DAYANANDA SAGAR
COLLEGE OF ENGINEERING

An Autonomous Institution
Affiliated to VTU
Approved by AICTE & UGC
Accredited by NBA
Accredited by NAAC with 'A' grade

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

PYTHON PROGRAMMING LABORATORY MANUAL

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTE AFFILIATED TO VTU, BELAGAVI)
Shavige Malleshwara Hills, Kumaraswamy Layout, Bangalore-560078

Course Name and Course Code	: Python Programming Laboratory (21CGL46)
Year and Semester	: II year, IV semester
Name of the Faculty's	: Dr. Vinod H C, Harshitha H R

VISION AND MISSION OF THE INSTITUTION

INSTITUTION VISION

To impact quality technical education with a focus on Research and Innovation emphasizing on Development of Sustainable and Inclusive Technology for the benefit of society.

INSTITUTION MISSION

- To provide an environment that enhances creativity and Innovation in pursuit of Excellence.
- To nurture teamwork in order to transform individuals as responsible leaders and entrepreneurs.
- To train the students to the changing technical scenario and make them to understand the importance of Sustainable and Inclusive technologies.

VISION AND MISSION OF CSD DEPARTMENT

DEPARTMENT VISION

Computer Science and Design Engineering Department shall architect the most innovative programs to deliver competitive and sustainable solutions using cutting edge technologies and implementations, for betterment of society and research.

DEPARTMENT MISSION

- To adopt the latest industry trends in teaching learning process in order to make students competitive in the job market
- To encourage forums that enable students to develop skills in multidisciplinary areas and emerging technologies
- To encourage research and innovation among students by creating an environment of learning through active participation and presentations
- To collaborate with industry and professional bodies for the students to gauge the market trends and train accordingly.
- To create an environment which fosters ethics and human values to make students responsible citizens.

COURSE INFORMATION SHEET

PROGRAMME: Computer Science and Design / Bachelors in Engineering						
COURSE NAME: PYTHON PROGRAMMING LABORATORY	**COURSE CODE: 21CGL46**					
COURSE TYPE (CORE / ELECTIVE): PCCL	**CONTACT HOURS: 24**					
CORRESPONDING LAB COURSE (IF ANY):Formal Lab	**CREDIT**	**L**	**T**	**P**	**S**	**Total**
		-	-	2	-	1
COURSE INSTRUCTOR(S) NAME: Dr. Vinod H C **Harshitha H R**	**CONTACT DETAILS:** **Dr. Vinod H.C** **Harshitha H R** vinodhc-csd@dayanandasagar.edu harshithahr-csd@dayanandasagar.edu **Room No. 404**, 4th Floor, Building no. 22, CSD Department, DSCE.					

COURSE OUTCOMES (CO)

Sl. No.	DESCRIPTION	REVISED BLOOM'S TAXONOMY (RBT)LEVEL
1.	**CO1:** Write Python programs that effectively utilize data types, operators, flow control, and exception handling to solve various programming problems.	L3
2.	**CO2:** Create functions in Python and understand how to pass parameters and return values, enabling the writing of modular and reusable code.	L4
3.	**CO3:** Demonstrate a solid understanding of string manipulation techniques and apply them to solve real-world problems.	L3
4.	**CO4:** Become proficient in working with different collections like lists, tuples, and dictionaries, and choose the appropriate collection for a given task	L4
5.	**CO5:** Employ pattern recognition techniques, including regular expressions, to extract and manipulate data effectively. Also, read, write, and organize various types of files, including spreadsheets, web pages, PDFs, Word documents, and JSON files, to perform data processing tasks.	L5

COURSE OBJECTIVES:

CLO 1: Understand the fundamentals of Python programming language, including data types, operators, flow control, and exception handling.
CLO 2: Learn how to create and utilize functions in Python, including passing parameters and returning values.
CLO 3: Explore the manipulation of strings using string methods and learn how to work with different collections like lists, tuples, and dictionaries.
CLO 4: Gain proficiency in pattern recognition, both with and without regular expressions, and apply them in practical scenarios.
CLO 5: Develop skills in reading, writing, and organizing files, as well as working with different file formats such as spreadsheets, web pages, PDFs, Word documents, and JSON files.

Note: two hours tutorial is suggested for each laboratory sessions.

Prerequisite

- Students should be familiarized about Python installation and setting Python environment
- Usage of IDLE or IDE like PyCharm should be introduced
  - ➢Python Installation: https://www.youtube.com/watch?v=Kn1HF3oD19c
  - ➢PyCharm Installation: https://www.youtube.com/watch?v=SZUNUB6nz3g

MAPPING OF CO's WITH PO's AND PSO's:

	PO												PSO		
CO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO1	3	-	-	2	-	1	-	-	2	-	1	2	-	2	1
CO2	2	-	-	1	2	-	-	-	3	2	-	2	-	2	1
CO3	2	2	-	1	-	1	-	-	1	2	-	2	-	2	1
CO4	3	-	-	2	-	-	-	-	2	2	-	2	-	2	1
CO5	2	3	-	2	-	2	-	-	2	2	-	2	-	2	1
Average	2.4	1	0	1.6	0.4	0.8	0	0	2	1.6	0.2	2	0	2	1

TEXTBOOKS:

1. Al Sweigart, **“Automate the Boring Stuff with Python”**,1stEdition, No Starch Press, 2015.(Available under CC-BY-NC-SA license at https://automatetheboringstuff.com/)
2. Reema Thareja “**Python Programming Using Problem Solving Approach**” Oxford UniversityPress.
3. Allen B. Downey, **“Think Python: How to Think Like a Computer Scientist”,** 2nd Edition, Green Tea Press, 2015. (Available under CC-BY-NC license at http://greenteapress.com/thinkpython2/thinkpython2.pdf)

LISTS OF EXPERIMENTS

Sl. No.	Name of the Experiment	Course Outcome(CO's)
1	**Aim:** Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python. a) **Write a Python program that accepts an integer from the user and determines whether it is a prime number or not.** b) **Write a Python program that reads a list of numbers from the user and calculates the sum of all even numbers in the list.**	CO1
2	**Aim:** Demonstrating creation of functions, passing parameters and return values. a) **Write a Python program that defines a function calculate_Fibonacci which accepts a positive integer n as input. The function calculates the Fibonacci number at position n using the formula Fn = Fn-1 + Fn-2, where F0 = 0 and F1 = 1. The program should display the Fibonacci number at position n or a suitable error message if the input value is not a positive integer.** b) **Write a Python function called find_maximum that accepts a list of numbers as a parameter and returns the maximum value from the list.**	CO2
3	**Aim:** Demonstration of manipulation of strings using string methods. a) **Write a Python program that accepts a sentence from the user and counts the number of occurrences of a specific word in the sentence. The program should prompt the user to enter both the sentence and the word to search for.** b) **Write a Python program that calculates the string similarity between two given strings provided by the user. The program should prompt the user to enter both strings and then calculate the similarity score between them.**	CO3
4	**Aim:** Discuss different collections like list, tuple and dictionary. a) **Write a Python program that takes a list of numbers as input from the user and calculates the sum of all the numbers in the list. The program should keep accepting numbers until the user enters a negative number.** b) **Write a Python program that reads a sentence from the user and counts the frequency of each word in the sentence. The program should store the word frequencies in a dictionary, where the words are the keys and the frequencies are the values.**	CO4
5	**Aim:** Demonstration of pattern recognition with and without using regular expressions. a) **Write a Python program that takes a list of words as input from the user and filters out words that start with a specific prefix. The program should prompt the user to enter the list of words and the prefix to filter.**	CO5

	b) **Write a Python program that takes a sentence as input from the user and extracts all the email addresses present in the sentence. The program should use regular expressions to identify and extract the email addresses.**	
6	**Aim:** Demonstration of reading, writing and organizing files. a) **Write a Python program that reads the content of a text file named "input.txt" and counts the number of occurrences of each word in the file. Then, write the word count results to a new file named "output.txt" in the format "word: count" for each word found in the input file. Ensure that the program ignores punctuation and treats uppercase and lowercase letters as the same word.** b) **Create a Python program that takes a directory path as input from the user and lists all the files and subdirectories within that directory. Additionally, the program should write the names of all the files in a separate text file named "file_list.txt".**	CO5
7	**Aim:** Demonstration of the concepts of classes, methods, objects and inheritance. a) **Create a class called Rectangle with the following attributes and methods:** **Attributes: length and width** **Methods:** **area(): Calculate and return the area of the rectangle.** **perimeter(): Calculate and return the perimeter of the rectangle.** **Create an object of the Rectangle class, initialize its attributes, and display its area and perimeter.** b) **Create a base class called Animal with the following attributes and methods:** **Attributes: name and age** **Methods:** **speak(): Display a message saying "The animal speaks."** **Create a derived class called Dog that inherits from the Animal class. Add a new method to the Dog class called bark(), which displays a message saying "The dog barks."** **Create an object of the Dog class, initialize its attributes, and demonstrate both the speak() and bark() methods.**	CO2
8	**Aim:** Demonstration of classes and methods with polymorphism and overriding. a) **Create a class called Shape with a method calculate_area(). The Shape class should be the base class for three other classes: Rectangle, Triangle, and Circle. Each subclass should have its own implementation of the calculate_area() method to calculate and return the area specific to that shape. Demonstrate polymorphism by creating objects of each class and calling the calculate_area() method on them.**	CO2
9	**Aim:** Demonstration of working with excel spreadsheets and web scraping. a) **Excel Spreadsheet Manipulation:** **Write a Python program that reads data from an Excel spreadsheet and performs the following tasks:** **Open the "data.xlsx" file and read the data from the "Sheet1".**	CO5

	Calculate the average of the numbers in column A and print it. **Find the maximum value in column B and print it.** **Create a new Excel file named "output.xlsx" and write the calculated average and maximum value in separate cells in the "Sheet1".** **b) Web Scraping with BeautifulSoup:** **Write a Python program that performs web scraping using BeautifulSoup library to extract data from a webpage. Follow these steps:** **Use the requests library to send a GET request to the URL: "https://www.example.com".** **Parse the HTML content of the response using BeautifulSoup.** **Extract all the <h2> tags from the HTML and print their text.** **Find all the links (<a> tags) in the HTML and print their href attributes.** **Extract the content of the first paragraph (<p> tag) and print it.**	
10	**Aim:** Demonstration of working with PDF, word and JSON files. **a) Write a python program to combine select pages from many PDFs.** **b) Write a python program to fetch current weather data from the JSON file.**	CO5

PYTHON PROGRAMMING LABORATORY MANUAL

SOFTWARE REQUIREMENTS:

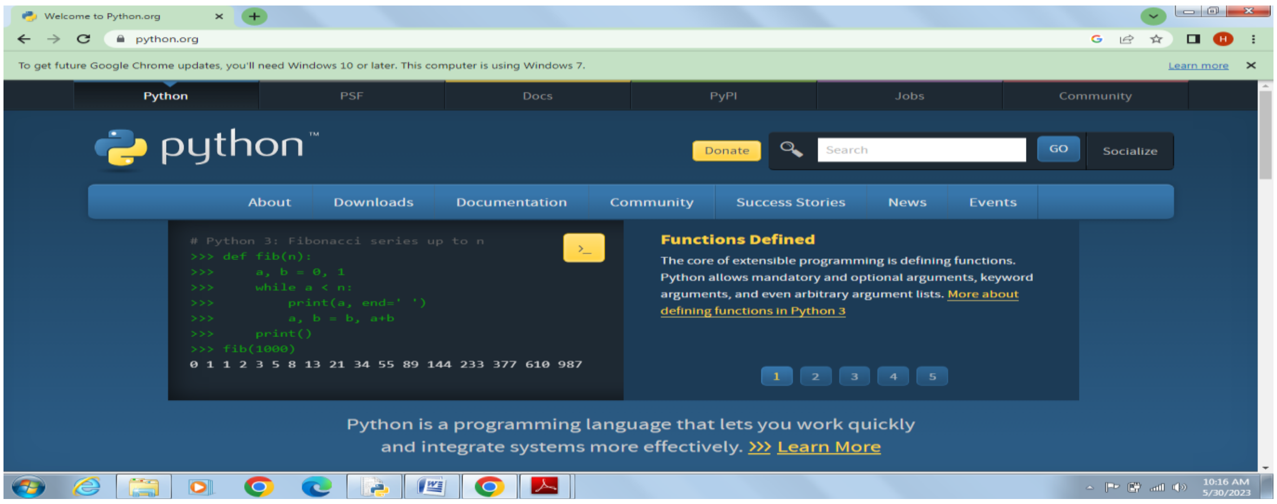
Python 3.11.3, IDLE editor

HARDWARE REQUIREMENTS:

PC – Intel / AMD Core i2 or above, 2 GB RAM, 500 GB Hard disk.

STEPS FOR INSTALLATION OF PYTHON 3.11.3

Step-1: Go for python.org



To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7.

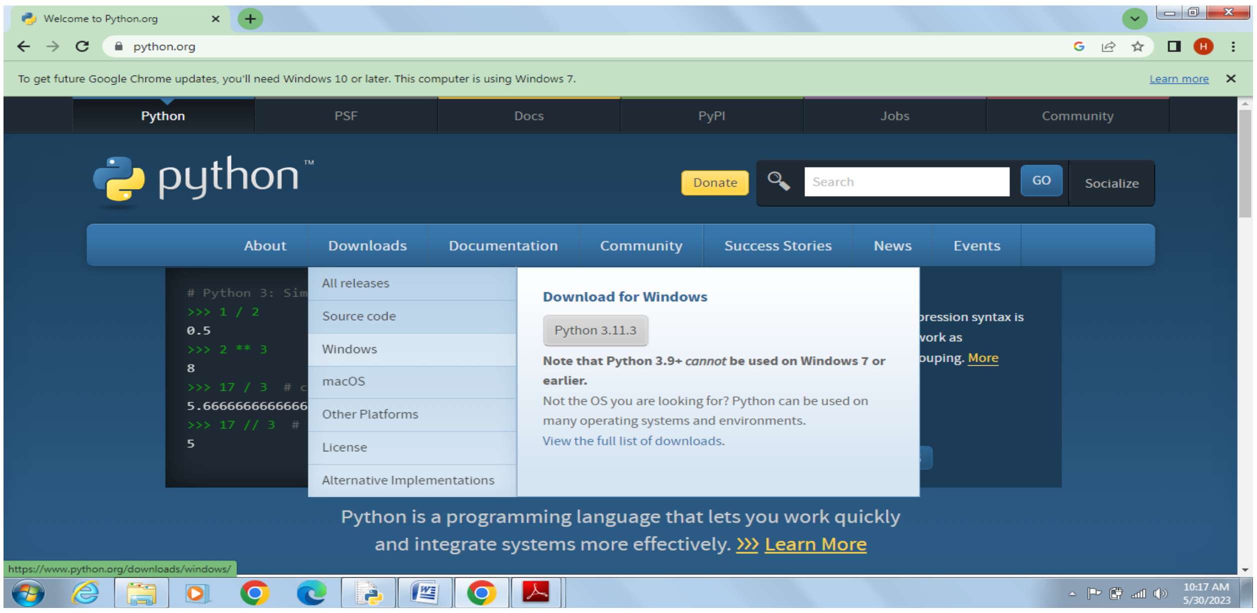
```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Step-2:Go to Downloads and download the latest version of python for windows, here latest version is Python 3.11.3



To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7.

```
# Python 3: Sim
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # c
5.6666666666666
>>> 17 // 3  #
5
```

Download for Windows

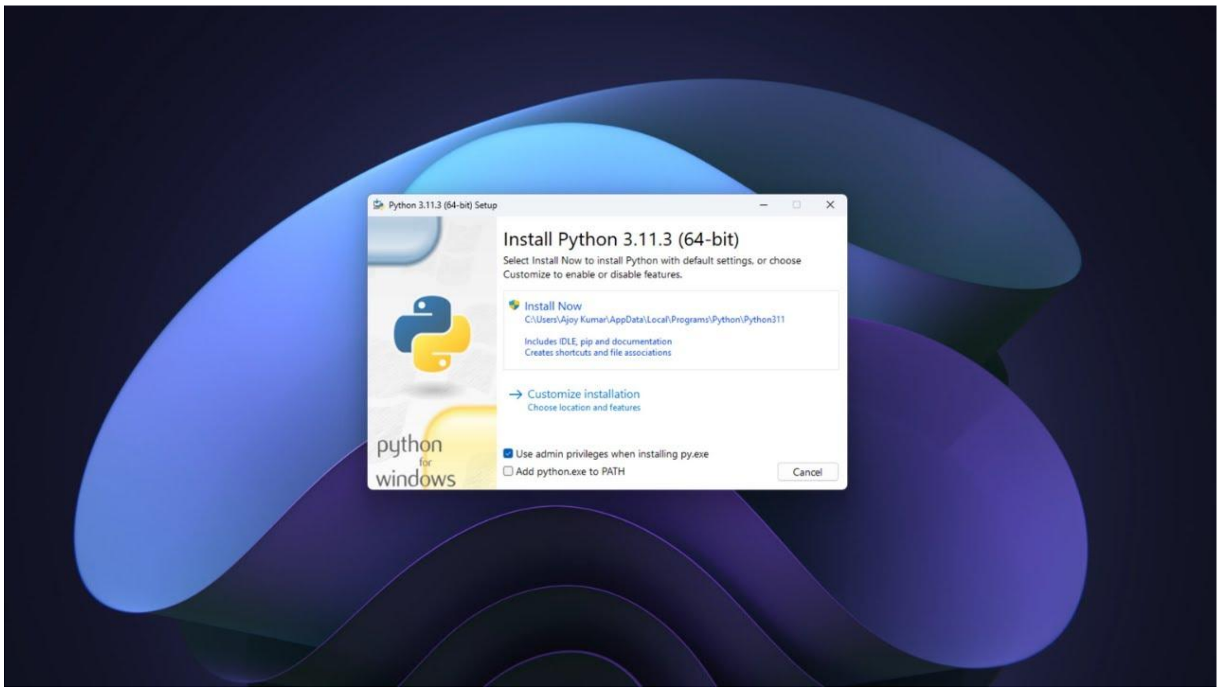
Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

pression syntax is
work as
ouping. More

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More


Python 3.11.3 (64-bit) Setup

Install Python 3.11.3 (64-bit)

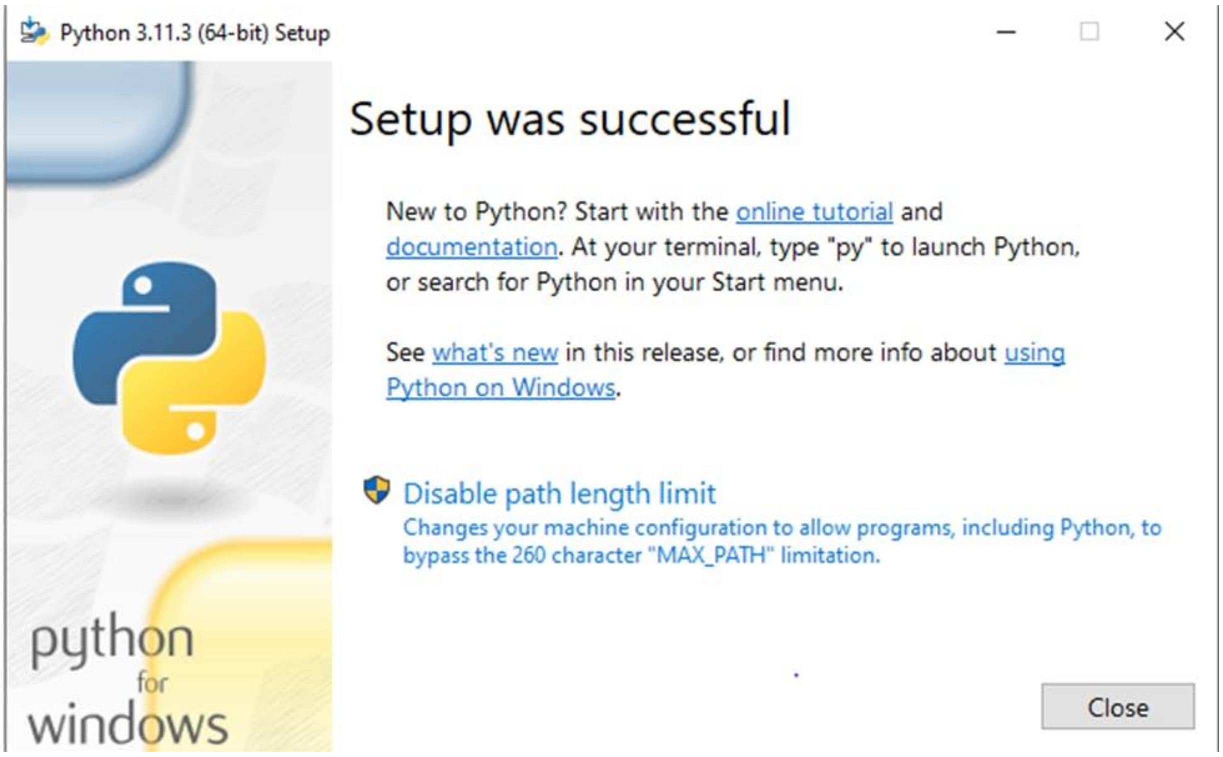
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Ajoy Kumar\AppData\Local\Programs\Python\Python311

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

[✓] Use admin privileges when installing py.exe
[ ] Add python.exe to PATH


Python 3.11.3 (64-bit) Setup

Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

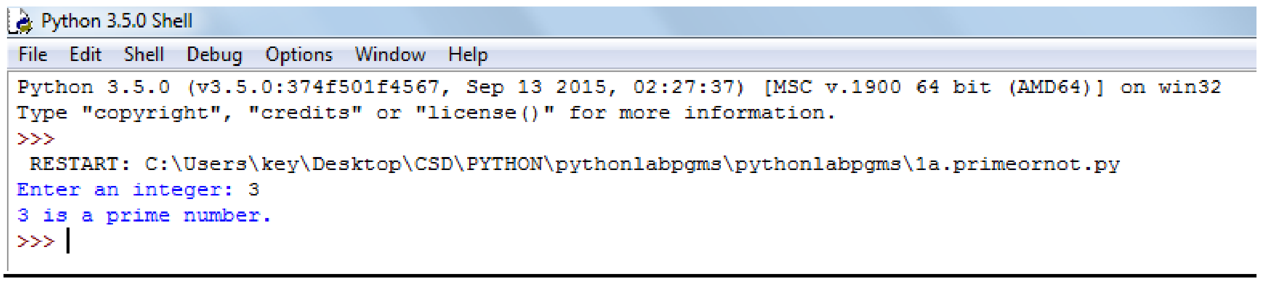
Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

PROGRAM -1

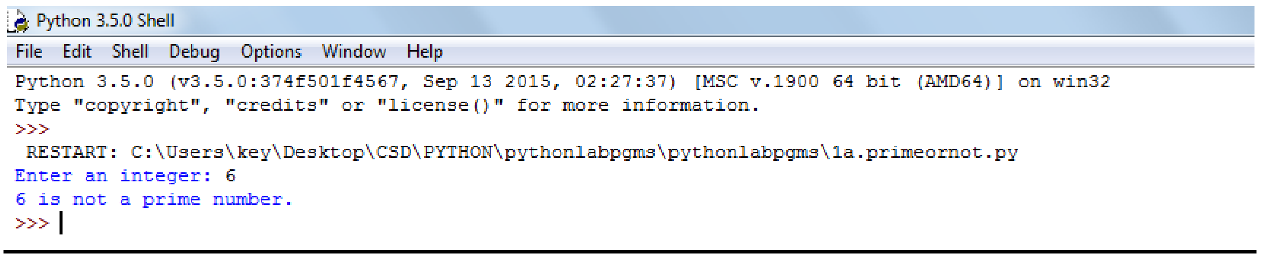
(A). Write a Python program to check whether a given number is prime or not.

```
def is_prime(number):
    if number < 2:
        return False

    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            return False

    return True

# Accept an integer from the user
num = int(input("Enter an integer: "))

# Check if the number is prime
if is_prime(num):
    print(num, "is a prime number.")
else:
    print(num, "is not a prime number.")
```

OUTPUT



```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\1a.primeornot.py
Enter an integer: 3
3 is a prime number.
>>>
```



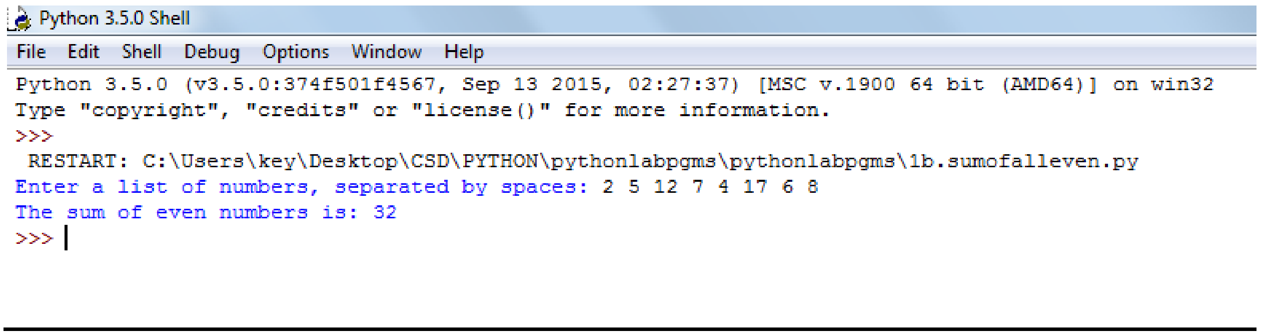
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\1a.primeornot.py
Enter an integer: 6
6 is not a prime number.
>>>
```

PROGRAM -1

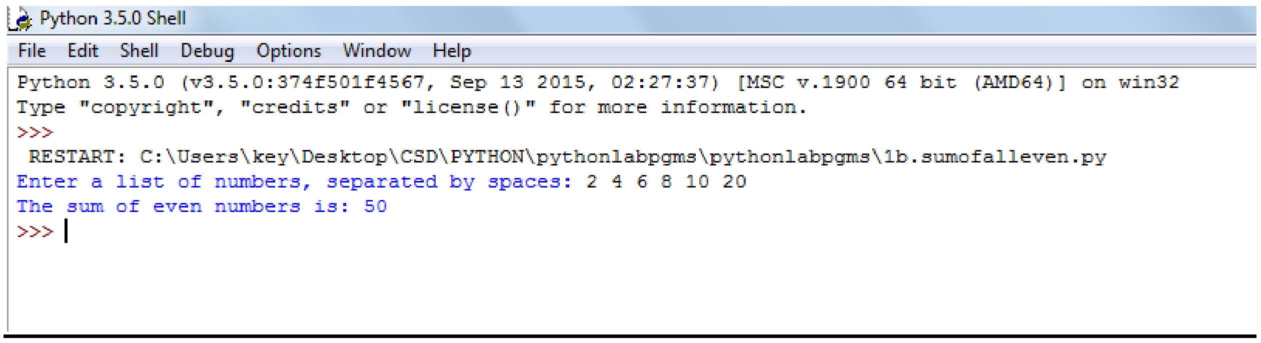
(B). Write a Python program to get a list of numbers and calculate sum of all even numbers.

```
# Accept input from the user
numbers = input("Enter a list of numbers, separated by spaces: ").split()

# Convert the input values to integers
numbers = [int(num) for num in numbers]

# Calculate the sum of even numbers
even_sum = sum(num for num in numbers if num % 2 == 0)

# Output the result
print("The sum of even numbers is:", even_sum)
```

OUTPUT



```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\1b.sumofalleven.py
Enter a list of numbers, separated by spaces: 2 5 12 7 4 17 6 8
The sum of even numbers is: 32
>>> 
```



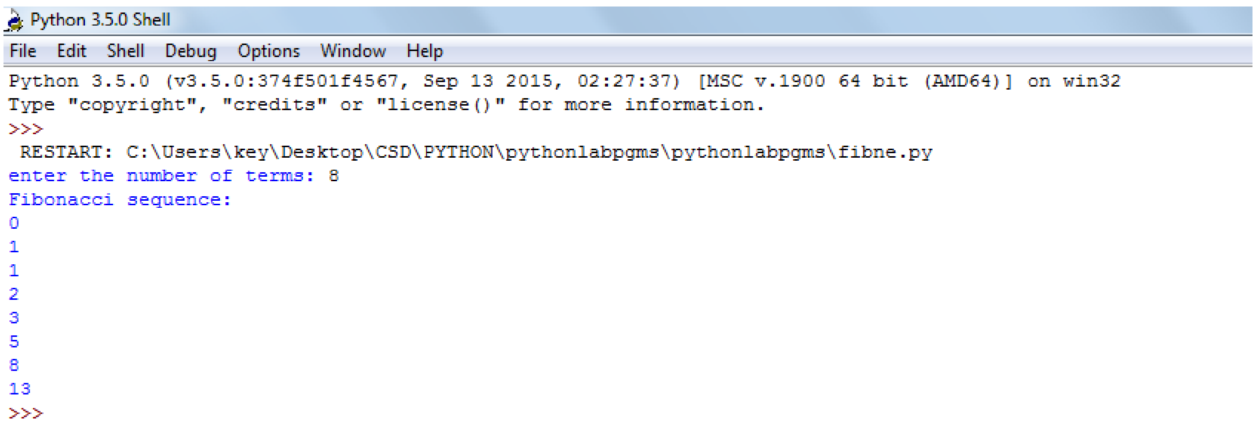
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\1b.sumofalleven.py
Enter a list of numbers, separated by spaces: 2 4 6 8 10 20
The sum of even numbers is: 50
>>> 
```

PROGRAM -2

(A). Write a Python program in which each new term in the Fibonacci sequence is generated by adding the previous two terms.

```
def recur_fibo(n):
   if n <= 1:
       return n
   else:
       return(recur_fibo(n-1) + recur_fibo(n-2))
# take input from the user
nterms = int(input("enter the number of terms: "))
# check if the number of terms is valid
if nterms <= 0:
   print("Plese enter a positive integer")
else:
   print("Fibonacci sequence:")
   for i in range(nterms):
       print(recur_fibo(i))
```

OUTPUT



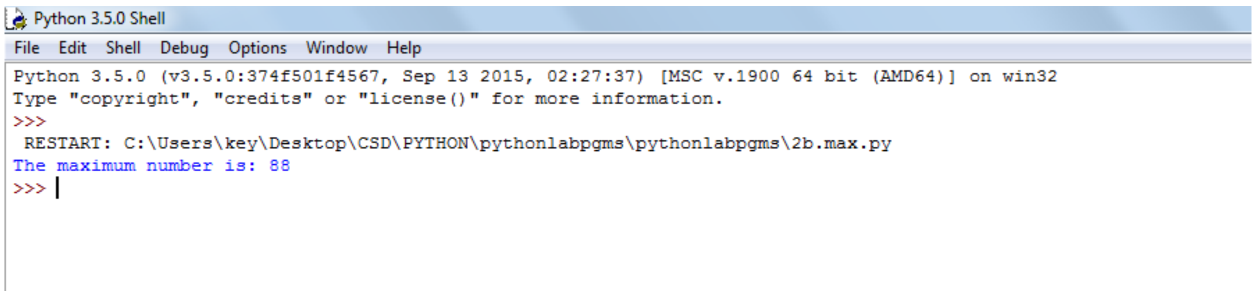
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\fibne.py
enter the number of terms: 8
Fibonacci sequence:
0
1
1
2
3
5
8
13
>>>
```

PROGRAM -2

(B). Write a Python program to get maximum number from the given list of numbers.

```
def find_maximum(numbers):
    if not numbers:
        raise ValueError("The list is empty.")
    maximum = numbers[0]  # Initialize maximum with the first element
    for num in numbers:
        if num > maximum:
            maximum = num
    return maximum
# Call the function with a list of numbers and store the result in a variable
numbers = [1, 40, 2, 88, 3]
max_num = find_maximum(numbers)
# Print the result
print("The maximum number is:", max_num)
```

OUTPUT



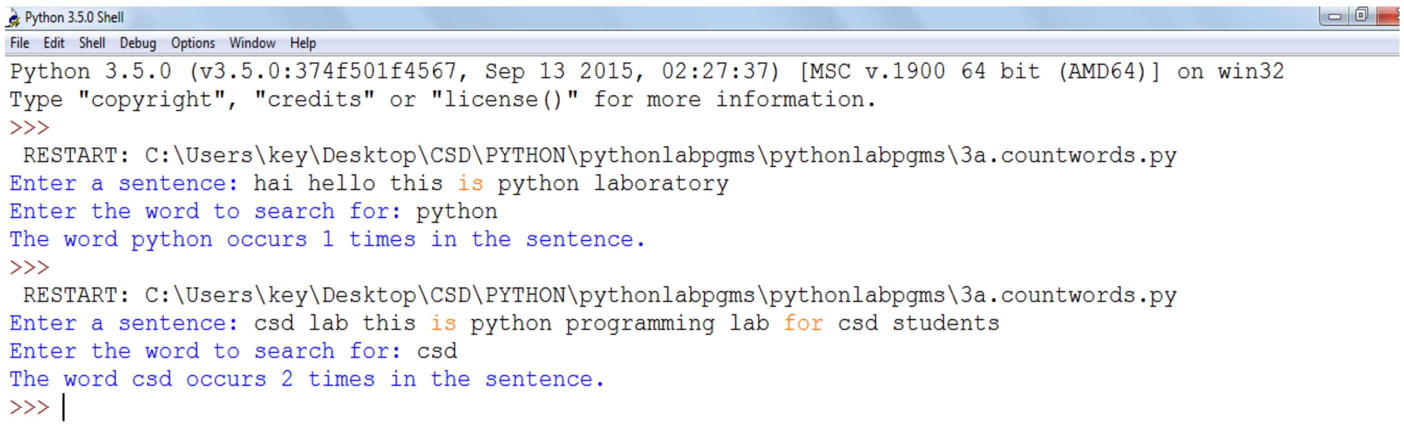
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\2b.max.py
The maximum number is: 88
>>> 
```

PROGRAM -3

(A).Write a Python program that accepts a sentence from the user and counts the number of occurrences of a specific word in the sentence. The program should prompt the user to enter both the sentence and the word to search for.

```
def count_word_occurrences(sentence, word):
    # Convert the sentence and word to lowercase for case-insensitive matching
    sentence = sentence.lower()
    word = word.lower()

    # Split the sentence into individual words
    words = sentence.split()

    # Count the occurrences of the word in the sentence
    count = words.count(word)

    return count

# Accept input from the user
sentence = input("Enter a sentence: ")
word = input("Enter the word to search for: ")

# Call the function and get the count of word occurrences
occurrences = count_word_occurrences(sentence, word)

# Display the result
print("The word",word,"occurs",occurrences, "times in the sentence.")
```

OUTPUT



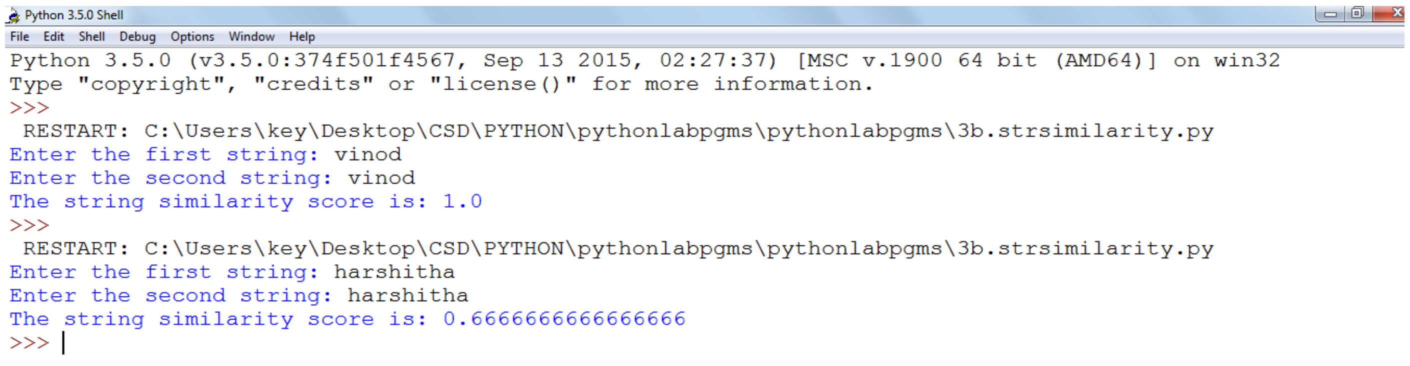
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\3a.countwords.py
Enter a sentence: hai hello this is python laboratory
Enter the word to search for: python
The word python occurs 1 times in the sentence.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\3a.countwords.py
Enter a sentence: csd lab this is python programming lab for csd students
Enter the word to search for: csd
The word csd occurs 2 times in the sentence.
>>>
```

PROGRAM -3

(B). Write a Python program that calculates the string similarity between two given strings provided by the user. The program should prompt the user to enter both strings and then calculate the similarity score between them.

```
def calculate_string_similarity(string1, string2):
    # Convert the strings to lowercase for case-insensitive matching
    string1 = string1.lower()
    string2 = string2.lower()

    # Calculate the similarity score
    common_chars = set(string1) & set(string2)
    similarity = len(common_chars) / max(len(string1), len(string2))

    return similarity

# Accept input from the user
string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")

# Call the function to calculate the similarity score
similarity_score = calculate_string_similarity(string1, string2)

# Display the result
print("The string similarity score is:", similarity_score)
```

OUTPUT



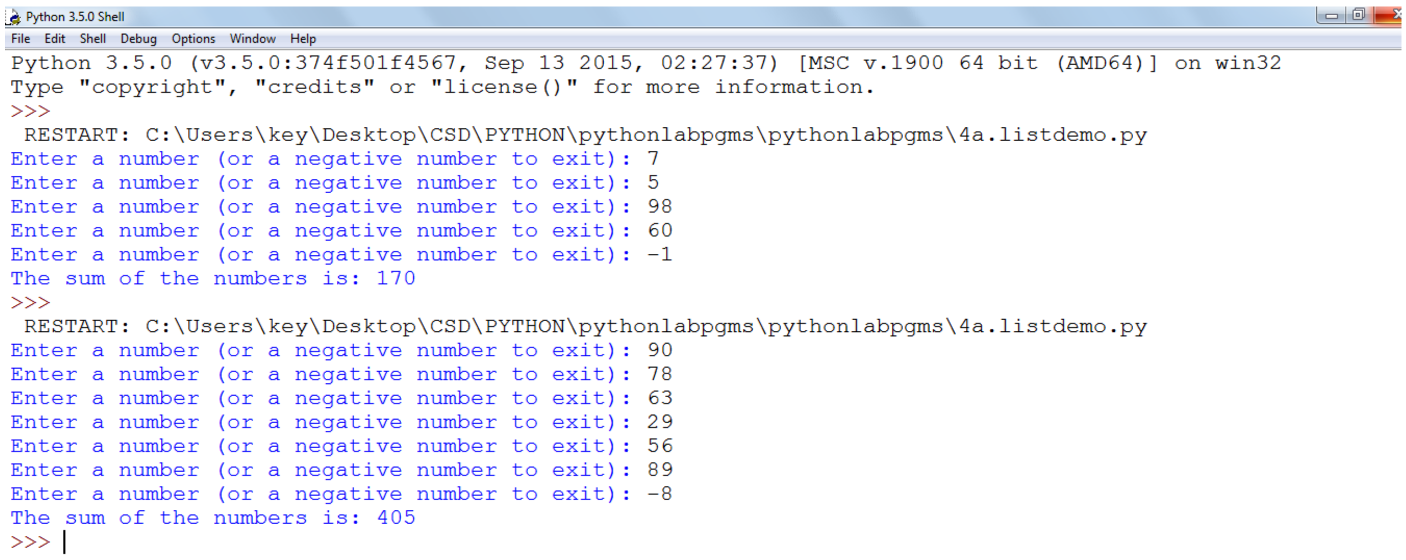
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\3b.strsimilarity.py 
Enter the first string: vinod
Enter the second string: vinod
The string similarity score is: 1.0
>>> 
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\3b.strsimilarity.py 
Enter the first string: harshitha
Enter the second string: harshitha
The string similarity score is: 0.6666666666666666
>>> |
```

PROGRAM -4

(A). Write a Python program that takes a list of numbers as input from the user and calculates the sum of all the numbers in the list. The program should keep accepting numbers until the user enters a negative number.

```
numbers = []

# Accept numbers from the user
while True:
    num = int(input("Enter a number (or a negative number to exit): "))
    if num < 0:
        break
    numbers.append(num)

# Calculate the sum of the numbers
total = sum(numbers)

# Display the result
print("The sum of the numbers is:", total)
```

OUTPUT



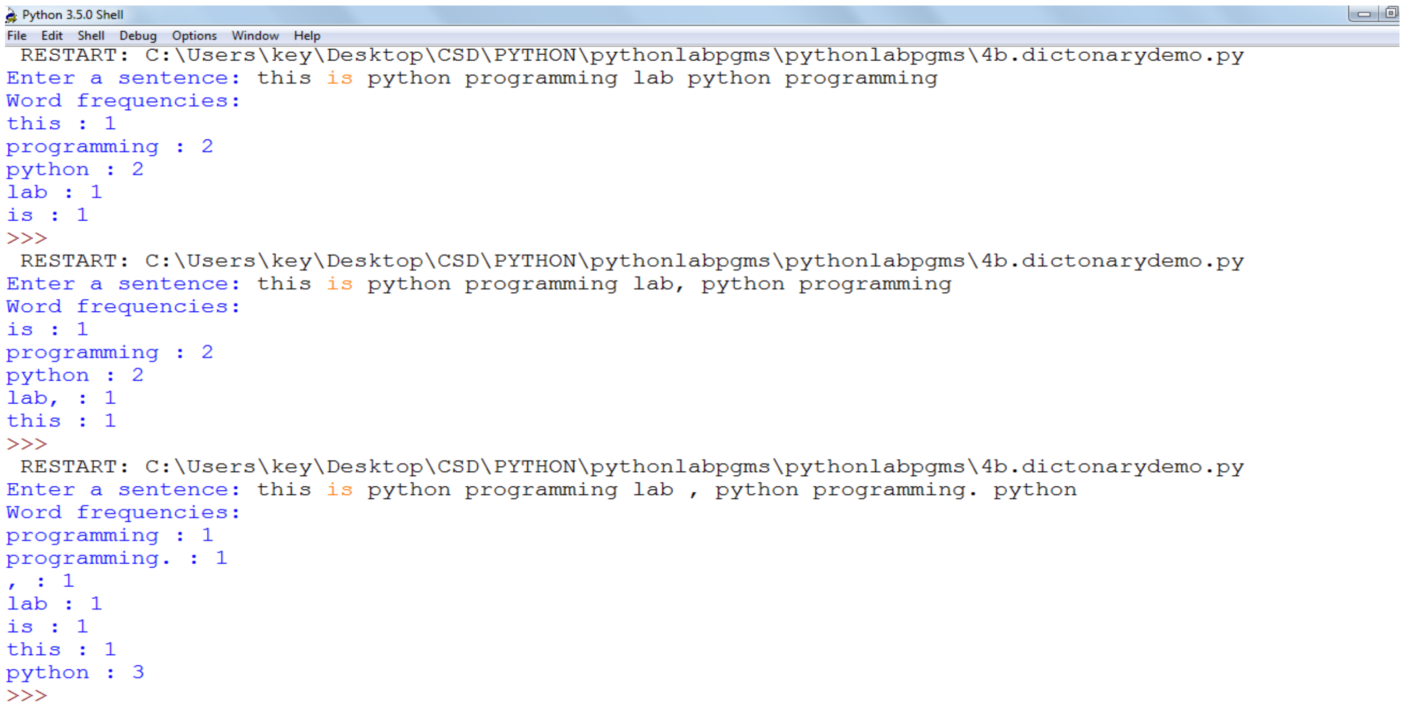
```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\4a.listdemo.py
Enter a number (or a negative number to exit): 7
Enter a number (or a negative number to exit): 5
Enter a number (or a negative number to exit): 98
Enter a number (or a negative number to exit): 60
Enter a number (or a negative number to exit): -1
The sum of the numbers is: 170
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\4a.listdemo.py
Enter a number (or a negative number to exit): 90
Enter a number (or a negative number to exit): 78
Enter a number (or a negative number to exit): 63
Enter a number (or a negative number to exit): 29
Enter a number (or a negative number to exit): 56
Enter a number (or a negative number to exit): 89
Enter a number (or a negative number to exit): -8
The sum of the numbers is: 405
>>> |
```

PROGRAM -4

(B). Write a Python program that reads a sentence from the user and counts the frequency of each word in the sentence. The program should store the word frequencies in a dictionary, where the words are the keys and the frequencies are the values.

```
def count_word_frequencies(sentence):
   # Convert the sentence to lowercase and split into words
   words = sentence.lower().split()

   # Create an empty dictionary to store word frequencies
   frequencies = {}

   # Count the frequencies of each word
   for word in words:
      if word in frequencies:
         frequencies[word] += 1
      else:
         frequencies[word] = 1

   return frequencies

# Accept input from the user
sentence = input("Enter a sentence: ")

# Call the function to count word frequencies
word_frequencies = count_word_frequencies(sentence)

# Display the result
print("Word frequencies:")
for word, frequency in word_frequencies.items():
   print(word, ":", frequency)
```

OUTPUT



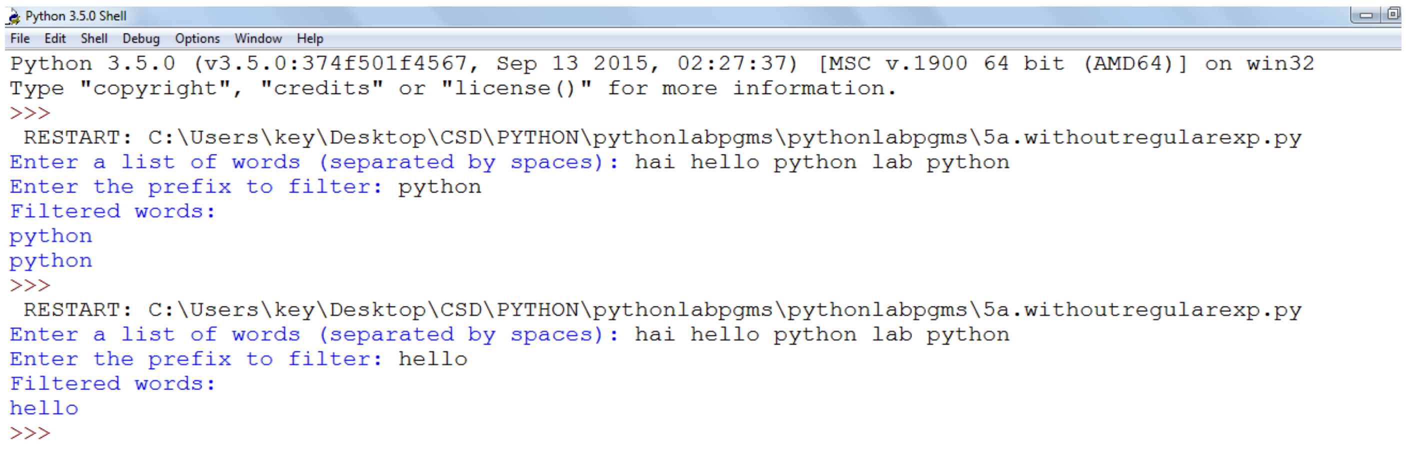
```
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\4b.dictonarydemo.py
Enter a sentence: this is python programming lab python programming
Word frequencies:
this : 1
programming : 2
python : 2
lab : 1
is : 1
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\4b.dictonarydemo.py
Enter a sentence: this is python programming lab, python programming
Word frequencies:
is : 1
programming : 2
python : 2
lab, : 1
this : 1
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\4b.dictonarydemo.py
Enter a sentence: this is python programming lab , python programming. python
Word frequencies:
programming : 1
programming. : 1
, : 1
lab : 1
is : 1
this : 1
python : 3
>>>
```

PROGRAM -5

(A). Write a Python program that takes a list of words as input from the user and filters out words that start with a specific prefix. The program should prompt the user to enter the list of words and the prefix to filter.

```
def filter_words_with_prefix(word_list, prefix):
    filtered_words = [word for word in word_list if word.startswith(prefix)]
    return filtered_words

# Accept input from the user
word_list = input("Enter a list of words (separated by spaces): ").split()
prefix = input("Enter the prefix to filter: ")

# Call the function to filter words
filtered_words = filter_words_with_prefix(word_list, prefix)

# Display the result
print("Filtered words:")
for word in filtered_words:
    print(word)
```

OUTPUT



```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\5a.withoutregularexp.py
Enter a list of words (separated by spaces): hai hello python lab python
Enter the prefix to filter: python
Filtered words:
python
python
>>>
 RESTART: C:\Users\key\Desktop\CSD\PYTHON\pythonlabpgms\pythonlabpgms\5a.withoutregularexp.py
Enter a list of words (separated by spaces): hai hello python lab python
Enter the prefix to filter: hello
Filtered words:
hello
>>>
```

PROGRAM -5

(B). Write a Python program that takes a sentence as input from the user and extracts all the email addresses present in the sentence. The program should use regular expressions to identify and extract the email addresses.

```
import re

def extract_email_addresses(sentence):
    pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b'
    email_addresses = re.findall(pattern, sentence)
    return email_addresses

# Accept input from the user
sentence = input("Enter a sentence: ")

# Call the function to extract email addresses
email_addresses = extract_email_addresses(sentence)

# Display the result
print("Email addresses found:")
for email in email_addresses:
    print(email)
```

OUTPUT

PROGRAM -6

(A). Write a Python program that reads the content of a text file named "input.txt" and counts the number of occurrences of each word in the file. Then, write the word count results to a new file named "output.txt" in the format "word: count" for each word found in the input file. Ensure that the program ignores punctuation and treats uppercase and lowercase letters as the same word.

```
import string

def count_words(input_file, output_file):
    word_count = {}

    # Read input file and count occurrences of each word
    with open(input_file, 'r') as file:
        for line in file:
            # Remove punctuation and convert to lowercase
            line = line.translate(str.maketrans('', '', string.punctuation))
            line = line.lower()

            # Split the line into words
            words = line.split()

            # Count the occurrences of each word
            for word in words:
                if word in word_count:
                    word_count[word] += 1
                else:
                    word_count[word] = 1

    # Write word count results to the output file
    with open(output_file, 'w') as file:
        for word, count in word_count.items():
            file.write(f"{word}: {count}\n")

# Usage example
input_file = "input.txt"
output_file = "output.txt"
count_words(input_file, output_file)
```

OUTPUT

PROGRAM -6

(B). Create a Python program that takes a directory path as input from the user and lists all the files and subdirectories within that directory. Additionally, the program should write the names of all the files in a separate text file named "file_list.txt".

```
import os

def list_files_and_subdirectories(directory):
    file_list = []
    for root, directories, files in os.walk(directory):
        for file in files:
            file_list.append(os.path.join(root, file))

    with open("file_list.txt", "w") as file:
        file.write("\n".join(file_list))

    print("List of files and subdirectories:")
    for item in file_list:
        print(item)

# Get directory path from the user
directory_path = input("Enter the directory path: ")

# Call the function
list_files_and_subdirectories(directory_path)
```

OUTPUT

PROGRAM -7

(A). Create a class called Rectangle with the following attributes and methods:

Attributes: length and width

Methods:

Area (): Calculate and return the area of the rectangle.

Perimeter (): Calculate and return the perimeter of the rectangle.

Create an object of the Rectangle class, initialize its attributes, and display its area and perimeter.

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)


# Creating an object of the Rectangle class
rectangle = Rectangle(4, 6)

# Displaying the area and perimeter of the rectangle
print("Area:", rectangle.area())
print("Perimeter:", rectangle.perimeter())
```

OUTPUT

PROGRAM -7

(B). Create a base class called Animal with the following attributes and methods:
Attributes: name and age
Methods:
speak(): Display a message saying "The animal speaks."
Create a derived class called Dog that inherits from the Animal class. Add a new method to the Dog class called bark(), which displays a message saying "The dog barks."

Create an object of the Dog class, initialize its attributes, and demonstrate both the speak() and bark() methods.

```
class Animal:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def speak(self):
        print("The animal speaks.")


class Dog(Animal):
    def bark(self):
        print("The dog barks.")


# Creating an object of the Dog class
dog = Dog("Max", 3)

# Demonstrating the speak() method of the Animal class
dog.speak()

# Demonstrating the bark() method of the Dog class
dog.bark()
```

OUTPUT

PROGRAM -8

(A). Create a class called Shape with a method calculate_area(). The Shape class should be the base class for three other classes: Rectangle, Triangle, and Circle. Each subclass should have its own implementation of the calculate_area() method to calculate and return the area specific to that shape. Demonstrate polymorphism by creating objects of each class and calling the calculate_area() method.

```
import math

class Shape:
    def calculate_area(self):
        pass

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def calculate_area(self):
        return self.length * self.width

class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height

    def calculate_area(self):
        return 0.5 * self.base * self.height

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        return math.pi * self.radius ** 2

# Demonstrate polymorphism
shapes = [Rectangle(4, 5), Triangle(3, 6), Circle(2)]
for shape in shapes:
    print("Area:", shape.calculate_area())
```

OUTPUT

PROGRAM -9

(A). Excel Spreadsheet Manipulation:

Write a Python program that reads data from an Excel spreadsheet and performs the following tasks:
Open the "data.xlsx" file and read the data from the "Sheet1".
Calculate the average of the numbers in column A and print it.
Find the maximum value in column B and print it.
Create a new Excel file named "output.xlsx" and write the calculated average and maximum value in separate cells in the "Sheet1".

```
import pandas as pd

# Read data from the Excel spreadsheet
df = pd.read_excel('data.xlsx', sheet_name='Sheet1')

# Calculate the average of column A
average = df['A'].mean()
print('Average of column A:', average)

# Find the maximum value in column B
maximum = df['B'].max()
print('Maximum value in column B:', maximum)

# Create a new Excel file and write the results
output_df = pd.DataFrame({'Average': [average], 'Maximum': [maximum]})
output_df.to_excel('output.xlsx', sheet_name='Sheet1', index=False)

print('Results written to output.xlsx')
```

OUTPUT

PROGRAM -9

(B). Web Scraping with BeautifulSoup:
Write a Python program that performs web scraping using BeautifulSoup library to extract data from a webpage. Follow these steps:
Use the requests library to send a GET request to the URL: "https://www.example.com".
Parse the HTML content of the response using BeautifulSoup.
Extract all the <h2> tags from the HTML and print their text.
Find all the links (<a> tags) in the HTML and print their href attributes.
Extract the content of the first paragraph (<p> tag) and print it.

```
import requests
from bs4 import BeautifulSoup

# Send GET request to the URL
url = "https://www.example.com"
response = requests.get(url)

# Parse the HTML content
soup = BeautifulSoup(response.content, 'html.parser')

# Extract and print all the <h2> tags
h2_tags = soup.find_all('h2')
for h2 in h2_tags:
    print(h2.text)

# Find and print the href attributes of all the links
links = soup.find_all('a')
for link in links:
    print(link.get('href'))

# Extract and print the content of the first paragraph
first_paragraph = soup.find('p').text
print("First paragraph:", first_paragraph)
```

OUTPUT