1. **That takes the radius of a sphere (a floating point number) as input and outputs the sphere’s diameter, circumference, surface area and volume.**

| import mathr=float(input("enter the radius r of a sphere"))d=2\*rprint("the diameter of the sphere is {:.2f}".format(d))SA=4\*math.pi\*r\*rprint("the surface area of the sphere is {:.2f}".format(SA))vol=(4/3)\*math.pi\*r\*r\*rprint("the volume of the sphere is {:.2f}".format(vol)) |
| --- |

Output

enter the radius r of a sphere5

the diameter of the sphere is 10.00

the surface area of the sphere is 314.16

the volume of the sphere is 523.60

1. **To circulate the n values stored in an list**

| n=int(input("enter number of terms : "))#read valueslist1=[]for i in range(0,n): ele=int(input("enter integer : ")) list1.append(ele)#circulate and display valuesprint("circulating the elements of list",list1)for val in range(0,n): ele=list1.pop(0) list1.append(ele) print(list1) |
| --- |

Output

 enter number of terms : 5

enter integer : 10

enter integer : 20

enter integer : 30

enter integer : 40

enter integer : 50

('circulating the elements of list', [10, 20, 30, 40, 50])

[20, 30, 40, 50, 10]

[30, 40, 50, 10, 20]

[40, 50, 10, 20, 30]

[50, 10, 20, 30, 40]

[10, 20, 30, 40, 50]

------------------

(program exited with code: 0)

Press any key to continue . . .

1. **That prints each character in this string followed by its ASCII value.Assume that the variable teststring refers to a string.**

| print("enter the string")teststring=input()for i in teststring: ascii\_value=ord(i) print(i,"\t",ascii\_value) |
| --- |

Output

enter the string

welcome

w 119

e 101

l 108

c 99

o 111

m 109

e 101

1. **To construct the truth tables for the following Boolean expressions.**
	1. **not (A or B)**
	2. **not A and not B**
2. **Not (A or B)**

| print('|{0} | {1} | {2} | {3} |'.format('A','B','A OR B','not(A or B)'))print("-"\*50)for p in False,True: for q in False,True: x=int(p or q) y=int(not(x)) print('|{0} | {1} | {2} | {3} |'.format(int(p),int(q),x,y)) |
| --- |

**Output**

**|A | B | A OR B | not(A or B) |**

**--------------------------------------------------**

**|0 | 0 | 0 | 1 |**

**|0 | 1 | 1 | 0 |**

**|1 | 0 | 1 | 0 |**

**|1 | 1 | 1 | 0 |**

1. **Not A and not B**

| print('|{0} | {1} | {2} | {3} | {4} |'.format('A','B','~A','~B','~A and ~B'))print("-"\*50)for p in False,True: for q in False,True: x=int(not(p)) y=int(not(q)) z=x and y print('|{0} | {1} | {2} | {3} | {4} |'.format(int(p),int(q),x,y,z)) |
| --- |

**output**

**|A | B | ~A | ~B | ~A and ~B |**

**--------------------------------------------------**

**|0 | 0 | 1 | 1 | 1 |**

**|0 | 1 | 1 | 0 | 0 |**

**|1 | 0 | 0 | 1 | 0 |**

**|1 | 1 | 0 | 0 | 0 |**

5. **A mapping that generates a list of absolute values of the numbers in a list named numbers.**

| def absvalue(x): if x>0: return x else: return -(x)lst=[-5,12,-56,90,-9,-8,-3]abslst=list(map(absvalue,lst))print("original list is = ",lst)print("list of absoulute value is =",abslst) |
| --- |

Output

original list is = [-5, 12, -56, 90, -9, -8, -3]

list of absoulute value is = [5, 12, 56, 90, 9, 8, 3]

6. **That reads an integer that are divisible by 5 between 0 and 1000 and adds all the digits in the integer.**

| n=input("enter the value divisible by 5 : ")def sum\_digits(n): digit\_tot=0 for i in (n): digit\_tot=digit\_tot+int(i) return(digit\_tot)print("the sum of digits of number {} is {}".format(n,sum\_digits(n)))  |
| --- |

Output

enter the value divisible by 5 : 23455

the sum of digits of number 23455 is 19

Another version

| n=int(input("enter the value divisible by 5 : "))def sum\_digits(n): digit\_tot=0 while (n!=0): digit\_tot=digit\_tot+ (n%10) n=n//10 return(digit\_tot)print("the sum of digits of number {} is {}".format(n,sum\_digits(n))) |
| --- |

enter the value divisible by 5 : 345

the sum of digits of number 345 is 12

7. **That classifies a given amount of money into smaller monetary units. (if the amount is 11.56 – the output should give how many dollars(100 cents), quarters(25 cents), dimes(10 cents), nickels(5 cents), pennies(1 cent)**

| amt=11.56dollar=100tot\_cent=11\*dollar+56nickel=tot\_cent/5penny=tot\_cent/1dime=tot\_cent/10quarter=tot\_cent/25print("the given amount is {}$".format(amt))print("the given amount in terms of cent is {}$".format(tot\_cent))print("the value equivalent in quarter is {} ".format(quarter) )print("the value equivalent in dime is {} ".format(dime) )print("the value equivalent in nickel is {} ".format(nickel) )print("the value equivalent in penny is {} ".format(penny) ) |
| --- |

Output

the given amount is 11.56$

the given amount in terms of cent is 1156$

the value equivalent in quarter is 46.24

the value equivalent in dime is 115.6

the value equivalent in nickel is 231.2

the value equivalent in penny is 1156.0

8. **To print the first n prime numbers.**

| def isPrime(n): n\_divisor=0 for i in range(1,n+1): if(n%i==0): n\_divisor=n\_divisor+1 if(n\_divisor<3): return True else: return Falsed=[]x=int(input("enter the number "))for i in range(1,x+1): if (isPrime(i)): d.append(i) print(d) |
| --- |

Output

enter the number 30

[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29]

9. **Use functions to compute the following summation:**

$\frac{1}{1+\sqrt{2}}+\frac{1}{\sqrt{2}+\sqrt{3}}+\frac{1}{\sqrt{3}+\sqrt{4}}+\frac{1}{\sqrt{4}+\sqrt{5}}+…+\frac{1}{\sqrt{99}+\sqrt{100}}$

| import mathdef term(x,y): denom=math.sqrt(x)+math.sqrt(y) return(1/denom)n=100sum\_value=0for i in range(1,n+1): sum\_value=sum\_value+term(i,i+1)print("the series value is ",sum\_value) |
| --- |

Output

the series value is 9.049875621120892

10. **Using functions to convert decimal to hexadecimal.**

| conversion\_table=['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F']decimal=int(input("enter the decimal number :"))decimal1=decimalhexadecimal=""while (decimal>0): remainder=decimal%16  hexadecimal=conversion\_table[remainder]+hexadecimal  decimal=decimal//16print("hexadecimal of ",decimal1," is =",hexadecimal,"H") |
| --- |

Output

enter the decimal number :44

hexadecimal of 44 is = 2C H

11. **To calculate the BMI of a person given the following specifications. The class BMI is defined as follows**

| **BMI** |
| --- |
| **-name : str****-age : int****-weight : float****-height : float****getName()****getAge()****getWeight()****getHeight()** |
| **BMI(name: str, age: int, weight: float, height: float)****getBMI():float****getStatus(): str** |

Program

| class BMI: def \_\_init\_\_(self,name,age,weight,height): self.\_\_name=name self.\_\_age=age self.\_\_weight=weight self.\_\_height=height  def getBMI(self): bmi=self.\_\_weight/(self.\_\_height\*self.\_\_height) return bmi  def getStatus(self): bmi=self.getBMI() if bmi<18.5: return("Underweight") elif bmi<25: return("Normal weight") elif bmi<30: return("Overweight") else: return("obese")  def getName(self): return self.\_\_name  def getAge(self): return self.\_\_age  def getWeight(self): return self.\_\_weight  def getHeight(self): return self.\_\_height bmi1=BMI("rani",25,75,1.58)print(" the BMI for {0} is {1} and the status is {2}".format(bmi1.getName(),bmi1.getBMI(),bmi1.getStatus())) |
| --- |

Output

 the BMI for rani is 30.043262297708697 and the status is obese

12. **Design a class named QuadraticEquation for a quadratic equation** $ax^{2}+bx+c=0$**. The class contains:**

* **The private data fields a,b and c the three co-efficients**
* **A constructor with arguments**
* **Three get methods**
* **A method named getDiscriminant() that returns the discriminant**
* **The methods named getRoot1() and getRoot2() returning the two roots of the equation. Note:- These methods are valid only if discriminant is nonnegative.Implement the above class and test it with different inputs.**

$r1=\frac{-b+ \sqrt{b^{2}^{}-4ac^{}^{}}}{2a}$$r2=\frac{-b - \sqrt{b^{2}^{}-4ac^{}^{}}}{2a}$

| import mathclass QuadraticEquation: def \_\_init\_\_(self,a,b,c): self.\_\_a=a self.\_\_b=b self.\_\_c=c def getA(self): return self.\_\_a def getB(self): return self.\_\_b def getC(self): return self.\_\_c def getDiscriminant(self): return((self.\_\_b\*self.\_\_b)-(4\*self.\_\_a\*self.\_\_c)) def getRoot1(self): x=self.getDiscriminant() if x<0: return 0 else: return(-self.\_\_b + math.sqrt(x) )/(2\*self.\_\_a) def getRoot2(self): x=self.getDiscriminant() if x <0: return 0 else: return(-self.\_\_b - math.sqrt(x))/(2\*self.\_\_a) a=int(input("enter the value for a"))b=int(input("enter the value for b"))c=int(input("enter the value for c"))equation=QuadraticEquation(a,b,c)disc=equation.getDiscriminant()if disc<0: print("the equation has no roots")elif disc==0: print("the root is ",equation.getRoot1())else: print("the roots are ",equation.getRoot1()," and ", equation.getRoot2())  |
| --- |

4x2 + 3x - 27 = 0

a=4

b=3

c=-27

Output

enter the value for a4

enter the value for b3

enter the value for c-27

the roots are 2.25 and -3.0

Another input

15x2 - 26x - 21 = 0

a=15, b= -26, c=-21

Output

enter the value for a15

enter the value for b-26

enter the value for c-21

the roots are 2.3333333333333335 and -0.6

Input 3

21x2 - 12x + 1 = 0

a=21, b=-12, c=1

Output

enter the value for a21

enter the value for b-12

enter the value for c1

the roots are 0.47014206410511505 and 0.10128650732345633

13. **A function to check whether a string is a valid password where the following are the rules of valid password.**

* 1. **A password must have at least eight characters**
	2. **A password must consist of only letters and digits**
	3. **A password must contain at least two digits**

| def passcheck(str): x=len(str) if (x< 8): print("your password is not valid. must contain minimum 8 characters") return(0) if len([x for x in str if x.isdigit()])<2: print("password must contains atleast 2 digits") return(0) if (str.isalnum()==0): print("your password must consists of alphabets and numerical only") return(0)str=input("enter the password")y=passcheck(str)if y!=0: print("your password is correct")else: print("password incorrect") |
| --- |

Output

enter the password rada

your password is not valid. must contain minimum 8 characters

password incorrect

14. **Create a module named RandomCharacter.py with five functions that randomly generate specific types of characters. 1. Generate a random character between ch1 and ch2. 2. Generate a random lowercase letter 3. Generate a random uppercase letter 4. Generate a random digit character 5. Generate a random character.**

**Create a list of random lowercase characters using the above module and write a program to count the occurrences of each alphabet.**

Module : RandomCharacter.py

| import randomimport stringdef generatechar(ch1,ch2): a=[] for i in range(ord(ch1),ord(ch2)+1): a.append(chr(i)) return(a)def randomchar(a): return(random.choice(a))def randomlower(): return(random.choices(string.ascii\_lowercase))def randomupper(): return(random.choices(string.ascii\_uppercase))def randomdigit(): return(random.choices(string.digits))def randomletter(): return(random.choices(string.ascii\_letters))def randomlower\_many(): return(''.join(random.choices(string.ascii\_lowercase,k=10))) |
| --- |

Main program : exercise14.py

| import RandomCharacter as rcch1=input("enter the starting character: ")ch2=input("enter the ending character: ")a=rc.generatechar(ch1,ch2)print(a)b=rc.randomchar(a)print("random character within given range =",b)r\_lower=rc.randomlower()print("random lower case character =",r\_lower)r\_upper=rc.randomupper()print("random upper case character =",r\_upper)r\_digit=rc.randomdigit()print("random digit =",r\_digit)r\_letter=rc.randomletter()print("random character =",r\_letter)r\_lower\_array=rc.randomlower\_many()print("the generated random lower case letters is =", r\_lower\_array)dict1={x:r\_lower\_array.count(x) for x in r\_lower\_array}print(dict1) |
| --- |

Output

enter the starting character: a

enter the ending character: n

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n']

random character within given range = c

random lower case character = ['w']

random upper case character = ['Q']

random digit = ['4']

random character = ['U']

the generated random lower case letters is = lhmadbxjmu

{'l': 1, 'h': 1, 'm': 2, 'a': 1, 'd': 1, 'b': 1, 'x': 1, 'j': 1, 'u': 1}

output

enter the starting character: h

enter the ending character: x

['h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x']

random character within given range = j

random lower case character = ['c']

random upper case character = ['Q']

random digit = ['9']

random character = ['F']

the generated random lower case letters is = mqhdaqfuqx

{'m': 1, 'q': 3, 'h': 1, 'd': 1, 'a': 1, 'f': 1, 'u': 1, 'x': 1}

15. **Suppose the weekly hours for all employees are stored in a table. Each row records an employee’s seven-day work hours with seven columns. Write a program that displays employees and their total hours in decreasing order of the total hours.**

| def workHours\_matrix(): workHours=[[2,4,3,4,5,8,8], [7,3,4,3,3,4,4], [3,3,4,3,3,2,2], [9,3,4,7,3,4,1], [3,5,4,3,6,3,8], [3,4,4,6,3,4,4], [3,7,4,8,3,8,4], [6,3,5,9,2,7,9]] weeklyHours=[] for i in range(len(workHours)): weeklyHours.append([sum(workHours[i]),i]) for i in range(len(workHours)): print(weeklyHours[i]) weeklyHours.sort()#Display result for empHours in weeklyHours: print("Employee : " + str(empHours[1]) + " : " + str(empHours[0]))workHours\_matrix()  |
| --- |

Output

[34, 0]

[28, 1]

[20, 2]

[31, 3]

[32, 4]

[28, 5]

[37, 6]

[41, 7]

Employee : 2 : 20

Employee : 1 : 28

Employee : 5 : 28

Employee : 3 : 31

Employee : 4 : 32

Employee : 0 : 34

Employee : 6 : 37

Employee : 7 : 41

16. **Design a class Polygon with the following:-**

**● Number of sides**

**● Magnitude of the sides as a list**

**● inputSides()**

**● dispSides()**

**Define another class Triangle which will inherit Polygon and has one extra function**

**findArea() to calculate the area of a triangle.(Any extra methods can be defined if required)**

| class Polygon: def \_\_init\_\_(self,no\_of\_sides): self.n=no\_of\_sides self.sides=[0 for i in range(no\_of\_sides)] def inputSides(self): for i in range(self.n): self.sides[i]=float(input("enter side")) def dispSides(self): for i in range(self.n): print("side {0} is {1}".format(i,self.sides[i]))class Triangle(Polygon): def \_\_init\_\_(self): Polygon.\_\_init\_\_(self,3) def findArea(self): a,b,c=self.sides s=(a+b+c)/2 area=(s\*(s-a)\*(s-b)\*(s-c)) print("the area of triangle is ",area)obj1=Triangle()obj1.inputSides()obj1.dispSides()obj1.findArea() |
| --- |

Output

enter side3

enter side4

enter side5

side 0 is 3.0

side 1 is 4.0

side 2 is 5.0

the area of triangle is 36.0

17. **A program that removes all the occurrences of a specified string from a text file and counts the same. Your program should prompt the user to enter a filename and a string to be removed.**

| filename=input("enter the name of the file with the correct path")t=input("enter the existing string")with open(filename,"r") as f: filedata=f.read()word\_count=filedata.count(t)print("the word occurs ",word\_count,"times")filedata=filedata.replace(t,"")with open("c:/pythonprgs/test.txt","w") as f: f.write(filedata) |
| --- |

18. **To illustrate ZeroDivisionError, KeyError, ValueError using exception handling.**

1. **ZeroDivisionError**

| try: print("this is try block") x=int(input("enter a divident number")) y=int(input("enter the divisor")) z=x/yexcept ZeroDivisionError: print("divide by zero error occured. divisor should not be zero")else: print("no error occured. this is in else block") print("result of division is =",z)finally: print("this is finally block")  |
| --- |

1. **KeyError**

| #KeyErrorages={'anitha':20,'arthi':21,'banu':19}x=input('enter the age for')try: print('{0} is {1} years old '.format(x,ages[x]))except KeyError: print('{0}s age is not known'.format(x)) |
| --- |

1. **ValueError**

| try: age=int(input("enter age")) if(age<18): raise ValueError else: print("you are eligible to vote")except ValueError: print("not eligible to vote") |
| --- |

19. **That reads an unspecified number of integers and finds the ones that have the most occurrences using a dictionary.**

| my\_list=[1,2,3,5,5,3,1,3,3,1,4,4,4,2,2,2,2,7]dict1={x:my\_list.count(x) for x in my\_list}print(dict1) |
| --- |

Output

{1: 3, 2: 5, 3: 4, 5: 2, 4: 3, 7: 1}

20. **A recursive function for the following 1. To generate the Fibonacci series. 2. To reverse the given string.**

1. **To generate the fibonacci series**

| #recursive function for fibonacci seriesdef fibonacci(n): if(n<=1): return n else: return(fibonacci(n-1)+fibonacci(n-2)) n=int(input("enter the number of terms"))print("Fibonacci sequence")for i in range(n): print(fibonacci(i)) |
| --- |

Output

enter the number of terms15

Fibonacci sequence

0

1

1

2

3

5

8

13

21

34

55

89

144

233

377

20.2 **To reverse the given string**

| # recursive function for reversel of a stringdef reverse(s): if len(s)==0: return s else: return reverse(s[1:])+s[0]str=input("enter the string : ")print("the reversed string is : ")print(reverse(str)) |
| --- |

Output

enter the string : python

the reversed string is : nohtyp