**FEDERAL INSTITUTE OF**

**SCIENCE AND TECHNOLOGY**

**(FISAT)TM**

**HORMIS NAGAR, MOOKKANNOOR**

**ANGAMALY-683577**



‘**FOCUS ON EXCELLENCE’**

**LABORATORY RECORD**

**20MCA131 - PROGRAMMING LAB**

**Name:** **AKASH SANTHOSH**

**Branch: MASTER OF COMPUTER APPLICATION**

**Semester: 1 Batch: 2021 A Roll No: 8**

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(FISAT)TM**

**HORMIS NAGAR, MOOKKANNOOR**

**ANGAMALY-683577**



‘**FOCUS ON EXCELLENCE’**

**Name : AKASH SANTHOSH**

**Branch : MASTER OF COMPUTER APPLICATIONS**

**Semester : 1 Roll No: 8**

**University Exam.Reg. No:** **FIT21MCA-2008**

**CERTIFICATE**

Certified that this is the Bonafide record of the Practical work donebyMr. **AKASH SANTHOSH** in the **20MCA131- PROGRAMMING** Laboratory of the Federal Institute of Science and Technology during the academic year 2021-2022.

Signature of Staff in Charge Signature of H.O.D

Name: Name:

Date:

**Date of University practical examination ………………………**

Signature of Signature of

Internal Examiner External Examiner

**CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SI No:** | **Date :** | **Name of Experiment:** | **Page No:** | **Signature of**  **Staff –In –Charge:** |
| **1** | **10/11/2021** | Display future leap years from current year to a final year entered by user. |  |  |
| **2** | **10/11/2021** | List comprehensions: (a) Generate positive list of numbers from a given list of integers (b) Square of N numbers (c) Form a list of vowels selected from a given word (d) List ordinal value of each element of a word (Hint: use ord() to get ordinal values) |  |  |
| **3** | **11/11/2021** | Count the occurrences of each word in a line of text. |  |  |
| **4** | **11/11/2021** | Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead |  |  |
| **5** | **11/11/2021** | Store a list of first names. Count the occurrences of ‘a’ within the list |  |  |
| **6** | **17/11/2021** | Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both. |  |  |
| **7** | **25/11/2021** | Get a string from an input string where all occurrences of first character replaced with ‘$’, except first character |  |  |
| **8** | **25/11/2021** | Create a string from given string where first and last characters exchanged. [eg: python - > nythop] |  |  |
| **9** | **28/10/2021** | Accept the radius from user and find area of circle. |  |  |
| **10** | **28/10/2021** | Find biggest of 3 numbers entered. |  |  |
| **11** | **25/11/2021** | Accept a file name from user and print extension of that. |  |  |
| **12** | **17/11/2021** | Create a list of colors from comma-separated color names entered by user. Display first and last colors. |  |  |
| **13** | **17/11/2021** | Accept an integer n and compute n+nn+nnn. |  |  |
| **14** | **17/11/2021** | Print out all colors from color-list1 not contained in color-list2. |  |  |
| **15** | **25/11/2021** | Create a single string separated with space from two strings by swapping the character at position 1. |  |  |
| **16** | **2/12/21** | Merge two dictionaries. |  |  |
| **17** | **25/11/21** | Find gcd of 2 numbers. |  |  |
| **18** | **17/11/2021** | From a list of integers, create a list removing even numbers. |  |  |
| **19** | **2/12/2021** | Program to find the factorial of a number |  |  |
| **20** |  | Generate Fibonacci series of N terms |  |  |
| **21** |  | Find the sum of all items in a list |  |  |
| **22** |  | Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square. |  |  |
| **23** |  | Display the given pyramid with step number accepted from user. |  |  |
| **24** |  | Count the number of characters (character frequency) in a string. |  |  |
| **25** |  | Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’. |  |  |
| **26** |  | Accept a list of words and return length of longest word. |  |  |
| **27** |  | Construct following pattern using nested loop.  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \* |  |  |
| **28** |  | Generate all factors of a number. |  |  |
| **29** |  | Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import \* statements) |  |  |
| **30** |  | Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area. |  |  |
| **31** |  | Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank. |  |  |
| **32** |  | Create a class Rectangle with private attributes length and width. Overload ‘<’ operator to compare the area of 2 rectangles. |  |  |
| **33** |  | Create a class Time with private attributes hour, minute and second. Overload ‘+’ operator to find sum of 2 time |  |  |
| **34** |  | Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no\_of\_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding. |  |  |
| **35** |  | Write a Python program to read a file line by line and store it into a list. |  |  |
| **36** |  | Write a Python program to read each row from a given csv file and print a list of strings. |  |  |

**COURSE OUTCOME 1**

1. **Display future leap years from current year to a final year entered by**

**User.**

**Source code**

print("Enter leap year between given two years"); startyear=2021

endyear=int(input("Enter end year")) print("list of leap years")

for year in range(startyear,endyear): if(0==year%4):

print(year)

**Output**Text

Description automatically generated

1. **List comprehensions:**
2. **Generate positive list of numbers from a given list of integers.**

**Source code**

list=[-1,1,7,25,-34,38]

print("Elements in the list are:",list) print("Positive numbers in the list")

for num in list:

if num>=0:

print(num)

**Output**

1. **Square of N numbers**

**Source code**

n=int(input('enter range:'))

for num in range(1,n+1):

num=num\*num

print(num)

Chart, waterfall chart

Description automatically generated **Output**

1. **Form a list of vowels selected from a given word.**

**Source code**

s=input("Enter a string: ")

list=[]

for i in s:

if i in "aeiouAEIOU":

list.append(i)

print("vowels in the list are:")

print(list)

Text

Description automatically generated with medium confidence**Output**

1. **List ordinal values of each element of a word.**

**Source code**

print("String: Welcome")

print("Ordinal Values")

for i in 'W','e','l','c','o','m','e':

x=ord(i)

print(x)

Graphical user interface, text, application

Description automatically generated **Output**

1. **Count the occurrences of each word in a line of text.**

**Source code**

list1=[]

list2=[]

x=input("Enter a string:")

for i in x.split(" "):

list1.append(i)

if i not in list2:

list2.append(i)

for i in list2:

print(i,"\t",list1.count(i))

Chart, waterfall chart

Description automatically generated**Output**

1. **Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead.**

**Source code**

list=[]

while True:

n=int(input('Enter an integer: '))

if(n<=100):

list.append(n)

else:

list.append('over')

print(list)

Text

Description automatically generated with medium confidence**Output**

1. **Store a list of first names. Count the occurrences of ‘a’ within the list.**

**Source code**

list=['anil','amal','nill'] print("Elements in the list are:")

print(list)

count=0

for word in list:

for i in word:

if i=='a':

count+=1

print("count of 'a' is:", count)

**Output**

1. **Enter 2 lists of integers.Check**
2. **whether list are of same length**
3. **whether list sums of same value**
4. **whether any value occur in both.**

**Source code**

l1=[1,2,3,4]

l2=[5,8,7]

print("List 1",l1)

print("List 2",l2)

x=len(l1)

y=len(l2)

if x==y:

print("List are of same length")

else:

print("Length of lists are different")

s1=0

s2=0

for i in range(x):

s1=s1+l1[i]

print("Sum of elements of List1:",s1)

for j in range(y):

s2=s2+l2[j]

print("Sum of elememts of List2:",s2)

if s1==s2:

print("Sum of list elements is same")

else:

print("Sum of list elements is not same")

print("Common elements are:")

for i in range(x):

for j in range(y):

if l1[i]==l2[j]:

print(l1[i])

Text

Description automatically generated**Output**

1. **Get a string from an input string where all occurrences of first character replaced with ‘$’,except first character.[eg:onion->oni$n]**

**Source code**

str=input("Enter a string: ")

print("Original string: ",str)

char=str[0]

str=str.replace(char,'$')

str=char+str[1:]

print("String: ",str)

Text

Description automatically generated with low confidence**Output**

1. **Create a string from given string where first and last characters exchanged. [eg:python->nythop]**

**Source code**

s=input("Enter a string: ")

t=s[0]

t1=s[-1]

n=len(s)

ns=t1+s[1:n-1]+t

print(ns)

Text

Description automatically generated**Output**

1. **Accept the radius from the user and find the area of the circle.**

**Source code**

r=int(input('Enter the radius: '))

A=3.14\*r\*r

print(A)

Graphical user interface, text, application

Description automatically generated**Output**

1. **Find the biggest of 3 numbers**

**Source code**

a=int(input('Enter the value of a:'))

b=int(input('Enter the value of b:'))

c=int(input('Enter the value c:'))

if a>b and a>c:

print(a)

if b>a and b>c:

print(b)

if c>a and c>b:

print(c)

Graphical user interface, application

Description automatically generated**Output**

1. **Accept a file name from user and print extension of that.**

**Source code**

import os

a=input("Enter filename:")

print("The extension of file",a,"is",os.path.splitext(a))

Text

Description automatically generated with medium confidence**Output**

1. **Create a list of colors from comma-separated color names entered by user. Display first and last colors.**

**Source code**

colors=[]

str=(input("Enter color names:"))

for i in str.split(','):

colors.append(i)

print(colors)

print("first color:",colors[0],"Last color:",colors[-1])

Chart

Description automatically generated**Output**

1. **Accept an integer n and compute n+nn+nnn.**

**Source code**

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

a=n\*1

b=n\*11

c=n\*111

s=a+b+c

print(n,"+",n,"\*",n,"+",n,"\*",n,"\*",n,"=",s)

**Output**

1. **Print out all color from color-list1 not contained in color-list2**

**Source code**

l1=['red','green','blue','yellow','black']

l2=['red','green','yellow']

print(l1)

print(l2)

print("Colors that are not in l1:

")

for i in l1:

if i not in l2:

print(i)

Text

Description automatically generated with medium confidence**Output**

1. **Create a single string separated with space from two strings by swapping the character at position 1.**

**Source code**

str1=input("Enter first string:")

str2=input("Enter second string:")

str3=str2[0]+str1[1:]+" "+str1[0]+str2[1:]

print(str3)

Text

Description automatically generated with medium confidence**Output**

1. **Merge two dictionaries.**

**Source code**

D1={"Name":"Anil","Age":"21"}

print("Directory 1",D1)

D2={"Gender":"male","Qualification":"BCA"}

print("Directory 2",D2)

D1.update(D2)

print("After merging...")

print(D1)

Text

Description automatically generated**Output**

1. **Find gcd of 2 numbers**

**Source code**

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

b=int(input("Enter first number: "))

x=min(a,b)

gcd=0

for i in range (1,x+1):

if((a%x==0) and (b%x==0)):

gcd=i

print("GCD is",i)

Text

Description automatically generated**Output**

1. **From a list of integers, create a list removing even numbers.**

**Source code**

l1=[1,2,3,4,5,6,7,8,9,10]

print(l1)

l2=[]

for i in range(len(l1)):

if l1[i]%2!=0:

l2.append(l1[i])

print("List after removing even elements")

print(l2)

**Output**

**COURSE OUTCOME 2**

1. **Program to find the factorial of a number.**

**Source code**

n=int(input('enter the value:'))

fact=1

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

fact=fact\*i

print(fact)

Graphical user interface

Description automatically generated with low confidence**Output**

1. **Generate fibonacci series of N terms.**

**Source code**

n=int(input('enter the value:'))

a=0

b=1

print(a)

print(b)

for i in range (2,n):

c=a+b

print(c)

a=b

b=c

Graphical user interface, text, application

Description automatically generated**Output**

1. **Find the sum of all items in a list.**

**Source code**

list=[1,6,3,4,1]

print("List elements are:",list)

sum=0

for i in list:

sum=sum+i

print("The sum of list elements is:",sum)

Text

Description automatically generated with medium confidence**Output**

1. **Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.**

**Source code**

limit1=1000

limit2=9999

list1=[]

for i in range(limit1,limit2):

j=i

digit=[]

while(i!=0):

digit.append(i%10)

i=int(i/10)

count=0

for n in digit:

if n%2==0:

count=count+1

if count==4:

for k in range(31,100):

if((k\*\*2)==j):

list1.append(j)

print(k)

print(list1)

A picture containing chart

Description automatically generated**Output**

1. **Display the given pyramid with step number accepted from user.**

**Source code**

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

for j in range(0,n+1):

for i in range(1,j+1):

i=j\*i

print(i,end=" ")

print("\n")

A picture containing chart

Description automatically generated**Output**

1. **Count the number of characters (character frequency) in a string.**

**Source code**

string=input("Enter a string:")

list1=[]

for i in string:

if i not in list1:

list1.append(i)

for i in list1:

count=0

for j in string:

if(i==j):

count=count+1

print(i,"\t:",count)

A picture containing table

Description automatically generated**Output**

1. **Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’.**

**Source code**

string=input("Enter a string:")

if(string[-3:]=="ing"):

string+="ly"

else:

string+="ing"

print(string)

Text

Description automatically generated with medium confidence**Output**

1. **Accept a list of words and return length of longest word.**

**Source code**

lis=[]

n=int(input("Enter the range:"))

print("Enter the words:")

for i in range(0,n):

lis.append(input(""))

longest=lis[0]

for i in range(1,n):

if(len(lis[i])>len(longest)):

longest=lis[i]

print("Length of longest word is",len(longest))

A picture containing text

Description automatically generated**Output**

1. **Construct following pattern using nested loop.**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**Source code**

for i in range(1,6):

for j in range(1,i+1):

print("\*",end=" ")

print("\n")

for i in range(4,0,-1):

for j in range(1,i+1):

print("\*",end=" ")

print("\n")

A picture containing table

Description automatically generated**Output**

1. **Generate all factors of a number.**

**Source code**

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

print("Factors are")

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

if(n%i==0):

print(i)

Text

Description automatically generated with low confidence**Output**

**COURSE OUTCOME 3**

1. **Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import \* statements)**

**Source code**

**Graphice\circle.py**

from math import pi

def area\_circle(radius):

return pi\*radius\*radius

def perimeter\_circle(radius):

return 2\*pi\*radius

**Graphics\rectangle.py**

def area\_rec(length,width):

return length\*width

def perimeter\_rec(length,width):

return 2\*(length+width)

**Graphics\tdgraphics\cuboid.py**

def area\_cuboid(l,b,h):

return 2\*(l\*h + b\*h + l\*b)

def volume\_cuboid(l,b,h):

return l\*b\*h

**Graphics\tdgraphics\sphere.py**

from math import pi

def area\_sphere(radius):

return 4\*(pi\*radius\*radius)

def perimeter\_sphere(radius):

return 2\*pi\*radius

**graphics.py (driver code)**

import Graphics

from Graphics import circle,rectangle

from Graphics.tdgraphics import cuboid,sphere

from Graphics.circle import \*

print("Area of a circle with radius 10 is : ",circle.area\_circle(10))

print("Permeter of a circle with radius 10 is ",circle.perimeter\_circle(10))

print("\n")

print("Area of a Rectangle with length and width 10 is : ",rectangle.area\_rec(10,10))

print("Permeter of a Rectangle with length and width 10 is : ",rectangle.perimeter\_rec(10,10))

print("\n")

print("Area of a cuboid with length,width,height 10 is : ",cuboid.area\_cuboid(10,10,10))

print("Volume of a cuboid with length,width,height 10 is : ",cuboid.volume\_cuboid(10,10,10))

print("\n")

print("Area of a spere with radius 10 is : ",sphere.area\_sphere(10))

print("Permeter of a spere with radius 10 is ",sphere.perimeter\_sphere(10))

Text

Description automatically generated**Output**

**COURSE OUTCOME 4**

1. **Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.**

**Source code**

class Rectangle:

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

self.length = length

self.breadth = breadth

def area(self):

return self.length \* self.breadth

def perimeter(self):

return 2\*(self.length + self.breadth)

l=int(input("Enter length of rectangle1: "))

b=int(input("Enter breadth of rectangle1: "))

rect1 = Rectangle(l,b)

a1=rect1.area()

p1=rect1.perimeter()

print("Area:",a1)

print("Perimeter:",p1)

l=int(input("Enter length of rectangle2: "))

b=int(input("Enter breadth of rectangle2: "))

rect2 = Rectangle(l,b)

a2=rect2.area()

p2=rect2.perimeter()

print("Area:",a2)

print("Perimeter:",p2)

if (a1>a2):

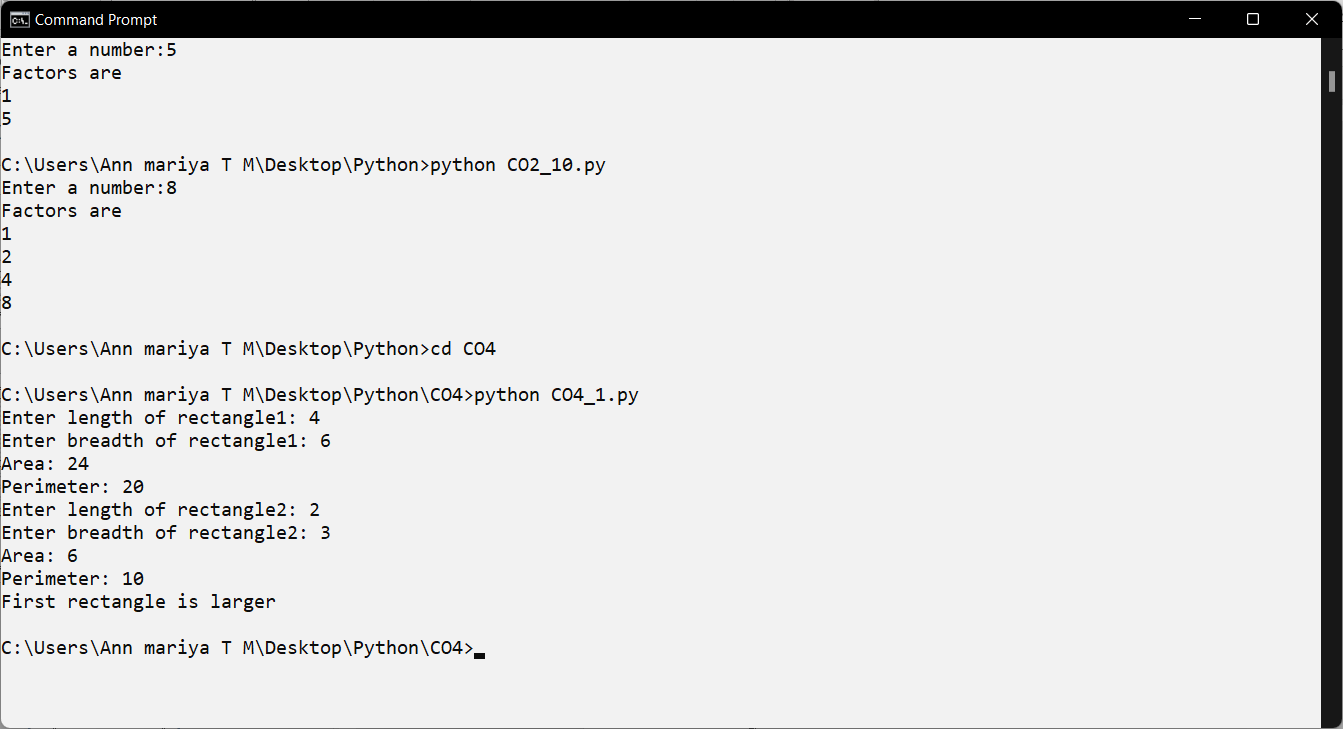
print("First rectangle is larger")

elif a1==a2:

print("Rectangles are of same area")

else:

print("Second rectangle is larger")

**Output**

1. **Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank.**

**Source code**

class bank:

def \_\_init\_\_(self,acc\_no,name,acc\_type,bal):

self.acc\_no=acc\_no

self.name=name

self.acc\_type=acc\_type

self.bal=bal

def deposit(self):

self.bal=self.bal+y

return self.bal

def withdraw(self):

return self.bal-y

def display\_balance(self):

return self.bal

acc1=bank("b11","Ann","Savings",50000)

while(1):

print("1.Deposit\n2.Withdraw\n3.Display balance\n4.Exit\n")

ch=int(input("Enter your choice:"))

if ch==1:

amt=int(input("Enter the amount:"))

b=acc1.deposit(amt)

print("Current balance:",b)

elif ch==2:

amt=int(input("Enter the amount:"))

b=acc1.withdraw(amt)

print("Current balance:",b)

elif ch==3:

cb=acc1.display\_balance()

print("Current balance:",cb)

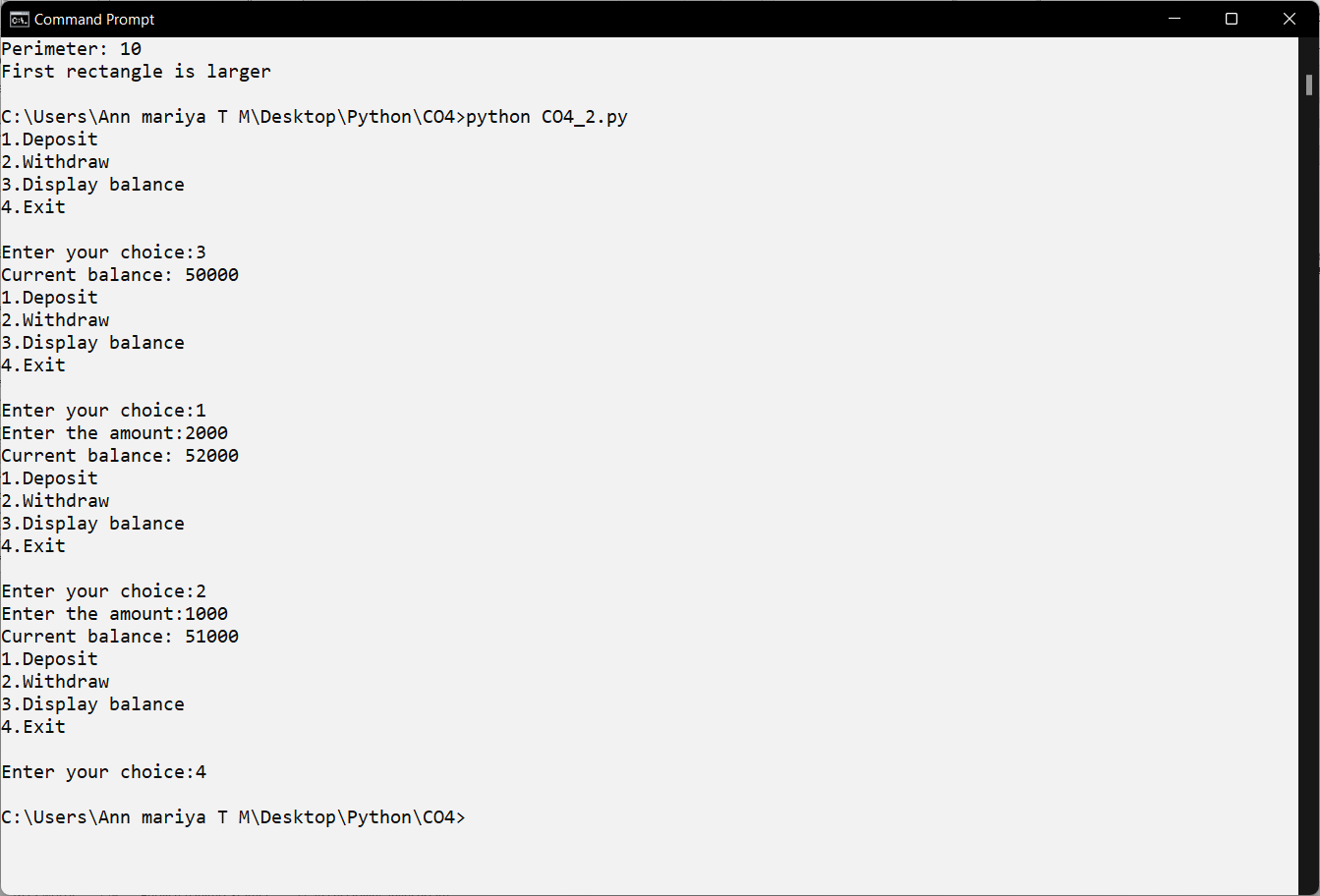
elif ch==4:

exit(1)

else:

print("Invalid choice")

**Output**



1. **Create a class Rectangle with private attributes length and width. Overload ‘<’ operator to compare the area of 2 rectangles.**

**Source code**

class Rectangle:

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

self.\_\_length = length

self.\_\_breadth = breadth

def \_\_lt\_\_ (self,rect2):

if self.\_\_length\*self.\_\_breadth < rect2.\_\_length\*rect2.\_\_breadth:

return True

else:

return False

l=int(input("Enter length of rectangle1: "))

b=int(input("Enter breadth of rectangle1: "))

rect1 = Rectangle(l,b)

l=int(input("Enter length of rectangle2: "))

b=int(input("Enter breadth of rectangle2: "))

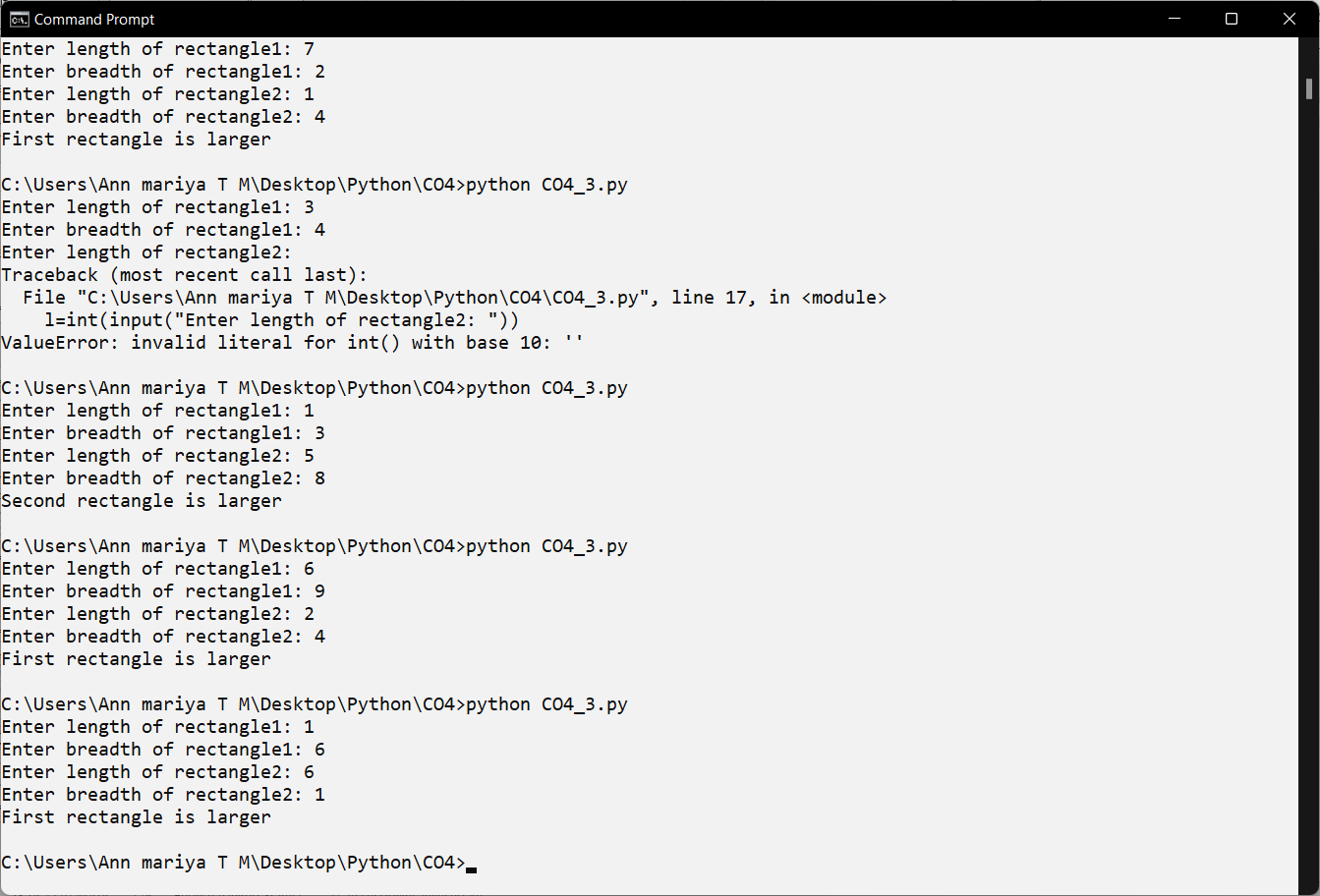
rect2 = Rectangle(l,b)

if rect1 < rect2:

print("Second rectangle is larger")

else:

print("First rectangle is larger")

**output**

1. **Create a class Time with private attributes hour, minute and second. Overload ‘+’ operator to find sum of 2 time.**

**Source code**

class Time:

def \_\_init\_\_(self,hr,min,sec):

self.\_\_hr=hr

self.\_\_min=min

self.\_\_sec=sec

def \_\_add\_\_(t1,t2):

hr=t1.\_\_hr+t2.\_\_hr

min=t1.\_\_min+t2.\_\_min

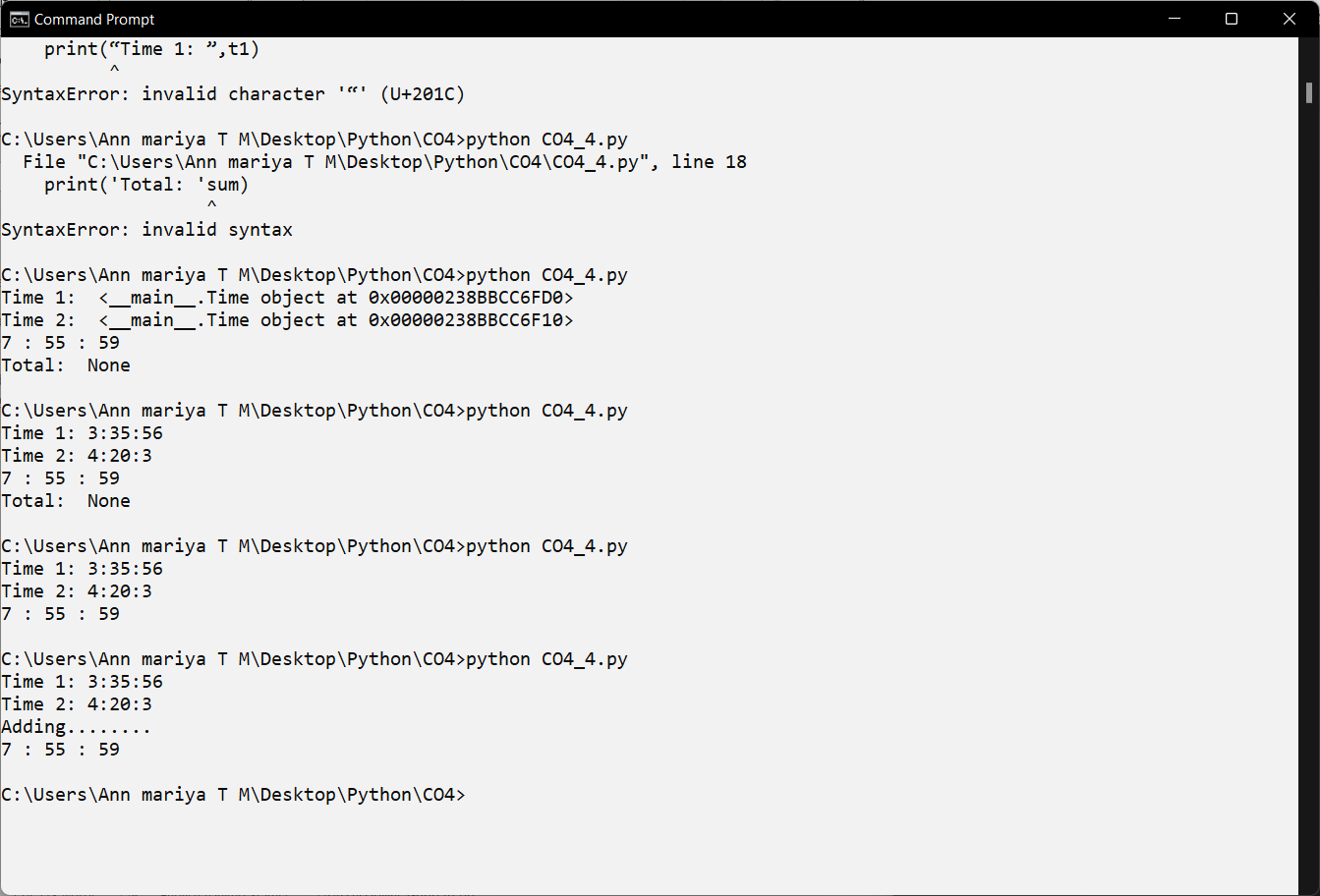
sec=t1.\_\_sec+t2.\_\_sec

print(hr,":",min,":",sec)

t1=Time(3,45,56)

t2=Time(4,20,3)

t1+t2

**Output**

1. **Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no\_of\_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.**

**Source code**

class Publisher(object):

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

self.name=name

def display1(self):

print(self.title)

print(self.author)

class Book(Publisher):

def \_\_init\_\_(self,name,title,author):

super().\_\_init\_\_(name)

self.title=title

self.author=author

def display2(self):

#super().display1()

print(self.title)

print(self.author)

class Python(Book):

def \_\_init\_\_(self,name,title,author,price,no\_of\_pages):

super().\_\_init\_\_(name,title,author)

self.price=price

self.no\_of\_pages=no\_of\_pages

def display3(self):

super().display2()

print(self.price)

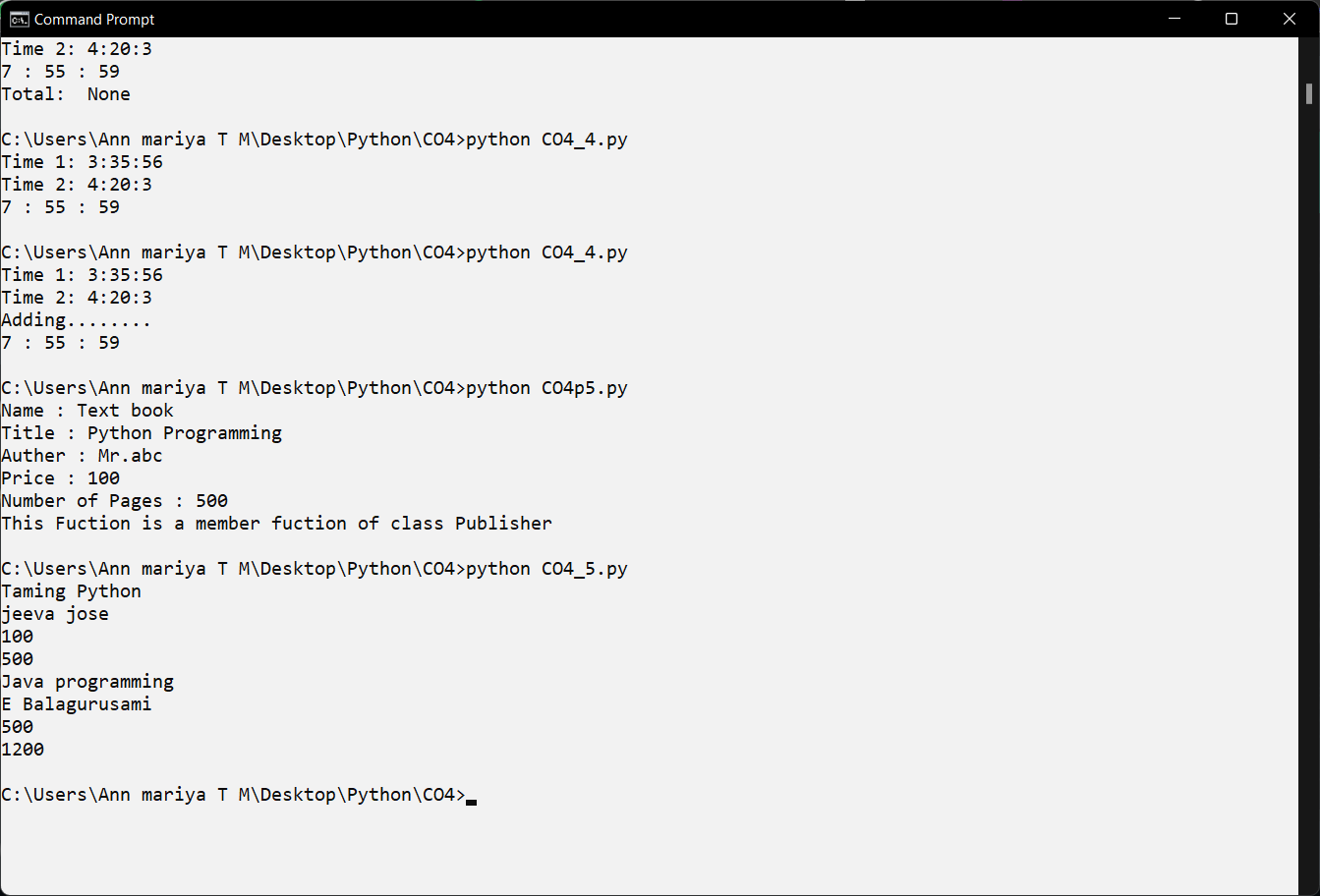
print(self.no\_of\_pages)

p=Python("ABC Publications","Taming Python","jeeva jose",100,500)

p.display3()

q=Python("XYZ Publications","Java programming","E Balagurusami",500,1200)

q.display3()

**Output**

**COURSE OUTCOME 5**

1. Write a Python program to read a file line by line and store it into a list.

**Source code**

fp=open("text\_file.txt",'r')

lines=[]

for line in fp:

lines.append(line.strip())

print(lines)

**Output**

1. **Write a Python program to read each row from a given csv file and print a list of strings.**

**Source code**

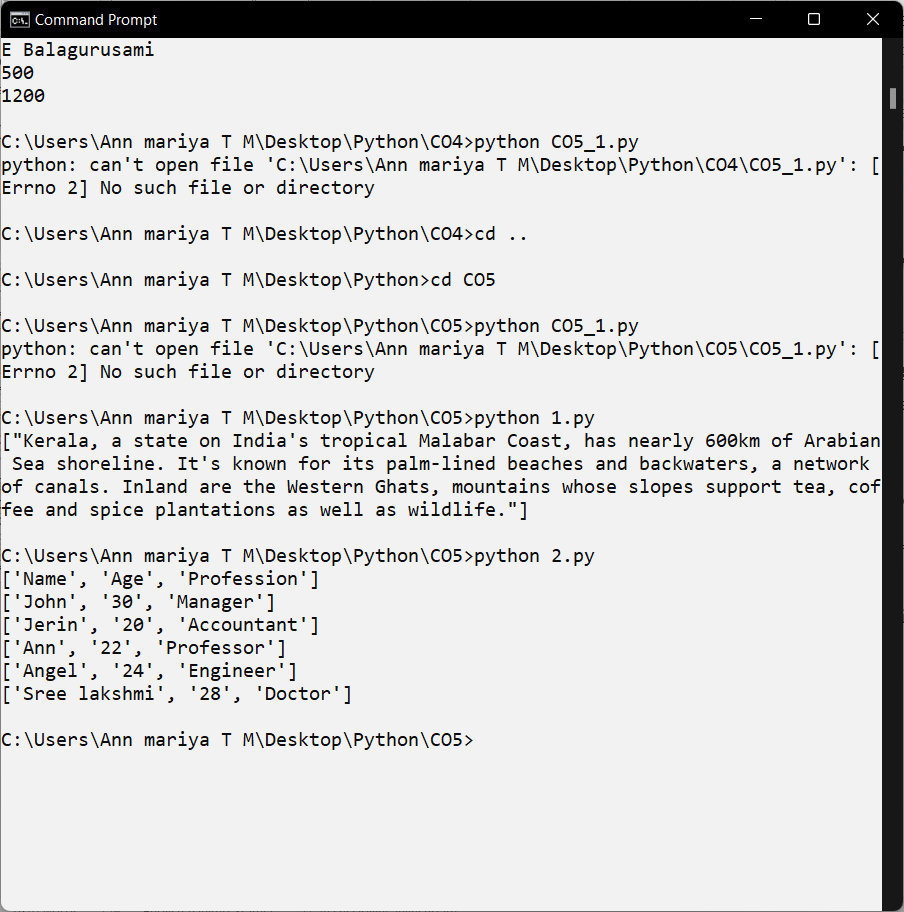
import csv

with open('people.csv', 'r') as file:

reader = csv.reader(file)

for row in reader:

print(row)

**Output**