#1(a)

def fact(a):

if a == 0:

return 1

elif a == 1:

return 1

else:

return a\*fact(a-1)

#test

print("Factorial:",fact(3))

#1(b)

def fibo(a):

if a == 0:

return 0

elif a == 1:

return 1

else:

return fibo(a-1)+fibo(a-2)

#test

print("n-th Fibonacci number:",fibo(3))

#1(c)

def recursive\_array(list, a, b): #b = size, a = index

if a >= b:

return

else:

print(list[a])

recursive\_array(list, a+1, b)

#test

a = [1, 2, 3, 4, 5]

printArray(a, 0, 5)

#1(d)

def power(num, a):

if a==0:

return 1

else:

m = power(num, a-1)

return num\*m

#test

print("Power Value: ",power(5, 2))

#2(a)

def deciTobinary(n):

if n > 1:

deciTobinary(int(n/2))

print(n%2, end="")

deciTobinary(25)

print()

#2(b)

class Node:

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

self.value = value

self.next = next

class new\_Node:

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

self.head = None

tail = None

for i in a:

if a == None:

raise Exception("Invalid index")

else:

n = Node(i, None)

if self.head is None:

self.head = n

tail = n

else:

tail.next = n

tail = n

def sum(self,node):

if node == None:

return 0

return node.value +self.sum(node.next)

#test

b = new\_Node(a)

print(b.sum(b.head))

#2(c)

def reverseList(node):

if node is not None:

reverseList(node.next)

print(node.value)

#Test

a = [2, 3, 4, 5]

reverse\_list(b.head)

#3

def hocBuilder(height, a=8):

if height<=0:

return "No need to build a house at all"

elif height==1:

return a

else:

h = height-1

a = a + 5\*(h)

hocBuilder(h)

return a

print(hocBuilder(2))

#4(a)

def pattern(a, num=1, i=1):

if a!=0:

pattern(a-1)

if i<=a:

for j in range(a):

print(num,end=" ")

num+=1

i+=1

print()

#test

pattern(5)

#4(b)

def pattern(a, j, i=1):

num = 1

x = j + 1

if i < x:

if a != 0:

for row in range(x-1, 0, -1):

if row > i:

print(" ", end=" ")

else:

print(num, end=" ")

num += 1

print("")

else:

return

else:

return

pattern(a - 1, j, i + 1)

pattern(5, 5)

#5

class FinalQ:

def print(self,array,idx):

if(idx<len(array)):

profit = self.calcProfit(array[idx])

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

self.print(array, idx+1)

def calcProfit(self,investment):

if investment == 25000:

return 0.0

elif investment > 25000 and investment < 26000:

return 4.5 + float(self.calcProfit(investment-100))

elif investment >= 26000 and investment <= 100000:

return 45 + float(self.calcProfit(investment-1000))

elif investment > 100000 and investment < 101000:

return 8 + float(self.calcProfit(investment-100))

elif investment>=101000:

return 80 + float(self.calcProfit(investment-1000))

#Tester

array = [25000,100000,250000,350000]

f = FinalQ()

f.print(array, 0)