#task-01(a)

def factorial(n):

if n == 1:

return 1

else:

return n\*factorial(n-1)

print(factorial(3))

#task-01(b)

def fibonacci(n):

if n <= 2 and n > 0:

return 1

elif n == 0:

return 0

else:

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

print(fibonacci(5))

#task-01(c)

def print\_array(ary,start):

if start == len(ary)-1:

return ary[len(ary)-1]

else:

print(ary[start],end=',')

return print\_array(ary,start+1)

print(print\_array([2,3,4,5,6,7,8],0))

#task-01(d)

def power(base,n):

if n == 0:

return 1

else:

return base \* power(base,n-1)

print(power(3,3))

#task-02(b)

class Node:

def \_\_init\_\_(self,elm,next):

self.element = elm

self.next = next

class node\_create:

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

self.head = Node(ary[0],None)

current\_head = self.head

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

n = Node(ary[i],None)

current\_head.next = n

current\_head = current\_head.next

def return\_head(self):

return self.head

#task - 2(a)

def sum\_linkedlist(node):

if node == None:

return 0

else:

return node.element + sum\_linkedlist(node.next)

#task - 2(b)

def print\_reverse(node):

if node == None:

return

else:

print\_reverse(node.next)

print(node.element,end=',')

obj1 = node\_create([1,2,3,4,5,6,7,8])

head = obj1.return\_head()

print(sum\_linkedlist(head))

print\_reverse(head)

#task-03

def hocBuilder(height):

if height == 1:

return 8

elif height == 0:

return 0

else:

return 5 + hocBuilder(height-1)

hocBuilder(5)

#Task\_4(a)

def pattern(i,j,n):

if i <= n:

if j<=i:

print(j, end="")

pattern(i,j+1,n)

return

print()

pattern(i+1,1,n)

return

n=5

pattern(1,1,n)

#Task\_4(b)

def pattern(i,j,n, flag = False):

if i <= n:

if flag is False:

print(" "\*(n-i),end="")

if j<=i:

print(j,end="")

return pattern(i,j+1,n,True)

print()

return pattern(i+1,1,n, False)

n=5

pattern(1,1,n)

#task-05

class FinalQ:

def print(self,array,idx):

if(idx<len(array)):

profit = self.calcProfit(array[idx])

print(str(idx+1) + ". Investment: " + str(array[idx]) + "; Profit: " + str(profit))

self.print(array, idx+1)

else:

return

def calcProfit(self,investment):

if investment <= 25000:

return 0.0

else:

net = investment-100000

net = net/100

i = 1

def profit\_count(net,i):

if i > 8:

return 0

else:

return net + profit\_count(net,i+1)

return 3375 + profit\_count(net,i)

array=[25000,100000,250000,350000]

f = FinalQ()

f.print(array,0)