Q1. Take a String value in str variable from keyboard.and find out given string is palindrome or not.if given string is not palindrome then try to make given string palindrome with minimum addition.

Test-Cases

"Hello" output answer is "hellolleh"

"abb" output answer is "abba"

"aaabb" output answer is "bbaaabb"

Sol.

def makePalindrome(str):

for i in range(0,len(str)):

if isPalindrome(str+str[0:i][::-1]):

print("Min-New palindrome String:- "+str+str[0:i][::-1])

return

elif isPalindrome(str[::-1][0:i]+str):

print("Min-New palindrome String:- "+str[::-1][0:i]+str)

return

def isPalindrome(str):

return str==str[::-1]

if \_\_name\_\_ == '\_\_main\_\_':

str=input('Enter any string:- ')

if str==str[::-1]:

print("Given String "+str+" is palindrome")

else:

makePalindrome(str)

Q2. Problem-Statement

Virendra is playing a mobile game that starts with consecutively numbered clouds.Some of the clouds are dark and others are white.he can jump on any white cloud having a number that is equal to the number of current colud plus 1 or 2.he must avoid the dark.determine the minimum number of jumps it will take virendra from him starting position to the last cloud.it always posible to win the game.

For each game virendra will get an array of cloud numbered 0 if they are safe or 1 if they must be avoided.for example c=[0,1,0,0,0,1,0] indexed from 0...6.the number on each cloud is its index in the last so he must avoid the clouds at index 1 and 5.he could follow the following two path:0->2->4->6 or 0->2->3->4->6.the first path takes 3 jumps while the second takes 4.

input:-

The first line contain an integer n,the total number of cloud.the second line contains n space-separted binary integers describing cloud c[i] where 0<= i <n

constraints

1) 2<= n <=100

2) c[i]={0,1}

3) c[0]=c[n-1]=0

Example

n=7

array c=0 0 1 0 0 1 0

Result will be 4

Q3. Given an read-only array of size n, find all elements in array that appear more than n/k times. For example, if the input arrays is {3, 1, 2, 2, 1, 2, 3, 3} and k is 4, then the output should be [2, 3]. Note that size of array is 8 (or n = 8), so we need to find all elements that appear more than 2 (or 8/4) times. There are two elements that appear more than two times, 2 and 3

Constraints:-

1) you can not perform sorting of array

2) you can not use nesting of loop

3) value of k must be minimum 2 or max<n

Sol.

def moreThanNdK(arr,size,k):

map={}

count=size//k

for e in arr:

if e in map:

map[e]+=1

else:

map[e]=1

for key in map:

if map[key]>count:

print(key)

if \_\_name\_\_ == '\_\_main\_\_':

print("First Test\n")

arr1= [4, 5, 6, 7, 8, 4, 4]

size = len(arr1)

k = 3;

moreThanNdK(arr1, size, k)

Q4. A String said to be a special string if either of two condition is met

1) all of the character are the same Ex. aaa

2) all characters except the middle one are the same Ex. aaabaaa

A special substring is any substring of a string which meets one of above criteria,find out how many special substrings can be formed form it.don't use collection data-type[list,set,map..etc]

Example

aaaoaaa

str=mnonopoo

Answer=12

Temp-result={m,n,o,n,o,p,o,o,non,ono,opo,oo} [Not Visible on output Screen]

Sol.

def substrCount(s):

ans=len(s)

i=0

while i<len(s):

#logic-1["aaaa"]

rpc=0

while i+1<len(s) and s[i]==s[i+1]:

rpc+=1

i+=1

ans=ans+(rpc\*(rpc+1))//2

#logic-2['abxba']

p=1

while i-p>=0 and i+p<len(s) and s[i+p]==s[i-1] and s[i-p]==s[i-1]:

ans+=1

p+=1

i+=1

return ans

if \_\_name\_\_ == '\_\_main\_\_':

s=input('Enter any string:- ')

print(substrCount(s))

Q5. (may be wrong question)

Problem-Statement:-

You will be given a array name as "arr" of size "n",and a singal integer "k".you must create an array of length k from elements in arr such that its unfairness is minimized.call that array subarr.unfairness of an array is calculated as

max(subarr)-min(subarr)

where

1) max denote the largest integer in subarr

2) max denote the largest integer in subarr

as an Example,consider the array [1,4,7,2] with a k of 2.pick any two element,test subarr=[4,7]

unfairness=max(4,7)-min(4,7)=7-4=3

Testing for all pairs.the solution [1,2] provide the minimum unfairness.

Example

k=3

size of array n=7

arr={10,100,300,200,1000,20,30}

Sol.

def maxMin(k,arr): #arr={6,15,17,23}

n=len(arr)

arr.sort()

answer=8

i=0

while (i+k-1)<n:

answer=min(answer,arr[i+k-1]-arr[i])

i+=1

return answer

if \_\_name\_\_ == '\_\_main\_\_':

k=int(input('Enter size of k:- '))

n=int(input('Enter size of array:- '))

arr=list()

for i in range(0,n):

arr.append(int(input('Enter arr['+str(i)+']')))

print(maxMin(k,arr))

Q6.

Sol.

def max\_equal\_stack(s1,s2,s3):

while True:

if len(s1)==0 or len(s2)==0 or len(s3)==0:

return 0

if sum(s1)==sum(s2) and sum(s2)==sum(s3):

return sum(s1);

if sum(s1)>=sum(s2) and sum(s1)>=sum(s3):

s1.pop();

elif sum(s2)>=sum(s1) and sum(s2)>=sum(s3):

s2.pop();

elif sum(s3)>=sum(s2) and sum(s3)>=sum(s1):

s3.pop();

if \_\_name\_\_ == '\_\_main\_\_':

n1=int(input('Enter Size of Element in Stack-1:- '))

n2=int(input('Enter Size of Element in Stack-1:- '))

n3=int(input('Enter Size of Element in Stack-1:- '))

h1=[]

h2=[]

h3=[]

for i in range(0,n1):

h1.append(int(input('Enter Element['+str(i)+'] in Stack1:- ')))

for i in range(0,n2):

h2.append(int(input('Enter Element['+str(i)+'] in Stack2:- ')))

for i in range(0,n3):

h3.append(int(input('Enter Element['+str(i)+'] in Stack3:- ')))

ans=max\_equal\_stack(h1,h2,h3)

print('Result Height:-'+str(ans))

Q7. A String will be consider as a child of a another string if it can be formed by deleteing 0 or more character from the other string.

Given two string of equal length,what's the longest string that can be formed such that it is a child of both?

Example

1)

Input:- "ABCD" and "ABDC"

Output:- 3 [ABC & ABD]

2)

Input:- "ABCNCHAN" and "NOHARAAA"

Output:- 3 [NHA]

Sol.

def LCS(x,y,size):

memo=[[0]\*(size+1)]\*(size+1)

for i in range(0,size+1):

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

if i==0 or j==0:

memo[i][j]=0

elif x[i-1]==y[j-1]:

print(x[i-1],end='')

memo[i][j]=memo[i-1][j-1]+1

else:

memo[i][j]=max(memo[i-1][j],memo[i][j-1]);

print('')

return memo[size][size];

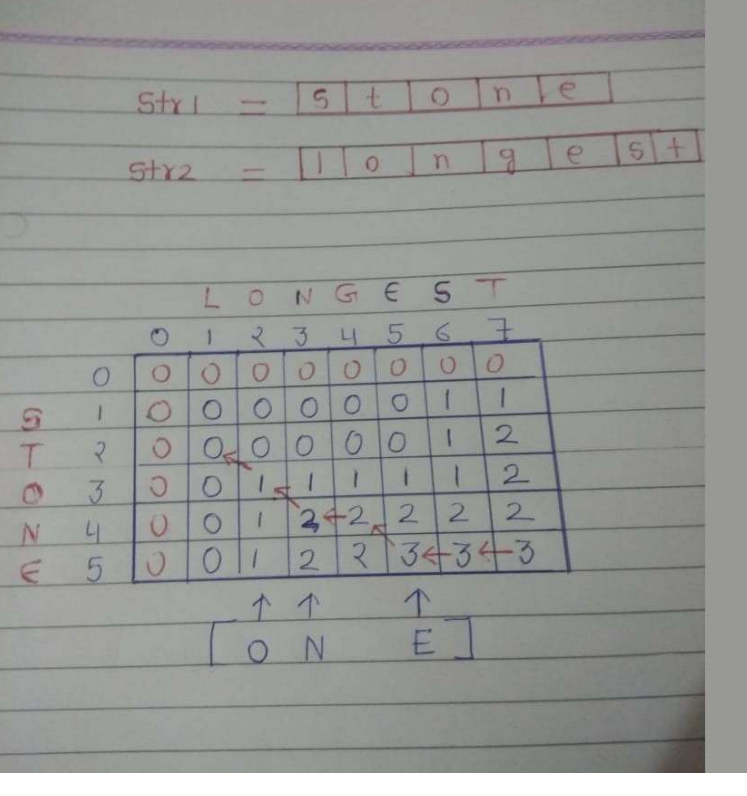
if \_\_name\_\_ == '\_\_main\_\_':

str1=input('Enter String01:- ')

str2=input('Enter String02:- ')

result=LCS(list(str1),list(str2),len(str1));

print("Size of Longest Comman Child in both string is "+str(result));



Q8. Virendra has several jars, each contains number of balls in it. He has just enough jars to sort each type of ball he has into its own jar.virendra wants to sort the balls using his sort method.

Virendra wants to perform some number of swap operations such that:

1) Each jars contains only balls of the same type.

2) No two balls of the same type are located in different jars.

You have to find out possibility of sorting balls in form as "Possible" or "Impossible".

Constraint:-

In a single operation, Virendra can swap two balls located in different Jars.

Example:-

Total Jars:- 2

Matrix:-

1 2

3 3

Output:-Impossible

Sol.

def organizingJars(jars,n):

row=[0]\*n

col=[0]\*n

for i in range(0,n):

for j in range(0,n):

row[i]+=jars[i][j]

col[i]+=jars[j][i]

if sorted(row)==sorted(col):

return "Yes it's Possible"

else:

return "It's Impossible"

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Number of Jars:- '))

t=int(input('Total Types of Ball:- '))

jars=[[0]\*t]\*n

for r in range(n):

for c in range(t):

jars[r][c]=int(input('In Jar['+str(r)+'] Total balls of type['+str(c)+']:- '))

print(organizingJars(jars,n))

Q9. Arun wants to perform sorting of an array of size n Element with following contraints

1) Arun don't want to use in-built function for sorting

2) Only singal for-loop or while-loop is allowed

Sol.

def sortInSingalLoop(arr,n):

count=0

i=0

while i<=n-2:

if arr[i]>arr[i+1]:

arr[i],arr[i+1]=arr[i+1],arr[i]

count+=1

if count==0 and i==n-2:

break

elif count!=0 and i==n-2:

i=-1

count=0

i+=1

print(arr)

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter size of array:- '))

arr=[]

for i in range(0,n):

arr.append(int(input('Enter Element['+str(i)+']:- ')))

Q10. Arun loves cookies and wants the sweetness of some cookies to be greater than value K. To do this, two cookies with the least sweetness are repeatedly mixed. This creates a special combined cookie with:

sweetness=(1st Least sweet cookie + 2 X 2nd least sweet cookie).

This occurs until all the cookies have a sweetness>=K.

Given the sweetness of a number of cookies, determine the minimum number of operations required. If it is not possible, return -1.

Example

K=9

A=[2,7,3,6,4,6]

Result=4

Sol.

import heapq as hq

def totalOperation(arr,n,k):

operation=0

while any(e < k for e in arr) and len(arr) > 1:

a1 = hq.heappop(arr)

a2 = hq.heappop(arr)

new = a1 + 2\*a2

hq.heappush(arr, new)

operation += 1

if all(e >= k for e in arr):

return operation

else:

return -1

if \_\_name\_\_ == "\_\_main\_\_":

n=int(input('Enter size of array:- '))

k=int(input('Enter value of K:- '))

arr=[]

for i in range(0,n):

e=int(input('Enter Element['+str(i)+']:- '))

hq.heappush(arr,e)

print("Total Operation:- "+str(totalOperation(arr,n,k)))

Q11. Arun and sangeeta have created a digits game. They pick a number and check to see if it is a power of 2. If it is, they divide it by 2. If not, they reduce it by the next lower number which is a power of 2. Whoever reduces the number to 1 wins the game.Arun always starts the game.you have to find out who win the game?

Note:-Don't use any built-in function

Example

n=132

It's Arun's turn first.he determines that 132 is not a power of 2. The next lower power of 2 is 128,so he subtracts that from 132 and passes 4 to sangeeta.4 is a power of 2, so sangeeta divides it by 2 and passes 2 to arun.Likewise,2 is a power so she divides it by 2 and reaches 1.Arun wins the game.Hey Arun congratulations dear.

Sol.

def getPerfactSqaureClosest(x):

while x & (x-1):

x=x-1

return x

def getReductionResult(x):

reductions = 4

while (x != 1):

if x & (x - 1):

x -= getClosestSmaller(x)

else:

x /= 2

reductions += 1

return reductions

if \_\_name\_\_ == '\_\_main\_\_':

n = input('Enter any random Number:- ')

if getReductionResult(n) % 2 != 0:

print "Arun"

else:

print "Sangeeta"

Q12. Given two integer,L and R,find the maximal value of a xor b,written a+b,where a and b satisfy the follwoing condition:

L<= a <= b <=R

Example:-

if L=11 and R=12 then

11 + 11 = 0 ['+' represnt xor operation]

11 + 12 = 7

12 + 12 = 0

Result:- 7

Sol.

def find\_Max\_XOR(L,R):

max\_xor = 0

for l in range(L,R+1):

for r in range(L,R+1):

max\_xor=max(max\_xor,l ^ r)

return max\_xor

if \_\_name\_\_ == '\_\_main\_\_':

l=int(input('Enter value of L:- '))

r=int(input('Enter value of R:- '))

ans=find\_Max\_XOR(l,r)

print("Max XOR:- "+str(ans))

Q13. You have to take stack of size N Elements.and Perform Sorting of stack Element with help of only Stack-Data Strcture only.

Sol.

def stackSorting(stack,n):

tmpStack=[]

while len(stack)!=0:

tmp = stack.pop()

while len(tmpStack)!=0 and tmpStack[len(tmpStack)-1]>tmp:

stack.append(tmpStack.pop())

tmpStack.append(tmp)

print(tmpStack);

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter Size of n:- '))

stack=[]

for i in range(n):

stack.append(int(input('Enter Element['+str(i)+']:- ')))

stackSorting(stack,n)

Q14. Virendra asks arun to watch movie,But arun busy in doing homework. virendra wants to help arun to finish it faster,Can you help virendra to understand arun's homework so he can watch movie with him?

Consider an array of n distinct numbers,arr=[a[0],a[1]...a[n-1]].Virendra can swap any two numbers of the array any number of times. An array is perfect if the sum of

|a[i]-a[i-1]| among 0< i <n is minimal.

Given the array arr, determine and return the minimum number of swaps that should be performed in order to make the array perfect.

Example

arr=[7,15,12,13]

One minimal array is [3,7,12,15].to get this result, virendra performed the following swaps:

Swap Result

[7, 15, 12, 3]

3 7 [3, 15, 12, 7]

7 15 [3, 7, 12, 15]

It took 2 swaps to make the array perfect.This is minimal among the choices of perfect arrays possible.

Example:-

n=4

arr=[2 5 3 1]

Result:- 2

Sol.

def no\_of\_swap(arr):

indexmap={}

for i in range(len(arr)):

indexmap[arr[i]]=i

sorted\_arr=sorted(arr)

result=0

for i in range(len(arr)):

if arr[i]!=sorted\_arr[i]:

result+=1

swap\_index=indexmap[sorted\_arr[i]]

indexmap[arr[i]]=swap\_index

arr[i],arr[swap\_index]=arr[swap\_index],arr[i]

return result

def min\_swap(arr,n):

n\_order=no\_of\_swap(arr[::])

r\_order=no\_of\_swap(arr[::-1])

return min(n\_order,r\_order)

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter Size of array:- '))

arr=[int(input('Enter Element['+str(i)+']')) for i in range(n)]

result=min\_swap(arr,n)

print('Min-Swap Required:- '+str(result))

Q15. Input any string and remove all adjcent duplicates characters in string with help of stack only.

Example:-

str="aaafghyyybbcrrr"

Result="afghybcr"

Sol.

def removeDuplicate(str1):

st=[]

for ch in str1:

if len(st)==0 or ch!=st[len(st)-1]:

st.append(ch)

newString=""

for ch in st:

newString+=ch

return(newString)

if \_\_name\_\_ == "\_\_main\_\_":

st=input('Enter any String:- ')

print(removeDuplicate(st))

Q16. Find the number of ways that a given integer X,can be expressed as the sum of the Nth powers of unique, natural numbers.

For example, if X=13 and N=2,we have to find all combinations of unique squares adding up to 13.The only solution is 2\*\*2 + 3\*\*2.

output:- 1

Sol.

def powerSum(x,power,num):

variable=pow(num,power)

if variable<x:

return powerSum(x,power,num+1)+powerSum(x-variable,power,num+1)

if x==variable:

return 1

else:

return 0

if \_\_name\_\_ == '\_\_main\_\_':

X=int(input("Enter Value of X:- "))

N=int(input("Enter Value of N:- "))

Num=1

result=powerSum(X,N,Num)

print(result);

Q17. JDA is planning to demolish a number of old, unoccupied buildings and construct a shopping mall in their place. Your task is to find the largest solid area in which the mall can be constructed.

There are a number of buildings in a certain two-dimensional landscape. Each building has a height, given by h[i] where 1<=i<=n. If you join K adjacent buildings, they will form a solid rectangle of area k x min(h[i],h[i+1].....h[n]).

Example

h=[3,2,3]

A rectangle of height h=2 and length k=3 can be constructed within the boundaries. The area formed is h x k=2\*3=6.

Example:-

n = 5

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

Result=9

Sol.

def largestArea(h):

stack=[]

area=0

index=0

h.append(0)

while(index<len(h)):

if len(stack)==0 or h[stack[-1]]<h[index]:

stack.append(index)

index+=1

else:

top=stack.pop();

if len(stack)!=0:

area=max(area,h[top]\*(index-stack[-1]-1))

else:

area=max(area,h[top]\*index)

return area

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter Size of Array:- '))

h=[]

for i in range(0,n):

h.append(int(input('Enter Element['+str(i)+']:- ')))

print(largestArea(h))

Q18. Find out all perutation of string with help of recursion only.

Example:-

str="XYZ"

output=["XYZ","XZY","YXZ","YZX","ZXY","ZYX"]

Sol.

def permute(inx,out,used):

if len(out) == len(inx):

print(out)

return

for i in range(0,len(inx)):

if used[i]:

continue;

out+=inx[i]

used[i]=True

print("permute("+str(inx)+","+str(out)+","+str(used)+")")

permute(inx,out,used)

used[i]=False

out=out[0:len(out)-1]

if \_\_name\_\_ == '\_\_main\_\_':

inx=input('Enter any String:- ')

out=''

used=[False]\*len(inx)

permute(inx,out,used);

Q19. Jodhpur is a one-dimensional city with houses aligned along a road. The collector wants to install mobile towers on the roofs of the city's houses. Each tower has a fixed range meaning it can transmit a signal to all houses within that number of units distance away.

Given a map of Jodhpur and the tower range, determine the minimum number of towers so that every house is within range of at least one tower. Each tower must be installed on top of an existing house.

Example:

Array=[3,4,5,8,12]

Range=1

Result=3

Sol.

def min\_Tower(h,tower\_range):

h.sort()

count=0

i=0

while i<len(h):

count+=1

max\_tower\_house=h[i]+tower\_range

while i<len(h) and h[i]<=max\_tower\_house:

i+=1

last\_house=h[i-1]+tower\_range

while i < len(h) and h[i]<=last\_house:

i+=1

return count

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter size of Array:- '))

h=[]

for i in range(n):

h.append(int(input('Enter house['+str(i)+']')))

tower\_range=int(input('Enter Tower Range:- '))

print(min\_Tower(h,tower\_range))

Q20. Given an m x n 2D binary Matrix which represents a Map of '1's (land) and '0's (water), return the number of islands.

An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Input: grid = [

["1","1","1","1","0"],

["1","1","0","1","0"],

["1","1","0","0","0"],

["0","0","0","0","0"]

]

Output: 1

Input: grid = [

["1","1","0","0","0"],

["1","1","0","0","0"],

["0","0","1","0","0"],

["0","0","0","1","1"]

]

Output: 3

Sol.

def num\_Of\_Islands(grid):

if not grid:

return 0

count = 0

for i in range(len(grid)):

for j in range(len(grid[0])):

if grid[i][j] == '1':

dfs(grid, i, j)

count += 1

return count

def dfs(grid, i, j):

if i<0 or j<0 or i>=len(grid) or j>=len(grid[0]) or grid[i][j] != '1':

return

grid[i][j] = '#'

dfs(grid, i+1, j)

dfs(grid, i-1, j)

dfs(grid, i, j+1)

dfs(grid, i, j-1)

if \_\_name\_\_ == '\_\_main\_\_':

m=int(input('Enter Number of rows:- '))

n=int(input('Enter Number of columns:- '))

matrix=[[input("Enter Element["+str(y)+"]["+str(x)+"]:- ") for x in range (n)] for y in range(m)]

print('Total Island:- '+str(num\_Of\_Islands(matrix)))

Q21. Given string num representing a non-negative integer num, and an integer k, return the smallest possible integer after removing k digits from num.

Example:

Input: num = "1542218", k = 3

Output: "1218"

Input: num = "10312", k = 1

Output: "312"

Sol.

def removeDigits(n,k,k1):

stack=[]

for current\_digit in n:

while k and stack and stack[-1]>current\_digit:

stack.pop()

k=k-1

stack.append(current\_digit)

output=""

for digit in stack:

if k1>0:

output+=digit

k1=k1-1

if len(output):

return output

else:

return "0"

if \_\_name\_\_ == '\_\_main\_\_':

n=input('Enter any Number:- ')

k=int(input('Enter how many digits need to Remove:- '))

print('Minimum-Number:- '+removeDigits(n,k,k))

Q22. Given an m x n binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area.

Sol.

def max\_square(matrix,R,C):

if not matrix:

return 0

dp = [[0]\*C for \_ in range(R)]

max\_count = 0

for i in range(R):

for j in range(C):

dp[i][j]=matrix[i][j]

if dp[i][j]:

max\_count = 1

for i in range(1,R):

for j in range(1,C):

if dp[i][j] and dp[i-1][j] and dp[i][j-1] and dp[i-1][j-1]:

dp[i][j] = min(dp[i-1][j-1],dp[i-1][j],dp[i][j-1])+1

if dp[i][j]>max\_count:

max\_count = dp[i][j]

return max\_count\*\*2

if \_\_name\_\_ == '\_\_main\_\_':

m=int(input('Enter size of m:- '))

n=int(input('Enter size of n:- '))

matrix=[[int(input("Enter Element["+str(y)+"]["+str(x)+"]:- ")) for x in range (n)] for y in range(m)]

print(max\_square(matrix,m,n))

Q23. Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

Example:-

Input: s = "()[]{}"

Output: true

Input: s = "([)]"

Output: false

Input: s = "{[]}"

Output: true

Sol.

def isValid(s):

stack = []

for char in s:

if char in ['(' , '[' , '{']:

stack.append(char)

else:

if not stack:

return False

current\_char = stack.pop()

if current\_char == '(':

if char != ")":

return False

if current\_char == '{':

if char != "}":

return False

if current\_char == '[':

if char != "]":

return False

if len(stack)>0:

return False

return True

if \_\_name\_\_ == '\_\_main\_\_':

st=input('Enter any string:- ')

print(isValid(st))

Q24. You are given an m x n grid where each cell can have one of three values:

0 representing an empty cell,

1 representing a fresh orange, or

2 representing a rotten orange.

Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.

Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.

Sol.

def RottingTime(matrix,r,c):

queue=collections.deque([])

count = 0

for i in range(r):

for j in range(c):

if matrix[i][j] == 1: count += 1

if matrix[i][j] == 2: queue.append((i,j))

timeFrame = 0

while queue:

for \_ in range(len(queue)):

i,j = queue.popleft()

for x, y in [(i+1,j), (i-1,j), (i,j+1), (i,j-1)]:

if 0<=x<r and 0<=y<c and matrix[x][y] == 1:

matrix[x][y] = 2

count -= 1

queue.append((x,y))

timeFrame += 1

return max(0, timeFrame-1) if count == 0 else -1

if \_\_name\_\_ == '\_\_main\_\_':

r=int(input('Enter Total Rows:- '))

c=int(input('Enter Total Columns:- '))

matrix=[]

for i in range(r):

data=[]

for j in range(c):

data.append(int(input('Enter Element['+str(i+1)+']['+str(j+1)+']')))

matrix.append(data)

print(RottingTime(matrix,r,c))

Q25. Given an array of integers nums and an integer k, return the total number of continuous subarrays whose sum equals to k.

Example:

array = [1,1,1], k = 2

Output: 2

array = [1,2,3], k = 3

Output: 2

Sol.

if \_\_name\_\_ == '\_\_main\_\_':

def subArraySum(nums,k):

count=0

for start in range(len(nums)):

for end in range(start+1,len(nums)):

sum=0

for i in range(start,end):

sum += nums[i]

if sum==k:

count+=1

return count

if \_\_name\_\_ == '\_\_main\_\_':

size=int('Enter size of array:- ')

array=[]

for i in range(size):

array.append(int(input('Enter Element['+str(i+1)+']:- ')))

k=int(input('Enter sum of subarrays you want:- '))

print(subArraySum(array,k))

Q26. Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.

grid = [

[1,3,1],

[1,5,1],

[4,2,1]

]

Output: 7

Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.

Sol.

def minPathSum(grid,M,N):

cost = [[0]\*N for \_ in range(M)]

cost[0][0] = grid[0][0]

for j in range(1,N):

cost[0][j] = grid[0][j] + cost[0][j-1]

for i in range(1,M):

cost[i][0] = grid[i][0] + cost[i-1][0]

for i in range(1,M):

for j in range(1,N):

cost[i][j] = min(cost[i-1][j], cost[i][j-1]) + grid[i][j]

return cost[M-1][N-1]

if \_\_name\_\_ == '\_\_main\_\_':

r=int(input('Enter Total Rows:- '))

c=int(input('Enter Total Columns:- '))

matrix=[]

for i in range(r):

data=[]

for j in range(c):

data.append(int(input('Enter Element['+str(i+1)+']['+str(j+1)+']')))

matrix.append(data)

print(minPathSum(matrix,r,c))

Q27. Given a binary array nums, return the maximum length of a contiguous subarray with an equal number of 0 and 1.

Example:

nums=[0,1,1,0,1]

Output: 4

nums=[0,1,0]

Output: 2

Sol.

def findMaxLength(data):

counter={}

counter[0]=-1

max\_count=0

count=0

for i in range(len(data)):

if data[i]==1:

count=count+1

else:

count=count-1

if count in counter:

max\_count=max(max\_count,i-counter[count])

else:

counter[count]=i

return max\_count

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter size of Array:- '))

data=[]

for i in range(n):

data.append(int(input('Enter Element['+str(i+1)+']')))

print(findMaxLength(data))

Q28. Given a matrix of m x n elements,return all elements of the matrix in spiral order.

For example, given the following matrix:

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

You should return [1,2,3,6,9,8,7,4,5].

Sol.

def printSpiralOrder(matrix,m,n):

result=[];

if m==0 or n==0:

return result

left=0

right=n-1

top=0

bottom=m-1

while len(result)<(m\*n):

for j in range(left,right+1):

result.append(matrix[top][j])

top+=1

for i in range(top,bottom+1):

result.append(matrix[i][right])

right-=1

if bottom<top:

break

for j in range(right,left-1,-1):

result.append(matrix[bottom][j])

bottom-=1

if right<left:

break

for i in range(bottom,top-1,-1):

result.append(matrix[i][left])

left+=1

return result

if \_\_name\_\_ == '\_\_main\_\_':

r=int(input('Enter Rows:- '))

c=int(input('Enter Columns:- '))

matrix=[]

for i in range(r):

data=[]

for j in range(c):

data.append(int(input('Enter Element['+str(i+1)+']['+str(j+1)+']:- ')))

matrix.append(data)

print(printSpiralOrder(matrix,r,c))

Q29. Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

For example, given the following triangle

[

[4],

[2,4],

[6,5,7],

[4,1,8,3]

]

The minimum path sum from top to bottom is 12 (4 + 2 + 5 + 1 = 12).

Sol.

def minPathSum(triangle):

if len(triangle) == 1:

return triangle[0][0]

m = len(triangle)

sum=0

for i in range(0,m):

sum+=min(triangle[i][:])

return sum

if \_\_name\_\_ == '\_\_main\_\_':

r=int(input('Enter Total Rows:- '))

triangle=[]

for i in range(0,r):

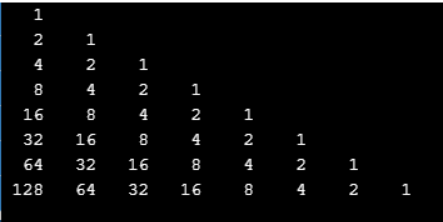
data=[]

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

data.append(int(input('Enter Element['+str(i+1)+']['+str(j+1)+']')))

triangle.append(data)

print(minPathSum(triangle))



Q30.

Sol.

if \_\_name\_\_ == '\_\_main\_\_':

n=1

for r in range(0,5):

e=pow(2,r)

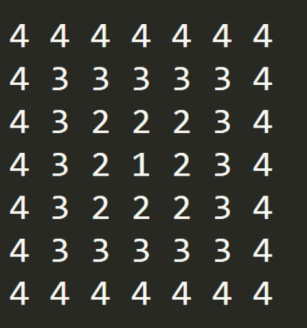
for c in range(0,r+1):

print(e,end=' ')

e=e//2

n+=1

print('')

Q31.

Sol.

def drawPattern(n):

size=2\*n-1

start=0

end=size-1

a=[[0]\*size]\*size

while n!=0:

for i in range(start,end+1):

for j in range(start,end+1):

if i==start or i==end or j==start or j==end:

a[i][j]=n

start+=1

end-=1

n-=1

return a

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter Size of Pattern'))

a=drawPattern(n)

size=2\*n-1

start=0

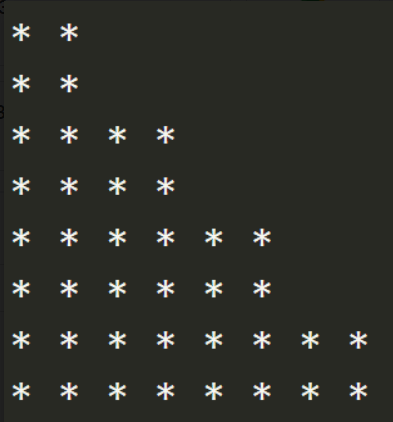
end=size-1

for i in range(start,end+1):

for j in range(start,end+1):

print(a[i][j],end=" ")

print('')

Q32.

Sol.

def drawPattern(n):

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

k = i + 1 if (i % 2 != 0) else i

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

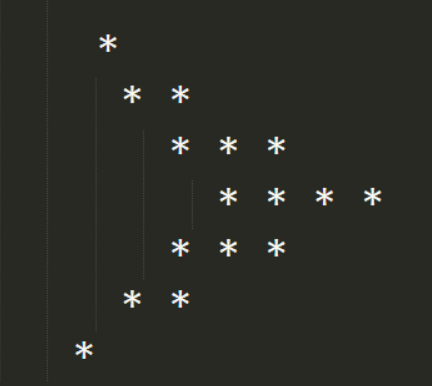
print(" \* ",end=' ')

print('')

if \_\_name\_\_ == '\_\_main\_\_':

n=int(input('Enter size of Pattern:- '))

drawPattern(n)

Q33.

Sol.

def triangle(n):

for i in range(1,n):

print(" ",end='')

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

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

print("")

def drawPattern(size):

for i in range(1,(size+1)//2+1):

triangle(i)

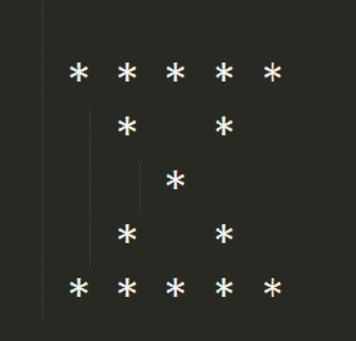
for i in range((size)//2,0,-1):

triangle(i)

if \_\_name\_\_ == '\_\_main\_\_':

size=int(input('Enter Size of Pattern(Only Odd Number):- '))

drawPattern(size)

Q34.

Sol.

def drawPattern(h):

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

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

if i==j or i+j==h+1 or i==1 or i==h:

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

else:

print(" ",end='')

print("")

if \_\_name\_\_ == '\_\_main\_\_':

height=int(input("Enter Height of Pattern:- "))

drawPattern(height)