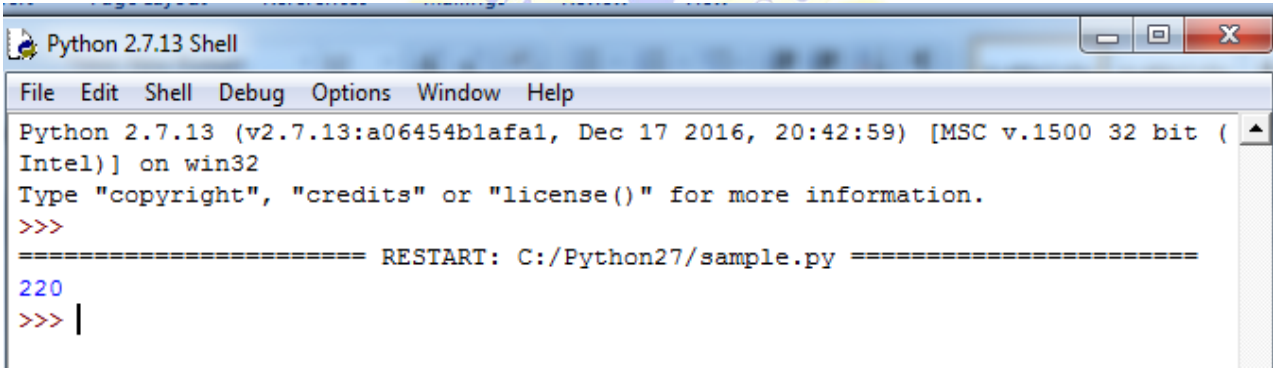
Description:

The **knapsack problem** or **rucksack problem** is a problem in combinatorial optimization. Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Applications of 0/1 Knapsack problem

Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials, selection of investments and portfolios, selection of assets for asset-backed securitization, and generating keys for the Merkle–Hellman and other knapsack cryptosystems.

Output:



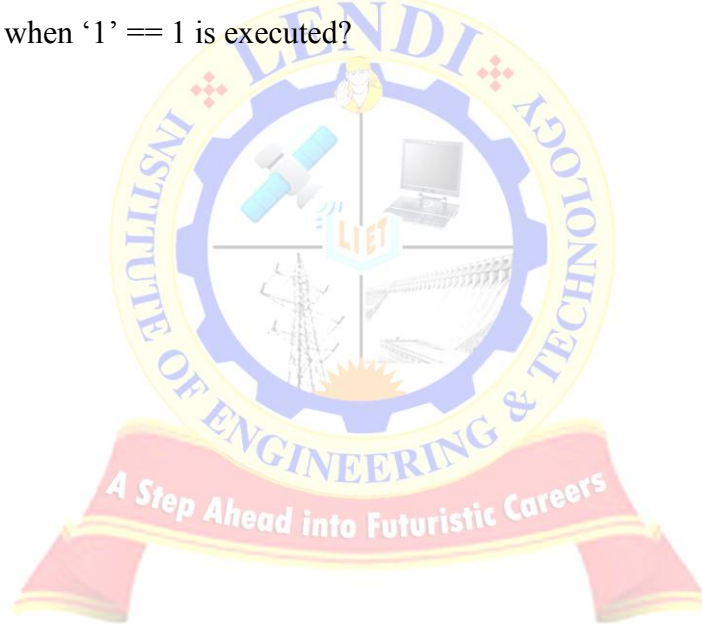
```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Python27/sample.py =====
220
>>> |
```

Conclusion:

Student gets the knowledge on data structures like stacks and Implementation of 0/1 knapsack problem. This experiment maps with CO2,CO4 and students can attain PO1,PO2, PO3,PO4, PO5,PO6,PO9,PO10, PO11,PO12 and PSO1,PSO2,PSO3.

Viva Questions:

1. What is stack?
2. Give real time and system examples of stack?
3. What is meant by push and pop?
4. When overflow will occur in stack?
5. What is the significance of top pointer?
6. How many stacks are needed to evaluate an expression without any embedded function calls ?
7. Which of the following method can be used to solve 0/1 knapsack problem?
8. What is another name for knapsack problem?
9. What is the time complexity for Knapsack problem?
10. What happens when '1' == 1 is executed?



Experiments beyond the Syllabus



Exercise – 1

Aim:

Write a program to implement DML operations using cx_Oracle.

Description:

Introduction to cx_Oracle:

cx_Oracle is a python extension module that enables access to oracle database. It conforms to the python database API specification. It is distributed under an open-source license. We can install it from the python command line by pip.

pip install cx_Oracle

Importing cx_Oracle:

Import the module in the python script as given below:

```
import cx_Oracle
```

connect():

Make a connection to an Oracle database by passing in the appropriate user/password to the following connection string:

```
connection = cx_Oracle.connect('sde/sde@localhost/XE')
```

Example:

```
connection = cx_Oracle.connect('system/password@localhost/XE')
```

cursor():

Define a parameter to access the cursor method.

```
cursor = connection.cursor()
```

execute():

For execution of a statement in database from python we were provided with execute. We need to pass a querystring as a parameter.

```
cursor.execute(querystring)
```

Example:

```
query_string = "SELECT * FROM parcels"
```

```
cursor.execute(query_string)
```


fetchall():

This method helps to retrieve data from the database as a tuple of tuples, such that each tuple is a row.

The 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.

```
rows = cursor.fetchall()
```

Example:

```
cursor.execute("SELECT * FROM parcels")
```

```
rows = cursor.fetchall()
```

commit():

This method sends a commit statement in order to commit the current transaction. Since by default connector/python does not autocommit, it is important to call this method after transaction that modifies data for tables that use transactional storage engines.

```
connection.commit()
```

close():

This method closes the cursor, resets all results, and ensures that the cursor object has no reference to its original connection object.

```
cursor.close()
```

Basic **DML Operations** those can be performed on database are:

insert:

In order to insert data into a table, insert command is used.

```
insert_stat = "INSERT INTO student(sid, sname) VALUES (104, 'Venkat')"
```

```
cursor.execute(insert_stat)
```

update:

This command updates existing data within a table.

```
update_stat = "UPDATE student SET sname = 'Raju' WHERE sid = 104"
```

```
cursor.execute(update_stat)
```

select:

This command inserts data into a table.

```
select_stat = "SELECT * from STUDENT"
```

```
cursor.execute(select_stat)
```

delete:

This command will delete all/some records from a database table.

```
delete_stat = "DELETE FROM STUDENT WHERE sid = 104"
```

```
cursor.execute(delete_stat)
```

Program-1(a): (insert operation)

```
import cx_Oracle
```

```
connection = cx_Oracle.connect('system/pavani@localhost/XE')
```

```
cursor = connection.cursor()
```

```
insert_stat = "INSERT INTO STUDENT(sid, sname, marks) VALUES (104, 'Venkat', 94)"
```

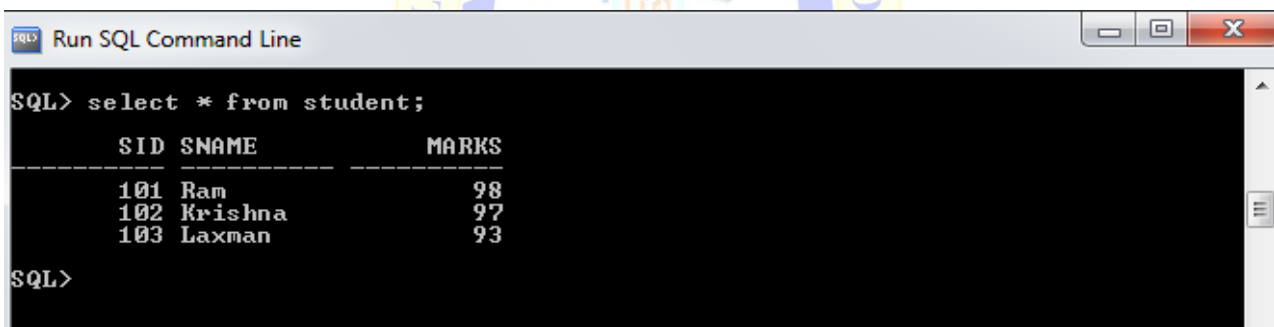
```
cursor.execute(insert_stat)
```

```
connection.commit()
```

```
cursor.close()
```

Output:

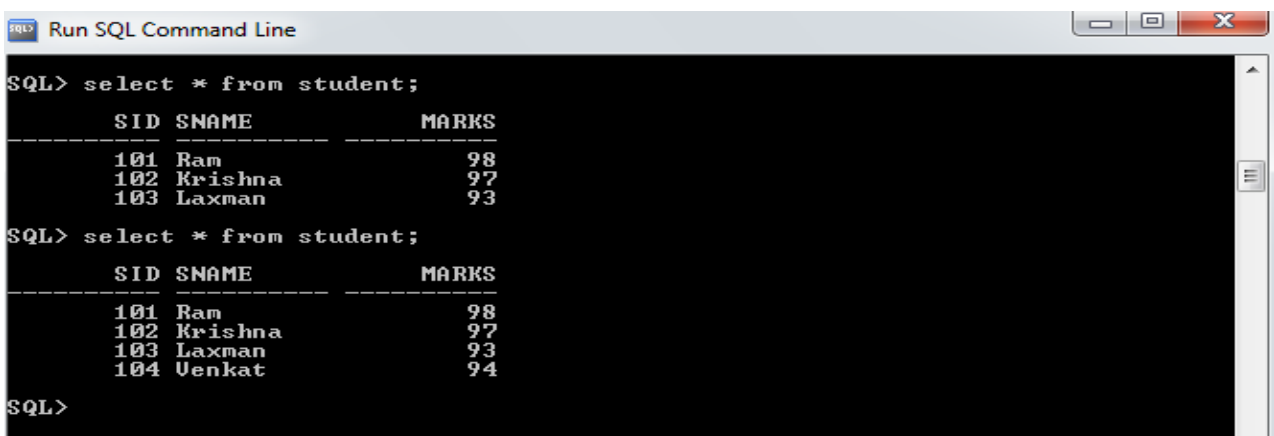
Student table in database before executing Program:



The screenshot shows a window titled "Run SQL Command Line" with a black background and white text. The prompt "SQL>" is followed by the command "select * from student;". The output is a table with three columns: SID, SNAME, and MARKS. The data rows are: 101 Ram 98, 102 Krishna 97, and 103 Laxman 93. The prompt "SQL>" is at the bottom.

SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93

Student table in database after executing Program:



The screenshot shows a window titled "Run SQL Command Line" with a black background and white text. The prompt "SQL>" is followed by the command "select * from student;". The output is a table with three columns: SID, SNAME, and MARKS. The data rows are: 101 Ram 98, 102 Krishna 97, 103 Laxman 93, and 104 Venkat 94. The prompt "SQL>" is at the bottom.

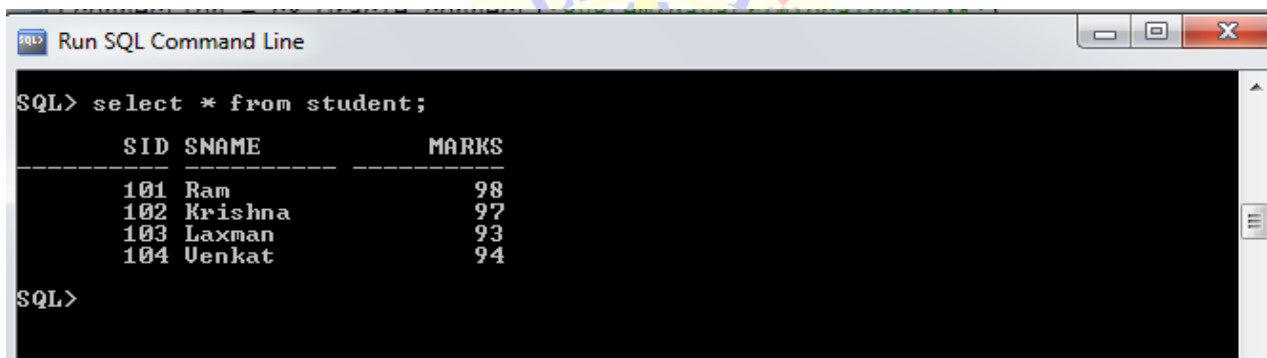
SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93
104	Venkat	94

Program-1(b): (update operation)

```
import cx_Oracle
connection = cx_Oracle.connect('system/pavani@localhost/XE')
cursor = connection.cursor()
update_stat = "UPDATE STUDENT SET marks = 99 WHERE sid = 104"
cursor.execute(update_stat)
connection.commit()
cursor.close()
```

Output:

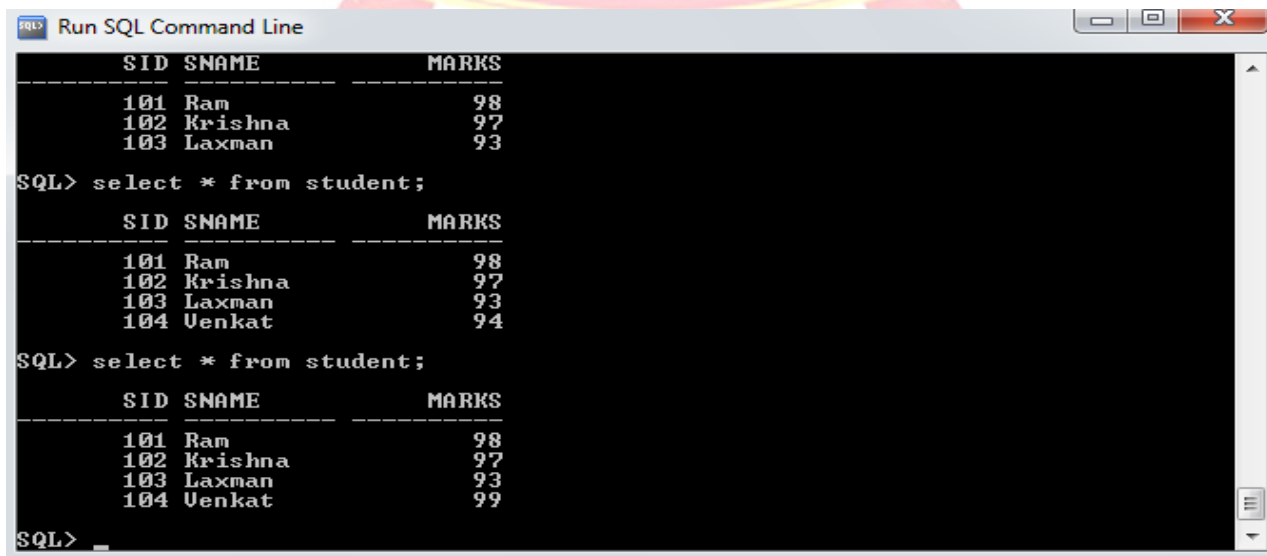
Student table in database before executing Program:



The screenshot shows a window titled "Run SQL Command Line" with a black background and white text. The prompt "SQL>" is followed by the command "select * from student;". Below the command, the output is displayed as a table with three columns: SID, SNAME, and MARKS. The data rows are: 101 Ram 98, 102 Krishna 97, 103 Laxman 93, and 104 Venkat 94. The prompt "SQL>" is shown again at the bottom.

SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93
104	Venkat	94

Student table in database after executing Program:



The screenshot shows the same "Run SQL Command Line" window. It displays the same table as before, but with an additional query. The first query and its output are the same. The second query is "SQL> select * from student;" and its output is the same table, but with the mark for student 104 (Venkat) updated to 99. The prompt "SQL>" is shown at the bottom.

SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93
104	Venkat	94

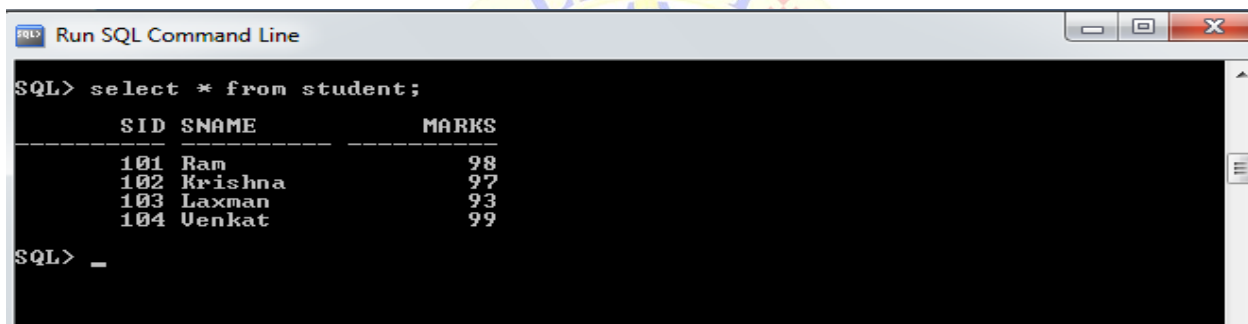
SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93
104	Venkat	99

Program-1(c): (delete operation)

```
import cx_Oracle  
connection = cx_Oracle.connect('system/pavani@localhost/XE')  
cursor = connection.cursor()  
delete_stat = "DELETE FROM STUDENT WHERE sid = 104"  
cursor.execute(delete_stat)  
connection.commit()  
cursor.close()
```

Output:

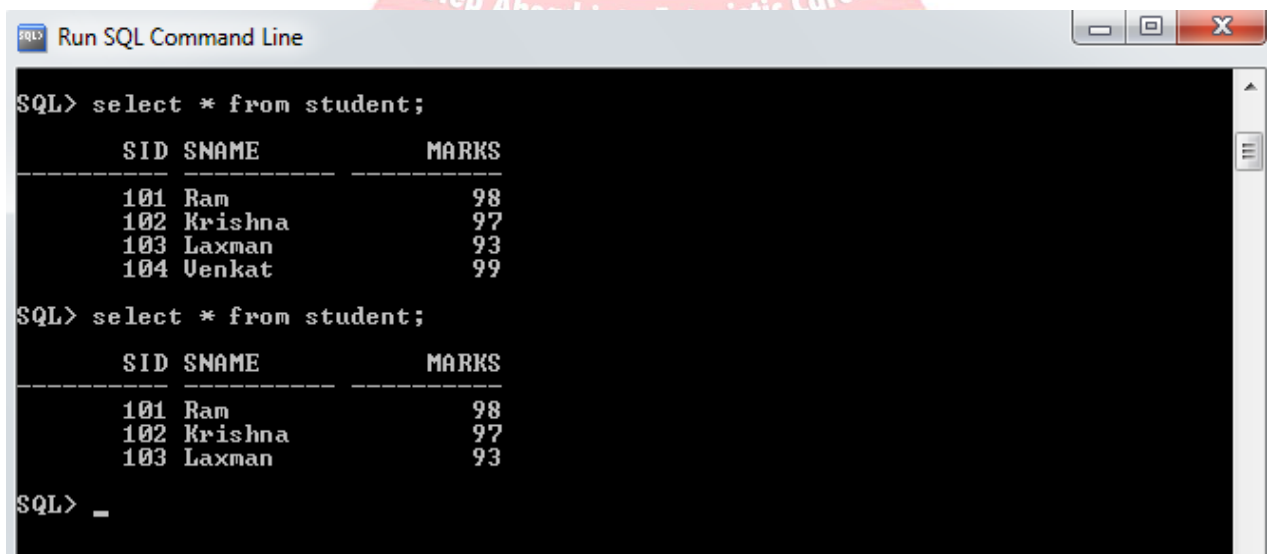
Student table in database before executing Program:



The screenshot shows a window titled "Run SQL Command Line". The command prompt shows the command "SQL> select * from student;" and the output is a table with three columns: SID, SNAME, and MARKS. The data rows are: 101 Ram 98, 102 Krishna 97, 103 Laxman 93, and 104 Venkat 99. The prompt "SQL> _" is visible at the bottom.

SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93
104	Venkat	99

Student table in database after executing Program:



The screenshot shows the same "Run SQL Command Line" window. The first command "SQL> select * from student;" shows the same four rows as before. The second command "SQL> select * from student;" shows only three rows: 101 Ram 98, 102 Krishna 97, and 103 Laxman 93. The row with SID 104 has been deleted. The prompt "SQL> _" is visible at the bottom.

SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93

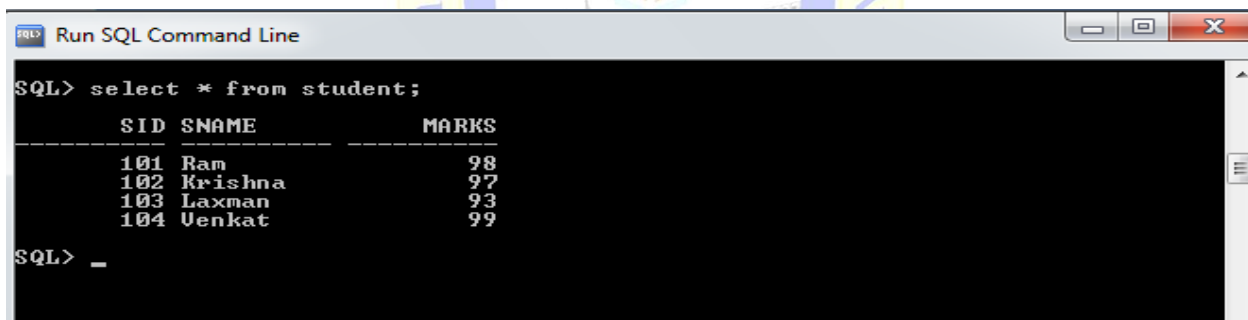
Program-1(d): (select operation)

```
import cx_Oracle
connection = cx_Oracle.connect('system/pavani@localhost/XE')
cursor = connection.cursor()

select_stat = "SELECT * from STUDENT"
cursor.execute(select_stat)
rows = cursor.fetchall()
for sid, sname, marks in rows:
    print sid, sname, marks
connection.commit()
cursor.close()
```

Output:

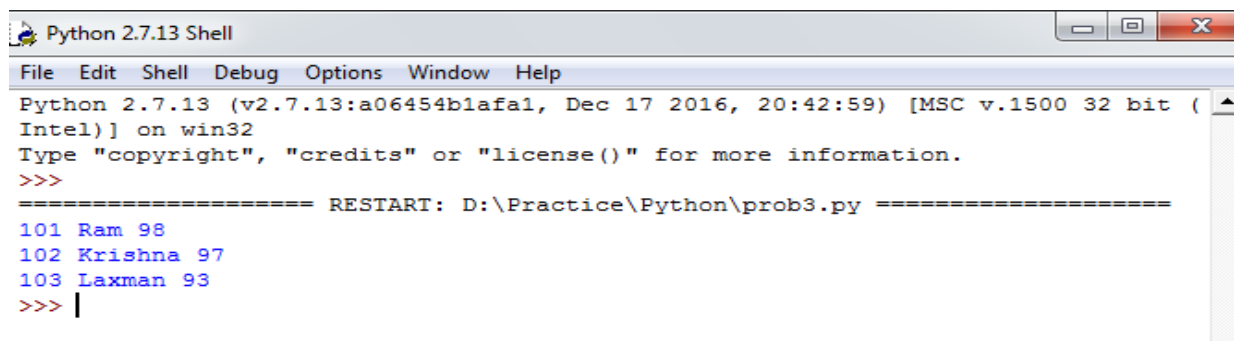
Student table in database is:



The screenshot shows a window titled "Run SQL Command Line". The command entered is "SQL> select * from student;". The output is a table with three columns: SID, SNAME, and MARKS. The data rows are: 101 Ram 98, 102 Krishna 97, 103 Laxman 93, and 104 Venkat 99.

SID	SNAME	MARKS
101	Ram	98
102	Krishna	97
103	Laxman	93
104	Venkat	99

Retrieving the rows from student table and displaying the same in python shell after executing above program:



The screenshot shows a window titled "Python 2.7.13 Shell". The output of the script is: "Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", followed by a restart message and the output of the script: "101 Ram 98", "102 Krishna 97", "103 Laxman 93".

```
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Practice\Python\prob3.py =====
101 Ram 98
102 Krishna 97
103 Laxman 93
>>> |
```

Conclusion:

Student gets the knowledge on how DML operations are performed on database using cx_Oracle. Students can attain PO1,PO2,PO3,PO4,PO5,PO6,PO9,PO10,PO11,PO12 and PSO1, PSO2, PSO3.

