Week – 1: DICTIONARY:

a. Write a program to count the numbers of characters in the string and store them in a dictionary data structure

```
string = input()
char_dict = {}

for items in string:
    if items in char_dict.keys():
        if(items == ' '):
            char_dict["whitespace"] += 1
        else:
            char_dict[items] += 1
    else:
        if(items == ' '):
            char_dict["whitespace"] = 1
        else:
            char_dict[items] = 1

print(char_dict)
```

Output:

```
hey there
{'h': 2, 'e': 3, 'y': 1, 'whitespace': 1, 't': 1, 'r': 1}
```

b. Write a program to use split and join methods in the string and trace a birthday with a dictionary data structure.

```
personal_dict = {}
for _ in range(int(input("Enter the number of Entries: "))):
    string = input("Enter Name DOB(DD-MM-YYYY). Follow the format!:
").split()

    name = string[0]

    dob_lst = string[1].split('-')

    personal_dict["/".join(dob_lst)] = name

dob_search = input("Enter dob of person: ")
if dob_search in personal_dict.keys():
    print(f'Name of person with DOB of {dob_search} is:
{personal_dict[dob_search]}')
else:
    print(f"DOB with {dob_search} not present in records!")
```

Output:

```
Enter the number of Entries: 2
Enter Name DOB(DD-MM-YYYY). Follow the format!: Nishant 14-11-2002
Enter Name DOB(DD-MM-YYYY). Follow the format!: Crazyboiii 15-07-2002
Enter dob of person: 14/11/2002
Name of person with DOB of 14/11/2002 is: Nishant
```

c. Write a program combine_lists that combines these lists into a dictionary.

```
dictionary =
dict(zip(list(map(int,input().split())),list(map(int,input().split()))))

print(dictionary)
```

Output:

```
1 2 3 4 5
10 20 30 40 50
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
```

d. Write a Python program to find shortest list of values with the keys in a given dictionary.

```
dictionary = {}
for _ in range(int(input("Enter no of lists: "))):
  lst = list(map(int,input(f"Enter list{_+1} data separated by space:
").split()))

  length = len(lst)

  if(length not in dictionary.keys()):
    dictionary[length] = [lst]
  else:
    dictionary[length].append(lst)

min_length = min(dictionary.keys())
no_lists = len(dictionary[min_length])
if(no_lists>1):
  print(f"There are {no_lists} lists that have min length of {min_length}:
")
  for items in dictionary[min_length]:
    print(items)
else:
  print("Minimum list is: ")
  print(*dictionary[min_length])
```

Output:

```
Enter no of lists: 2
Enter list1 data separated by space: 1 0
Enter list2 data separated by space: 1 0 1 1 0 10 20
Minimum list is:
[1, 0]
```

Week – 2: NESTED LISTS

a. Write a program to read a 3 X 3 matrix and find the transpose

```
rows,columns = map(int,input('Enter rows and columns of Matrix A(Separated by space): ').split())
matA = []
for _ in range(rows):
    rows_lst = []
    for __ in range(columns):
        rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
    matA.append(rows_lst)

print("MatA: ")

for items in matA:
    print(*items)

transpose = []

for index in range(columns):
    transp_rows = []
    for index2 in range(rows):
        transp_rows.append(matA[index2][index])
    transpose.append(transp_rows)

print("Transpose: ")
for items in transpose:
    print(*items)
```

Output:

```
Enter rows and columns of Matrix A(Separated by space): 3 3
Row 1 Element 1: 2
Row 1 Element 2: 3
Row 1 Element 3: 4
Row 2 Element 1: 1
Row 2 Element 2: 2
Row 2 Element 3: 3
Row 3 Element 1: 4
Row 3 Element 2: 5
Row 3 Element 3: 6
MatA:
2 3 4
1 2 3
4 5 6
Transpose:
2 1 4
3 2 5
4 3 6
```

b. Write a program to perform addition, subtraction of two 3 X 3 matrices.

```
rowsA,columnsA = map(int,input('Enter rows and columns of Matrix A
(Separated by space): ').split())

rowsB,columnsB = map(int,input('Enter rows and columns of Matrix
B(Separated by space): ').split())

if(rowsA == rowsB and columnsA == columnsB):
```

```
    matA = []
    matB = []

    for _ in range(rowsA):
        rows_lst = []
        for __ in range(columnsA):
            rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
        matA.append(rows_lst)

    for _ in range(rowsB):
        rows_lst = []
        for __ in range(columnsB):
            rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
        matB.append(rows_lst)

    print("MatA: ")

    for items in matA:
        print(*items)

    print("MatB: ")

    for items in matB:
        print(*items)

    add = []
    sub = []

    for index in range(rowsA):
        add_row = []
        sub_row = []
        for index2 in range(columnsA):
            add_row.append(matA[index][index2]+matB[index][index2])
            sub_row.append(matA[index][index2]-matB[index][index2])
        add.append(add_row)
        sub.append(sub_row)

    print("Matrix Add Result: ")
    for items in add:
        print(*items)

    print("Matrix Sub Result: ")
    for items in sub:
        print(*items)
else:
    print("Matrix addition and subtraction is not possible!")
```

Output:

```
Enter rows and columns of Matrix A(Separated by space): 2 2
Enter rows and columns of Matrix B(Separated by space): 2 2
Row 1 Element 1: 1
Row 1 Element 2: 1
Row 2 Element 1: 1
Row 2 Element 2: 1
Row 1 Element 1: 2
Row 1 Element 2: 2
Row 2 Element 1: 2
Row 2 Element 2: 2
MatA:
1 1
1 1
MatB:
2 2
2 2
Matrix Add Result:
3 3
3 3
Matrix Sub Result:
-1 -1
-1 -1
```

c. Write a program to perform multiplication of two 3 X 3 matrices

```
rowsA,columnsA = map(int,input('Enter rows and columns of Matrix A(Separated
by space): ').split())
rowsB,columnsB = map(int,input('Enter rows and columns of Matrix B(Separated
by space): ').split())
if(columnsA == rowsB):
    matA = []
    matB = []

    for _ in range(rowsA):
        rows_lst = []
        for __ in range(columnsA):
            rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
        matA.append(rows_lst)

    for _ in range(rowsB):
        rows_lst = []
        for __ in range(columnsB):
            rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
        matB.append(rows_lst)

    print("MatA: ")
```

```
    for items in matA:
        print(*items)

    print("MatB: ")

    for items in matB:
        print(*items)

    mult = []

    for i in range(rowsA):
        mult_row = []
        for j in range(columnsB):
            adder = 0
            for k in range(columnsA):
                adder += matA[i][k]*matB[k][j]
            mult_row.append(adder)
        mult.append(mult_row)

    for items in mult:
        print(*items)
else:
    print("Matrix Multiplication not possible")
```

Output:

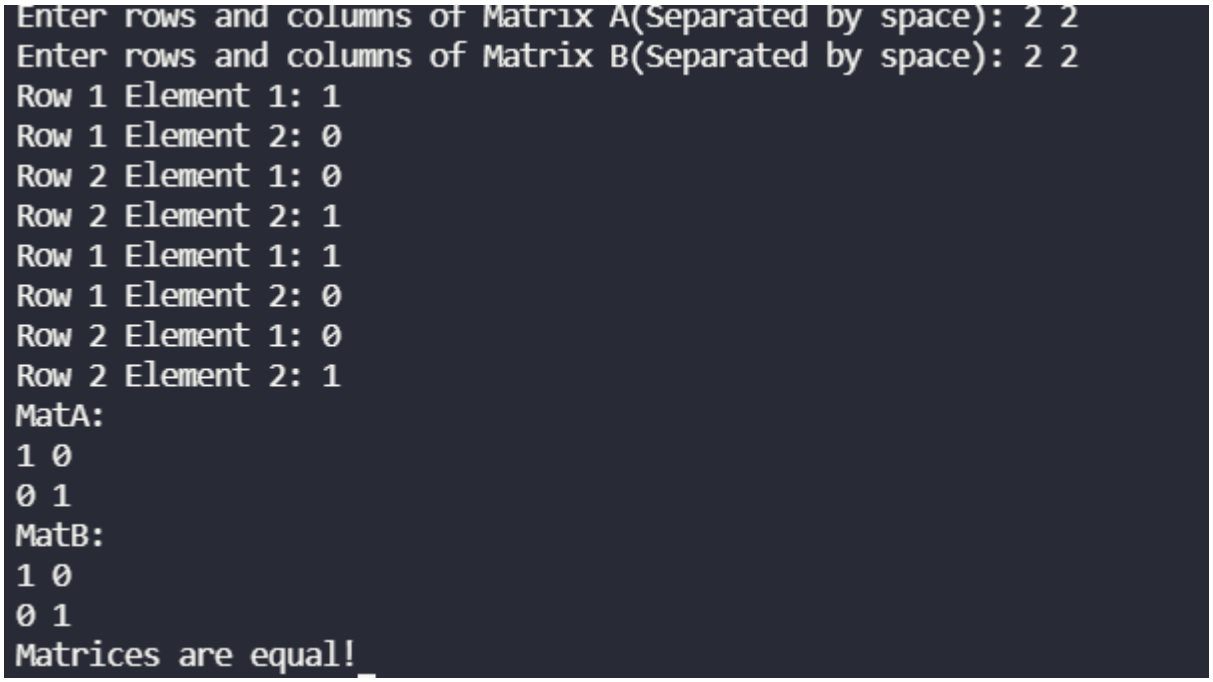
```
Enter rows and columns of Matrix A(Separated by space): 2 3
Enter rows and columns of Matrix B(Separated by space): 3 2
Row 1 Element 1: 1
Row 1 Element 2: 2
Row 1 Element 3: 3
Row 2 Element 1: 4
Row 2 Element 2: 5
Row 2 Element 3: 6
Row 1 Element 1: 7
Row 1 Element 2: 8
Row 2 Element 1: 9
Row 2 Element 2: 0
Row 3 Element 1: 1
Row 3 Element 2: 2
MatA:
1 2 3
4 5 6
MatB:
7 8
9 0
1 2
28 14
79 44
```

d. Write a program to check whether two given 3 X 3 matrices are identical or not.

```
rowsA,columnsA = map(int,input('Enter rows and columns of Matrix
A(Separated by space): ').split())
rowsB,columnsB = map(int,input('Enter rows and columns of Matrix
B(Separated by space): ').split())
if(rowsA == rowsB and columnsA == columnsB):
```

```
    matA = []
    matB = []

    for _ in range(rowsA):
        rows_lst = []
        for __ in range(columnsA):
            rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
        matA.append(rows_lst)

    for _ in range(rowsB):
        rows_lst = []
        for __ in range(columnsB):
            rows_lst.append(int(input(f'Row {_+1} Element {__+1}: ')))
        matB.append(rows_lst)

    print("MatA: ")

    for items in matA:
        print(*items)

    print("MatB: ")

    for items in matB:
        print(*items)

    status = True

    for index in range(rowsA):
        for index2 in range(columnsA):
            if(matA[index][index2] != matB[index][index2]):
                status = False
                break

    if(status == True):
        print("Matrices are equal!")
    else:
        print("Matrices are unequal!")
else:
    print("Matrices are unequal!")
```

Output:


Enter rows and columns of Matrix A(Separated by space): 2 2
Enter rows and columns of Matrix B(Separated by space): 2 2
Row 1 Element 1: 1
Row 1 Element 2: 0
Row 2 Element 1: 0
Row 2 Element 2: 1
Row 1 Element 1: 1
Row 1 Element 2: 0
Row 2 Element 1: 0
Row 2 Element 2: 1
MatA:
1 0
0 1
MatB:
1 0
0 1
Matrices are equal!

Week – 3: USER DEFINED FUNCTIONS:

a. Write a function ball_collide that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding. Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius If (distance between two balls centers) <= (sum of their radii) then (they are colliding)

```
import math as m

def ball_collide(ball1:tuple,ball2:tuple):
    x1,y1,r1 = ball1
    x2,y2,r2 = ball2

    distance_balls = m.sqrt(pow((x1-x2),2)+pow((y1-y2),2))

    rad_sum = r1+r2

    if(distance_balls<=rad_sum):
        return True
    else:
        return False

ball1 = tuple(map(int,input('x1 y1 r1: ').split()))
ball2 = tuple(map(int,input('x2 y2 r2: ').split()))
if(ball_collide(ball1,ball2)):
    print("Balls are colliding")
else:
    print("Collision not Possible")
```

Output:

```
x1 y1 r1: 2 3 10
x2 y2 r2: 2 4 5
Balls are colliding
```

b. Write a function to find mean, median, mode for the given set of numbers in a list.

```
def mmm(numbers:list):
    numbers.sort()
    length_num = len(numbers)
    total = sum(numbers)
    mean = total/length_num
    #median:
    if(length_num%2==0):
        median = (numbers[(length_num//2)-1]+numbers[(length_num//2)])/2
    else:
        median = numbers[length_num//2]
    #mode
```

```
    counter = 0
    dictionary = {}

    for index in range(len(set(numbers))):
        counter = 1
        for index2 in range(index+1,len(numbers)):
            if(numbers[index] == numbers[index2]):
                counter += 1
        dictionary[counter] = numbers[index]

    mode = dictionary[max(dictionary.keys())]

    return (mean,median,mode)

numbers = list(map(int,input().split()))

print("Mean Median Mode:")
print(*mmm(numbers))
```

Output:

```
1 2 2 2 2 3 4 5
Mean Median Mode:
2.625 2.0 2
```

c. Write a function nearly_equal to test whether two strings are nearly equal. Two strings a and b are nearly equal when a can be generated by a single mutation on b. (HINT:- Nearly equal string : Python – Jython, Perl – Pearl)

```
def nearly_equal(str1:str,str2:str):
    count=0
    index1 = index2 = 0
    while(index1<len(str1) and index2<len(str2)):
        if(str1[index1]!=str2[index2]):
            count += 1
            if(len(str1)>len(str2)):
                index1=index1+1
            if(len(str1)<len(str2)):
                index1=index1-1
        if(count>1):
            return False
        index1 += 1
        index2 += 1
    if(count<2):
            return True

word1 = input()
word2 = input()
```

```
if(nearly_equal(word1,word2)):
    print("Nearly Equal")
else:
    print("Not Nearly Equal")
```

Output:

```
Python
Jython
Nearly Equal
```

d.

The mathematician Srinivasa Ramanujan found an infinite series that can be used to generate a numerical approximation of $1/\pi$:

Write a function called estimate_pi that uses this formula to compute and return an estimate of π. It should use a while loop to compute terms of the summation until the last term is smaller than 1e-15 (which is Python notation for 10^{-15}).

You can check the result by comparing it to math.pi.

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{(k!)^4 396^{4k}}$$

```
# Mission - Estimate pi:

import math as m
def estimate_pi():
    constant = (2*m.sqrt(2))/9801
    k = 0
    estimate = 0
    while(1):
        element1 = (m.factorial(4*k))*(1103+(26390*k))
        element2 = pow(m.factorial(k),4)*pow(396,(4*k))

        element = constant*(element1/element2)

        if(element<= 1e-15):
            break
        else:
            estimate += element

        k+= 1

    return 1/estimate

print(estimate_pi())
print(m.pi)
```

Output:

```
3.1415926535897936
3.141592653589793
```

Week – 4: MODULES:

a. Install packages requests, flask and explore using (pip)

```
pip install requests
pip install Flask
pip install explore
```

b. Write a Python program that imports requests and fetch content from wiki page

```
import requests

r = requests.get("www.google.com")

print(r.content)
```

c. Write a Python program to generate a series of unique random numbers by using random module

```
import random
n = int(input('Enter no of integers for series: '))
counter = 0
lst = []

while(counter < n):
  random_no = random.randint(1,pow(2,n))

  if random_no not in lst:
    lst.append(random_no)
    counter += 1
  else:
    continue

for items in lst:
  print(items,end=', ')
```

d. Write a Python program to find the substrings within a string using re module.

```
import re

string = input('Enter string: ')
substring = input('Substring: ')

x = re.findall(substring,string)

print(x)
```

Week – 5: DATE AND TIME:

a. Demonstrate Basic date and time classes, Different time formats, Converting between formats, Formatting dates and times, Parsing date/time information.

```
import datetime
import time

#different date and time classes

x = datetime.datetime.now() #datetime class

time_secs = time.time()
print(time_secs)

print(str(x))

#Formating

print(x.strftime("%H:%M:%S")) #Prints 24 hour time
print(x.strftime("%d %B %Y, %A %I:%M:%S %p")) #Local Version of date time

#Converting B/w Formats:

ch = int(input('Enter 1 for 24hr and 2 for 12hr: '))

if(ch == 1):
    time_input = str(input('Enter 24hr format: '))

    x = datetime.datetime.strptime(time_input,"%H:%M:%S")
    time_12 = x.strftime("%I:%M:%S %p")

    print(time_12)
if(ch == 2):
    time_input = str(input('Enter 12hr format: '))

    x = datetime.datetime.strptime(time_input,"%I:%M:%S %p")
    time_24 = x.strftime("%H:%M:%S")

    print(time_24)
#Parsing Date time:

datetime_input = input('Enter Date(DD MM YYYY) time(HH MM SS)')

date_time = datetime.datetime.strptime(datetime_input,"%d %m %Y %H %M %S")

print(date_time.strftime("%d %B %Y, %H:%M:%S"))
```

Output:

```
1649213769.6720846
2022-04-06 08:26:09.672084
08:26:09
06 April 2022, Wednesday 08:26:09 AM
Enter 1 for 24hr and 2 for 12hr: 1
Enter 24hr format: 19:00:00
07:00:00 PM
Enter Date(DD MM YYYY) time(HH MM SS)14 11 2022 05 30 30
14 November 2022, 05:30:30
```

b. Write a script that reads the current time and converts it to a time of day in hours, minutes, and seconds, plus the number of days since the epoch.

```
import math

time_epoch = time.time() #secs

mins = time_epoch/60

hours = mins/60

days = math.floor(hours/24)

date = datetime.datetime.now()

date.strftime("%H %M %S")

print(f'Today is {date.hour} hours, {date.minute} minutes, {date.second}
seconds and {days} days since epoch')
```

Output:

```
Today is 8 hours, 29 minutes, 32 seconds and 19088 days since epoch
```

c. Design a Python script to determine the difference in date for given two dates in YYYY:MM:DD format(0 <= YYYY <= 9999, 1 <= MM <= 12, 1 <= DD <= 31) following the leap year rules

```
date1 = input()
date2 = input()

d1 = datetime.datetime.strptime(date1,"%d/%m/%Y")
d2 = datetime.datetime.strptime(date2,"%d/%m/%Y")

print(abs(d1-d2).days,'days')
```

Output:

```
14/11/2020
09/09/2020
66  days
```

d. Design a Python Script to determine the time difference between two given times in HH:MM:SS format.(0 <= HH <= 23, 0 <= MM <= 59, 0 <= SS <= 59).

```
time1 = input()
time2 = input()

t1 = datetime.datetime.strptime(time1,"%H:%M:%S")
t2 = datetime.datetime.strptime(time2,"%H:%M:%S")

print(abs(t1-t2))
```

Output:

```
10:14:02
11:12:01
0:57:59
```

Week – 6: CLASS AND OBJECTS

a. Create a class ATM and define ATM operations to create account, deposit, check_balance, withdraw and delete account. Use constructor to initialize members.

```
import random

class ATM:
    name = ''
    def __init__(self):
        self.pin = 0
        self.balance = 0
        self.accountNo = 0

    def createAccount(self,pin,name):
        self.accountNo = random.randint(900000000000,1000000000000)
        self.name = name
        self.pin = pin
        print(f'Account Created! Thank You {self.name}!')

    def deposit(self,amount:int):
        self.balance += amount
        print(f'{amount} deposited successfully!')

    def withdraw(self,amount:int):
        if(amount>self.balance):
            print(f'Your balance is low - Rs. {self.balance}')
        else:
            print(f'{amount} withdrawn from Account!')
            self.balance -= amount

    def check_Balance(self):
        print(f'Your balance is: Rs. {self.balance}!')

    def deleteAccount(self):
        print(f'Account {self.accountNo} deleted successfully!')

    def check_AccOpen(self,pin):
        if(self.pin == pin):
            print(f'Welcome {self.name}!')
            print(f'Opening Menu, please wait!')
        else:
            print('Incorrect pin! ')

#Driver Code:

name = input('Enter your Full name: ')
pin = int(input('Please create pin: '))
```

```
atm = ATM()
atm.createAccount(pin,name)

check_pin = int(input('Enter your pin to open account: '))
atm.check_AccOpen(check_pin)

print('''#Select an Option:\n
1. Deposit Amount
2. Check Balance
3. Withdraw Amount
4. Delete Account
''')

while(True):
    ch = int(input('Enter your choice: '))

    if(ch == 1):
        amount = int(input('Enter deposit amount: '))
        atm.deposit(amount)

    elif(ch == 2):
        atm.check_Balance()

    elif(ch == 3):
        amount = int(input('Enter withdrawal amount: '))
        atm.withdraw(amount)

    elif(ch == 4):
        atm.deleteAccount()
        del atm
        print('Terminating Program!')
        break
    else:
        print('Exiting Program!')
        break
```

Output:

```
Enter your Full name: Nishant Ghosh
Please create pin: 241421
Account Created! Thank You Nishant Ghosh!
Enter your pin to open account: 241421
Welcome Nishant Ghosh!
Opening Menu, please wait!
#Select an Option:

1. Deposit Amount
2. Check Balance
3. Withdraw Amount
4. Delete Account

Enter your choice: 1
Enter deposit amount: 1000
1000 deposited successfully!
Enter your choice: 2
Your balance is: Rs. 1000!
Enter your choice: 3
Enter withdrawal amount: 2
2 withdrawn from Account!
Enter your choice: 4
Account 962623002517 deleted successfully!
Terminating Program!
```

b. Make a class Employee with a name and salary. Make a class Manager inherit from Employee. Add an instance variable, named department. Write a method that prints manager's name, department and salary. Make a class Executive inherit from Manager. Write a method that prints the string "Executive" followed by the information stored in the Manager super class object.

```
class Employee:
    salary = 0
    name = ''

class Manager(Employee):
    department = ""
    def __init__(self):
        Employee.name = input('Enter name: ')
        Employee.salary = int(input('Enter salary: '))
        Manager.department = input('Enter department: ')

    def showDetails(self):
        print(f'Name: {Employee.name}, Salary: {Employee.salary},
Department: {Manager.department}')

class Executive(Manager):
    def __init__(self):
```

```
        pass

    def display(self):
        print('Executive')
        super().showDetails()

#Driver Code:

manager = Manager()

manager.showDetails()

exe = Executive()
exe.display()
```

Output:

```
Enter name: Nishant
Enter salary: 200000
Enter department: CSE
Name: Nishant, Salary: 200000, Department: CSE
Executive
Name: Nishant, Salary: 200000, Department: CSE
```

c. A hospital wants to create a database regarding its indoor patients. The information to store include a) Name of the patient b) Date of admission c) Disease d) Date of discharge. Create a structure to store the date (year, month and date as its members). Create a base class to store the above information. The member function should include functions to enter information and display a list of all the patients in the database. Create a derived class to store the age of the patients. List the information about to store the age of the patients. List the information about all the pediatric patients (less than twelve years in age).

```
import random
dictionary_hosp = {}

def getPatientNo():
    patient_no = random.randint(pow(10, 6), pow(10, 8))
    while(patient_no in dictionary_hosp):
        patient_no = random.randint(pow(10, 6), pow(10, 8))

    return patient_no

class Patient:
    dod = 'DD/MM/YYYY'

    def get_Details(self):
        patient_no = getPatientNo()
```

```
        self.name = input('Enter name: ')
        self.disease = input('Name of disease: ')
        self.doa = input('Enter doa in DD/MM/YY: ')

        dictionary_hosp[patient_no] = {'name': self.name, 'disease':
self.disease,
                                       'Date of Admission': self.doa, 'Date
of Discharge': self.dod}
        print('Your patient no is:', patient_no, 'please remember!')

    def display_details(self, patient_no: int):
        if patient_no in dictionary_hosp:
            print(dictionary_hosp[patient_no])
        else:
            print('Invalid Patient No.')

class AgeUpdate(Patient):
    def updateAge(self, patient_no: int, age: int):
        if patient_no in dictionary_hosp:
            dictionary_hosp[patient_no].update({'age': age})
        else:
            print('Invalid Patient No.')

    def list_pediatric(self):
        for patients in dictionary_hosp.keys():
            if dictionary_hosp[patients]['age'] < 12:
                print(dictionary_hosp[patients])

# Driver Code:

print('1. Enter Patient Information\n2.Update age\n3.List pediatric\n4.List
details\n5.Quit')
patientOb = Patient()
update_age = AgeUpdate()
while(True):
    ch = int(input('Enter your choice: '))

    if ch == 1:
        patientOb.get_Details()

    if ch == 2:
        update_age.updateAge(
            int(input('Enter patient no.')), int(input('Enter age')))

    if ch == 3:
        update_age.list_pediatric()

    if ch == 4:
```

```
        patientOb.display_details(int(input('Patient No: ')))

    if ch == 5:
        break
```

Output:

```
1. Enter Patient Information
2.Update age
3.List pediatric
4.List details
5.Quit
Enter your choice: 1
Enter name: Nishant
Name of disease: Meh
Enter doa in DD/MM/YY: 12/11/2002
Your patient no is: 73798497 please remember!
Enter your choice: 2
Enter patient no.73798497
Enter age20
Enter your choice: 3
Enter your choice: 4
Patient No: 73798497
{'name': 'Nishant', 'disease': 'Meh', 'Date of Admission': '12/11/2002', 'Date of Discharge': 'DD/MM/YYYY', 'age': 20}
Enter your choice: 2
Enter patient no.73798497
Enter age11
Enter your choice: 3
{'name': 'Nishant', 'disease': 'Meh', 'Date of Admission': '12/11/2002', 'Date of Discharge': 'DD/MM/YYYY', 'age': 11}
```

Week 8 Exception Handling:

a. Read two numbers n1 and n2. Write a function to compute n1/n2 and use try/except to catch the exceptions.

```
def compute():
    n1 = int(input())
    n2 = int(input())

    try:
        print(n1/n2)
    except ArithmeticError:
        print('Dividing By Zero Error!')


#Driver:
compute()
```

Output:

```
1
0
Dividing By Zero Error!
```

b. Write a Python program to detect and handle the exception while solving the quadratic equation.

```
import math as m
lst = input().split()

try:
    a,b,c = map(int,lst)

    d = b*b - (4*a*c)

    if d>0:
        r1 = (-b + m.sqrt(d))/(2*a)
        r2 = (-b - m.sqrt(d))/(2*a)
        print('%.2f %.2f'%(r1,r2))
    elif d == 0:
        r1 = -b/(2*a)
        print('%.2f'%r1)
    else:
        print('Complex roots!')
except ValueError as v:
    print("Error:",v)
```

Output:

```
a 1 2
Error detected. Please check input:  invalid literal for int() with base 10: 'a'
```

c. Write a Python program to handle the run time errors while doing file handling operation

```
try:
    filename = input('Enter filename with .txt extension: ')
    fptr = open(filename,'r')
    print(fptr.read())
except FileNotFoundError:
    print('Error! The requested file isn\'t found')
```

Output:

```
Enter filename with .txt extension: a.txt
Error! The requested file isn't found
```

d. Write a Python program to create and raise user defined exception

```
class MyException(Exception):
    def __init__(self,exception_message):
        self.exception_message = exception_message

try:
    exception_ob = MyException(input('Enter your exception message: '))
    raise exception_ob
except MyException as err:
    print('Error Message:',err)
```

Output:

```
Enter your exception message: Hey broooo
Error Message: Hey broooo
```

Week 9 File Handling:

a.Write a python program to create two threads to keep a count of number of even numbers entered by the user

```
from threading import Thread

print('Enter the list of numbers below')
lst = list(map(int, input().split()))

def even_check(lst):
      c = 0
      for i in lst:
            if i%2 == 0:
                  c+=1
      print(f'Number of Even numbers in the list: {c}')

def odd_check(lst):
      c = 0
      for i in lst:
            if i%2 != 0:
                  c+=1
      print(f'Number of Odd numbers in the list: {c}')

odd_thread = Thread(target=odd_check(lst))
even_thread = Thread(target=even_check(lst))

odd_thread.start()
even_thread.start()
```

Input:

Enter the list of numbers below

1 2 3 4 5 6

Output:

Number of Odd numbers in the list: 3

Number of Even numbers in the list: 3

b. Write a test-case to check the function even_numbers which return True on passing a list of all even numbers.

```
print("Enter the list of even numbers below:")
lst = list(map(int, input().split()))

def even_numbers(lst):
      count  = 0
      n = len(lst)
      for i in lst:
            if i%2 == 0 :
                  count += 1
      if n == count:
            return True
      else:
            return False

print(even_numbers(lst))
```

Input:

Enter the list of even numbers below:

2 4 6 8 10

Output:

True

c.Write a test-case to check the function reverse_string which returns the reversed string.

```
s = input('Enter the string: ').strip()

def reverse_string(s):
      return s[::-1]

print(reverse_string(s))
```

Input:

Enter the string: hello

Output:

olleh

Week 10 – NumPy:

a. Using Numpy, write a basic array of operations on a single array to add x to each element of array and subtract y from each element of array.

```
import numpy as np

def add_to_array(arr, x):
      for i in range(len(arr)):
            arr[i] += x
      print(arr)

def sub_from_array(arr, y):
      for i in range(len(arr)):
            arr[i] -= y
      print(arr)

print("Enter the list of elements below: ")
lst = list(map(int, input().split()))
arr = np.array(lst)

while True:
      print("1: Add to Array Elements")
      print("2: Subtract from Array Elements")
      print("3: End Program")
      print("Select any one from above:")
      option = int(input())
      if option == 1:
            x = int(input("Enter element to be added: "))
            add_to_array(arr, x)
      if option == 2:
            y = x = int(input("Enter element to be subtracted: "))
            sub_from_array(arr,y)
      if option == 3:
            break
```

Output:

```
Enter the list of elements below:

1 2 3 4 5 6 7 8 9
```

```
1: Add to Array Elements

2: Subtract from Array Elements

3: End Program

Select any one from above:

1

Enter element to be added: 2

[ 3  4  5  6  7  8  9 10 11]

1: Add to Array Elements

2: Subtract from Array Elements

3: End Program

Select any one from above:

2

Enter element to be subtracted: 2

[1 2 3 4 5 6 7 8 9]

1: Add to Array Elements

2: Subtract from Array Elements

3: End Program

Select any one from above:

3
```

b. Using Numpy, write a program to add, subtract and multiply two matrices.

```
import numpy as np

# creating first matrix
A = np.array([[1, 2], [3, 4]])

# creating second matrix
```

```
B = np.array([[4, 5], [6, 7]])

print("Printing elements of first matrices")
print(A)
print("Printing elements of second matrices")
print(B)

# adding two matrices
print("Addition of two matrices")
print(np.add(A, B))

# subtracting two matrices
print("Subtraction of two matrices")
print(np.subtract(A, B))

# This will return product of two matrices
res = A @ B
print("Multiplication of two matrices")
print(res)
```

Output:

```
Printing elements of first matrices

[[1 2]

 [3 4]]

Printing elements of second matrices

[[4 5]

 [6 7]]

Addition of two matrices

[[ 5  7]

 [ 9 11]]

Subtraction of two matrices
```

```
[[-3 -3]

 [-3 -3]]

Multiplication of two matrices

[[16 19]

 [36 43]]
```

c. Create multi-dimensional arrays and find its shape and dimension

```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])

# Print shape of ndarray
print(f'Shape of the ndrray is {x.shape}')

# Print dimensions of ndarray
print(f'Dimensions of the ndarray is {x.ndim}')
```

Output:

```
Shape of the ndrray is (2, 3)

Dimensions of the ndarray is 2
```

Week 11 – GUI:

a. Design a GUI based calculator to perform arithmetic operations like addition, subtraction, multiplication and division.

```
# import everything from tkinter module
from tkinter import *

# globally declare the expression variable
expression = ""

def press(num):
    global expression
    expression = expression + str(num)
    equation.set(expression)

# Function to evaluate the final expression
def equalpress():
    try:

        global expression
        total = str(eval(expression))

        equation.set(total)
        expression = ""
    except:

        equation.set(" error ")
        expression = ""

def clear():
    global expression
    expression = ""
    equation.set("")


# Driver code
if __name__ == "__main__":
    # create a GUI window
    gui = Tk()

    # set the background colour of GUI window
    gui.configure(background="light blue")
```

```
# set the title of GUI window
gui.title("Simple Calculator")
gui.geometry("270x150")
equation = StringVar()
expression_field = Entry(gui, textvariable=equation)
expression_field.grid(columnspan=4, ipadx=70)

button1 = Button(gui, text=' 1 ', fg='black', bg='white',
                 command=lambda: press(1), height=1, width=7)
button1.grid(row=2, column=0)

button2 = Button(gui, text=' 2 ', fg='black', bg='white',
                 command=lambda: press(2), height=1, width=7)
button2.grid(row=2, column=1)

button3 = Button(gui, text=' 3 ', fg='black', bg='white',
                 command=lambda: press(3), height=1, width=7)
button3.grid(row=2, column=2)

button4 = Button(gui, text=' 4 ', fg='black', bg='white',
                 command=lambda: press(4), height=1, width=7)
button4.grid(row=3, column=0)

button5 = Button(gui, text=' 5 ', fg='black', bg='white',
                 command=lambda: press(5), height=1, width=7)
button5.grid(row=3, column=1)

button6 = Button(gui, text=' 6 ', fg='black', bg='white',
                 command=lambda: press(6), height=1, width=7)
button6.grid(row=3, column=2)

button7 = Button(gui, text=' 7 ', fg='black', bg='white',
                 command=lambda: press(7), height=1, width=7)
button7.grid(row=4, column=0)

button8 = Button(gui, text=' 8 ', fg='black', bg='white',
                 command=lambda: press(8), height=1, width=7)
button8.grid(row=4, column=1)

button9 = Button(gui, text=' 9 ', fg='black', bg='white',
                 command=lambda: press(9), height=1, width=7)
button9.grid(row=4, column=2)

button0 = Button(gui, text=' 0 ', fg='black', bg='white',
```

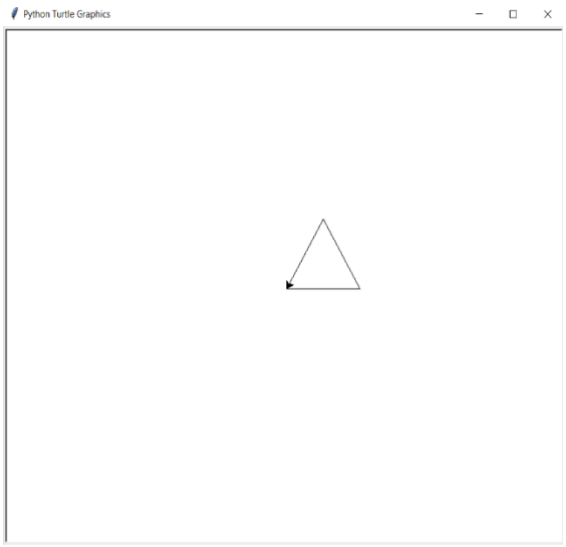
```
                command=lambda: press(0), height=1, width=7)
button0.grid(row=5, column=0)

plus = Button(gui, text=' + ', fg='black', bg='white',
              command=lambda: press("+"), height=1, width=7)
plus.grid(row=2, column=3)

minus = Button(gui, text=' - ', fg='black', bg='white',
              command=lambda: press("-"), height=1, width=7)
minus.grid(row=3, column=3)

multiply = Button(gui, text=' * ', fg='black', bg='white',
                command=lambda: press("*"), height=1, width=7)
multiply.grid(row=4, column=3)

divide = Button(gui, text=' / ', fg='black', bg='white',
                command=lambda: press("/"), height=1, width=7)
divide.grid(row=5, column=3)

equal = Button(gui, text=' = ', fg='black', bg='white',
              command=equalpress, height=1, width=7)
equal.grid(row=5, column=2)

clear = Button(gui, text='Clear', fg='black', bg='white',
              command=clear, height=1, width=7)
clear.grid(row=5, column='1')

Decimal= Button(gui, text='.', fg='black', bg='white',
                command=lambda: press('.'), height=1, width=7)
Decimal.grid(row=6, column=0)
# start the GUI
gui.mainloop()
```

Week 12 – GRAPHICS:

a. Consider turtle object. Write functions to draw triangle, rectangle, polygon, circle and sphere. Use object oriented approach.

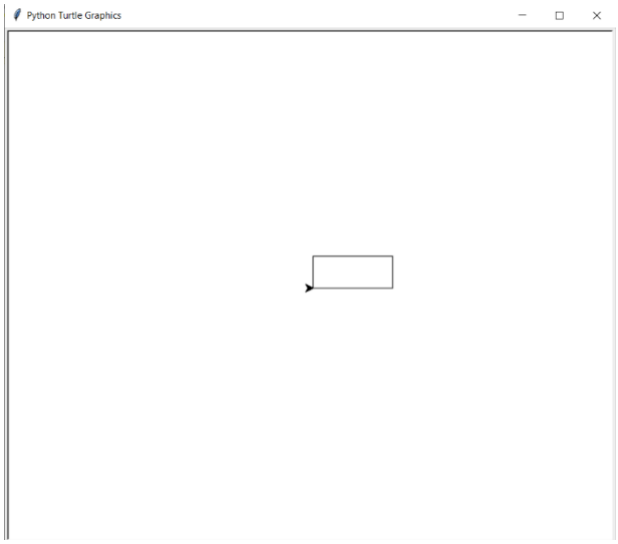
```
# Triangle
import turtle

tur = turtle.Turtle()
tur.forward(100)
tur.left(120)
tur.forward(100)
tur.left(120)
tur.forward(100)
turtle.done()
```

Output:



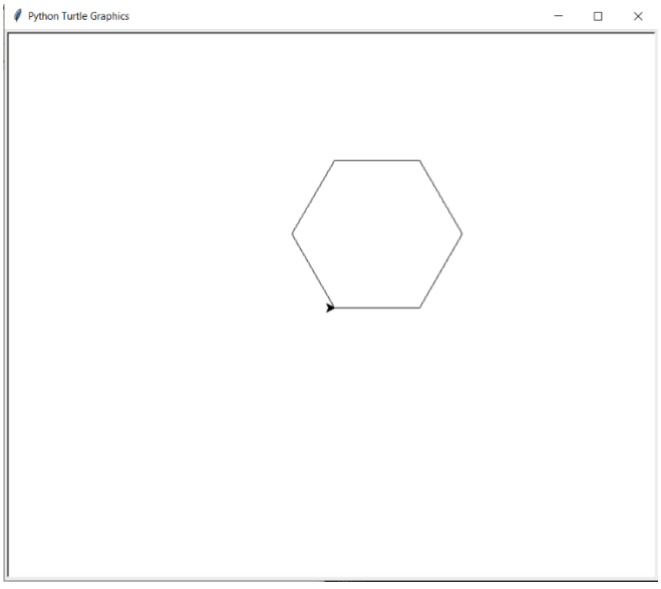

```
#Rectangle
import turtle
t = turtle.Turtle()
l = 100
w = 40
for _ in range(4):
    if _% 2 == 0:
        t.forward(l)
        t.left(90)
    else:
        t.forward(w)
        t.left(90)
turtle.done()
```

Output:



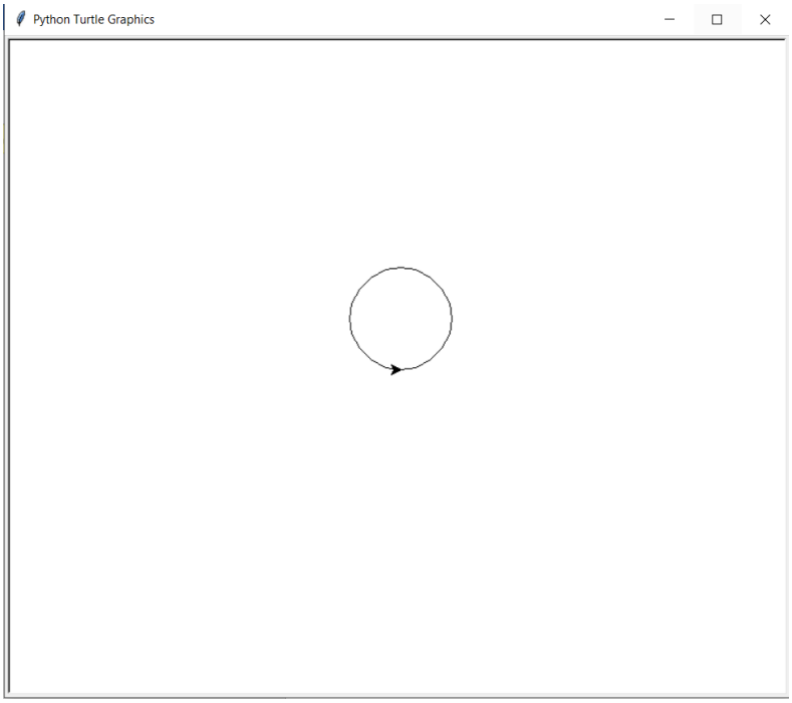

```
#Polygon
import turtle
polygon_ = turtle.Turtle()
for i in range(6):
    polygon_.forward(100)
    polygon_.right(300)
turtle.done()
```

Output:




```
#Circle
import turtle
t = turtle.Turtle()
r = 50
t.circle(r)
turtle.done()
```

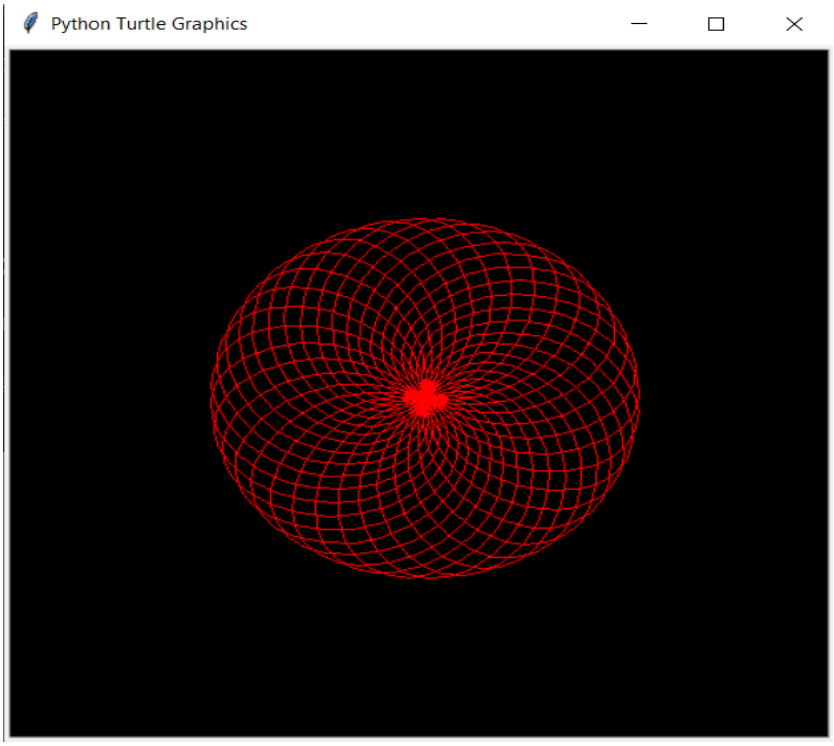
Output:




```
# Sphere
import turtle as tur

screen = tur.Screen()


def draw(rad):

    for i in range(2):
        tur.circle(rad, 90)
        tur.circle(rad//2, 90)


screen.setup(500, 500)

screen.bgcolor('black')

tur.color('red')

val = 10
ind = 0

tur.speed(100)

for i in range(36):

    tur.seth(-val)

    if ind == 5:
```

```
        ind = 0
    else:
        ind += 1

    draw(80)

    val += 10

tur.hideturtle()
tur.done()
```

Output:



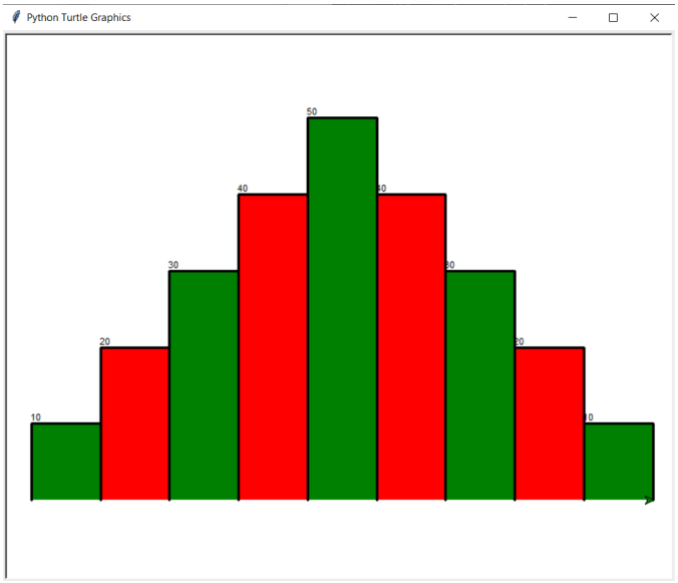
b. Design a Python program using the Turtle graphics library to construct a turtle bar chart representing the grades obtained by N students read from a file categorizing them into distinction, first class, second class, third class and failed.

```
import turtle
def drawBar(t, height, color):
    # Start filling this shape
    t.fillcolor(color)
    t.begin_fill()
    t.left(90)
    t.forward(height)
    t.write(str(height))
    t.right(90)
    t.forward(40)
    t.right(90)
    t.forward(height)
```

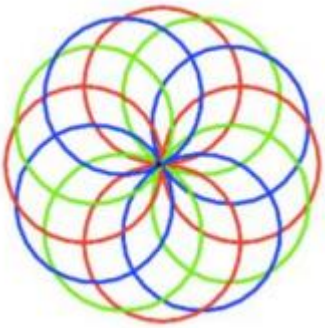
```
    t.left(90)
    # stop filling the shape
    t.end_fill(
# Driver Code

xs = list(map(int,input().split()))
clrs = ["green", "red"]

maxheight = max(xs)
numbers = len(xs)
border = 10
wn = turtle.Screen()
wn.setworldcoordinates(
    0 - border, 0 - border, 40 * numbers + border, maxheight + border
)
# Create tess and set some attributes
tess = turtle.Turtle()
tess.pensize(3)
for i in range(len(xs)):
    drawBar (tess, xs[i],
             clrs[i%2])
wn.exitonclick()
```

Input: 10 20 30 40 50 40 30 20 10

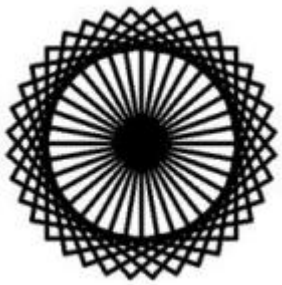
Output:



c. Write a python program to implement the following figures using turtle



```
import turtle as tt
tt.pensize(2)
tt.speed(0)
for i in range(12):
    for color in ('red', 'blue', 'light green'):
        tt.color(color)
        tt.circle(100)
        tt.left(30)
    # Hide the cursor(or turtle) which drew the circle
    tt.hideturtle()
tt.done()
```



```
t = turtle.Turtle()
t.pensize(4)
t.speed(0)
for i in range(36):
    t.left(10)
    t.forward(100)
    t.left(90)
    t.forward(100)
    t.left(90)
    t.forward(100)
    t.left(90)
    t.forward(100)
    t.left(90)
turtle.done()
```