**Program :1**

def fact\_rec(n):  
 if n==0 or n==1:  
 return 1  
 else:  
 return n\* fact\_rec(n-1)  
number=int(input("enter a value"))   
res=fact\_rec(number)  
print("the factroial of {}is{}.".format(number,res))

**program:2**

def isLeapYear(year):  
 if (year % 4 == 0 and year % 100 != 0)or year % 400 == 0:  
 return True  
 else:  
 return False  
year = int(input("enter a year:"))   
if isLeapYear(year):  
 print('{}is a leap year.'.format(year))  
else:  
 print('{} is not a leap year.'. format(year))

**program:3**

# Python program to create Bankaccount class

# with both a deposit() and a withdraw() function

class Bank\_Account:

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

self.balance=0

print("Hello!!! Welcome to the Deposit & Withdrawal Machine")

def deposit(self):

amount=float(input("Enter amount to be Deposited: "))

self.balance += amount

print("\n Amount Deposited:",amount)

def withdraw(self):

amount = float(input("Enter amount to be Withdrawn: "))

if self.balance>=amount:

self.balance-=amount

print("\n You Withdrew:", amount)

else:

print("\n Insufficient balance ")

def display(self):

print("\n Net Available Balance=",self.balance)

# Driver code

# creating an object of class

s = Bank\_Account()

# Calling functions with that class object

s.deposit()

s.withdraw()

s.display()

**program 4:**

#Define the base class playerclass player:

class player:

def player(self):

print("The player is playing cricket.")

#Define the derived class Batsman

class Batsman(player):

def play(self):

print("The Batsman is batting.")

#Define the derived class Bowler

class Bowler(player):

def play(self):

print("The bowler is bowling.")

#create objects of Batsman and Bowler classes

batsman = Batsman()

bowler = Bowler()

#call the play() method for each object

batsman.play()

bowler.play()

PROGRAM:5

"""

Write a function called linear\_search\_product that takes the list of products and a target product

name as input. The function should perform a linear search to find the target product in the list and

return a list of indices of all occurrences of the product if found, or an empty list if the product is not

found.

"""

def linearSearchProduct(productList, targetProduct):

indices = []

for index, product in enumerate(productList):

if product == targetProduct:

indices.append(index)

return indices

# Example usage:

products = ["shoes", "boot", "loafer", "shoes", "sandal", "shoes"]

target = "shoes"

target2 = 'apple'

result = linearSearchProduct(products, target)

print(result)

PROGRAM 6

'''

Implement a function called sort\_students that takes a list of student objects as input and sorts the

list based on their CGPA (Cumulative Grade Point Average) in descending order. Each student object

has the following attributes: name (string), roll\_number (string), and cgpa (float). Test the function

with different input lists of students.

'''

class Student:

def \_\_init\_\_(self, name, roll\_number, cgpa):

self.name = name

self.roll\_number = roll\_number

self.cgpa = cgpa

def sort\_students(student\_list):

# Sort the list of students in descending order of CGPA

sorted\_students = sorted(student\_list,

key=lambda student: student.cgpa,

reverse=True)

# Syntax - lambda arg:exp

return sorted\_students

# Example usage:

students = [

Student("Hari", "A123", 7.8),

Student("Srikanth", "A124", 8.9),

Student("Saumya", "A125", 9.1),

Student("Mahidhar", "A126", 9.9),

]

sorted\_students = sort\_students(students)

# Print the sorted list of students

for student in sorted\_students:

print("Name: {}, Roll Number: {}, CGPA: {}".format(student.name,

student.roll\_number,

student.cgpa))