1A)

Input :

m1 = int(input("Enter marks for test1 : "))

m2 = int(input("Enter marks for test2 : "))

m3 = int(input("Enter marks for test3 : "))

if m1 <= m2 and m1 <= m3:

avgMarks = (m2+m3)/2

elif m2 <= m1 and m2 <= m3:

avgMarks = (m1+m3)/2

elif m3 <= m1 and m2 <= m2:

avgMarks = (m1+m2)/2

print("Average of best two test marks out of three test’s marks is", avgMarks);

output :

Enter marks for test1 : 45

Enter marks for test2 : 39

Enter marks for test3 : 48

Average of best two test marks out of three test’s marks is 46.5

1B)

Input :

val = int(input("Enter a value : "))

str\_val = str(val)

if str\_val == str\_val[::-1]:

print("Palindrome")

else:

print("Not Palindrome")

for i in range(10):

if str\_val.count(str(i)) > 0:

print(str(i),"appears", str\_val.count(str(i)), "times");

Output :

Enter a value : 1234234

Not Palindrome

1 appears 1 times

2 appears 2 times

3 appears 2 times

4 appears 2 times

2A)

Input :

def fn(n):

if n == 1:

return 0

elif n == 2:

return 1

else:

return fn(n-1) + fn(n-2)

num = int(input("Enter a number : "))

if num > 0:

print("fn(", num, ") = ",fn(num) , sep ="")

else:

print("Error in input")

Output :

Enter a number : 5

fn(5) = 3

Enter a number : -1

Error in input

2B)

def bin2Dec(val):

rev=val[::-1]

dec = 0

i = 0

for dig in rev:

dec += int(dig) \* 2\*\*i

i += 1

return dec

def oct2Hex(val):

rev=val[::-1]

dec = 0

i = 0

for dig in rev:

dec += int(dig) \* 8\*\*i

i += 1

list=[]

while dec != 0:

list.append(dec%16)

dec = dec // 16

nl=[]

for elem in list[::-1]:

if elem <= 9:

nl.append(str(elem))

else:

nl.append(chr(ord('A') + (elem -10)))

hex = "".join(nl)

return hex

num1 = input("Enter a binary number : ")

print(bin2Dec(num1))

num2 = input("Enter a octal number : ")

print(oct2Hex(num2))

Output:

Enter a binary number : 10111001

185

Enter a octal number : 675

1BD

3B)

sentence = input("Enter a sentence : ")

wordList = sentence.split(" ")

print("This sentence has", len(wordList), "words")

digCnt = upCnt = loCnt = 0

for ch in sentence:

if '0' <= ch <= '9':

digCnt += 1

elif 'A' <= ch <= 'Z':

upCnt += 1

elif 'a' <= ch <= 'z':

loCnt += 1

print("This sentence has", digCnt, "digits", upCnt, "upper case letters", loCnt, "lower case letters")

Output :

Enter a sentence : Rama went to Devaraja market to pick 2 kgs of vegetable

This sentence has 11 words

This sentence has 1 digits 2 upper case letters 42 lower case letters

3B)

str1 = input("Enter String 1 \n")

str2 = input("Enter String 2 \n")

if len(str2) < len(str1):

short = len(str2)

long = len(str1)

else:

short = len(str1)

long = len(str2)

matchCnt = 0

for i in range(short):

if str1[i] == str2[i]:

matchCnt += 1

print("Similarity between two said strings:")

print(matchCnt/long)

Output:

Enter String 1

Python Exercises

Enter String 2

Python Exercises

Similarity between two said strings:

1.0

Enter String 1

Python Exercises

Enter String 2

Python Exercise

Similarity between two said strings:

0.9375

4A)

import random

def merge\_sort(lst):

if len(lst) > 1:

mid = len(lst) // 2

left\_half = lst[:mid]

right\_half = lst[mid:]

merge\_sort(left\_half)

merge\_sort(right\_half)

i = j = k = 0

while i < len(left\_half) and j < len(right\_half):

if left\_half[i] < right\_half[j]:

lst[k] = left\_half[i]

i += 1

else:

lst[k] = right\_half[j]

j += 1

k += 1

while i < len(left\_half):

lst[k] = left\_half[i]

i += 1

k += 1

while j < len(right\_half):

lst[k] = right\_half[j]

j += 1

k += 1

return lst

def insertion\_sort(arr):

for i in range(1, len(arr)):

key = arr[i]

j = i - 1

while j >= 0 and key < arr[j]:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

my\_list = []

for i in range(10):

my\_list.append(random.randint(0, 999))

print("\nUnsorted List")

print(my\_list)

print("Sorting using Insertion Sort")

insertion\_sort(my\_list)

print(my\_list)

my\_list = []

for i in range(10):

my\_list.append(random.randint(0, 999))

print("\nUnsorted List")

print(my\_list)

print("Sorting using Merge Sort")

merge\_sort(my\_list)

print(my\_list)

Output :

Unsorted List

[932, 111, 226, 685, 543, 589, 918, 539, 294, 717]

Sorting using Insertion Sort

[111, 226, 294, 539, 543, 589, 685, 717, 918, 932]

Unsorted List

[613, 176, 828, 265, 65, 326, 359, 919, 514, 868]

Sorting using Merge Sort

[65, 176, 265, 326, 359, 514, 613, 828, 868, 919]

4B)

def roman2Dec(romStr):

roman\_dict ={'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}

romanBack = list(romStr)[::-1]

value = 0

rightVal = roman\_dict[romanBack[0]]

for numeral in romanBack:

leftVal = roman\_dict[numeral]

if leftVal < rightVal:

value -= leftVal

else:

value += leftVal

rightVal = leftVal

return value

romanStr = input("Enter a Roman Number : ")

print(roman2Dec(romanStr))

Output :

Enter a Roman Number : XVII

17

Enter a Roman Number : MLXVI

1066

5A)

import re

def isphonenumber(numStr):

if len(numStr) != 12:

return False

for i in range(len(numStr)):

if i==3 or i==7:

if numStr[i] != "-":

return False

else:

if numStr[i].isdigit() == False:

return False

return True

def chkphonenumber(numStr):

ph\_no\_pattern = re.compile(r'^\d{3}-\d{3}-\d{4}$')

if ph\_no\_pattern.match(numStr):

return True

else:

return False

ph\_num = input("Enter a phone number : ")

print("Without using Regular Expression")

if isphonenumber(ph\_num):

print("Valid phone number")

else:

print("Invalid phone number")

print("Using Regular Expression")

if chkphonenumber(ph\_num):

print("Valid phone number")

else:

print("Invalid phone number")

Output :

Enter a phone number : 444-654-5656

Without using Regular Expression

Valid phone number

Using Regular Expression

Valid phone number

Enter a phone number : 45A4-444-878

Without using Regular Expression

Invalid phone number

Using Regular Expression

Invalid phone number

5B)

import re

phone\_regex = re.compile(r'\+\d{12}')

email\_regex = re.compile(r'[A-Za-z0-9.\_]+@[A-Za-z0-9]+\.[A-Z|a-z]{2,}')

with open('example.txt', 'r') as f:

for line in f:

matches = phone\_regex.findall(line)

for match in matches:

print(match)

matches = email\_regex.findall(line)

for match in matches:

print(match)

Output:

+918151894220

sample@gmail.com

6A)

import os.path

import sys

fname = input("Enter the filename : ")

if not os.path.isfile(fname):

print("File", fname, "doesn't exists")

sys.exit(0)

infile = open(fname, "r")

lineList = infile.readlines()

for i in range(20):

print(i+1, ":", lineList[i])

word = input("Enter a word : ")

cnt = 0

for line in lineList:

cnt += line.count(word)

print("The word", word, "appears", cnt, "times in the file")

Output:

Enter the filename : example.txt

1 : +918151894220

2 : sample@gmail.com

3 : 123456

4 : samlegmail

Enter a word : sample

The word the appears 2 times in the file

6B)

import os

import sys

import pathlib

import zipfile

dirName = input("Enter Directory name that you want to backup : ")

if not os.path.isdir(dirName):

print("Directory", dirName, "doesn't exists")

sys.exit(0)

curDirectory = pathlib.Path(dirName)

with zipfile.ZipFile("myZip.zip", mode="w") as archive:

for file\_path in curDirectory.rglob("\*"):

archive.write(file\_path, arcname=file\_path.relative\_to(curDirectory))

if os.path.isfile("myZip.zip"):

print("Archive", "myZip.zip", "created successfully")

else:

print("Error in creating zip archive")

Output:

Enter Directory name that you want to backup : zipDemo

Archive myZip.zip created successfully

7A)

import math

class Shape:

def \_\_init\_\_(self):

self.area = 0

self.name = ""

def showArea(self):

print("The area of the", self.name, "is", self.area, "units")

class Circle(Shape):

def \_\_init\_\_(self,radius):

self.area = 0

self.name = "Circle"

self.radius = radius

def calcArea(self):

self.area = math.pi \* self.radius \* self.radius

class Rectangle(Shape):

def \_\_init\_\_(self,length,breadth):

self.area = 0

self.name = "Rectangle"

self.length = length

self.breadth = breadth

def calcArea(self):

self.area = self.length \* self.breadth

class Triangle(Shape):

def \_\_init\_\_(self,base,height):

self.area = 0

self.name = "Triangle"

self.base = base

self.height = height

def calcArea(self):

self.area = self.base \* self.height / 2

c1 = Circle(5)

c1.calcArea()

c1.showArea()

r1 = Rectangle(5, 4)

r1.calcArea()

r1.showArea()

t1 = Triangle(3, 4)

t1.calcArea()

t1.showArea()

Output:

The area of the Circle is 78.53981633974483 units

The area of the Rectangle is 20 units

The area of the Triangle is 6.0 units

7B)

class Employee:

def \_\_init\_\_(self):

self.name = ""

self.empId = ""

self.dept = ""

self.salary = 0

def getEmpDetails(self):

self.name = input("Enter Employee name : ")

self.empId = input("Enter Employee ID : ")

self.dept = input("Enter Employee Dept : ")

self.salary = int(input("Enter Employee Salary : "))

def showEmpDetails(self):

print("Employee Details")

print("Name : ", self.name)

print("ID : ", self.empId)

print("Dept : ", self.dept)

print("Salary : ", self.salary)

def updtSalary(self):

self.salary = int(input("Enter new Salary : "))

print("Updated Salary", self.salary)

e1 = Employee()

e1.getEmpDetails()

e1.showEmpDetails()

e1.updtSalary()

Output:

Enter Employee name : raman

Enter Employee ID : 105

Enter Employee Dept : CSE

Enter Employee Salary : 10000

Employee Details

Name : raman

ID : 105

Dept : CSE

Salary : 10000

Enter new Salary : 99800

Updated Salary 99800

8A)

class PaliStr:

def \_\_init\_\_(self):

self.isPali = False

def chkPalindrome(self, myStr):

if myStr == myStr[::-1]:

self.isPali = True

else:

self.isPali = False

return self.isPali

class PaliInt(PaliStr):

def \_\_init\_\_(self):

self.isPali = False

def chkPalindrome(self, val):

temp = val

rev = 0

while temp != 0:

dig = temp % 10

rev = (rev\*10) + dig

temp = temp //10

if val == rev:

self.isPali = True

else:

self.isPali = False

return self.isPali

st = input("Enter a string : ")

stObj = PaliStr()

if stObj.chkPalindrome(st):

print("Given string is a Palindrome")

else:

print("Given string is not a Palindrome")

val = int(input("Enter a integer : "))

intObj = PaliInt()

if intObj.chkPalindrome(val):

print("Given integer is a Palindrome")

else:

print("Given integer is not a Palindrome")

Output:

Enter a string : madam

Given string is a Palindrome

Enter a integer : 567587

Given integer is not a Palindrome

Enter a string : INDIA

Given string is not a Palindrome

Enter a integer : 6789876

Given integer is a Palindrome

9A)

import requests

import os

from bs4 import BeautifulSoup

url = 'https://xkcd.com/1/'

if not os.path.exists('xkcd\_comics'):

os.makedirs('xkcd\_comics')

while True:

res = requests.get(url)

res.raise\_for\_status()

soup = BeautifulSoup(res.text, 'html.parser')

comic\_elem = soup.select('#comic img')

if comic\_elem == []:

print('Could not find comic image.')

else:

comic\_url = 'https:' + comic\_elem[0].get('src')

print(f'Downloading {comic\_url}...')

res = requests.get(comic\_url)

res.raise\_for\_status()

image\_file = open(os.path.join('xkcd\_comics', os.path.basename(comic\_url)), 'wb')

for chunk in res.iter\_content(100000):

image\_file.write(chunk)

image\_file.close()

prev\_link = soup.select('a[rel="prev"]')[0]

if not prev\_link:

break

url = 'https://xkcd.com' + prev\_link.get('href')

print('All comics downloaded.')

Output:

Downloading https://imgs.xkcd.com/comics/barrel\_cropped\_(1).jpg...

Downloading https://imgs.xkcd.com/comics/radians\_are\_cursed.png...

Downloading https://imgs.xkcd.com/comics/presents\_for\_biologists.png...

Downloading https://imgs.xkcd.com/comics/launch\_window.png...

Downloading https://imgs.xkcd.com/comics/obituary\_editor.png...

Downloading https://imgs.xkcd.com/comics/fanservice.png...

Downloading https://imgs.xkcd.com/comics/hand\_dryers.png...

9B)

from openpyxl import Workbook

from openpyxl.styles import Font

wb = Workbook()

sheet = wb.active

sheet.title = "Language"

wb.create\_sheet(title = "Capital")

lang = ["Kannada", "Telugu", "Tamil"]

state = ["Karnataka", "Telangana", "Tamil Nadu"]

capital = ["Bengaluru", "Hyderabad", "Chennai"]

code =['KA', 'TS', 'TN']

sheet.cell(row = 1, column = 1).value = "State"

sheet.cell(row = 1, column = 2).value = "Language"

sheet.cell(row = 1, column = 3).value = "Code"

ft = Font(bold=True)

for row in sheet["A1:C1"]:

for cell in row:

cell.font = ft

for i in range(2,5):

sheet.cell(row = i, column = 1).value = state[i-2]

sheet.cell(row = i, column = 2).value = lang[i-2]

sheet.cell(row = i, column = 3).value = code[i-2]

wb.save("demo.xlsx")

sheet = wb["Capital"]

sheet.cell(row = 1, column = 1).value = "State"

sheet.cell(row = 1, column = 2).value = "Capital"

sheet.cell(row = 1, column = 3).value = "Code"

ft = Font(bold=True)

for row in sheet["A1:C1"]:

for cell in row:

cell.font = ft

for i in range(2,5):

sheet.cell(row = i, column = 1).value = state[i-2]

sheet.cell(row = i, column = 2).value = capital[i-2]

sheet.cell(row = i, column = 3).value = code[i-2]

wb.save("demo.xlsx")

srchCode = input("Enter state code for finding capital ")

for i in range(2,5):

data = sheet.cell(row = i, column = 3).value

if data == srchCode:

print("Corresponding capital for code", srchCode, "is", sheet.cell(row = i, column = 2).value)

sheet = wb["Language"]

srchCode = input("Enter state code for finding language ")

for i in range(2,5):

data = sheet.cell(row = i, column = 3).value

if data == srchCode:

print("Corresponding language for code", srchCode, "is", sheet.cell(row = i, column = 2).value)

wb.close()

Output:

Enter state code for finding capital KA

Corresponding capital for code KA is Bengaluru

Enter state code for finding language TS

Corresponding language for code TS is Telugu

10A)

from PyPDF2 import PdfWriter, PdfReader

num = int(input("Enter page number you want combine from multiple documents "))

pdf1 = open('birds.pdf', 'rb')

pdf2 = open('birdspic.pdf', 'rb')

pdf\_writer = PdfWriter()

pdf1\_reader = PdfReader(pdf1)

page = pdf1\_reader.pages[num - 1]

pdf\_writer.add\_page(page)

pdf2\_reader = PdfReader(pdf2)

page = pdf2\_reader.pages[num - 1]

pdf\_writer.add\_page(page)

with open('output.pdf', 'wb') as output:

pdf\_writer.write(output)

Output:

Enter page number you want combine from multiple documents 3

birdsDownload

birdspicDownload

outputDownload

10B)

Fetch weather data from the JSON

import json

with open('weather\_data.json') as f:

data = json.load(f)

current\_temp = data['main']['temp']

humidity = data['main']['humidity']

weather\_desc = data['weather'][0]['description']

print(f"Current temperature: {current\_temp}°C")

print(f"Humidity: {humidity}%")

print(f"Weather description: {weather\_desc}")

JSON File :

{

"coord": {

"lon": -73.99,

"lat": 40.73

},

"weather": [

{

"id": 800,

"main": "Clear",

"description": "clear sky",

"icon": "01d"

}

],

"base": "stations",

"main": {

"temp": 15.45,

"feels\_like": 12.74,

"temp\_min": 14.44,

"temp\_max": 16.11,

"pressure": 1017,

"humidity": 64

},

"visibility": 10000,

"wind": {

"speed": 4.63,

"deg": 180

},

"clouds": {

"all": 1

},

"dt": 1617979985,

"sys": {

"type": 1,

"id": 5141,

"country": "US",

"sunrise": 1617951158,

"sunset": 1618000213

},

"timezone": -14400,

"id": 5128581,

"name": "New York",

"cod": 200

}

Output:

Current temperature: 15.45°C

Humidity: 64%

Weather description: clear sky

PART-B

1)

Input :

from datetime import date

perName = input("Enter the name of the person:")

perDOB = int(input("Enter his year of birth:"))

curYear = date.today().year

perAge = curYear - perDOB

if(perAge>60):

print(perName,"aged",perAge,"years is a senior citizen.")

else:

print(perName,"aged",perAge,"years is not a senior citizen.")

Output:

Enter the name of the person:Ajay

Enter his year of birth:2001

Ajay aged 22 years is not a senior citizen.

Enter the name of the person:vijay

Enter his year of birth:1947

Vijay aged 76 years is a senior citizen

2)

Input:

def fact(num):

if num == 0:

return 1

else:

return num\*fact(num-1)

n = int(input("Enter the value of N:"))

r = int(input("Enter the value of R:"))

nCr = fact(n)/(fact(r)\*fact(n-r))

print(n,'C',r,"=","%d"%nCr,sep = "")

Output:

Enter the value of N:5

Enter the value of R:2

5 C 2 = 10

3)

Input:

from math import sqrt

myList = []

num = int(input("Enter the number of elements in your list :"))

for i in range(num):

val = int(input("Enter the element :"))

myList.append(val)

print("The length of list is", len(myList))

print('List Contents are',myList)

total = 0

for elem in myList:

total += elem

mean = total/num

total = 0

for elem in myList:

total += (elem - mean)\*(elem - mean)

variance = total/num

stdDev = sqrt(variance)

print("Mean =",mean)

print("Variance =",variance)

print("Standard Deviation =","%.2f"%stdDev)

Output:

Enter the number of elements in your list :4

Enter the element :10

Enter the element :6

Enter the element :15

Enter the element :12

The length of list is 4

List Contents are [10, 6, 15, 12]

Mean = 10.75

Variance = 10.6875

Standard Deviation = 3.27

4)

Inout:

num = input("Enter a number :")

print("The number entered is:",num)

uniqDig = set(num)

for elem in uniqDig:

print(elem,"occurs",num.count(elem),"times")

Output:

Enter a number :345634231

The number entered is: 345634231

6 occurs 1 times

1 occurs 1 times

5 occurs 1 times

2 occurs 1 times

4 occurs 2 times

3 occurs 3 times

5)

Input:

import sys

def DivExp(a,b):

assert a>0,"a should be greater than 0"

try:

c=a/b

except ZeroDivisionError:

print("value of b cannot be zero")

sys.exit(0)

else:

return c

val1 = int(input("Enter a value for a:"))

val2 = int(input("Enter a value for b:"))

val3 = DivExp(val1,val2)

print(val1,"/",val2,"=",val3)

Output:

Enter a value for a:25

Enter a value for b:5

25 / 5 = 5.0