Guru Nanak Dev Engineering college , Ludhiana

Practical file of python

Subject Code-LPCIT-105

SUBMITTED To :

Pf. Rupinder Kaur

SUBMITTED By :

ISHA ARORA

URN-2004930

CRN-2021051

D2-ITA2

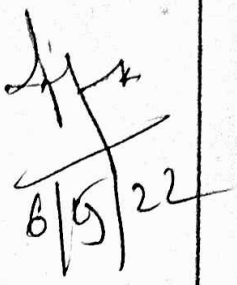
S.NO	Name of the Experiment	DATE OF EXP.	Signature
	Exercises	18-2-22	
1.	Datatypes, operators, expressions.	25-2-22	
	Program 1.1- Program to calculate the momentum of an object using its mass and velocity		
	Program 1.2- Program to calculate diameter, circumference, surface area and volume of a sphere using radius.		good 4/3/2022
	Program 1.3- Program to calculate no. of minutes in a year.		
2.	LOOPS and selection statements.	4-3-22	
	Program 2.1- TO indicate whether a triangle is an equilateral triangle or not		
	Program 2.2- TO calculate the total distance travelled by the bouncing of a ball.		
	Program 2.3 TO display a prediction of the total population.		

3.	Strings and Text files:- Program 3.1 - Program to write a script that inputs a line of plain text and a distance value and outputs an encrypted text using a ceasar cipher Program 3.2 - A script that inputs a line of encrypted text and a distance value & outputs plain text using ceasar cipher. Program 3.3 - TO write the left shift bit scripts and Right shift bit script.	11-3-22	
4.	Lists and Dictionaries:- Program 4.1 - Program to navigate the lines of text in a file. Program 4.2 - TO define a funition decimal to Rep that returns the representation of an integer in a given base Program 4.3 - To write a program that prints the unique words in the file in alphabetical order.	22-3-22	

5.	Design with functions :- Program 5.1 - A list is sorted in ascending order if it is empty or each item except the last one is less than or equal to its succession. Define a predicate isSorted that expects a list as an argument and returns True if the list is sorted, or returns False otherwise Program 5.2 - Add a command to this chapter's case study program that allows the user to view the contents of a file in the current working directory. Program 5.3 - write a recursive function that expects a pathname if pathname refers to a file, its name is displayed otherwise directory	25-3-22	
6	Graphical user interface :- Program-6.1 - write a GUI based program that implements the tax calculator program Program 6-2 - A GUI program that allows the user to open, edit and save text files Program 6.3 - A GUI program that implements an image browser for your computer's file system.	30-3-22	

7. Design with classes:-

4-4-22



Program 7.1 →

Adds 3 methods to the student class that compare two student objects. One method to check for quality, other for less than and third for greater than or equal to.

Program 7.2 →

Place several student objects into a list and shuffle it. Then run the sort method with this list & display all the students.

Program 7.3 →

To write an ATM program that allows a user an indefinte number of attempts to log in.

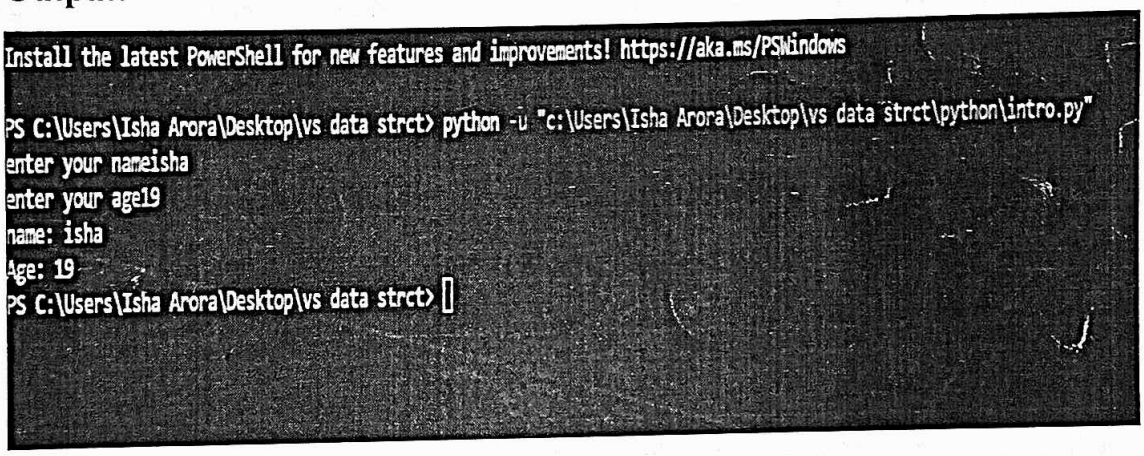
Exercise:

- **Write a program to print your name and age:**

Input code-

```
name = input ("enter your name") name = input("enter your name")
age = input ("enter your age")
print ("name: { }\n Age: {}".format(name, age))
age = input ("enter your age")
print ("name: {}\n Age: {}".format(name, age))
```

Output:



- **Write a program to print sum of two numbers:**

Input code-

```
n1 = int (input ('Enter first number : '))
n2 = int (input ('Enter second number : '))
sum = n1 + n2
print ('Sum =', sum)
```

Output:

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Isha Arora\Desktop\vs data strct> python -u "c:\Users\Isha Arora\Desktop\vs data strct\python\intro.py"
area of triangle:28.0
PS C:\Users\Isha Arora\Desktop\vs data strct> python -u "c:\Users\Isha Arora\Desktop\vs data strct\python\intro.py"
Enter first number : 76
Enter second number : 89
Sum = 165
PS C:\Users\Isha Arora\Desktop\vs data strct>
```

- **Write a program to calculate the area of rectangle:**

Input code-

```
length=3
breadth=7
area=length*breadth
print ("length= {0} , breadth={1}".format(length ,breadth))
print ("area of rectangle={}".format(area))
```

Output:

```
PROBLEMS (3)   OUTPUT   DEBUG CONSOLE   TERMINAL

le.py"
length=3 , breadth=7
area of rectangle=21
PS C:\Users\Isha Arora\Desktop\vs data strct>
```

- **Write a program to calculate area of a traingle:**

Input code-

```
base=8
```

```
height=7
area=0.5*base*height
print("base=" , base)
print("height=" , height)
print("area of triangle:{}".format(area))
```

Output:

```
PS C:\Users\Isha Arora\Desktop\vs data strct> python -u "c:\Users\Isha Arora\Desktop\vs data strct\python\intro.py"
area of triangle
base= 8
height= 7
area of triangle:28.0
PS C:\Users\Isha Arora\Desktop\vs data strct>
```

- **Write a program to calculate area of a circle:**

Input code-

```
PI=3.14
r=float (input ("enter the radius of a circle:"))
area=PI*r*r
print ("area of circle=%.2f " %area)
```

Output:

```
PS C:\Users\Isha Arora\Desktop\vs data strct> python -u "c:\Users\Isha Arora\Desktop\vs data strct\python\intro.py"
enter the radius of a circle:6
area of circle=113.04
PS C:\Users\Isha Arora\Desktop\vs data strct>
```

Program-1.1

Input code/ Output:

```
#program:momentum.py
#project:1.1
"""Question:An object's momentum is its mass multiplied by its velocity. Write a program that
accepts an object's mass (in kilograms) and velocity (in meters per second) as inputs and
then outputs its momentum"""
"""
The inputs are : mass(m) in kg and velocity(v) in m/s of an object
Computations are : m *v
The output is: momentum(M) of an object

"""

#Taking mass and velocity of an object:
m=int (input ("enter mass of an object:"))
v=int (input ("enter velocity of an object: "))

# computations to calculate momentum
M=m*v

#printing result
Print ("momentum of an object is:", M ,"m/s")
```

```
surface area(S)
volume(V)

"""

#defining constants
pi=3.14

#taking radius as input
r=float(input("enter the radius of sphere in meter(m):"))

#calculating diameter of sphere
d=2*r

#calculating circumference
c=2* pi *r

#calculating surface area
S=4*pi*r**2

#calculating volume of sphere
V=(4/3)*pi*r**3

#Output:
print("diameter of sphere:",d,"m")
print("circumference of sphere:",c,"m")
print("surface area of sphere:",round(S,2),"m^2")
print("volume of sphere:",round(V,3),"m^3")
```

OUTPUT:

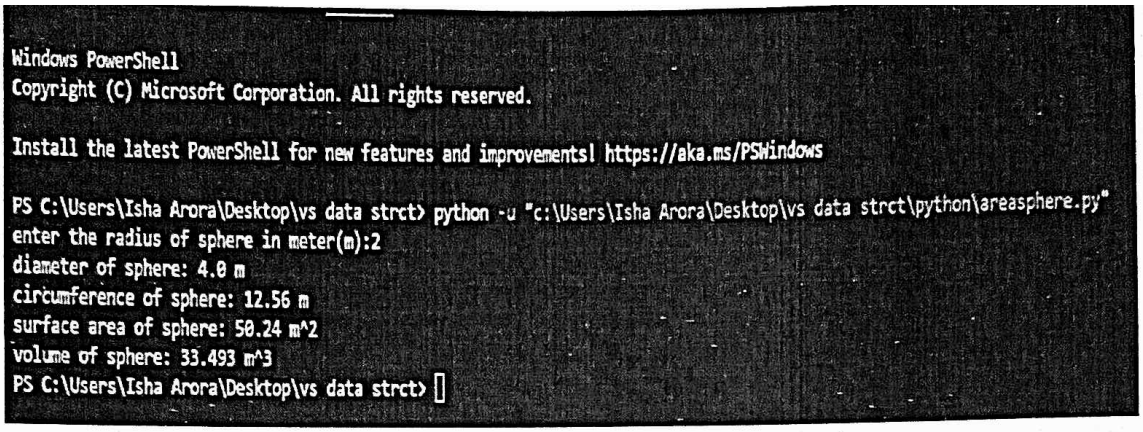
OUTPUT:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Isha Arora\Desktop\vs data strct> python -u "c:\Users\Isha Arora\Desktop\vs data strct\python\momentum.py"
enter mass of an object:40
enter velocity of an object: 4
momentum of an object is: 160 m/s
PS C:\Users\Isha Arora\Desktop\vs data strct>
```

Program-1.2

Input code/ Output:

```
#program:sphere.py
#project:1.2
#Question:
#Write a program that takes the radius of a sphere (a floating-point number) as input and
#then outputs the spheres diameter, circumference, surface area, and volume.

"""
#Input: Radius of sphere(r).

computations: diameter(d)=2r
circumference(C)=2pi *r
surface area(S)=4pi r**2
volume(V)=4/3pi r**3

output: Circumference(c)
diameter(d)
```



Program-1.3

Input code/ Output:

#program:sphere.py

#project:1.3

#Question:

#Write a program that calculates and prints the number of minutes in a year"

"""

Computations:

1 hour = 60 min

1 day = 24 hrs

1 day=24*60 min

1 year= 365 days

365 days=365*24*60 min

Output:

no. of minutes in a hour

no. of minutes in a day

no. of minutes in a year

"""

#calculating no. of minutes in a day

minutes=60

hours=24

dayhrs=minutes*hours

print("no. of minutes in a day=",dayhrs)

#calculating no. of minutes in a year

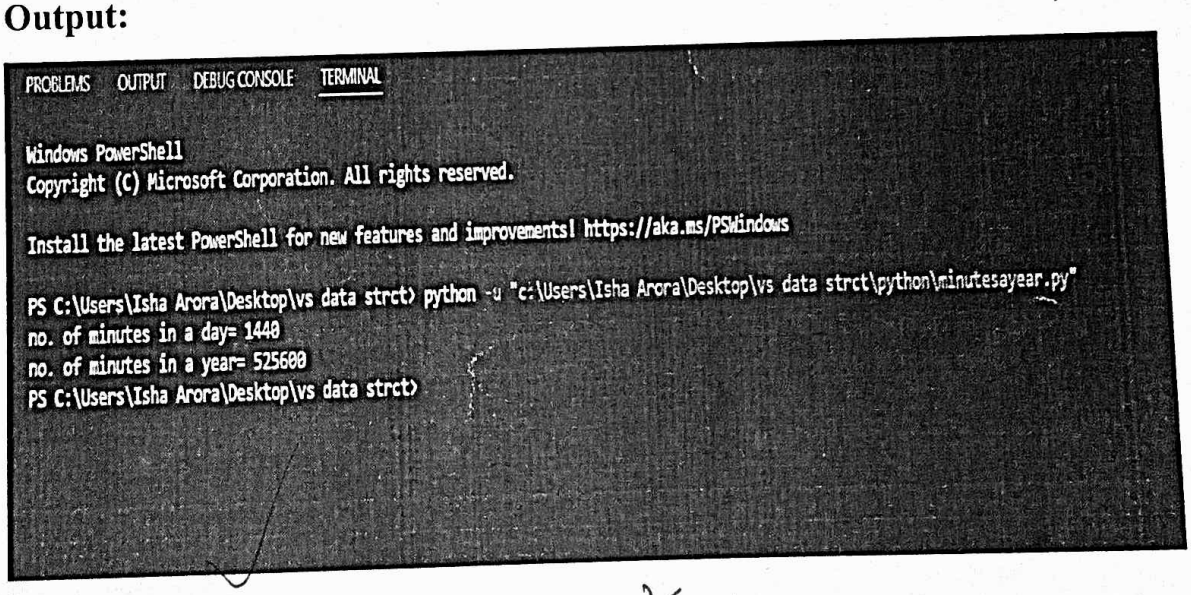
yeardays=365

yearmin=dayhrs*yeardays

#output

print("no. of minutes in a year=",yearmin)

Output:



14/3/2022

Program-2.1

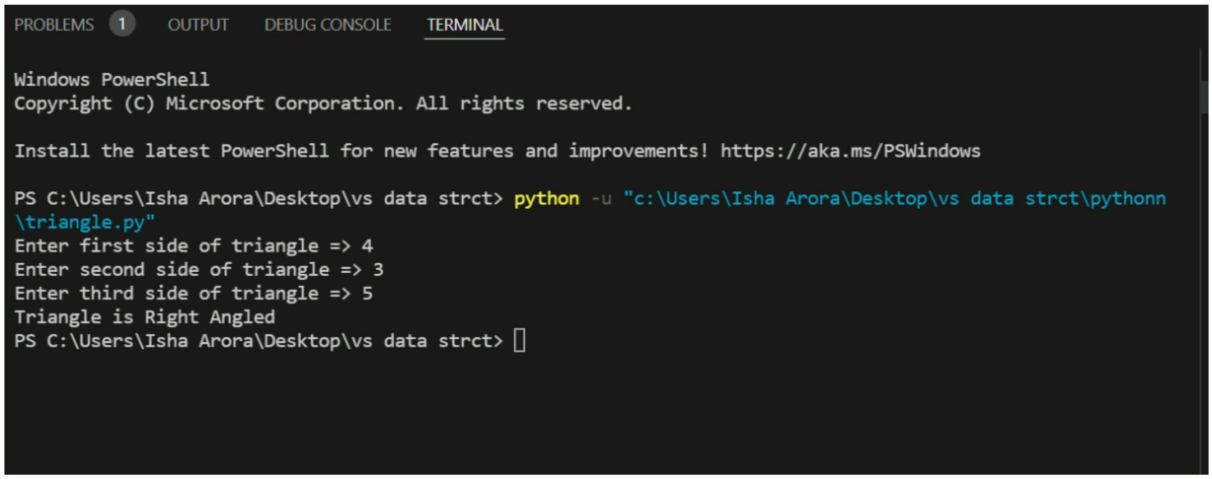
Input code:

```
#project:2.1

#Question:
#Write a program that should indicate whether or not the triangle is a right triangle.

"""
Input:

sides of traingle:
side 1,
side 2,
side 3,

Computations:

compute longest side of a triangle:
side1>side2 and side1>side3  => side 1 is greatest
similarly for side 2 and side 3

checking the condition for right triangle:
if side 1 is greatest=>
side2 **2 + side3**= side1**2(largest side)
if this is true than the triangle is right traingle

Output:

this triangle is  a right angled triangle
```

```
or
this is not a right angled triangle
"""

#taking input:
side1 = float(input("Enter first side of triangle => "))
side2 = float(input("Enter second side of triangle => "))
side3 = float(input("Enter third side of triangle => "))

#compute longest side of a triangle:

if (side1>side2) and (side1>side3):
  largest_side=side1

elif (side2>side1) and (side2>side3):
  largest_side=side2
else:
    largest_side=side3

# Applying Pythagorean theorem to check if triangle is Right Angled
# If side1 is largest side of triangle
if (largest_side == side1):
    if (side2**2 + side3**2 == side1**2): print("Triangle is Right Angled")
    else: print("Triangle is Not Right Angled")
# If side2 is largest side of triangle
if(largest_side == side2):
    if(side1**2 + side3**2 == side2**2):
        print("Triangle is Right Angled")
    else:
        print("Triangle is Not Right Angled")
```

```
# If side3 is largest side of triangle
if(largest_side == side3):
    if(side1**2 + side2**2 == side3**2):
        print("Triangle is Right Angled")
    else:
        print("Triangle is Not Right Angled")
```

Output:



Program-2.2

Input code:

```
"""
project:2.2
```

Question:

A local biologist needs a program to predict population growth. The inputs would be the initial number of organisms, the rate of growth (a real number greater than 0), the number of hours it takes to achieve this rate, and a number of hours during which the population grows. For example, one might start with a population of 500 organisms, a growth rate of 2, and a growth period to achieve this rate of 6 hours. Assuming that none of the

organisms die, this would imply that this population would double in size every 6 hours. Thus, after allowing 6 hours for growth, we would have 1000 organisms, and after 12 hours, we would have 2000 organisms. Write a program that takes these inputs and displays a prediction of the total population

Input:

initial number of organisms =population
the rate of growth =rate
the number of hours it takes to achieve this rate = period
number of hours during which the population grows = total_hours

Computations:
population *=rate
hours +=period

Output:
Total population = population

"""
#taking input of initial organisms ,
population=int(input("enter the initial no. of organisms"))

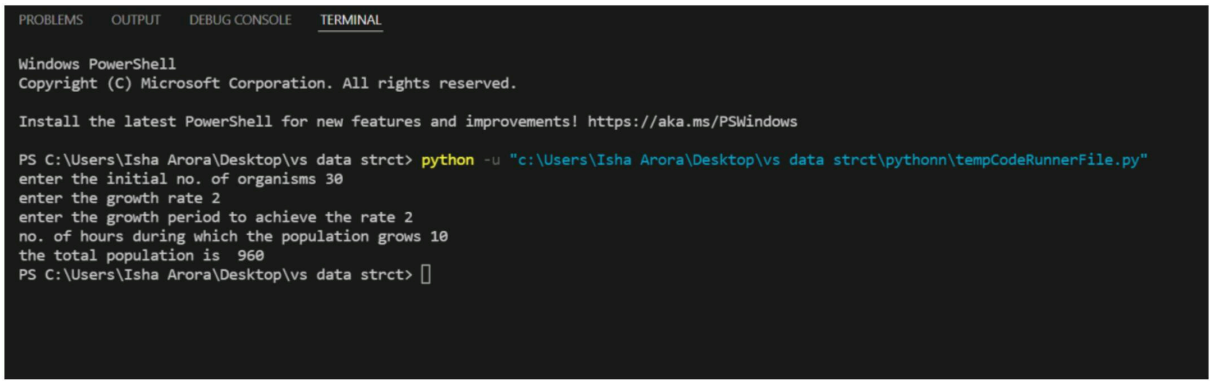
#input rate of growth
rate=int(input("enter the growth rate"))

#input the no. of hours to achieve this rate of growth
period=int(input("enter the growth period to achieve the rate"))

#enter no. of hours during which the population grows

```
total_hours=int(input("no. of hours during which the population grows"))

#initialising hours
hours=0

#the population should grow to the desired no. of hours
while hours <total_hours:
    #population will get increase by the no. of times as rate
    population *=rate
    #hours will increase by the period required to achieve the rate
    hours += period

#output:
print("the total population is " , (int(population)))
```

Output:



Program-2.3

Input code:

```
#Question:
"""
```

A standard science experiment is to drop a ball and see how high it bounces. Once the "bounciness" of the ball has been determined, the ratio gives a bounciness index.Write a program that lets the

user enter the initial height from which the ball is dropped and the number of times the ball is allowed to continue bouncing. Output should be the total distance traveled by the ball.

Input:

height from which the ball is dropped: height

Enter the bounciness index of the ball: index

Enter the number of times the ball is allowed to continue bouncing: times

Total distance traveled is: distance

Computations:

height*index

distance=distance +height

Output:

total distance travelled

"""

#taking inputs:

height=float(input("enter the height from which the ball is dropped"))

index=float(input("enter the bounciness index of the ball"))

times=int(input("enter the no. of times the ball is allowed to continue bouncing"))

#initialising distance value to zero

distance =0

```
# condition for no. of time sthe ball should bounce
while times>0:
    #calculating the new height

    height = height *index

    # calculating the distance before bouncing
    distance = distance + height

    #calculating the distance after bouncing
    distance= distance +height

    #decrementing the bounces
    times-=1

#displaying the result
print("total distance travelled =", distance)
```

Output:

Practical No.-3

Program-3.1

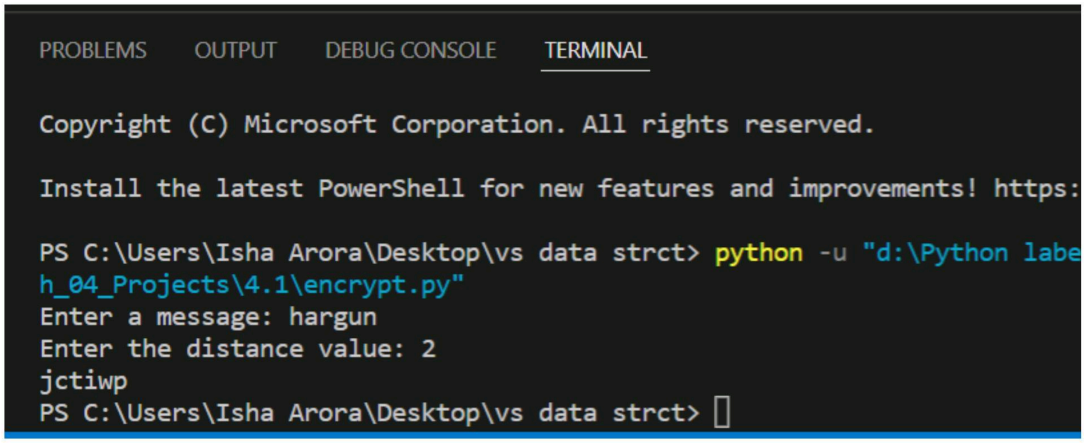
Question-:

Write a script that inputs a line of plaintext and a distance value and outputs an encrypted text using a Caesar cipher. The script should work for any printable characters.

Input:

```
"""
File: encrypt.py
Encypts an input string of the ASCII characters and prints
the result.  The other input is the distance value.
"""
# The ASCII values range from 0 through 127
plainText = input("Enter a message: ")
distance = int(input("Enter the distance value: "))
code = ""
for ch in plainText:
    ordValue = ord(ch)
    cipherValue = ordValue + distance
    if cipherValue > 127:
        cipherValue = distance - (127 - ordValue + 1)
    code +=  chr(cipherValue)
print(code)
```

Output:



Program-3.2

Question-:

Write a script that inputs a line of encrypted text and a distance value and outputs plaintext using a Caesar cipher. The script should work for any printable characters.

Input:

"""

File: decrypt.py

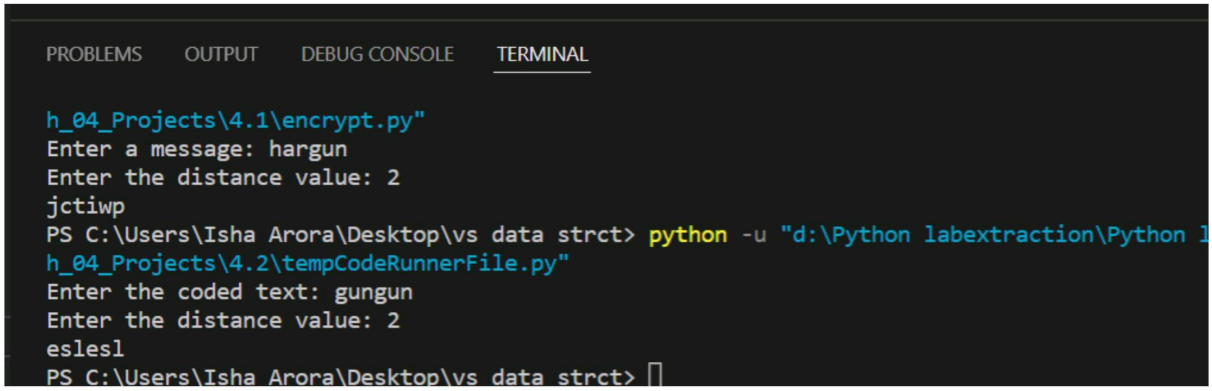
Decypts an input string characters and prints

the result. The other input is the distance value.

"""

```
# The ASCII values range from 0 through 127


code = input("Enter the coded text: ")
distance = int(input("Enter the distance value: "))
plainText = ''
for ch in code:
    ordValue = ord(ch)
    cipherValue = ordValue - distance
    if cipherValue < 0:
        cipherValue = 127 - \
                    (distance - (1 - ordValue))
    plainText +=  chr(cipherValue)
print(plainText)
```

Output:



Program-3.3

Question-:

A bit shift is a procedure whereby the bits in a bit string are moved to the left or to the right. For example, we can shift the bits in the string 1011 two places to the left to produce the string 1110. Note that the leftmost two bits are wrapped around to the right side of the string in this operation. Define two scripts, shiftLeft.py and shiftRight.py, that expect a bit string as an input. The script shift Left shifts the bits in its input one place to the left, wrapping the leftmost bit to the rightmost position. The script performs the inverse operation. Each script prints the resulting string.

Input:

"""

File: shiftleft.py

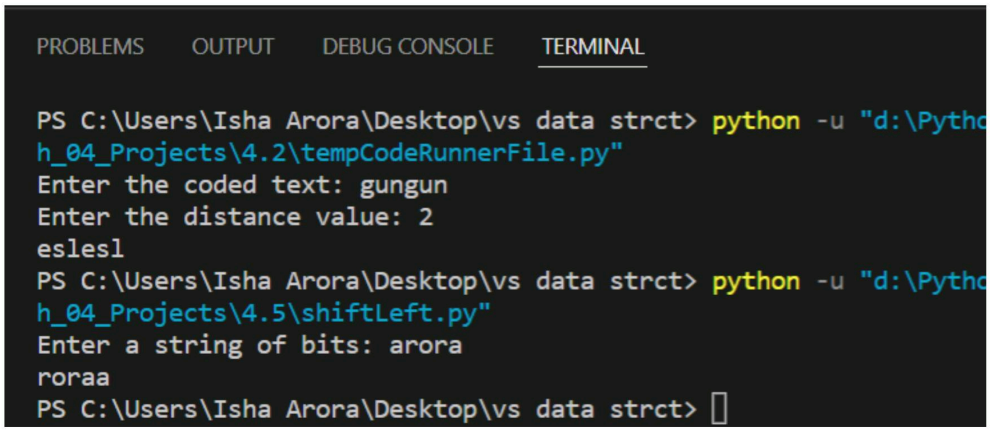
Shifts the bits in an input string one place to the left.

The leftmost bit wraps around to the rightmost position.

"""

```
bits = input("Enter a string of bits: ")
if len(bits) > 1:
    bits = bits[1:] + bits[0]
print(bits)
```

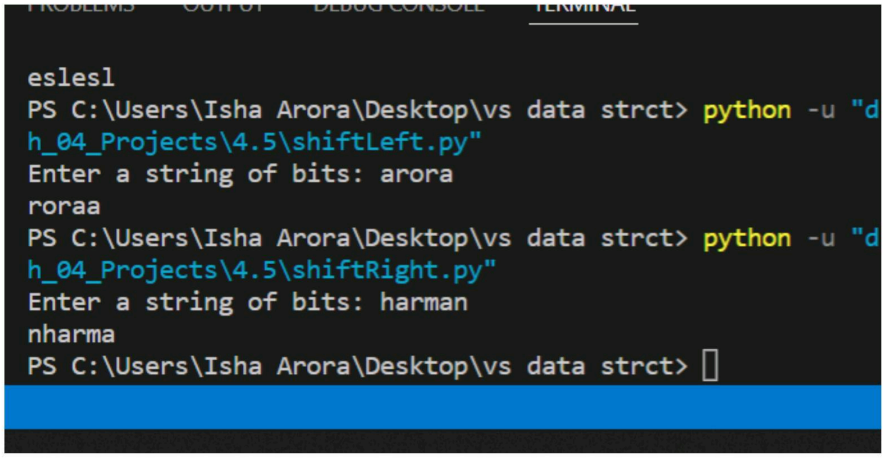
Output:



Input Code:

```
"""
File: shiftright.py
Shifts the bits in an input string one place to the right.
The rightmost bit wraps around to the leftmost position.
"""

bits = input("Enter a string of bits: ")
if len(bits) > 1:
    bits = bits[-1] + bits[:-1]
print(bits)
```

Output:



Practical No.-4

Program - 4.1

Question-:

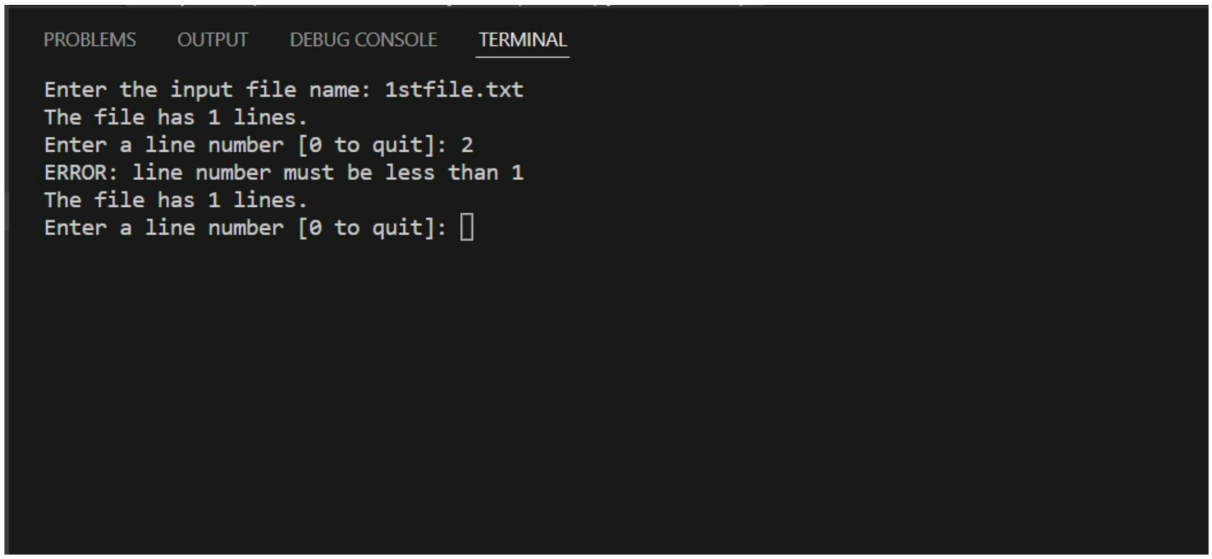
Write a program that allows the user to navigate the lines of text in a file. The program should prompt the user for a filename and input the lines of text into a list. The program then

enters a loop in which it prints the number of lines in the file and prompts the user for a line number. Actual line numbers range from 1 to the number of lines in the file. If the input is 0, the program quits. Otherwise, the program prints the line associated with that number.

INPUT:

```
"""
File: navigate.py
Allows the user to navigate to any line in a text file.
"""


# Take the input file name
inName = input("Enter the input file name: ")


# Open the input file and read the text
inputFile = open(inName, 'r')
lines = []
for line in inputFile:
    lines.append(line)


# Loop for line numbers from the user until she enters 0
# and prints the line's number followed by the line
while True:
    print("The file has", len(lines), "lines.")
    if len(lines) == 0:
        break
    lineNumber = int(input("Enter a line number [0 to quit]: "))
    if lineNumber == 0:
        break
    elif lineNumber >= len(lines):
        print("ERROR: line number must be less than", len(lines))
    else:
        print(lineNumber, ": ", lines[lineNumber])
```

Output:



Program -4.2

Question:

Define a function decimal to Rep that returns the representation of an integer in a given base. The two arguments should be the integer and the base. The function should return a string. It should use a lookup table that associates integers with digits. Include a main function that tests the conversion function with numbers in several bases

Input:

```
"""
File: convert.py
Defines a function repToDecimal that converts
a number in any base to decimal.
"""


# Table of digits in bases 2-16 with integer values
repTable = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4,
            '5': 5, '6':6, '7': 7, '8' : 8, '9' : 9,
            'A' : 10, 'B' : 11, 'C' : 12, 'D' : 13,
            'E' : 14, 'F' : 15}
```

```
def repToDecimal(rep, base):
    """Converts the string rep of a number in base
    to decimal and returns the decimal as an int."""
    decimal = 0
    exp = len(rep) - 1
    for digit in rep:
        decimal += repTable[digit] * base ** exp
        exp -= 1
    return decimal


def main():
    """Tests the function."""
    print(repToDecimal('10', 10))
    print(repToDecimal('10', 8))
    print(repToDecimal('10', 2))
    print(repToDecimal('10', 16))


main()
```

Output:

Program -4.3

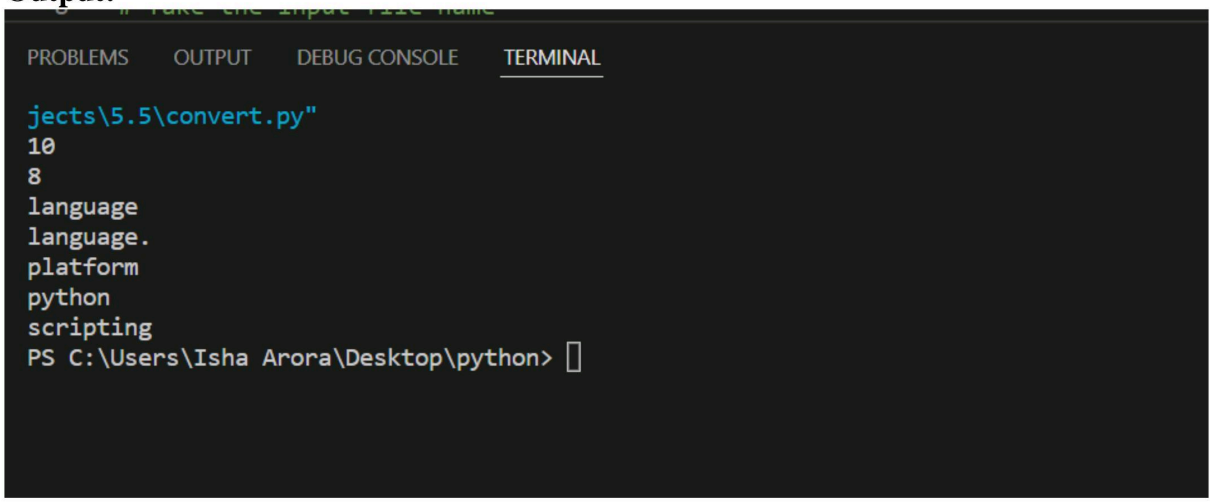
Question:

Write a program that inputs a text file. The program should print the unique words in the file in alphabetical order.

Input:

```
"""
File: unique.py

Prints the unique words in a text file in alphabetical order.
"""

# Take the input file name
inName = input("Enter the input file name: ")

# Open the input file and initialize list of unique words
inputFile = open(inName, 'r')
uniqueWords = []

# Add the unique words in the file to the list
for line in inputFile:
    words = line.split()
    for word in words:
        if not word in uniqueWords:
            uniqueWords.append(word)
uniqueWords.sort()

# Prints the unique words
for word in uniqueWords:
    print(word)
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
jects\5.5\convert.py"
10
8
language
language.
platform
python
scripting
PS C:\Users\Isha Arora\Desktop\python>

Practical No.-5

Program -5.1

Question:

A list is sorted in ascending order if it is empty or each item except the last one is less than or equal to its successor. Define a predicate is Sorted that expects a list as an argument and returns True if the list is sorted, or returns False otherwise.

Input:

"""

File: testsort.py

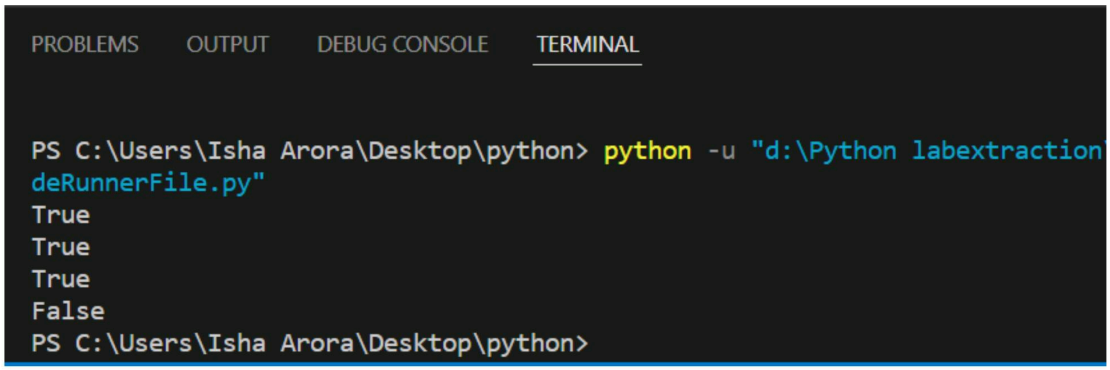
Defines a predicate to test lists for being sorted.

"""

```
def isSorted(lyst):
    """Returns True if lyst is sorted in ascending
    order or False otherwise."""
    if len(lyst) == 0 or len(lyst) == 1:
        return True
    else:
        for index in range(len(lyst) - 1):
            if lyst[index] > lyst[index + 1]:
                return False
    return True

def main():
    lyst = []
    print(isSorted(lyst))
    lyst = [1]
    print(isSorted(lyst))
```

```
    lyst = list(range(10))
    print(isSorted(lyst))
    lyst[9] = 3
    print(isSorted(lyst))

main()
```

Output:



Program -5.2

Question:

Add a command to this chapter's case study program that allows the user to view the contents of a file in the current working directory. When the command is selected, the program should display a list of filenames and a prompt for the name of the file to be viewed. Be sure to include error recovery

Input:

```
"""
File: filesys.py

Provides a menu-driven tool for navigating a file system
and gathering information on files.

Adds a command to view a file's contents.
"""

import os, os.path
```

```
QUIT = '8'

COMMANDS = ('1', '2', '3', '4', '5', '6', '7', '8')

MENU = """1   List the current directory
2   Move up
3   Move down
4   Number of files in the directory
5   Size of the directory in bytes
6   Search for a file name
7   View the contents of a file
8   Quit the program"""

def main():
    while True:
        print(os.getcwd())
        print(MENU)
        command = acceptCommand()
        runCommand(command)
        if command == QUIT:
            print("Have a nice day!")
            break

def acceptCommand():
    """Inputs and returns a legitimate command number."""
    while True:
        command = input("Enter a number: ")
        if not command in COMMANDS:
            print("Error: command not recognized")
        else:
            return command
```

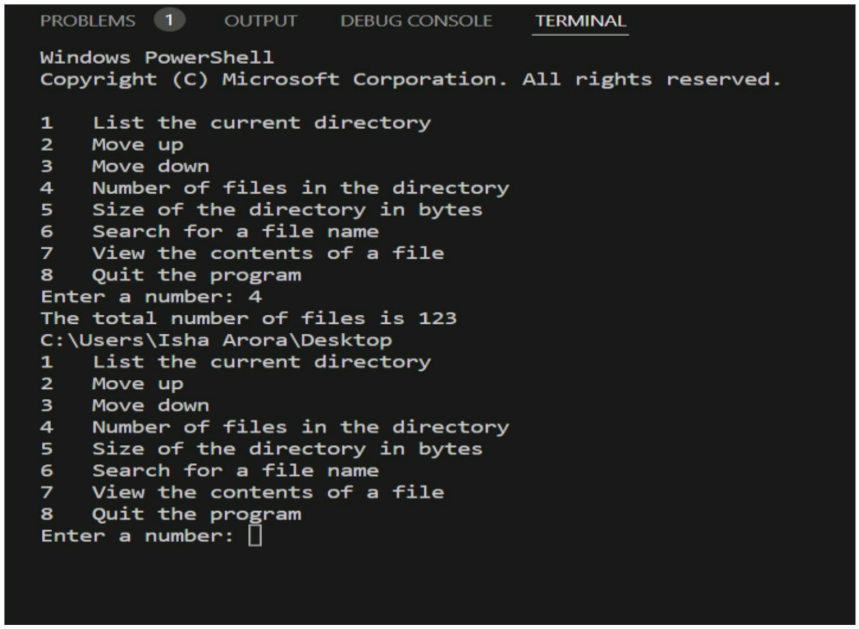
```
def runCommand(command):
    """Selects and runs a command."""
    if command == '1':
        listCurrentDir(os.getcwd())
    elif command == '2':
        moveUp()
    elif command == '3':
        moveDown(os.getcwd())
    elif command == '4':
        print("The total number of files is", \
              countFiles(os.getcwd()))
    elif command == '5':
        print("The total number of bytes is", \
              countBytes(os.getcwd()))
    elif command == '6':
        target = raw_input("Enter the search string: ")
        fileList = findFiles(target, os.getcwd())
        if not fileList:
            print("String not found")
        else:
            for f in fileList:
                print(f)
    elif command == '7':
        viewFile(os.getcwd())


def viewFile(dirName):
    lyst = list(filter(os.path.isfile, os.listdir(dirName)))
    if len(lyst) == 0:
        print("There are no files in this directory")
    else:
        while True:
```

```
            print("Files in " + dirName + ":")
            for element in lyst: print(element)
            fileName = input("Enter a file name from these names: ")
            if not fileName in lyst:
                print("Sorry, there is an error in your file name.")
            else:
                f = open(fileName, 'r')
                print(f.read())
                break

def listCurrentDir(dirName):
    """Prints a list of the cwd's contents."""
    lyst = os.listdir(dirName)
    for element in lyst: print(element)

def moveUp():
    """Moves up to the parent directory."""
    os.chdir("..")

def moveDown(currentDir):
    """Moves down to the named subdirectory if it exists."""
    newDir = input("Enter the directory name: ")
    if os.path.exists(currentDir + os.sep + newDir) and \
       os.path.isdir(newDir):
        os.chdir(newDir)
    else:
        print("ERROR: no such name")

def countFiles(path):
    """Returns the number of files in the cwd and
    all its subdirectories."""
    count = 0
```

```
    lyst = os.listdir(path)
    for element in lyst:
        if os.path.isfile(element):
            count += 1
        else:
            os.chdir(element)
            count += countFiles(os.getcwd())
            os.chdir("..")
    return count

def countBytes(path):
    """Returns the number of bytes in the cwd and
    all its subdirectories."""
    count = 0
    lyst = os.listdir(path)
    for element in lyst:
        if os.path.isfile(element):
            count += os.path.getsize(element)
        else:
            os.chdir(element)
            count += countBytes(os.getcwd())
            os.chdir("..")
    return count

def findFiles(target, path):
    """Returns a list of the file names that contain
    the target string in the cwd and all its subdirectories."""
    files = []
    lyst = os.listdir(path)
    for element in lyst:
        if os.path.isfile(element):
            if target in element:
```

```
            files.append(path + os.sep + element)
        else:
            os.chdir(element)
            files.extend(findFiles(target, os.getcwd()))
            os.chdir("..")
    return files


main()
```

Output:



Program -5.3

Question:

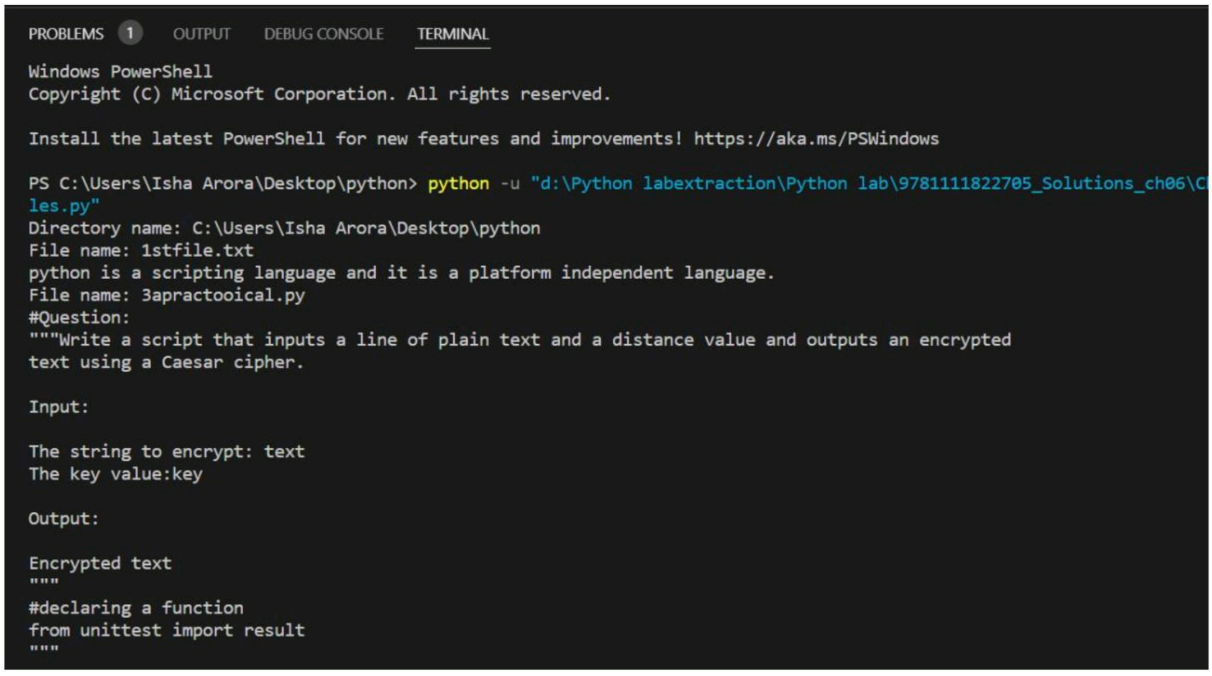
Write a recursive function that expects a pathname as an argument. The pathname can be either the name of a file or the name of a directory. If the pathname refers to a file, its name is displayed, followed by its contents. Otherwise, if the pathname refers to a directory, the function is applied to each name in the directory. Test this function in a new program

Input:

```
"""
File: viewfiles.py
Allows the user to visit all of the files in the current path and view them.
"""
import os, os.path
def main():
```

```
displayFiles(os.getcwd())
    def displayFiles(path):
    """Visits all of the files and directories in
    path and displays the files' contents."""
    if os.path.isfile(path):
        print("File name: " + path)
        f = open(path, 'r')
        print(f.read())
    else:
        print("Directory name: " + path)
        lyst = os.listdir(path)
        for element in lyst: displayFiles(element)
main()
```

Output:



Practical no.- 6

Program=6.1

Question:

Write GUI-based program that implements the tax calculator program

Input:

```
"""
File: calculator.py

Simulates a calculator with +, -, *, /, =, and C.
"""

from tkinter import *

class Calculator(Frame):

    def __init__(self):
        """Set up the window and widgets."""
        Frame.__init__(self)
        self.master.title("Calculator")
        self.grid()
        self._dataVar = StringVar()
        self._dataEntry = Entry(self,
                                textvariable = self._dataVar)
        self._dataEntry.grid(row = 0, column = 0,
                             columnspan = 4)
        self._sevenBtn = Button(self, text = "7",
                                command = self._seven)
        self._sevenBtn.grid(row = 1, column = 0)
        self._eightBtn = Button(self, text = "8",
                                command = self._eight)
```

```
self._eightBtn.grid(row = 1, column = 1)
self._nineBtn = Button(self, text = "9",
                       command = self._nine)
self._nineBtn.grid(row = 1, column = 2)
self._divBtn = Button(self, text = "/",
                      command = self._div)
self._divBtn.grid(row = 1, column = 3)
self._fourBtn = Button(self, text = "4",
                       command = self._four)
self._fourBtn.grid(row = 2, column = 0)
self._fiveBtn = Button(self, text = "5",
                       command = self._five)
self._fiveBtn.grid(row = 2, column = 1)
self._sixBtn = Button(self, text = "6",
                      command = self._six)
self._sixBtn.grid(row = 2, column = 2)
self._mulBtn = Button(self, text = "*",
                      command = self._mul)
self._mulBtn.grid(row = 2, column = 3)
self._oneBtn = Button(self, text = "1",
                      command = self._one)
self._oneBtn.grid(row = 3, column = 0)
self._twoBtn = Button(self, text = "2",
                      command = self._two)
self._twoBtn.grid(row = 3, column = 1)
self._threeBtn = Button(self, text = "3",
                        command = self._three)
self._threeBtn.grid(row = 3, column = 2)
self._subBtn = Button(self, text = "-",
                      command = self._sub)
```

```
        self._subBtn.grid(row = 3, column = 3)
        self._zeroBtn = Button(self, text = "0",
                               command = self._zero)
        self._zeroBtn.grid(row = 4, column = 0)
        self._equalsBtn = Button(self, text = "=",
                               command = self._equals)
        self._equalsBtn.grid(row = 4, column = 1)
        self._clearBtn = Button(self, text = "C",
                               command = self._clear)
        self._clearBtn.grid(row = 4, column = 2)
        self._addBtn = Button(self, text = "+",
                               command = self._add)
        self._addBtn.grid(row = 4, column = 3)
        self._clear()

    def _zero(self):
        self._insert("0")

    def _one(self):
        self._insert("1")

    def _two(self):
        self._insert("2")

    def _three(self):
        self._insert("3")

    def _four(self):
        self._insert("4")
```

```
def _five(self):
    self._insert("5")

def _six(self):
    self._insert("6")

def _seven(self):
    self._insert("7")

def _eight(self):
    self._insert("8")

def _nine(self):
    self._insert("9")

def _add(self):
    self._setOperator("+")

def _sub(self):
    self._setOperator("-")

def _mul(self):
    self._setOperator("*")

def _div(self):
    self._setOperator("/")

def _equals(self):
    self._calculate()
```

```
def _clear(self):
    """Restores the calculator to its startup state."""
    self._number1 = 0
    self._number2 = None
    self._operator = None
    self._dataVar.set(str(self._number1))

def _insert(self, s):
    """Updates the first number or the second number with
    the next digit and then updates the field."""
    if self._number2 == None:
        # Adjust number1
        if self._number1 == 0:
            # Was just 0, so replace it
            self._number1 = int(s)
        else:
            # Was not 0, so append to it
            self._number1 = int(str(self._number1) + s)
        self._dataVar.set(str(self._number1))
    else:
        # Adjust number2
        self._number2 = int(str(self._number2) + s)
        self._dataVar.set(str(self._number2))

def _setOperator(self, op):
    """Sets the operator symbol and makes the second number active."""
    self._operator = op
    self._number2 = 0

def _calculate(self):
```

```
        """Computes the result and displays it in the field.
        The result is also placed in the first number for further
        calculations."""
        # No effect if not ready for = yet
        if self._number2 is None or self._operator is None:
            return
        # Assume an error until proven OK
        result = "ERR"
        # Attempt to calculate a result
        if self._operator == '+':
            result = self._number1 + self._number2
        elif self._operator == '-':
            result = self._number1 - self._number2
        elif self._operator == '*':
            result = self._number1 * self._number2
        elif self._operator == '/' and self._number2 != 0:
            result = self._number1 / self._number2
        # Not an error, then put result in first number
        # for further calculations
        if result != "ERR":
            self._number1 = result
            self._number2 = None
            self._operator = None
        # Display result or ERR
        self._dataVar.set(str(result))
  def main():
    Calculator().mainloop()
main()
```

Output:

Program=6.2

Question:

Write a GUI-based program that allows the user to open, edit, and save text files. The GUI should include a labeled entry field for the filename and a multiline text widget for the text of the file. The user should be able to scroll through the text by manipulating a vertical scrollbar. Include command buttons labeled Open, Save, and New that allow the user to open, save, and create new files. The New command should then clear.

Input:

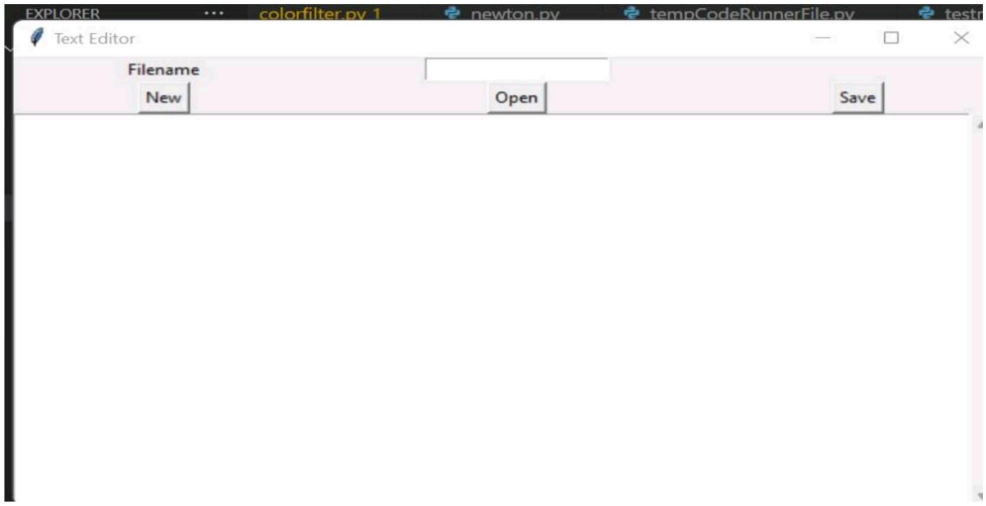
```
"""
File: editor.py


GUI for text editor program.
"""
from tkinter import *
import tkinter.messagebox
import os.path


class Editor(Frame):


    def __init__(self):
        """Set up the window and widgets."""
        Frame.__init__(self)
        self.master.title("Text Editor")
        self.grid()
        # Input fields
```

```
self._fileLabel = Label(self, text = "Filename")
self._fileLabel.grid(row = 0, column = 0)
self._fileVar = StringVar()
self._fileEntry = Entry(self,
                        textvariable = self._fileVar)
self._fileEntry.grid(row = 0, column = 1)
# Command buttons
self._newBtn = Button(self,
                      text = "New",
                      command = self._newFile)
self._newBtn.grid(row = 1, column = 0)
self._openBtn = Button(self,
                      text = "Open",
                      command = self._openFile)
self._openBtn.grid(row = 1, column = 1)
self._saveBtn = Button(self,
                      text = "Save",
                      command = self._saveFile)
self._saveBtn.grid(row = 1, column = 2)
# Frame for text box and scrollbar
self._textPane = Frame(self)
self._textPane.grid(row = 2, column = 0,
                    columnspan = 3,
                    sticky = N+S+E+W)
self._yScroll = Scrollbar(self._textPane,
                          orient = VERTICAL)
self._yScroll.grid(row = 0, column = 1,
                   sticky = N+S)
self._outputArea = Text(self._textPane,
                        width = 80,
```

```
                        height = 20,
                        yscrollcommand = self._yScroll.set)
    self._outputArea.grid(row = 0, column = 0,
                          sticky = W+E+N+S)
    self._yScroll["command"] = self._outputArea.yview

def _newFile(self):
    self._filename= ""
    self._fileVar.set("")
    self._outputArea.delete("1.0", END)

def _openFile(self):
    self._filename = self._fileVar.get()
    if os.path.exists(self._filename):
        self._file = open(self._filename, 'r')
        self._outputArea.delete("1.0", END)
        self._outputArea.insert("1.0", self._file.read())
    else:
        tkinter.messagebox.showerror(message = "File does not exist!",
                                     parent = self)

def _saveFile(self):
    self._filename = self._fileVar.get()
    if self._filename != '':
        self._file = open(self._filename, 'w')
        self._file.write(self._outputArea.get("1.0", END))
        self._file.close()
    else:
        tkinter.messagebox.showerror(message = "Must enter a file name!",
                                     parent = self)
```

```
def main():
    Editor().mainloop()


main()
```

Output:



Program No.- 6.3

Question:

Write a GUI-based program that implements an image browser for your computer's file system. The look, feel, and program logic should be like those of the simple text file bowser developed in this chapter. The file dialog should filter for GIF image files, and create and open a PhotoImage when a file is accessed

Input:

```
"""
File: imagebrowser.py
Browser for image (.gif) files.
"""from tkinter import *
import os
import os.path
class ImageBrowser(Frame):
def __init__(self):
        """Sets up the window and the widgets."""
        Frame.__init__(self)
```

```
self.master.title("Image Browser")
self.master.rowconfigure(0, weight = 1)
self.master.columnconfigure(0, weight = 1)
self.grid(sticky = W+E+N+S)

# Set up the pane and add the list box and its
# scroll bar
self._listPane = Frame(self)
self._listPane.grid(row = 0, column = 0,
                    sticky = N+S)
self._yScroll = Scrollbar(self._listPane,
                          orient = VERTICAL)
self._yScroll.grid(row = 0, column = 1,
                   sticky = N+S)
self._listBox = Listbox(self._listPane,
                        width = 20,
                        height = 10,
                        selectmode = SINGLE,
                        yscrollcommand = \
                        self._yScroll.set)
self._listBox.grid(row = 0, column = 0,
                   sticky = N+S)
self._yScroll["command"] = self._listBox.yview

# Set up the input field, image label, and button
self._pathVar = StringVar()
self._input = Entry(self,
                    textvariable = self._pathVar,
                    width = 70)
self._input.grid(row = 0, column = 1,
```

```
                    sticky = N)
        self._imageLabel = Label(self, text = "")
        self._imageLabel.grid(row = 1, column = 1)
        self._goButton = Button(self,
                                text = "Go",
                                command = self._go)
        self._goButton.grid(row = 1, column = 0)

        # Configure the list pane to expand
        self.rowconfigure(0, weight = 1)
        self._listPane.rowconfigure(0, weight = 1)

        self._getPathAndFiles()

    def _getPathAndFiles(self):
        """Sets the field and the list box with the path of the cwd
        and the directories and the .gif files in that directory,
        respectively."""
        self._path = os.getcwd()
        self._pathVar.set(self._path)
        fileList = filter(lambda f: ".gif" in f or os.path.isdir(f),
                          os.listdir(self._path))
        index = self._listBox.size() - 1
        while self._listBox.size() > 0:
            self._listBox.delete(index)
            index -= 1
        self._listBox.insert(END, "..")
        for f in fileList:
            self._listBox.insert(END, f)
        self._listBox.activate(0)
```

```
    def _go(self):
        """Moves to a directory or shows an image."""
        (index) = self._listBox.curselection()
        item = self._listBox.get(index)
        if os.path.isfile(item):
            # It's a file, so load the image
            self._image = PhotoImage(file = item)
            self._imageLabel["image"] = self._image
        else:
            # It's a directory, so move to it and refresh the view
            os.chdir(item)
            self._getPathAndFiles()
            self._imageLabel["image"] = None
            self._image = None
def main():
    """Instantiate and pop up the window."""
    ImageBrowser().mainloop()
main()
```

Output:

Practical No.-7

Program-7.1

Question:

Add three methods to the Student class that compare two Student objects. One method should test for equality. A second method should test for less than. The third method should test for greater than or equal to. In each case, the method returns the result of the comparison of the two students' names. Include a main function that tests all of the comparison operators.

Input:

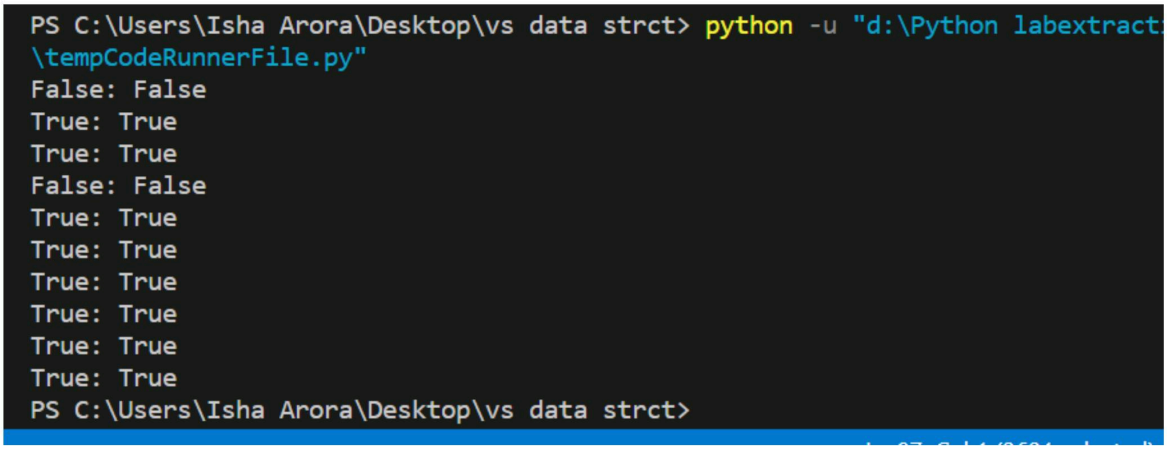
```
"""
File: student.py
Resources to manage a student's name and test scores.
Includes methods for comparisons and testing for equality.
"""


from functools import reduce


class Student(object):
    """Represents a student."""


    def __init__(self, name, number):
        """All scores are initially 0."""
        self._name = name
        self._scores = []
        for count in range(number):
            self._scores.append(0)


    def getName(self):
        """Returns the student's name."""
        return self._name


    def setScore(self, i, score):
```

```
    """Resets the ith score, counting from 1."""
    self._scores[i - 1] = score

def getScore(self, i):
    """Returns the ith score, counting from 1."""
    return self._scores[i - 1]

def getAverage(self):
    """Returns the average score."""
    sum = reduce(lambda x, y: x + y, self._scores)
    return sum / len(self._scores)

def getHighScore(self):
    """Returns the highest score."""
    return reduce(lambda x, y: max(x, y), self._scores)

def __str__(self):
    """Returns the string representation of the student."""
    return "Name: " + self._name  + "\nScores: " + \
           " ".join(map(str, self._scores))

def __lt__(self, other):
    """Returns self < other, with respect
    to names."""
    return self._name < other._name

def __gt__(self, other):
    """Returns self > other, with respect
    to names."""
    return not (self == other or self < other)
```

```
    def __le__(self, other):
        """Returns self <= other, with respect
        to names."""
        return self == other or self < other

    def __ge__(self, other):
        """Returns self >= other, with respect
        to names."""
        return self == other or self > other

    def __eq__(self, other):
        """Tests for equality."""
        if self is other:
            return True
        elif type(self) != type(other):
            return False
        else:
            return self._name == other._name

def main():
    """Tests equality and comparisons."""
    s1 = Student("Ken", 10)
    s2 = Student("Mary", 10)
    s3 = Student("Ken", 10)
    print("False:", s1 == s2)
    print("True:", s1 == s3)
    print("True:", s1 == s1)
    print("False:", s1 is s3)
    print("True:", s1 < s2)
```

```
print("True:", s2 > s1)
print("True:", s2 >= s1)
print("True:", s1 >= s3)
print("True:", s1 <= s2)
print("True:", s1 <= s3)
main()
```

Output:



Program-7.2

Question:

This project assumes that you have completed Project 1. Place several Student objects into a list and shuffle it. Then run the sort method with this list and display all of the students' information.

Input:

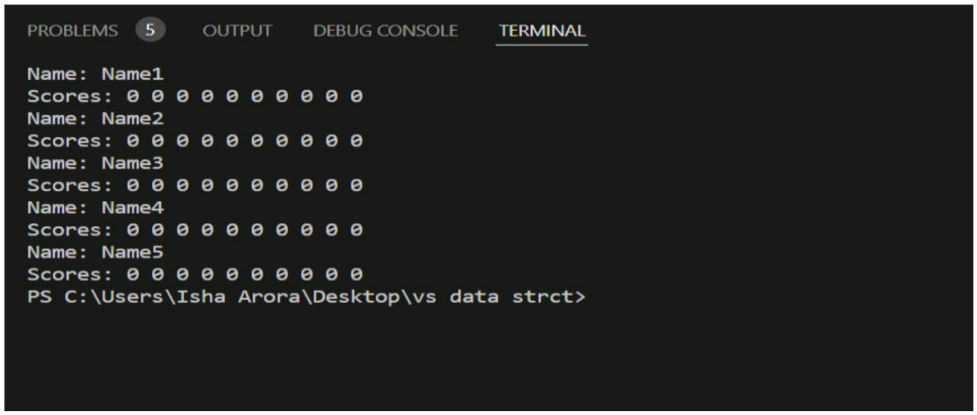
```
"""File: student.py

Resources to manage a student's name and test scores.

Includes methods for comparisons and testing for equality.

Tests the class by putting students into random order in a list

and then sorting them.
"""

from functools import reduce

class Student(object):
    """Represents a student."""
```

```
def __init__(self, name, number):
    """All scores are initially 0."""
    self._name = name
    self._scores = []
    for count in range(number):
        self._scores.append(0)

  def getName(self):
    """Returns the student's name."""
    return self._name

  def setScore(self, i, score):
    """Resets the ith score, counting from 1."""
    self._scores[i - 1] = score

  def getScore(self, i):
    """Returns the ith score, counting from 1."""
    return self._scores[i - 1]

  def getAverage(self):
    """Returns the average score."""
    sum = reduce(lambda x, y: x + y, self._scores)
    return sum / len(self._scores)

  def getHighScore(self):
    """Returns the highest score."""
    return reduce(lambda x, y: max(x, y), self._scores)

  def __str__(self):
    """Returns the string representation of the student."""
```

```
return "Name: " + self._name  + "\nScores: " + \
               " ".join(map(str, self._scores))

    def __lt__(self, other):
        """Returns self < other, with respect
        to names."""
        return self._name < other._name

    def __gt__(self, other):
        """Returns self > other, with respect
        to names."""
        return not (self == other or self < other)

    def __le__(self, other):
        """Returns self <= other, with respect
        to names."""
        return self == other or self < other

    def __ge__(self, other):
        """Returns self >= other, with respect
        to names."""
        return self == other or self > other

    def __eq__(self, other):
        """Tests for equality."""
        if self is other:
            return True
        elif type(self) != type(other):
            return False
        else:
```

```
return self._name == other._name
import random
def main():
    """Tests sorting."""
    # Create the list and put 5 students into it
    lyst = []
    for count in range(5):
        s = Student("Name" + str(count + 1), 10)
        lyst.append(s)
    # Shuffle and print the contents
    random.shuffle(lyst)
    print("Unsorted list of students:")
    for s in lyst:
        print(s)
    # Sort and print the contents
    lyst.sort()
    print("\nSorted list of students:")
    for s in lyst:
        print(s)
        main()
```

Output:



Program-7.3

Question:

The ATM program allows a user an indefinite number of attempts to log in. Fix the program so that it displays a popup message that the police will be called after a user has had three successive failures. The program should also disable the login button when this happens.

Input:

```
"""
File: atm.py


This module defines the ATM class and its application.


Modifies the user interface so that it prints a message and shuts down
if the user fails at three consecutive logins.


To test, launch from Idle and run


>>> createBank(5)
>>> main()


Can be modified to run as a script after a bank has been saved.
"""
from bank import Bank, SavingsAccount


class ATM(object):
    """This class represents terminal-based ATM transactions."""


    SECRET_CODE = "CloseItDown"


    def __init__(self, bank):
        self._account = None
```

```
    self._bank = bank
    self._methods = {}          # Jump table for commands
    self._methods["1"] = self._getBalance
    self._methods["2"] = self._deposit
    self._methods["3"] = self._withdraw
    self._methods["4"] = self._quit


def run(self):
    """Logs in users and processes their accounts."""
    failureCount = 0
    while True:
        name = input("Enter your name: ")
        if name == ATM.SECRET_CODE:
            break
        pin = input("Enter your PIN: ")
        self._account = self._bank.get(pin)
        if self._account == None:
            print("Error, unrecognized PIN")
            failureCount += 1
        elif self._account.getName() != name:
            print("Error, unrecognized name")
            self._account = None
            failureCount += 1
        else:
            failureCount = 0
            self._processAccount()
        if failureCount == 3:
            print("Shutting down and calling the cops!")
            break
```

```
def _processAccount(self):
    """A menu-driven command processor for a user."""
    while True:
        print("1  View your balance")
        print("2  Make a deposit")
        print("3  Make a withdrawal")
        print("4  Quit\n")
        number = input("Enter a number: ")
        theMethod = self._methods.get(number, None)
        if theMethod == None:
            print("Unrecognized number")
        else:
            theMethod()
            if self._account == None:
                break

def _getBalance(self):
    print("Your balance is $", self._account.getBalance())

def _deposit(self):
    amount = float(input("Enter the amount to deposit: "))
    self._account.deposit(amount)

def _withdraw(self):
    amount = float(input("Enter the amount to withdraw: "))
    message = self._account.withdraw(amount)
    if message:
        print(message)

def _quit(self):
```

```
        self._bank.save()
        self._account = None
        print("Have a nice day!")

# Top-level functions
def main():
    """Instantiate an ATM and run it."""
    bank = Bank("bank.dat")
    atm = ATM(bank)
    atm.run()

def createBank(number = 0):
    """Saves a bank with the specified number of accounts.
    Used during testing."""
    bank = Bank()
    for i in range(number):
        bank.add(SavingsAccount('Name' + str(i + 1),
                                str(1000 + i),
                                100.00))
    bank.save("bank.dat")
```

Output: