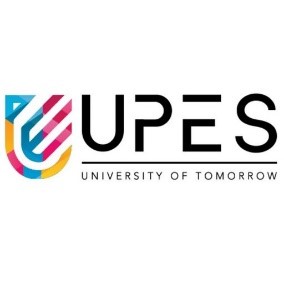
**Python Programming**

Lab Assignment

**Bachelor of Computer Application**

**Submitted by**

**Name:Devashish singh Roll No:65**



**Submitted to**

**Dr. Hemant Petwal**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, Uttarakhand**

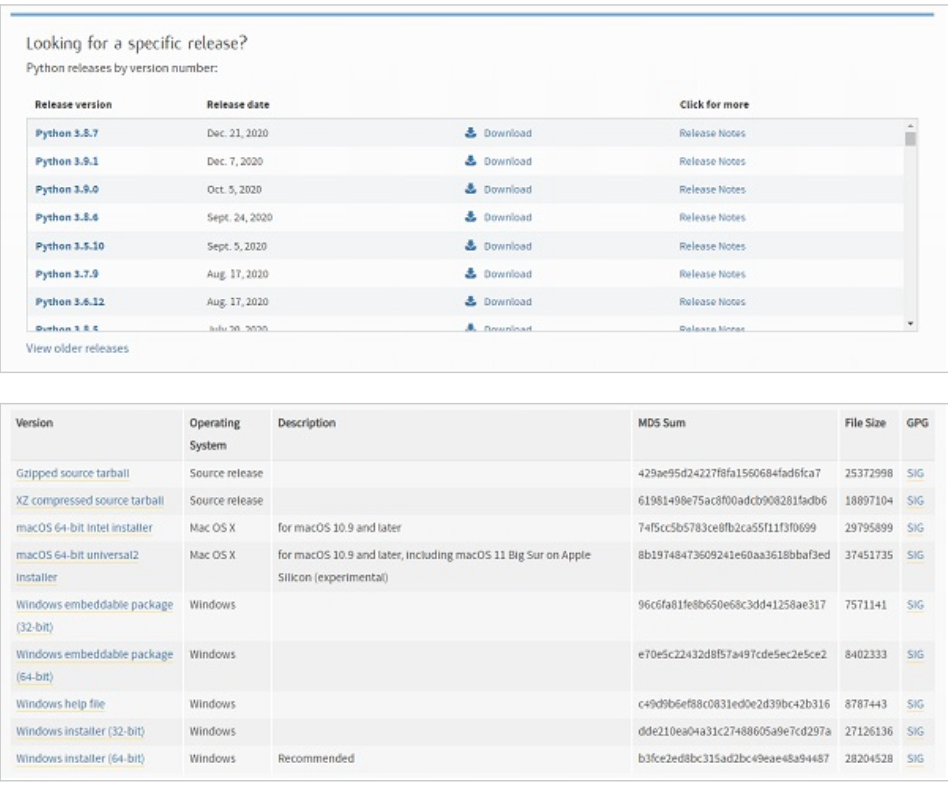
**Jan-May – 2024**

Expirenment 1:Python Installation

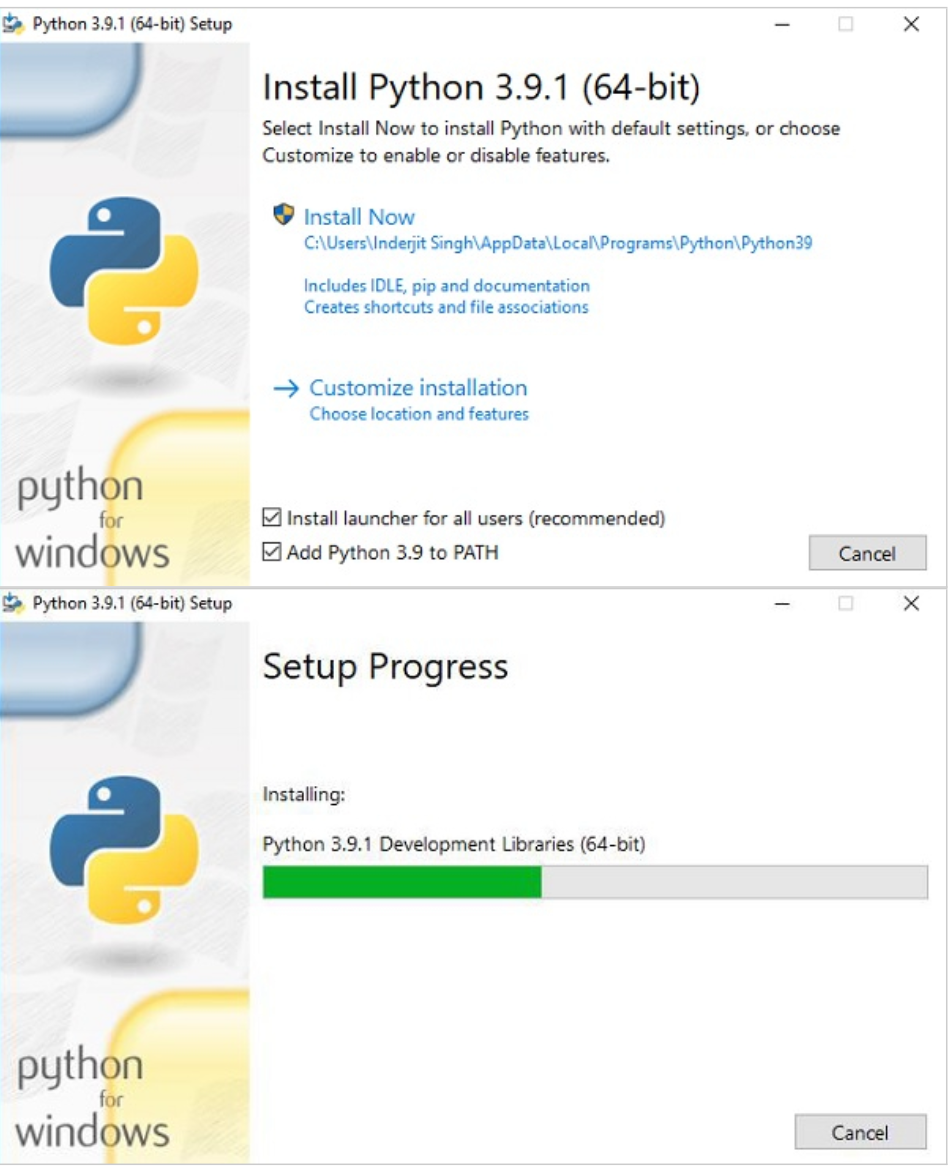
How to install python?

To install Python on your Windows machine using the Microsoft store, perform the following steps:

1. Open the Microsoft Store app on your Windows machine. You can do this by clicking the Start menu and searching for "Microsoft Store."



1. In the Microsoft Store app, search for "Python." You should see different "Python " apps in the search results. With referring to different versions of Python available on the Microsoft store.
2. Click on the app to open the app page.
3. Click the "Get" button to installation process.



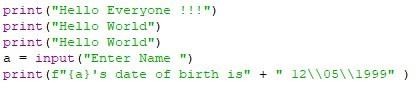
1. The Microsoft Store will download and install Python on your computer.
2. Once the installation is complete, follow the instructions in the section "Checking if Python is Already Installed on Your Windows Machine" to check that Python has been installed correctly.
3. This Python installation also comes with the IDLE Shell.

Experinment 2: Starting with python

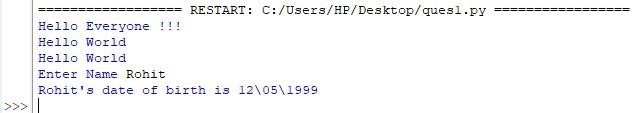
1. Install Python and understand difference between scripting and interactive modes in IDLE.

Script Mode, is used when the user is working with more than one single code or a block of code. Interactive mode is used when an user wants to run one single line or one block of code. If one needs to write a long piece of Python code or if the Python script spans multiple files, interactive mode is not recommended. In script mode, You write your code in a text file then save it with a .py extension.

1. Write Python programs to print strings in the given manner:
   1. Hello Everyone !!!
   2. Hello World
   3. Hello World
   4. Rohit's date of birth is 12\05\1999

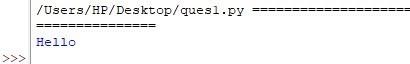
Input

output

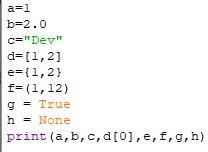


1. Declare a string variable called x and assign it the value "Hello". Print out the value of X

Input

Output

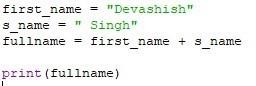
1. Take different data types and print values using print function.

Input

Output



1. Take two variables, a and b. Assign your first name and last name. Print your Name after adding your First name and Last name together.

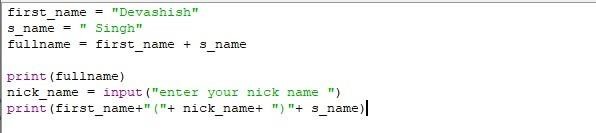
Input

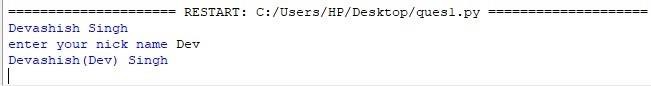
Output



1. Declare three variables, consisting of your first name, your last name and Nickname. Write a program that prints out your first name, then your nickname in parenthesis and then your last name.

Example output: George (woody) Washington.

Input

Output

7.Declare and assign values to suitable variables and print in the following way: NAME: NIKUNJ BANSAL

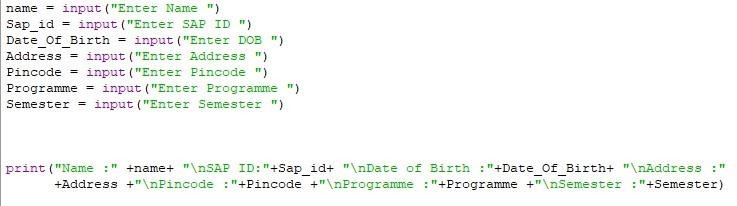
SAP ID: 500069944

DATE OF BIRTH: 13 Oct 1999

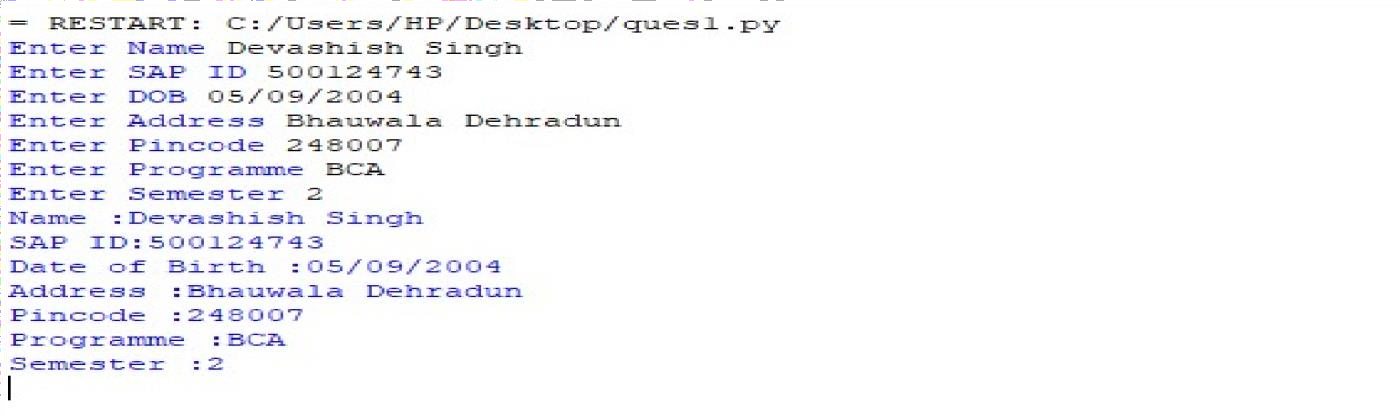
ADDRESS: UPES Bidholi Campus Pincode: 248007

Programme: AI & ML Semester: 2

Input



Output



EXPERIMENT 3: Use of input statements, operators

1. **Declare these variables (x, y and z) as integers. Assign a value of 9 to x. Assign a value of 7 to y, perform addition, multiplication, division, and subtraction on these two variables and print out the result.**

# Solution:

## Coding:

x=9 y=7 z=x+y

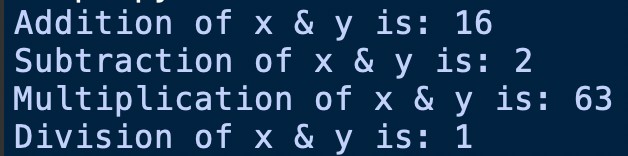
print("Addition of x & y is:",z) z=x-y

print("Subtraction of x & y is:",z) z=x\*y

print("Multiplication of x & y is:",z) z=x//y

print("Division of x & y is:",z)

## Output:



1. **Write a Program where the radius is taken as input to compute the area of a circle.**

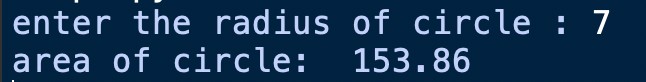
# Solution:

## Coding:

r=int(input("enter the radius of circle : ")) area=3.14\*r\*r

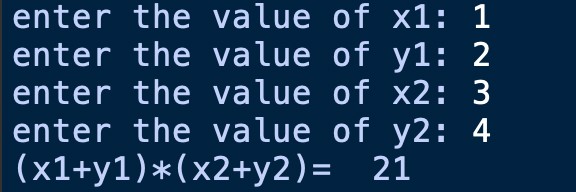
print("area of circle: ",area)

## Output:



1. **Write a Python program to solve (x1+y1)\*(x2+y2).**

# Solution:

x1=int(input("enter the value of x1: ")) y1=int(input("enter the value of y1: ")) x2=int(input("enter the value of x2: ")) y2=int(input("enter the value of y2: ")) z1=x1+y1

z2=x2+y2 result=z1\*z2

print("(x1+y1)\*(x2+y2)= ",result)

1. **Test data: x = 4, y = 3. Write a Program to perform any operation to get expected output: 49**

# Solution:

## Coding:

x=4 y=3 z=x+y

output=z\*z

print("output= ",output)

## Output:



1. **Write a program to find simple interests.**

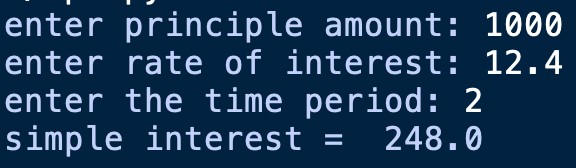
# Solution:

## Coding:

p=int(input("enter principle amount: ")) r=ﬂoat(input("enter rate of interest: ")) t=int(input("enter the time period: ")) si=(p\*t\*r)//100

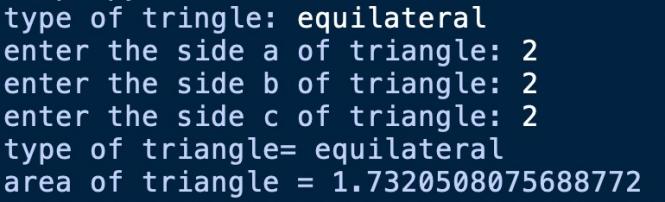
print("simple interest = ",si)

## Output:



1. **Write a program to find area of triangle when length of sides are given.**

# Solution:

\_type=input("type of tringle: ")

a=int(input("enter the side a of triangle: "))

b=int(input("enter the side b of triangle: ")) c=int(input("enter the side c of triangle: "))

s=(a+b+c)//2

area=(s\*(s-a)\*(s-b)\*(s-c))\*\*0.5 print("type of triangle=",\_type) print("area of triangle =",area)

1. **Write a program to convert minutes into seconds and hours**

# Solution:

## Coding:

minute=int(input("enter minutes: ")) seconds=minute\*60

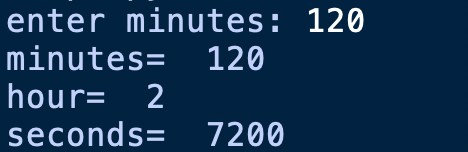
hour=minute//60

print("minutes= ",minute)

print("hour= ",hour)

print("seconds= ",seconds)

## Output:



1. **Write a program to swap two numbers without taking additional variable.**

# Solution:

## Coding:

x=3 y=2

print("before swapping") print("x=",x)

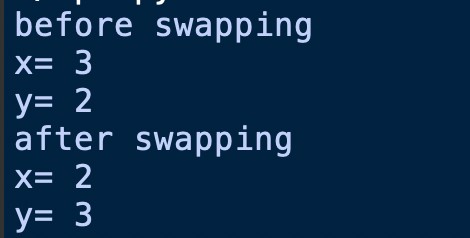
print("y=",y)

(x,y)=(y,x)

print("after swapping") print("x=",x)

print("y=",y)

## Output:



1. **Write a program to find sum of first n natural numbers.**

# Solution:

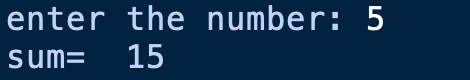
## Coding:

n=int(input("enter the number: ")) sum=0

for i in range(1,n+1): sum=sum+i

print("sum= ",sum)

## Output:



1. **Write a program to check whether a number is perfect square or not?**

# Solution:

## Coding:

n=int(input("enter the number: "))

ﬂag=0

for i in range(1,n):

if i\*i==n:

ﬂag=1

break

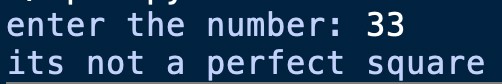
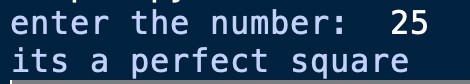
if ﬂag==0:

print("its not a perfect square")

else:

print("its a perfect square")

## Output:



1. **Write a program to covert a given series into perfect cube.**

## Coding:

num=eval(input("Enter a series of number: "))

n=[ ]

for i in num:

n.append(i\*\*3)

print("Cubic series are ")

print(n)

## Output:



1. **Write a program to print Fibonacci series.**

# Solution:

## Coding:

num = int(input("Enter the number of terms for the Fibonacci series: "))

flrst= 0

second= 1 sum=0

if num<=0:

print("enter num greater then 0: ")

else:

for i in range(0, num): print(sum,end=" ") flrst=second

second=sum

sum= flrst+second

## Output:



1. **Write a program to compute the length of the hypotenuse (c) of a right triangle using Pythagoras theorem.**

## Coding:

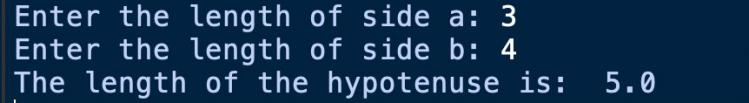
import math

a = int(input("Enter the length of side a: "))

b = int(input("Enter the length of side b: ")) hypotenuse = math.sqrt(a\*a + b\*b)

print("The length of the hypotenuse is: ",hypotenuse)

## Output:



1. **Write a program to print sum of even and odd numbers among n natural numbers**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **n = int(input("Enter the value of n: "))**  **even = 0**  **odd = 0**  **for i in range(1, n + 1):**  **if i % 2 == 0:**  **even += i else:**  **odd += i**  **print(f"Sum of even numbers from 1 to {n}:**  **{even}")**  **print(f"Sum of odd numbers from 1 to {n}: {odd}")** | **Output:** |

Expirenment 4 : Python lab Assignment

1. **Read an integer from keyboard using input and display it using print**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **n=int(input('Enter a number: '))**  **print(n)** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Read a float type number from a keyboard using input and display it using print.**

**Truncate the number to two decimal places.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **f=float(input('Enter a float number: '))**  **print('{:.2f}'.format(f))** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Any variable in Python has to have a name to identify its presence in the program. The**

**name assigned to that variable will become a "identifier". The identifier's name can start**

**with the alphabets A to Z or a to z or an underscore (\_). Also, numerals (0 to 9) can be**

**present in the variable name. And, the special symbols such as: !, #,@,%,$ cannot be used**

**in the identifiers. Write a program to assign different combinations of identifiers to the**

**variables and display the output of the values stored in identifiers and their type.**

|  |  |
| --- | --- |
| **Coding:**  **integer=10**  **decimal=10.11**  **num\_ber=-5**  **print(type(integer))**  **print(type(decimal))**  **print(type(num\_ber))** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Take input of a string “UPES University” and print 1st, fourth and last character of the string**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **string=str(input('Enter a string: '))**  **print(string[0])**  **print(string[3])**  **print(string[-1])** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Write a program to accept your personal details such as name and age and print it on the screen**

**using the formatter and the placeholder.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **name=str(input('Enter your name: '))**  **age=int(input('Enter your age: '))**  **new=('Hi {},you are {} years old!!!'.format(name,age))**  **print(new)** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Write a program to accept string as input and print it on the screen using formatter and**

**placeholder.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **a=str(input('Enter string: '))**  **b=str(input('Enter second string: '))**  **c=('I love {} programming, it is very Useful.'.format(a,b))**  **print(c)** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Mohit is a very cunning child, when his brother was away from his laptop, he changed his**

**original program to the following:**

x = input()

y = input()

if x>y:

output("x is greater than y")

else:

print("y is greater than x")

This program does not throw an error when it is run, rather it throws an error during runtime. These

kinds of errors are known as runtime errors.

If we give x=1, y=2, the program runs fine, but when we give x=2 and y=1, the program will throw an

error.

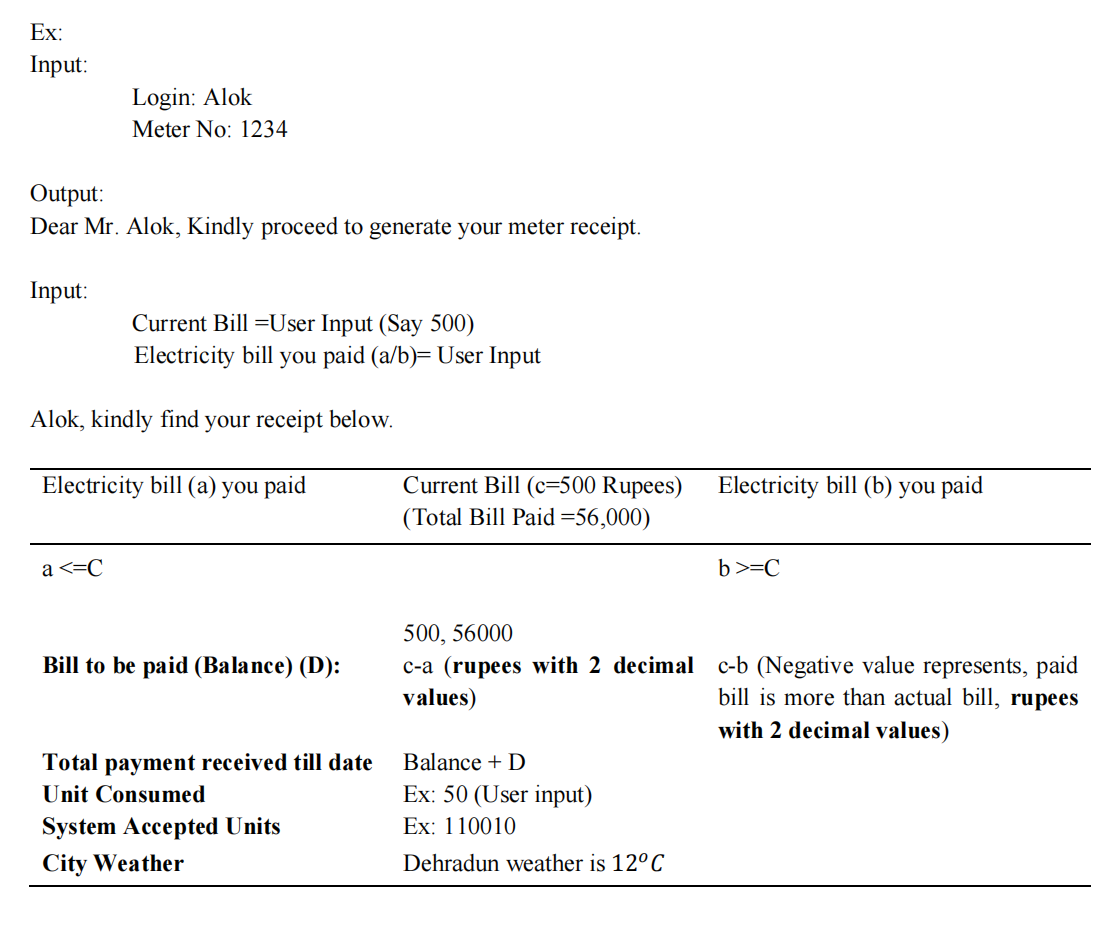
Correct this code so that it is error free.

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **x=int(input('enter a number: '))**  **y=int(input('enter 2nd number: '))**  **if x>y:**  **print("x is greater than y")**  **else:**  **print("y is greater than X")** | **Output:**  **Screenshot 2024-02-11 at 6.59.46 PM** |

1. **Below is Smart meter Prototype, write a program to implement this meter through python**

**programming.**

****

# Solution:

|  |  |
| --- | --- |
| **Coding:**  **login=str(input('LOGIN: '))**  **meter\_no=int(input('METER NO.: '))**  **print('Dear Mr.{}, Kindly proceed to generate your meter reciept'.format(login))**  **current=int(input('Current Bill:Rs '))**  **a=int(input('Electricity bill you paid(a):Rs '))**  **b=int(input('Electricity bill you paid(b):Rs '))**  **unit=int(input("Enter unit: "))**  **w=int(input('Enter weather of Dehradun: '))**  **total=56000**  **print('{},Kindly find your receipt below.'.format(login))**  **\_='\_'**  **space=' '**  **print(\_\*110)**  **print("Electricity bill(a)you paid {:<20} Current Bill=Rs{:<15} Electricity bill (b)you paid ".format(space,current))**  **print(space\*45,"Total bill paid=Rs 56000")**  **print(\_\*110)**  **if(a<=current) and (b>=current):**  **print('{}{:>90}'.format(a,b))**  **print("{:>55},56000".format(current))**  **print("Bill to be Paid(Balance)(D):{:^50.2f}{:>15.2f}".format((current-a),(current-b)))**  **x=total-(current-a)**  **y=total-(current-b)**  **print("Total payment received till date: {:>25.2f} {:>35.2f} ".format(x,y))**  **print("Unit Consumed {:^80}".format(unit))**  **print("System accepted units:{:^65b}".format(unit))**  **print("City Weather:{:<30}Dehradun weather is {}\u00b0 C".format(space,w))**  **print(\_\*110)** | **Output:**  **Screenshot 2024-02-11 at 8.40.04 PM** |

Expirenment 5 : Python lab Assignment

1. **Write a program to find left shift and right shift values of a given number.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  i=int(input("Enter a number :"))  L\_shift=i<<1  R\_shift=i>>1  print("Right Shift is",R\_shift)  print("Left Shift is",L\_shift) | **Output:**  **Screenshot 2024-02-13 at 4.29.22 PM** |

1. **Using membership operator find whether a given number is in sequence**

**(10,20,56,78,89)**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  list={10,20,56,78,89}  num=int(input("Enter a number to be found :"));  if num in list:  print("{} is in list".format(num))  else:  print("{} is not in the list".format(num))  if num not in list:  print("{} is not in the list".format(num))  else:  print("{} is in the list".format(num)) | **Output:**  **Screenshot 2024-02-13 at 11.02.30 PM**  **Screenshot 2024-02-13 at 11.01.13 PM** |

**3.Using membership operator find whether a given character is in a string.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  str1=str(input("Enter any word: "))  print(str1)  search=str(input("Enter any chracter to be founded in word :"))  if search in str1:  print("The character is present")  else:  print("The chracter is not present") | **Output:**  **Screenshot 2024-02-13 at 11.06.42 PM**  **Screenshot 2024-02-13 at 11.15.25 PM** |

**4.Check whether a given number is divisible by 3 and 5 both.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  input\_num = int(input("Enter a number: "))  if input\_num % 3 == 0 and input\_num % 5 == 0:  print("The number {} is divisible by both 3 and 5.".format(input\_num))  else:  print("The number {} is not divisible by both 3 and 5.".format(input\_num)) | **Output:**  **Screenshot 2024-02-13 at 11.13.53 PM**  **Screenshot 2024-02-13 at 11.14.19 PM** |

**5.Check whether a given number is multiple of 5 or not.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  num= int(input("enter a number : "))  if num%5 == 0:  print("num is multiple of 5")  else:  print("num is not multiple of 5") | **Output:**  **Screenshot 2024-02-13 at 11.34.44 PMScreenshot 2024-02-13 at 11.34.59 PM** |

**6.Find the greatest among two numbers. If numbers are equal than print “numbers are**

**equal”**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  a=int(input("enter value of a : "))  b=int(input("enter value of b : "))  print("a={} and b={}".format(a,b))  if a>b:  print("a is greater then b")  elif a==b:  print("both are equal")  else:  print("b is greater then a") | **Output:**  **Screenshot 2024-02-13 at 11.42.05 PM**  **Screenshot 2024-02-13 at 11.41.54 PM**  **Screenshot 2024-02-13 at 11.41.44 PM** |

**7.Find the greatest among three numbers assuming no two values are same.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  num1=int(input("enter num1 :"))  num2=int(input("enter num2 :"))  num3=int(input("enter num3 :"))  if num1 >= num2:  if num1 >= num3:  print("the greatest is = {}".format(num1))  elif num2 >= num1:  if num2 >= num3:  print("the greatest is = {}".format(num2))  else :  print("the greatest is = {}".format(num3)) | **Output**  **Screenshot 2024-02-14 at 12.04.26 AM** |

**8.Check whether the quadratic equation has real roots or imaginary roots. Display the**

**roots.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  print("Equation: ax^2 + bx + c ")  a=int(input("Enter a: "))  b=int(input("Enter b: "))  c=int(input("Enter c: "))  d=b\*\*2-4\*a\*c  d1=d\*\*0.5  if(d<0):  print("The roots are imaginary. ")  else:  print("The roots are real")  r1=(-b+d1)/2\*a  r2=(-b-d1)/2\*a  print("The first root: ",round(r1,2))  print("The second root: ",round(r2,2)) | **Output**  **Screenshot 2024-02-14 at 12.27.35 AM**  **Screenshot 2024-02-14 at 12.27.50 AM** |

1. **Write a program to find whether a given year is a leap year or not.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  year = int(input("Enter year to be checked: "))    if year % 4 == 0:  if year % 100 == 0:  if year % 400 == 0:  print("The year is a leap year!")  else:  print("The year is not a leap year!")  else:  print("The year is a leap year!")  else:  print("The year is not a leap year!") | **Output**  **Screenshot 2024-02-14 at 10.23.35 AM** |

**10. Print the grade sheet of a student for the given range of cgpa. Scan marks of five subjects**

**and calculate the percentage.**

**CGPA=percentage/10**

**CGPA range:**

**0 to 3.4 -> F**

**3.5 to 5.0->C+**

**5.1 to 6->B**

**6.1 to 7-> B+**

**7.1 to 8-> A**

**8.1 to 9->A+**

**9.1 to 10-> O (Outstanding)**

**Sample Grade sheet**

**Name: Rohit Kumar**

**Roll Number: R17234512**

**SAPID: 50005673**

**Sem: 1**

**Course: B.Tech. CSE AI & ML**

**Subject name: Marks**

**PDS: 70**

**Python: 80**

**Chemistry: 90**

**English: 60**

**Physics: 50 Percentage: 70%**

**CGPA:7.0**

**Grade: A**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  name=input("Enter your name: ")  roll\_num=input("Enter your roll number : ")  sap=input("Enter your sapid : ")  sem=input("Enter your Semester : ")  course=input("Enter your course : ")  sub1=int(input("Enter marks of PDS: "))  sub2=int(input("Enter marks of Python: "))  sub3=int(input("Enter marks of Chemistry: "))  sub4=int(input("Enter marks of English: "))  sub5=int(input("Enter marks of Physics: "))  percent=((sub1+sub2+sub3+sub4+sub5)/500)\*100  cgpa=percent/10  print(' SAMPLE GRADE SHEET ')  print('Name:{}'.format(name))  print('Roll Number:{}'.format(roll\_num))  print('SAP ID:{}'.format(sap))  print('Semester:{}'.format(sem))  print('Course:{}'.format(course))  print('Subject name :Mark')  print('PDS:{}'.format(sub1))  print('Python:{}'.format(sub2))  print('Chemistry:{}'.format(sub3))  print('English:{}'.format(sub4))  print('Physics:{}'.format(sub5))  print('Percentage:{}'.format(percent))  print("CGPA:{}".format(cgpa))  if cgpa>=0 and cgpa<=3.4:  print('Grade:F')  elif cgpa>=3.5 and cgpa<=5.0:  print('Grade:C+')  elif cgpa>=5.1 and cgpa<=6.0:  print('Grade:B')  elif cgpa>=6.1 and cgpa<=7.0:  print('Grade:B+')  elif cgpa>=7.1 and cgpa<=8.0:  print('Grade:A')  elif cgpa>=8.1 and cgpa<=9.0:  print('Grade:A+')  else:  print('Grade:O') | **Output** |

Python Lab Assignment 6

**1. Write a Python program that takes two strings as input and concatenates them**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  a=input("enter str1: ")  b=input("enter str2: ")  print(a +“ ”+ b) | **Output**  **Screenshot 2024-02-21 at 7.41.58 PM** |

**2. Write a Python program that takes a string as input and prints its reverse.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  a=input("enter string:")  x= a[::-1]  print(x) | **Output**  **Screenshot 2024-02-21 at 7.48.55 PM** |

**3. Write a Python program that takes a string as input and counts the number of**

**vowels in it.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  a=input("enter string:")  vowels="aeiouAEIOU"  i=0  for char in a:  if char in vowels:  i += 1  print("number of vowels: {} ".format(i)) | **Output**  **Screenshot 2024-02-21 at 7.52.13 PM** |

**4. Write a Python program that checks if a given string is a palindrome (reads the**

**same backward as forward).**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  string = input("Enter a string: ")  if string == string[::-1]:  print(string, "is a palindrome.")  else:  print(string, "is not a palindrome.") | **Output**  **Screenshot 2024-02-21 at 8.10.05 PM** |

**5. Write a Python program that takes a sentence as input and counts the number of**

**words in it**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  a=input("enter sentence: ")  count=len(a.split())  print("number of word in sentence: {}".format(count)) | **Output**  **Screenshot 2024-02-21 at 8.14.34 PM** |

**6. Write a Python program that takes a string as input and prints it in uppercase.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  str1 = input("enter string: ")  str2 = str1.upper()  print(str2) | **Output** |

**7. Write a Python program that checks if two given strings are anagrams (contain the**

**same characters with the same frequency).**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  a=input("Enter the first string: ")  b=input("Enter the second string: ")  x=[i for i in a]  x.sort()  y=[j for j in b]  y.sort()  if x==y:  print("Strings are anagrams")  else:  print("Strings are not anagrams") | **Output** |

**8. Write a Python program to perform basic string compression using the counts of**

**repeated characters. For example, the string "aabcccccaaa" would become**

**"a2b1c5a3".**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  s=str(input('Enter a string: '))  compressed = ""  current\_char = s[0]  count = 1  for i in range(1, len(s)):  if s[i] == current\_char:  count=count+ 1  else:  compressed =compressed+ current\_char + str(count)  current\_char = s[i]  count = 1  compressed=compressed + current\_char + str(count)  print(compressed) | **Output** |

**9. Write a Python program that takes a sentence as input and capitalizes the first**

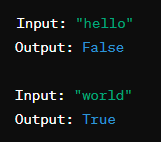
**letter of each word.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  str1 = input("Enter a string: ")  lst=str1.split()  count=0  for i in lst:  lst[count]=i.capitalize()  count+=1  str2=' '.join(lst)  print(str2) | **Output** |

**10. Write a Python function that takes a string as input and determines if it has all**

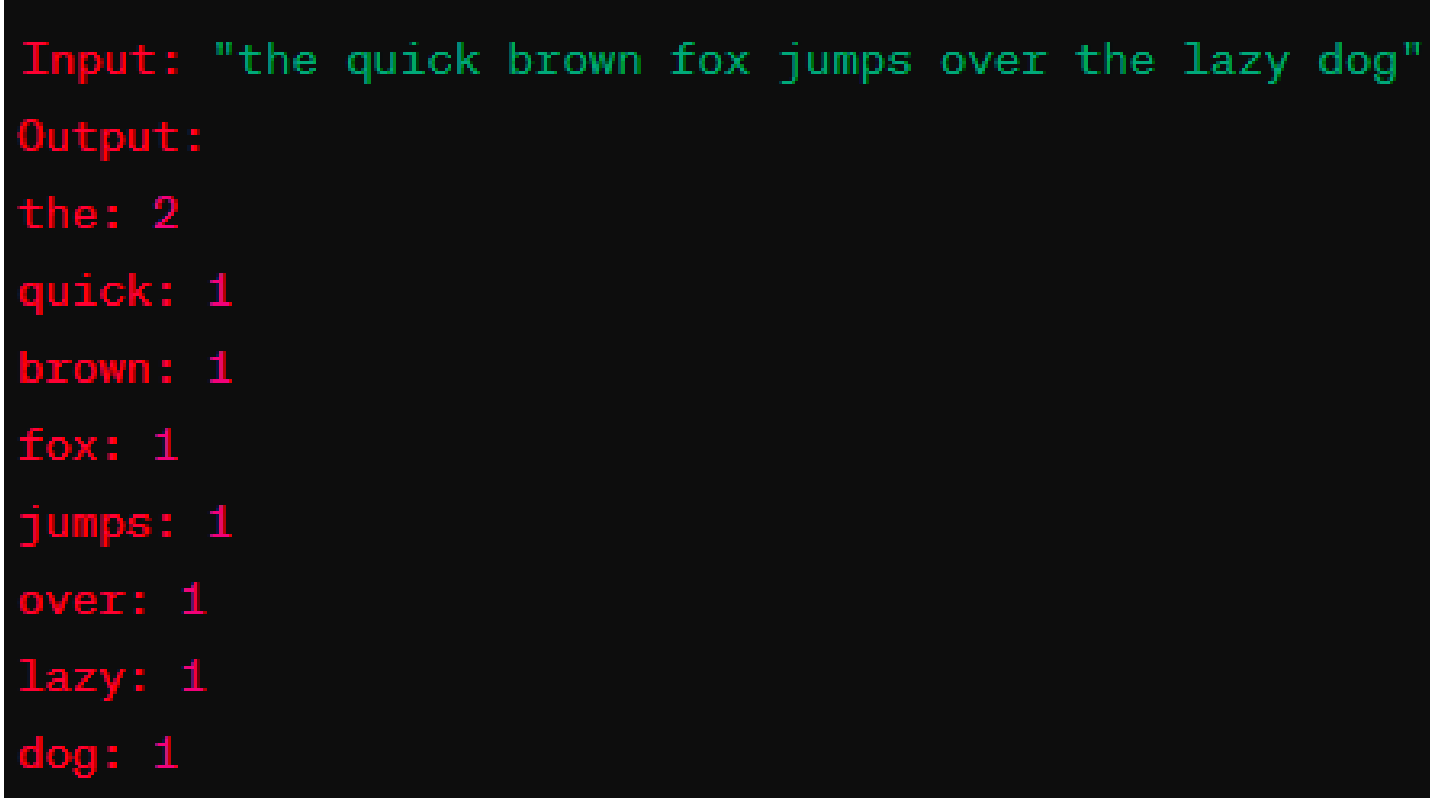
**unique characters.**



# Solution:

|  |  |
| --- | --- |
| **Coding:**  str1=input("Enter a string: ")  lst=list(str1)  count=0  for i in lst:  a=lst.count(i)  if a>1:  print("False")  count=0  break  else:  count+=1  if count>0:  print("True") | **Output** |

**11. Write a Python program that takes a sentence as input and prints the frequency of each word**

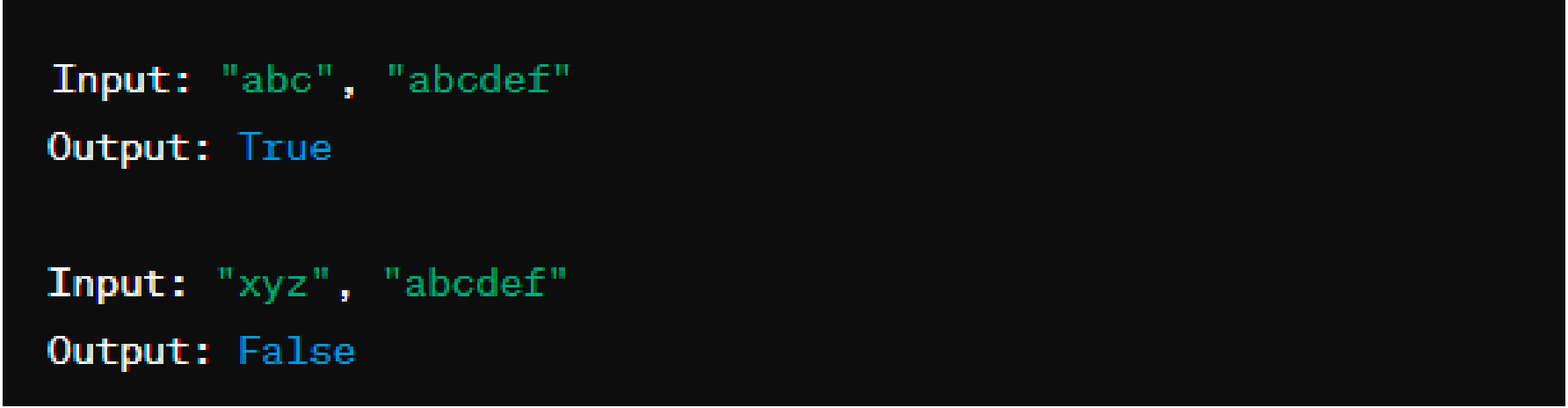


# Solution:

|  |  |
| --- | --- |
| **Coding:**  str1=input("Enter a sentence: ")  lst1=str1.split()  lst2=[]  for i in lst1:  if i not in lst2:  count=lst1.count(i)  print("{}: {}".format(i,count))  lst2.append(i) | **Output** |

**12. Write a Python function that takes two strings as input and determines if the first**

**string is a substring of the second string.**



# Solution:

|  |  |
| --- | --- |
| **Coding:**  str1=input("enter first string: ")  str2=input("enter second string: ")  if str1 in str2:  print("true")  else:  print("false") | **Output** |

1. **Write a Python program that takes a list of numbers as input and prints their sum.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  n=int(input('Enter the number of elements: '))  print('Enter the elements of list: ')  for i in range(n):  p=int(input())  lst.append(p)  print('List:',lst)  s=0  for i in range(0,len(lst)):  s=s+lst[i]  print('Sum of element in list :',s) | **Output** |

**14. Write a Python program that takes a list as input and prints its reversed order.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  list1=[]  n=int(input("enter number of elements in list: "))  print("enter the element of lists: ")  for i in range(n):  p=int(input())  list1.append(p)  print("list: ",list1)  list1.reverse()  print("reverse list :",list1) | **Output** |

**15. Write a Python function that takes a list as input and returns a new list containing**

**only the unique elements in the original list.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  n=int(input('Enter the number of elements of first list: '))  print('Enter the elements of first list: ')  for i in range(n):  p=int(input())  lst.append(p)  print('List 1:',lst)  lst2=[]  for i in lst:  if i not in lst2:  lst2.append(i)  print('List having unique element:',lst2) | **Output** |

**16. Write a Python program that takes two lists as input and prints their intersection**

**(common elements).**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  lst2=[]  n=int(input('Enter the number of elements of first list: '))  print('Enter the elements of first list: ')  for i in range(n):  p=int(input())  lst.append(p)  m=int(input('Enter the number of elements of second list: '))  print('Enter the elements of second list: ')  for i in range(m):  q=int(input())  lst2.append(q)  print('List 1:',lst)  print('List 2:',lst2)  lst3=[x for x in lst if x in lst2]  lst3.sort()  print('Intersected (common) element in both list:',lst3) | **Output** |

**17. Write a Python program that takes a list of numbers as input and prints the**

**elements from the third to the sixth (inclusive).**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  n=int(input('Enter the number of elements of first list: '))  print('Enter the elements of first list: ')  for i in range(n):  p=int(input())  lst.append(p)  print('List 1:',lst)  lst2=lst[2:7]  print('Element from 3rd to 6th(inclusive):',lst2) | **Output** |

**18. Write a Python program that takes two lists as input and uses the extend method**

**to combine them into a single list.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  lst2=[]  n=int(input('Enter the number of elements of first list: '))  print('Enter the elements of first list: ')  for i in range(n):  p=int(input())  lst.append(p)  m=int(input('Enter the number of elements of second list: '))  print('Enter the elements of second list: ')  for i in range(m):  q=int(input())  lst2.append(q)  print('List 1:',lst)  print('List 2:',lst2)  lst.extend(lst2)  print('Combined list:',lst) | **Output** |

**19. Write a Python program that takes a list of strings as input and prints the strings at**

**even indices in reverse order.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  txt=str(input("Enter a list of strings: "))  lst=txt.split()  for i in range(0,len(lst)):  if(i%2==0):  lst[i]=lst[i][::-1]  print(lst) | **Output** |

**20. Write a Python program that takes a list and a number n as input and repeats the**

**elements of the list n times.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  l1=[]  n=int(input("Enter number of repertation:"))  l=int(input("Enter length of list1:"))  for i in range(0,l):  e=input()  l1.append(e)  print("List1:",l1)  rl=l1\*n  print(rl) | **Output** |

**21. Write a Python program that takes a list of strings as input and sorts them in**

**alphabetical order.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  n=int(input('Enter the number of elements of list: '))  print('Enter the elements of list(string): ')  for i in range(n):  p=str(input())  lst.append(p)  print('List 1:',lst)  lst.sort()  print('Alphabetical order sorted list:',lst) | **Output** |

**22. Write a Python program that takes a list and an element as input and removes all**

**occurrences of that element from the list.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst=[]  n=int(input('Enter the number of elements of list: '))  print('Enter the elements of list: ')  for i in range(n):  p=int(input())  lst.append(p)  print('List 1:',lst)  m=int(input('Enter the number you want to remove:'))  i=0  while i<len(lst):  if lst[i]==m:  del lst[i]  else:  i=i+1  print('New list :',lst) | **Output** |

**23. Write a Python program that takes a list of numbers as input and uses list**

**comprehension to create a new list containing the squares of even numbers.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  l1=[]  l=int(input("Enter length of list1:"))  for i in range(0,l):  e=int(input())  l1.append(e)  print("List:",l1)  sl=[i\*\*2 for i in l1 if i%2==0]  print("Squared list:",sl) | **Output** |

**24. Write a Python function that takes a list of strings as input and uses list**

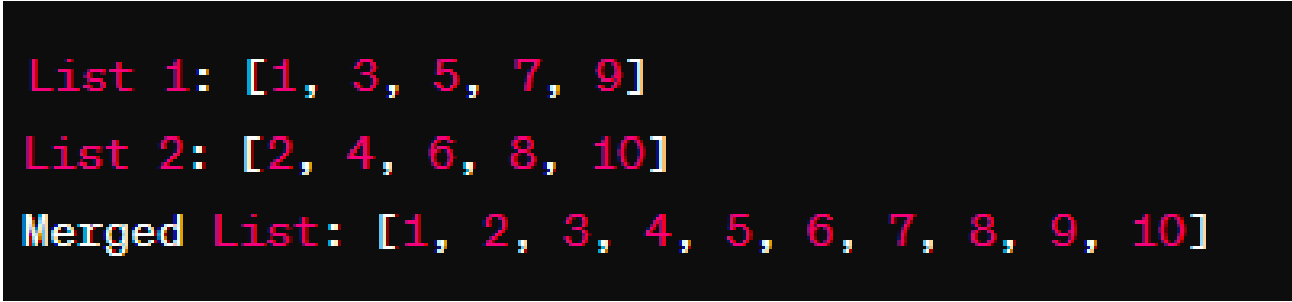
**comprehension to count the total number of vowels in all the strings combined.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  txt=str(input('Enter the elements of list(string): '))  lst=txt.split()  print('List:',lst)  count=0  vowels =['aeiouAEIOU']  for string in lst:  for char in string:  if char in vowels:  count=count+1    print('Vowels: ',count) | **Output** |

**25. Write a Python program that takes two sorted lists as input and merges them into**

**a single sorted list. Avoid using built-in functions or libraries for sorting.**



# Solution:

|  |  |
| --- | --- |
| **Coding:**  lst1=[]  n=int(input('Enter the number of elements of list: '))  print('Enter the elements of list: ')  for i in range(n):  p=int(input())  lst1.append(p)  print('List 1:',lst1)  i=0  lst2=[]  while i<len(lst1):  if lst1[i]%2==0:  lst2.append(lst1[i]\*\*2)  i=i+1    print('New list :',lst2) | **Output** |

Lab Experiment 7: Python

**Question 1: Tuple Basics**

**1. Create a tuple my\_tuple with elements (10, 20, 30, 40, 50).**

**2. Print the length of my\_tuple.**

**3. Print the element at index 2.**

**4. Print the last element of the tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  my\_tuple=(10, 20, 30, 40, 50)  #2  print("length of tuple",len(my\_tuple))  #3  print("element present in index 2 of tuple :",my\_tuple[2])  #4  print("last element of tuple",my\_tuple[-1]) | **Output**  Screenshot 2024-03-01 at 9.28.25 PM |

**Question 2: Tuple Manipulation**

**Write a Python program that does the following:**

**1. Create a tuple original\_tuple with elements (10, 20, 30, 40, 50).**

**2. Print the length of original\_tuple.**

**3. Print the element at index 3.**

**4. Print the last element of the tuple.**

**5. Convert original\_tuple to a list new\_list.**

**6. Add a new element 60 to new\_list.**

**7. Convert new\_list back to a tuple new\_tuple.**

**8. Print new\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  original\_tuple=(10, 20, 30, 40, 50)  #2  print("length of original\_tuple: ",len(original\_tuple))  #3  print("the element at index 3: ",original\_tuple[3])  #4  print("the last element of the tuple: ",original\_tuple[-1])  #5  new\_list=list(original\_tuple)  #6  new\_list.append(60)  #7  new\_tuple=tuple(new\_list)  #8  print("new\_tuple: ",new\_tuple) | **Output**  Screenshot 2024-03-01 at 9.41.40 PM |

**Question 3: Unpacking Tuples**

**Write a program that unpacks a tuple into multiple variables. The tuple is (10, 20, 30) and**

**the variables should be a, b, and c. Print the values of a, b, and c.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  my\_tuple=(10,20,30)  (a,b,c)=my\_tuple  print("a:",a)  print("b:",b)  print("c:",c) | **Output** |

**Question 4: Tuple Concatenation**

**Write a Python program that concatenates two tuples:**

**1. Create two tuples, tuple1 with elements (1, 2, 3) and tuple2 with elements (4, 5, 6).**

**2. Concatenate tuple1 and tuple2 into a new tuple result\_tuple.**

**3. Print result\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  tuple1=(1,2,3)  tuple2=(4,5,6)  #2  result\_tuple=tuple1+tuple2  #3  print("result\_tuple:",result\_tuple) | **Output** |

**Question 5: Tuple Packing and Unpacking**

**Write a Python program that demonstrates tuple packing and unpacking:**

**1. Create variables name, age, and country.**

**2. Pack these variables into a tuple called person.**

**3. Unpack the person tuple into name, age, and country variables.**

**4. Print these variables**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  name=input("enter your name: ")  age=int(input("enter your age: "))  country=input("enter your country: ")  #2  person=(name,age,country)  #3  (name,age,country)=person  #4  print("name: ",name)  print("age: ",age)  print("country: ",country) | **Output** |

**Question 6: Tuple Unpacking**

**Write a program that unpacks a tuple into variables a, b, and c. The tuple is (10, 20, 30).**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  my\_tuple=(10,20,30)  (a,b,c)=my\_tuple  print("a:",a)  print("b:",b)  print("c:",c) | **Output** |

**Question 7: Tuple Comprehension**

**Write a Python program that creates a tuple of squares of numbers:**

**1. Use tuple comprehension to create a tuple squares\_tuple that contains squares of**

**numbers from 1 to 10.**

**2. Print squares\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  squares\_tuple=tuple(i\*\*2 for i in range(1,11))  #2  print("square of numbers 1 to 10 :",squares\_tuple) | **Output** |

**Question 8: Frequency Count in Tuple**

**Write a Python program to count the frequency of elements in a tuple:**

**1. Create a tuple test\_tuple with repeated elements (1, 2, 3, 4, 1, 2, 1, 4, 5).**

**2. Count the frequency of each element and store it in a dictionary frequency\_dict.**

**3. Print frequency\_dict.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  tup1=(1, 2, 3, 4, 1, 2, 1, 4, 5)  #2&3  l=[]  for i in tup1:  if i not in l:  count=tup1.count(i)  print("{}: {}".format(i,count))  l.append(i)  tup2=tuple(l) | **Output** |

**Question 9: Convert List to Tuple**

**1. Create a list my\_list with elements [10, 20, 30].**

**2. Convert my\_list to a tuple my\_tuple.**

**3. Print my\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  my\_list=[10,20,30]  #2  my\_tuple=tuple(my\_list)  #3  print("tuple:",my\_tuple) | **Output** |

**Question 10: Reverse a Tuple**

**1. Create a tuple numbers with elements (1, 2, 3, 4, 5).**

**2. Write a program to reverse the elements of numbers.**

**3. Print the reversed tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  numbers = (1, 2, 3, 4, 5)  #2  reversed\_num = tuple(reversed(numbers))  #3  print("Original Tuple:", numbers)  print("Reversed Tuple:", reversed\_num) | **Output**  Screenshot 2024-03-01 at 10.49.19 PM |

**Question 11: Find Maximum Element**

**1. Create a tuple num\_tuple with elements (10, 20, 30, 40, 50).**

**2. Write a program to find the maximum element in num\_tuple.**

**3. Print the maximum element.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  num\_tuple = (10, 20, 30, 40, 50)  #2  max\_element = num\_tuple[0]  for num in num\_tuple:  if num > max\_element:  max\_element = num  #3  print("Maximum Element:", max\_element) | **Output**  **Screenshot 2024-03-01 at 10.51.10 PM** |

**Question 12: Tuple Membership Check**

**Write a Python program to check if an element exists in a tuple:**

**1. Create a tuple check\_tuple with elements (10, 20, 30, 40, 50).**

**2. Prompt the user to enter a number.**

**3. Check if the entered number exists in check\_tuple and print an appropriate message.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  check\_tuple=(10, 20, 30, 40, 50)  #2  num\_check=int(input("enter number you want to check :"))  #3  if num\_check in check\_tuple:  print("number {} present in tuple".format(num\_check))  else:  print("number {} not present in tuple".format(num\_check)) | **Output**  Screenshot 2024-03-01 at 10.52.58 PM |

**Question 13: Check Tuple Membership**

**1. Create a tuple test\_tuple with elements (1, 2, 3, 4, 5).**

**2. Prompt the user to enter a number.**

**3. Check if the entered number exists in test\_tuple and print an appropriate message.**

# Solution:

|  |  |  |  |
| --- | --- | --- | --- |
| **Coding:**   |  |  | | --- | --- | | #1  test\_tuple=(1,2,3,4,5)  #2  num=int(input("Enter a number:"))  #3  if num in test\_tuple:  print("{} is in tuple".format(num))  else:  print("{} is not in tuple".format(num)) | **Output**  Screenshot 2024-03-01 at 10.52.58 PM | | **Output**  Screenshot 2024-03-01 at 10.56.03 PM |

**Question 14: Zip and Unzip Tuples**

**Write a Python program that demonstrates zip and tuple unpacking:**

**1. Create two tuples names and ages containing names and corresponding ages.**

**2. Use zip to combine names and ages into a person\_info tuple.**

**3. Unpack person\_info into name and age variables.**

**4. Print name and age.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  names = ("Dev", "Sam", "Ram")  ages = (19, 30, 35)  person\_info = tuple(zip(names, ages))  print(f"after zipping:\n {person\_info}",)  name,age=zip(\* person\_info)  print(f"after unpacking: \n name: {name},age: {age}") | **Output**  Screenshot 2024-03-01 at 11.03.29 PM |

**Question 15: Tuple Sorting**

**Write a Python program that sorts a tuple of tuples based on the second element of each**

**tuple:**

**1. Create a tuple of tuples students with elements (("Alice", 22), ("Bob", 19), ("Charlie",**

**25)).**

**2. Sort students based on the second element of each tuple.**

**3. Print the sorted students.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  students=(("Alice", 22), ("Bob", 19), ("Charlie",25))  #2  l=list(students)  for i in range(0,len(l)):  for j in range(0,len(l)):  if l[i][1]<l[j][1]:  x=l[i]  l[i]=l[j]  l[j]=x  sorted\_tuple=tuple(l)  #3  print("Sorted Tuple: {}".format(sorted\_tuple)) |  |

**Output**

Screenshot 2024-03-01 at 11.08.10 PM

**Question 16: Tuple Slice**

**1. Create a tuple my\_tuple with elements (1, 2, 3, 4, 5, 6, 7, 8, 9, 10).**

**2. Slice my\_tuple to create a new tuple slice\_tuple with elements from index 3 to index 7.**

**3. Print slice\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  my\_tuple=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  #2  slice\_tuple=my\_tuple[3:7]  #3  print("slice\_tuple:",slice\_tuple) | **Output** |

**Question 17: Check Tuple Equality**

**1. Create two tuples tuple1 and tuple2 with the same elements.**

**2. Write a program to check if tuple1 and tuple2 are equal.**

**3. Print "Equal" or "Not Equal" based on the comparison.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  tuple1=(1,2,3,4,5)  tuple2=(1,2,3,4,5)  #2  l=[]  for i in tuple1:  if i in tuple2:  l.append(i)  t=tuple(l)  #3  if t==tuple1:  print("Equal")  else:  print("Not Equal") | **Output**  **Screenshot 2024-03-01 at 11.16.14 PM** |

**Question 18: Tuple Concatenation and Sorting**

**1. Create two tuples tuple1 with elements (5, 8, 2) and tuple2 with elements (3, 6, 1).**

**2. Concatenate tuple1 and tuple2 into a new tuple result\_tuple.**

**3. Sort result\_tuple in descending order.**

**4. Print the sorted result\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  tuple1=(5,8,2)  tuple2=(3,6,1)  #2  result\_tuple=tuple1+tuple2  #3  l=list(result\_tuple)  for i in range(0,len(l)):  for j in range(0,len(l)):  if l[i]>l[j]:  x=l[i]  l[i]=l[j]  l[j]=x  sorted\_tuple=tuple(l)  #4  print("Sorted Tuple: {}".format(sorted\_tuple)) | **Output**  **Screenshot 2024-03-01 at 11.21.32 PM** |

**Question 19: Unique Elements in Tuple**

**1. Create a tuple mixed\_tuple with elements (1, 2, 3, 4, 5, 1, 2, 3).**

**2. Write a program to create a new tuple unique\_tuple with unique elements from mixed\_tuple.**

**3. Print unique\_tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  mixed\_tuple=(1,2,3,4,5,1,2,3)  #2  x=[]  for a in mixed\_tuple:  if a not in x:  x.append(a)  unique\_tuple=tuple(x)  #3  print("Tuple of unique elements:{}".format(unique\_tuple)) | **Output** |

**Question 20: Tuple Intersection**

**1. Create two tuples tuple1 with elements (1, 2, 3, 4) and tuple2 with elements (3, 4, 5, 6).**

**2. Write a program to find the intersection of tuple1 and tuple2.**

**3. Print the intersection.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  tuple1=(1,2,3,4)  tuple2=(3,4,5,6)  #2  l=[]  for i in tuple1:  if i in tuple2:  l.append(i)  t=tuple(l)  #3  print("Intersection of tuple1 and tuple2 is:{}".format(t)) | **Output** |

**Question 21: Tuple Flattening**

1. **Create a tuple nested\_tuple with nested tuples ((1, 2), (3, 4), (5, 6)).**
2. **Write a program to flatten nested\_tuple into a single-level tuple.**

**3. Print the flattened tuple.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  nested\_tuple=((1,2),(3,4),(5,6))  #2  l=[]  for i in nested\_tuple:  l.extend(i)  t=tuple(l)  #3  print("Flattened Tuple:{}".format(t)) | **Output** |

**Question 22: Tuple Range Check**

**1. Create a tuple range\_tuple with elements (10, 20, 30, 40, 50).**

**2. Prompt the user to enter a number.**

**3. Check if the entered number falls within the range of elements in range\_tuple.**

**4. Print an appropriate message.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  #1  range\_tuple=(10,20,30,40,50)  #2  n=int(input("Enter a number:"))  #3  max=0  for i in range\_tuple:  if i>max:  max=i  min=max  for i in range\_tuple:  if i<min:  min=i  #4  if min<=n<=max:  print("The number is in range")  else:  print("The number is not in range") | **Output** |

Lab Experiment 8: Python

**Q1. Define a class Cab having following specifications: 1. Init method that initializes driver name, kms and rate/km. 2. Cab Class had a method rateperkm() that returns the running charges as kms\*rate 3. There are 3 drivers (driver1, driver2 and driver3) who have their own rate (rate1, rate2 and rate3) 4. Create three objects of the class Cab (firstcab, secondcab and thirdcab) and use to get the name of each driver along with the charges.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  class Cab:  def \_\_init\_\_(self, driver\_name, kms, rate\_per\_km):  self.driver\_name = driver\_name  self.kms = kms  self.rate\_per\_km = rate\_per\_km  def rateperkm(self):  return self.kms \* self.rate\_per\_km  kms = 25  driver\_1 = "Raju"  rate\_1 = 11  driver\_2 = "Rahul"  rate\_2 = 13  driver\_3 = "Viswas"  rate\_3 = 15  firstcab = Cab(driver\_1, kms, rate\_1)  secondcab = Cab(driver\_2, kms, rate\_2)  thirdcab = Cab(driver\_3, kms, rate\_3)  print("First Cab Driver: {}".format(firstcab.driver\_name))  print("First Cab Payment: {}".format(firstcab.rateperkm()))  print("Second Cab Driver: {}".format(secondcab.driver\_name))  print("Second Cab Payment: {}".format(secondcab.rateperkm()))  print("Third Cab Driver: {}".format(thirdcab.driver\_name))  print("Third Cab Payment: {}".format(thirdcab.rateperkm())) | **Output** |
|  |  |
|  |  |

**Q2. Given two user inputs x and n, calculate the value of xn .**

**For this expression evaluation design a class and an object to call the method to**

**implement this.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  class power:  def cal\_power(self,x,n):  return x\*\*n  cal=power()  x=int(input("enter the value of x:"))  n=int(input("enter the value of n:"))  result=cal.cal\_power(x,n)  print("output={}".format(result)) | **Output** |

**Q3. Define a class cars that uses init method to initialize the name, model and speed of a car.**

**1. Cars has two methods accelerate() and brakes(), that takes the value speed from init**

**method.**

**2. accelerate() returns the Cars.speed +70 whereas brakes() returns Cars.speed-20.**

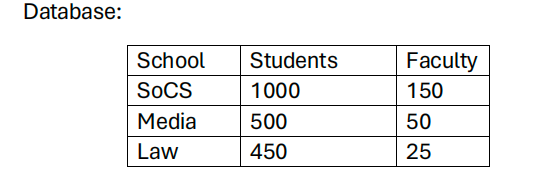
# Solution:

|  |  |
| --- | --- |
| **Coding:**  class Car:  def \_\_init\_\_(self,car\_name,car\_model,car\_speed):  self.car\_name = car\_name  self.car\_model = car\_model  self.car\_speed = car\_speed  def accelerate(self):  return self.car\_speed + 70  def brakes(self):  return self.car\_speed - 20  car\_name=input("enter car name: ")  car\_model=input("enter car model: ")  car\_speed=int(input("enter car speed: "))  car=Car(car\_name,car\_model,car\_speed)  result1=car.accelerate()  result2=car.brakes()  print("When the car accelerates,speed is {} ".format(result1))  print("Car brakes applied; speed is {} ".format(result2)) | **Output** |

**Q4. WAP to create a class UPES with three attributes, namely School name, number of students**

**and number of faculties. Add a method in the class to show these attributes. Create three objects**

**of this class UPES and show their details**



# Solution:

|  |  |
| --- | --- |
| **Coding:**  class UPES:  def \_\_init\_\_(self, schoolname, students, faculty):  self.schoolname = schoolname  self.students = students  self.faculty = faculty    def display(self):  """Display the details of the school."""  return "|{}\t\t|{}\t\t\t|{}\t\t\t|\n".format(str(self.schoolname), str(self.students), str(self.faculty))  soc = UPES('SoCS', 1000, 150)  media = UPES('Media', 500, 50)  law = UPES('Law', 450, 25)  with open("upes\_details.txt", "w") as file:  file.write("| School Name | Number of Students | Number of Faculties |\n")  file.write("+----------------+--------------------+------------------------+\n")  file.write(soc.display())  file.write(media.display())  file.write(law.display())  file.write("+----------------+--------------------+------------------------+\n")  print("Details written to 'upes\_details.txt' file.")  with open("upes\_details.txt", "r") as file:  content = file.read()  print(content) | **Output** |

**Q5. We have two circles with given coordinates of their centers C1(x1, y1) and C2(x2, y2) and**

**radius R1 and R2. Create a class with a method to check if the given circles**

**a) Inside the other**

**b) touch each other**

**c) Intersect each other**

**d) Do not overlap**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  class Circle:  def \_\_init\_\_(self, x, y, radius):  self.x = x  self.y = y  self.radius = radius  def distance(self, other\_circle):  dx = self.x - other\_circle.x  dy = self.y - other\_circle.y  return (dx\*\*2 + dy\*\*2)\*\*0.5  def relationship(self, other\_circle):  dx = abs(self.x - other\_circle.x)  dy = abs(self.y - other\_circle.y)  d = self.distance(other\_circle)    if d < abs(self.radius - other\_circle.radius):  return "One circle is inside the other"    elif d == abs(self.radius - other\_circle.radius):  return "Circles touch each other"  elif d < self.radius + other\_circle.radius:  return "Circles intersect each other"    else:  return "Circles do not overlap"  x1 = float(input("Enter x-coordinate for circle 1: "))  y1 = float(input("Enter y-coordinate for circle 1: "))  radius1 = float(input("Enter radius for circle 1: "))  x2 = float(input("Enter x-coordinate for circle 2: "))  y2 = float(input("Enter y-coordinate for circle 2: "))  radius2 = float(input("Enter radius for circle 2: "))  c1 = Circle(x1, y1, radius1)  c2 = Circle(x2, y2, radius2)  relationship = c1.relationship(c2)  print(relationship) | **Output** |

**Q6. Write a Python code to check a given number is odd or even using class. For this, design a**

**class namely “even\_odd” and a method “check” and create an object to check the number**

**using this function.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  class Even\_Odd:  def check(self, number):  if number % 2 == 0:  return "number is even."  else:  return "number is odd."  x = Even\_Odd()  num = int(input("Enter a number: "))  result = x.check(num)  print(result) | **Output** |

**Q7. A person has a list of words, where the words are written in small case letters. He wants to**

**convert each word of that list into uppercase letters. Write a python program (a function) that**

**converts small case word list to uppercase words list. For Example ['delhi', 'panjab'] will be**

**input and output will be ['DELHI', 'PANJAB'].**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  def convert\_to\_uppercase(word\_list):  uppercase\_list = [word.upper() for word in word\_list]  return uppercase\_list  list\_input = input("Enter words separated by commas: ").split(',')  list\_input = [word.strip() for word in list\_input]  uppercase = convert\_to\_uppercase(list\_input)  print("Uppercase words:", uppercase) | **Output** |

**Q8. Counting Upper and Lower case and Space symbols: Design a python module that will**

**count both upper, lower case symbols, and spaces in a given paragraph or sentences. Create a**

**module named case\_counting.py which has the funtion string\_test for performing the count.**

**Create an other program file main.py which import the case\_counting module.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  def string\_test(input\_string):  count\_dict = {'upper': 0, 'lower': 0, 'space': 0}  for char in input\_string:  if char.isupper():  count\_dict['upper'] += 1  elif char.islower():  count\_dict['lower'] += 1  elif char.isspace():  count\_dict['space'] += 1  return count\_dict  def main():  input\_string = input("Enter any sentence: ")  result = string\_test(input\_string)  print("\nOriginal String: ", input\_string)  print("No. of Upper case characters: ", result['upper'])  print("No. of Lower case Characters: ", result['lower'])  print("No. of spaces: ", result['space'])  main() | **Output** |

**Q9. Perfect number**

**Design and code a function viz., “perfect()” that determines if parameter number is a perfect**

**number. Use this function in a program that determines and prints all the perfect numbers**

**between 1 and N in a list.**

**[An integer number is said to be “perfect number” if its factors, including 1(but not the number**

**itself), sum to the number. E.g., 6 is a perfect number because 6=1+2+3].**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  def perfect(number):  sum = 1  i = 2  while i \* i <= number:  if number % i:  i += 1  else:  if i \* (number // i) == number:  sum += i + number // i  i += 1  return (sum == number and number != 1)  def perfect\_numbers(N):  perfect\_numbers\_list = []  for i in range(1, N + 1):  if perfect(i):  perfect\_numbers\_list.append(i)  return perfect\_numbers\_list  N = int(input("Enter the value of N: "))  print("Perfect numbers are: ", perfect\_numbers(N)) | **Output** |

**Q10. Practice Question: Student Database Management:**

**Develop a Python program to manage a student database. Implement functionalities to add**

**new students, display all students' information, search for a student by their ID, and save the**

**database to a file named "students.txt".**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  class Student:  def \_\_init\_\_(self, id, name, age):  self.id = id  self.name = name  self.age = age  def \_\_str\_\_(self):  return f"ID: {self.id}, Name: {self.name}, Age: {self.age}"  def save\_students\_to\_file(students, filename):  with open(filename, 'w') as file:  for student in students:  file.write(f"{student.id},{student.name},{student.age}\n")  def load\_students\_from\_file(filename):  students = []  with open(filename, 'r') as file:  for line in file:  data = line.strip().split(',')    student = Student(data[0], data[1], int(data[2]))  students.append(student)  return students  students = [Student("101", "Alice", 20), Student("102", "Bob", 21)]  save\_students\_to\_file(students, "students.txt")  loaded\_students = load\_students\_from\_file("students.txt")  for student in loaded\_students:  print(student) | **Output** |

**Q11. Practice Question: Employee Directory:**

**Implement a Python program to maintain an employee directory. Allow users to add new**

**employees, update employee information, delete employees, display all employees' details,**

**and save the directory to a file named "employees.txt".**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  class Employee:  def \_\_init\_\_(self, id, name, position):  self.id = id  self.name = name  self.position = position  def \_\_str\_\_(self):  return f"ID: {self.id}, Name: {self.name}, Position: {self.position}"  def save\_employees\_to\_file(employees, filename):  with open(filename, 'w') as file:  for emp in employees:  file.write(f"{emp.id},{emp.name},{emp.position}\n")  def load\_employees\_from\_file(filename):  employees = []  with open(filename, 'r') as file:  for line in file:  data = line.strip().split(',')  # Convert id from string to integer  emp = Employee(int(data[0]), data[1], data[2])  employees.append(emp)  return employees  # Usage example:  employees = [  Employee("001", "Alice", "Manager"),  Employee("002", "Bob", "Developer")  ]  save\_employees\_to\_file(employees, "employees.txt")  loaded\_employees = load\_employees\_from\_file("employees.txt")  for emp in loaded\_employees:  print(emp) | **Output** |

**Q12. Exception Handling:**

**1) Write a Python program that takes two numbers from the user and divides the first number**

**by the second number. Handle the ZeroDivisionError exception if the second number is zero.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  def divide\_numbers():  try:  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))    result = num1 / num2  print(f"The result is {result}")    except ZeroDivisionError:  print("Error: Division by zero is not allowed.")  divide\_numbers() | **Output** |

**2) Develop a Python program that reads the contents of a file specified by the user. Handle the**

**FileNotFoundError exception if the file does not exist.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  try:  file\_name = input("Please enter the file name: ")  with open(file\_name, 'r') as file:  contents = file.read()  print("The contents of the file are:")  print(contents)  except FileNotFoundError:  print("The file you specified does not exist. Please check the file name and try again.") | **Output** |

**3) Create a Python program that prompts the user to enter an integer. Handle the ValueError**

**exception if the input is not an integer**.

# Solution:

|  |  |
| --- | --- |
| **Coding:**  try:  user\_input\_str = input("Please enter an integer: ")  integer\_number = int(user\_input\_str)  print("The integer you entered is:", integer\_number)  except ValueError:  print("The input you provided is not a valid integer. Please check your input and try again.") | **Output** |

**4) Write a Python program that prompts the user to enter the index of a list and then prints the**

**element at that index. Handle the IndexError exception if the index is out of range.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  my\_fruits = ['apple', 'banana', 'cherry', 'date', 'elderberry']  try:  user\_index = int(input("Please enter the index of the element you want to access: "))  print("The element at index", user\_index, "is:", my\_fruits[user\_index])  except IndexError:  print("The index you provided is out of range. Please check your input and try again.")  except ValueError:  print("The input you provided is not a valid integer. Please check your input and try again.") | **Output** |

**5) Develop a Python program that defines a dictionary and prompts the user to enter a key to**

**retrieve the corresponding value. Handle the KeyError exception if the key does not exist in**

**the dictionary.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  my\_dict = {"apple": 1, "banana": 2, "cherry": 3}  try:  key = input("Enter the key: ")  print("The value for the key", key, "is:", my\_dict[key])  except KeyError:  print("Error: The key does not exist in the dictionary.") | **Output** |

**6) Design a Python program that writes user input to a file named "output.txt". Handle the**

**IOError exception if there is an error while writing to the file.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  try:  user\_text = input("Please enter some text: ")  with open('output.txt', 'w') as output\_file:  output\_file.write(user\_text)  print("The text was written to the file successfully.")  except IOError:  print("There was an error while writing to the file. Please check the file and try again.") | **Output** |

**7) Create a Python program that prompts the user to enter two numbers and then**

**concatenates them as strings. Recognize TypeError exception if the inputs are not**

**convertible to strings.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  try:  num1 = input("Enter the first number: ")  num2 = input("Enter the second number: ")  result = num1 + num2  print("The concatenated result is:", result)  except TypeError:  print("Error: The input could not be converted to strings.") | **Output** |

**8) Write a Python program that prompts the user to enter a number between 1 and 10. Handle**

**the ValueError exception if the input is not within the specified range.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  try:  user\_input = input("Please enter a number between 1 and 10: ")  number = int(user\_input)    if 1 <= number <= 10:  print("The number you entered is:", number)  else:  raise ValueError("The input you provided is not within the specified range. Please check your input and try again.")  except ValueError:  print("The input you provided is not within the specified range. Please check your input and try again.") | **Output** |

**9) Develop a Python program that performs arithmetic operations based on user input.**

**Recognize the ArithmeticError exception for invalid operations.**

# Solution:

|  |  |
| --- | --- |
| **Coding:**  try:  num1 = float(input("Please enter the first number: "))  num2 = float(input("Please enter the second number: "))  operation = input("Please enter the arithmetic operation (+, -, \*, /): ")  if operation == '+':  result = num1 + num2  elif operation == '-':  result = num1 - num2  elif operation == '\*':  result = num1 \* num2  elif operation == '/':  if num2 == 0:  raise ArithmeticError("Division by zero is not allowed.")  else:  result = num1 / num2  else:  raise ValueError("Invalid arithmetic operation.")  print("The result of the arithmetic operation is:", result)  except ValueError as ve:  print(ve)  print("Please check your input and try again.")  except ArithmeticError as ae:  print(ae)  print("Please check your input and try again.") | **Output** |