CO1

2. Display future leap years from current year to a final year entered by user.

```
x=int(input("enter a year to check leap year or not"))

if (x%400==0 or (x%4 and x%100!=0)):
    print(x," is a leap year")
else:
    print(x," not is leap year")
```

OUTPUT

```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe" "E:/MCA/
enter a year to check leap year or not2021
2021  is a leap year

Process finished with exit code 0
```

3.(a) Generate positive list of numbers from a given list of integers

```
list1 = [11, -21, 0, 45, 66, -93]
for num in list1:
    if num >= 0:
        print(num, end=" ")
```

OUTPUT

```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe" "E:/MCA/S1/prg lab/prg/myclass/co1-3a.py"
11 0 45 66
Process finished with exit code 0
```

11. Find biggest of 3 numbers entered.

```
print("Enter any three numbers :")
a=int(input())
b=int(input())
c=int(input())
if(a>b and a>c):
    print(a," is the biggest")
elif(b>c):
    print(b," is the biggest")
else:
    print(c," is the biggest")
```

OUTPUT

```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe"
Enter any three numbers :
22
33
44
44  is the biggest
```

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

```
count = input("Enter a list of colors from comma-separated color names \n")
colors = [word for word in count.split(",")]
print(colors[0],"  ",colors[-1])
```

OUTPUT

```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe" "E:/MCA/S1/pr
Enter a list of colors from comma-separated color names
red,pink,orange,blue,green
red    green
```

15. Print out all colors from color-list1 not contained in color-list2.

```
colorslist1=set(["Red","Green","Blue","pink"])
colorslist2=set(["Green","Orange","Blue"])
print("colorslist 1 : \n",colorslist1)
print("colorslist 2 : \n",colorslist2)
print("\nPrint out all colors from color-list1 not contained in color-
list2")
print(colorslist1.difference(colorslist2))
```

OUTPUT

```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe" "E:/MCA/S1/prg lab/prg/myclass/co1-15.py"
colorslist 1 :
 {'pink', 'Blue', 'Red', 'Green'}
colorslist 2 :
 {'Blue', 'Orange', 'Green'}

Print out all colors from color-list1 not contained in color-list2
{'Red', 'pink'}
```

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

```
string = input("Enter 2 string seperated by comma : ")
string = string.split(',')
print(string[1][0] + string[0][1:] +" "+ string[0][0] + string[1][1:])
```

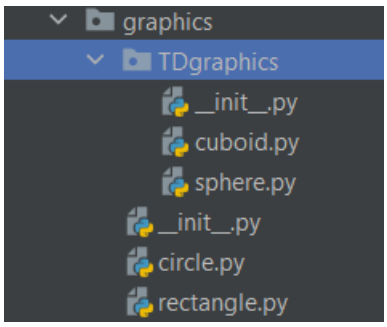
OUTPUT

```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe" "E:/MCA/S1/prg lab/prg/myclass/co1-16.py"
Enter 2 string seperated by comma : hi , hello
 i  hhello

Process finished with exit code 0
```

CO3

2. 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)

Packages:



Graphics package

circle

```
from math import pi

def area_circle(radius):
    return pi*radius*radius

def perimeter_circle(radius):
    return 2*pi*radius
```

Rectangle

```
def area_rec(length,width):
    return length*width

def perimeter_rec(length,width):
    return 2*(length+width)
```

3-D graphics package

Cuboid

```
def area_cuboid(l,b,h):
    return 2*(l*h + b*h + l*b)

def perimeter_cuboid(l,b,h):
    return 4*(l+b+h)
```

Sphere

```
from math import pi

def area_sphere(radius):
    return 4*(pi*radius*radius)

def perimeter_sphere(radius):
    return 2*pi*radius
```

Programs that finds area and perimeter of figures by different importing statements.

```
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("Area of a circle with radius 10 is : ",area_circle(10))

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("Area of a  cuboid with length,width,height 10 is :
",cuboid.area_cuboid(10,10,10))
print("Permeter of a cuboid with length,width,height 10 is :
",cuboid.perimeter_cuboid(10,10,10))

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))
```

OUTPUT

```
Permeter of a circle with radius 10 is  62.83185307179586
Area of a circle with radius 10 is :  314.1592653589793
Area of a Rectangle with length and width 10 is :  100
Permeter of a Rectangle with length and width 10 is :  40
Area of a  cuboid with length,width,height 10 is :  600
Permeter of a cuboid with length,width,height 10 is :  120
Area of a spere with radius 10 is :  1256.6370614359173
Permeter of a spere with radius 10 is  62.83185307179586
```

CO4

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

```
class rectangle():
    def __init__(self,length,breadth):
        self.length=length
        self.breadth=breadth
    def area(self):
        return self.breadth*self.length
    def peri(self):
        return 2*(self.length+self.breadth)
obj1=rectangle(6,2)
obj2=rectangle(4,3)
print("Area of a rectangle 1: ",obj1.area())
print("Perimeter of a rectangle 1: ",obj1.peri())
print("Area of a rectangle 2: ",obj2.area())
print("Perimeter of a rectangle 2: ",obj2.peri())
if(obj1.area()==obj2.area()):
    print(obj1.area(),"=",obj2.area())
else:
    print(obj1.area(),"not equal to ",obj2.area())
```

OUTPUT

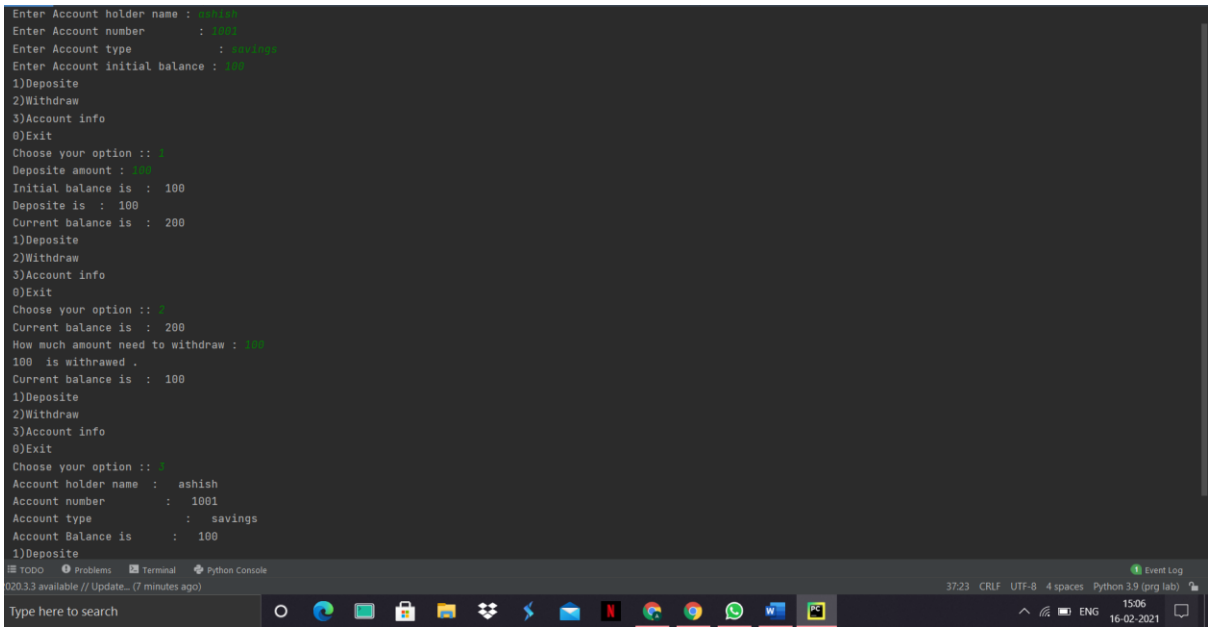
```
"E:\MCA\S1\prg lab\venv\Scripts\python.exe" "E:/MCA/S1/prg lab/prg/myclass/co4 1.py"
Area of a rectangle 1:  12
Perimeter of a rectangle 1:  16
Area of a rectangle 2:  12
Perimeter of a rectangle 2:  14
12 = 12
```

2. 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.

```
class bank:
    __acc_name = ""
    __acc_no = ""
    __acc_type = ""
    __acc_balance = 0

    def __init__(self, a_name, a_no, a_type, a_balance):
        self.__acc_name = a_name
        self.__acc_no = a_no
        self.__acc_type = a_type
        self.__acc_balance = a_balance

    def deposite(self, a_deposit):
        print("Initial balance is  : ", self.__acc_balance)
        print("Deposite is  : ", a_deposit)
        self.__acc_balance += a_deposit
        print("Current balance is  : ", self.__acc_balance)

    def withdraw(self):
        print("Current balance is  : ", self.__acc_balance)
        self.amount = int(input("How much amount need to withdraw : "))
        if self.amount > self.__acc_balance:
```

```
            print("You don't have enough balance to withdraw !!")
            print("Current balance is  : ", self.__acc_balance)
        else:
            print(self.amount, " is withrawed .")
            self.__acc_balance -= self.amount
            print("Current balance is  : ", self.__acc_balance)

    def acc_info(self):
        print("Account holder name    :  ", self.__acc_name)
        print("Account number         :  ", self.__acc_no)
        print("Account type           :  ", self.__acc_type)
        print("Account Balance is     :  ", self.__acc_balance)

def main():
    name = input("Enter Account holder name : ")
    no = input("Enter Account number        : ")
    atype = input("Enter Account type             : ")
    bal = int(input("Enter Account initial balance : "))
    holder = bank(name, no, atype, bal)

    while (True):

        opt = int(input("1)Deposite \n2)Withdraw \n3)Account info
\n0)Exit\nChoose your option :: "))

        if opt == 1:
            amount = int(input("Deposite amount : "))
            holder.deposite(amount)
        elif opt == 2:
            holder.withdraw()
        elif opt == 3:
            holder.acc_info()
        elif opt == 0:
            break
        else:
            print("Invalid Option !")


if __name__ == "__main__":
    while (True):
        main()
```

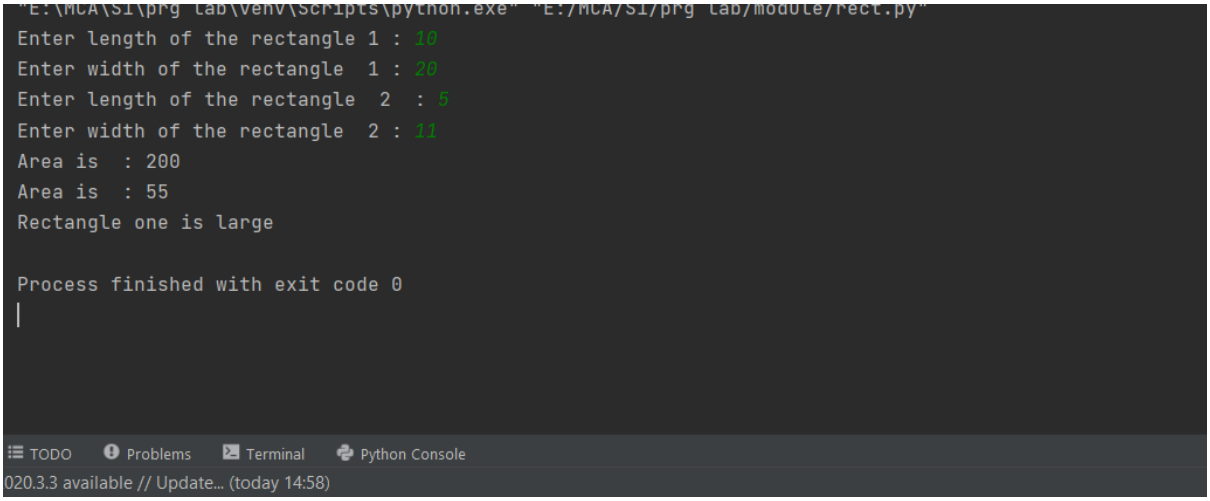
OUTPUT



3. Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.

```
class rectangle:
    __area = 0
    __perimeter = 0

    def __init__(self, length, width):
        self.__length = length
        self.__width = width

    def calc_area(self):
        self.__area = self.__length * self.__width
        print("Area is  :", self.__area)

    def __lt__(self, second):
        if self.__area < second.__area:
            return True
        else:
            return False


length1 = int(input("Enter length of the rectangle 1 : "))
width1 = int(input("Enter width of the rectangle  1 : "))
length2 = int(input("Enter length of the rectangle  2  : "))
width2 = int(input("Enter width of the rectangle  2 : "))

obj1 = rectangle(length1, width1)
obj2 = rectangle(length2, width2)
obj1.calc_area()
deobj2.calc_area()

if obj1 < obj2:
    print("Rectangle two is large")
else:
    print("Rectangle one is large")
```

OUTPUT



4.Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.

```
class Time:
    def __init__(self,hh=0,mm=0,ss=0):
        self.__hour=hh
        self.__minute=mm
        self.__second=ss
    def __add__(self,other):
        second=int((self.__second + other.__second)%60)
        minute=int((self.__minute + other.__minute)%60 + ((self.__second +
other.__second)/60))
        hour=int((self.__hour + other.__hour)%24 + (self.__minute +
other.__minute)/60)
        print('Time[hh:mm:ss] ',hour,':',minute,':',second)
T1=Time(11,35,15)
T2=Time(10,15,46)
T1 + T2
```

OUTPUT

5. 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.

```
class Publisher:
    def __init__(self,name1):
        self.name=name1
    def show(self):
        pass
class Book(Publisher):
    def __init__(self,title1,author1,name1):
        self.title=title1
        self.author=author1
        Publisher.__init__(self,name1)
    def show(self):
        pass
class Python(Book):
    def __init__(self,p,no,title1,author1,name1):
        self.price=p
        self.no_of_pages=no
        Book.__init__(self,title1,author1,name1)
    def show(self):
        print('Book title:',self.title)
        print('Author:',self.author)
        print('Publisher:',self.name)
        print('Price: Rs.',self.price)
        print('No of pages:',self.no_of_pages)
P1=Python(144.67,300,'Robinson Crusoe','Daniel Crusoe','Starbooks')
P1.show()
```

OUTPUT