Leap year

def CheckLeap(Year):  
 if ((Year % 400 == 0) or  
 (Year % 100 != 0) and  
 (Year % 4 == 0)):  
 print("Given Year is a leap Year")  
 else:  
 print("Given Year is not a leap Year")  
Year = int(input("Enter the number: "))  
CheckLeap(Year)

pascal triangle

num =int(input("enter the rows:"))  
num=int(input("enter the row number:"))  
def solve(n):  
 for i in range(n+1):  
 for j in range(n-i):  
 print(' ', end='')  
 C = 1  
 for j in range(1, i+1):  
 print(C, ' ', sep='', end='')  
 C = C \* (i - j) // j  
 print()  
solve(num)

sorted list

list1 =[8,9,10,20]  
list2 =[12,15,25,30]  
sorted\_list =[]  
while list1 and list2:  
 if list1[0] < list2[0]:  
 sorted\_list.append(list1.pop(0))  
 else:  
 sorted\_list.append(list2.pop(0))  
sorted\_list += list1  
sorted\_list += list2  
print(sorted\_list)

Reverse string

def reverse(str):  
 str = str[::-1]  
 return str  
s ="mahammad"  
print("The reversed string using extended slice operator is : ", reverse(s))

combination of two strings(remove common words)

from collections import Counter  
def removeCommonWords(sent1, sent2):  
 sentence1 = list(sent1.split())  
 sentence2 = list(sent2.split())  
 frequency1 = Counter(sentence1)  
 frequency2 = Counter(sentence2)  
 word = 0  
 for i in range(len(sentence1)):  
 if sentence1[word] in frequency2.keys():  
 sentence1.pop(word)  
 word = word - 1  
 word += 1  
 word = 0  
 for i in range(len(sentence2)):  
 if sentence2[word] in frequency1.keys():  
 sentence2.pop(word)  
 word = word - 1  
 word += 1  
 print(\*sentence1)  
 print(\*sentence2)  
sentence1 = input("Enter Your Sentence 1 : ")  
sentence2 = input("Enter Your Sentence 2 : ")  
removeCommonWords(sentence1, sentence2)

binary numbers

def add\_binary\_nums(x, y):  
 max\_len = max(len(x), len(y))  
 x = x.zfill(max\_len)  
 y = y.zfill(max\_len)  
 result = ''  
 carry = 0  
 for i in range(max\_len - 1, -1, -1):  
 r = carry  
 r += 1 if x[i] == '1' else 0  
 r += 1 if y[i] == '1' else 0  
 result = ('1' if r % 2 == 1 else '0') + result  
 carry = 0 if r < 2 else 1 # Compute the carry.  
 if carry != 0: result = '1' + result  
 return result.zfill(max\_len)  
print(add\_binary\_nums('11', '1'))

permutation

def permute(LIST):  
 length = len(LIST)  
 if length <= 1:  
 yield LIST  
 else:  
 for n in range(0, length):  
 for end in permute(LIST[:n] + LIST[n + 1:]):  
 yield [LIST[n]] + end  
for x in permute(["1", "2", "3"]):  
 print(x)

common two strings

def editDistance(str1, str2, m, n):  
 if m == 0:  
 return n  
 if n == 0:  
 return m  
 if str1[m-1] == str2[n-1]:  
 return editDistance(str1, str2, m-1, n-1)  
 return 1 + min(editDistance(str1, str2, m, n-1),  
 editDistance(str1, str2, m-1, n),  
 editDistance(str1, str2, m-1, n-1)  
 )  
str1 = "horse"  
str2 = "ros"  
print (editDistance(str1, str2, len(str1), len(str2)))

isomorphic

def areIsomorphic(str1, str2):  
 charCount = dict()  
 c = "a"  
 for i in range(len(str1)):  
 # if str1[i] is a key in charCount  
 if str1[i] in charCount:  
 c = charCount[str1[i]]  
 if c != str2[i]:  
 return False  
 elif str2[i] not in charCount.values():  
 charCount[str1[i]] = str2[i]  
 else:  
 return False  
 return True  
str1 = str(input("enter the string :"))  
str2 = str(input("enter the string 2:"))  
if (len(str1) == len(str2) and areIsomorphic(str1, str2)):  
 print("true")  
else:  
 print("false")

array of integer good pairs

def solve(nums):  
 count = 0  
 n = len(nums)  
 for i in range(n):  
 for j in range(i + 1, n):  
 if nums[i] == nums[j]:  
 count += 1  
 return count  
nums = [5, 6, 7, 5, 5, 7]  
print(solve(nums))

climb staries

def climbStairs(n):  
 steps = []  
 steps.append(1)  
 steps.append(2)  
 for i in range(2, n):  
 steps.append([i - 1] + [i - 2])  
 return steps[n - 1]  
n=int(input("enter the number of stairs:"))  
print(climbStairs(n))

greatest of three numbers

a=int(input("enter the number a:" ))  
b=int(input("enter the value b:"))  
c=int(input("enter the number c:"))  
if(a>b)and(a>c):  
 print("a is largest")  
elif(b>c)and(c>a):  
 print("b is largest")  
else:  
 print("c is largest")

max num of jumps

def minJumps(arr,l,h):  
 if (h==l):  
 return 0  
 if (arr[l]==0):  
 return float('inf')  
 min = float('inf')  
 for i in range(l + 1, h + 1):  
 if (i < l + arr[l] + 1):  
 jumps = minJumps(arr, i, h)  
 if (jumps != float('inf') and  
 jumps + 1 < min):  
 min = jumps + 1  
 return min  
arr = [1,3,5,8,9,2,6,7,6,8,9]  
n = len(arr)  
print('Minimum number of jumps to reach','end is', minJumps(arr, 0, n - 1))

parenthesis

def printParenthesis(str, n):  
 if (n > 0):  
 \_printParenthesis(str, 0,n, 0, 0)  
 return  
def \_printParenthesis(str, pos, n,open, close):  
 if (close == n):  
 for i in str:  
 print(i, end="")  
 print()  
 return  
 else:  
 if (open > close):  
 str[pos] = ')'  
 \_printParenthesis(str, pos + 1, n,open, close + 1)  
 if (open < n):  
 str[pos] = '('  
 \_printParenthesis(str, pos + 1, n, open + 1, close)  
n =int(input("enter the number:"))  
str = [""] \* 2 \* n  
printParenthesis(str, n)

possible combinations

def comb(L):  
 for i in range(3):  
 for j in range(3):  
 for k in range(3):  
 if (i != j and j != k and i != k):  
 print(L[i], L[j], L[k])  
comb([1, 2, 3])

profit

def maxProfit(price, n):  
 profit = [0]\*n  
 max\_price = price[n-1]  
 for i in range(n-2, 0, -1):  
 if price[i] > max\_price:  
 max\_price = price[i]  
 profit[i] = max(profit[i+1], max\_price - price[i])  
 min\_price = price[0]  
 for i in range(1, n):  
 if price[i] < min\_price:  
 min\_price = price[i]  
 profit[i] = max(profit[i-1], profit[i]+(price[i]-min\_price))  
 result = profit[n-1]  
 return result  
price = [7,1,5,3,6,4]  
print ("Maximum profit is:", maxProfit(price, len(price)))

pyramid

print("print equilaterial triangle pyramid using asterisk symbol")  
size = 7  
m = (2 \* size) - 2  
for i in range(0, size):  
 for j in range(0, m):  
 print(end=" ")  
 m = m - 1  
 for j in range(0, i + 1):  
 print("\* ", end=' ')  
 print(" ")

roman numbers

def value(r):  
 if (r == 'I'):  
 return 1  
 if (r == 'V'):  
 return 5  
 if (r == 'X'):  
 return 10  
 if (r == 'L'):  
 return 50  
 if (r == 'C'):  
 return 100  
 if (r == 'D'):  
 return 500  
 if (r == 'M'):  
 return 1000  
 return -1  
  
  
def romanToDecimal(str):  
 res = 0  
 i = 0  
  
 while (i < len(str)):  
  
 # Getting value of symbol s[i]  
 s1 = value(str[i])  
  
 if (i + 1 < len(str)):  
  
 # Getting value of symbol s[i + 1]  
 s2 = value(str[i + 1])  
  
 # Comparing both values  
 if (s1 >= s2):  
  
 # Value of current symbol is greater  
 # or equal to the next symbol  
 res = res + s1  
 i = i + 1  
 else:  
  
 # Value of current symbol is greater  
 # or equal to the next symbol  
 res = res + s2 - s1  
 i = i + 2  
 else:  
 res = res + s1  
 i = i + 1  
  
 return res  
print("Integer form of Roman Numeral is"),  
print(romanToDecimal("MCMIV"))

student details

a=int(input("total users: "))  
b=int(input("staff users: "))  
if(a<=0 or a==b or a==450 or a==600):  
 print("invalid input")  
else:  
 y=b/3  
 s=a-b-y  
 print("student users: ",s)

triangle

num =int(input("enter the rows:"))  
num=int(input("enter the row number:"))  
def solve(n):  
 for i in range(n+1):  
 for j in range(n-i):  
 print(' ', end='')  
 C = 1  
 for j in range(1, i+1):  
 print(C, ' ', sep='', end='')  
 C = C \* (i - j) // j  
 print()  
solve(num)

remove the letter

def delchar(s, c):  
 if len(c) == 1:  
 for i in s:  
 if c != i:  
 print(i, end="")  
 else:  
 print(s)  
delchar('hello world', 'l')

seasons

month = input("Input the month (January, February etc.): ")  
day = int(input("Input the day: "))  
if month in ('January', 'February'):  
 season = 'winter'  
elif month in ('April', 'May', 'June'):  
 season = 'spring'  
elif month in ('July', 'August', 'September','MARCH'):  
 season = 'summer'  
else:  
 season = 'autumn'  
if (month == 'March') and (day > 19):  
 season = 'SUMMER'  
elif (month == 'June') and (day > 20):  
 season = 'WINTER'  
elif (month == 'September') and (day > 21):  
 season = 'autumn'  
elif (month == 'December') and (day > 20):  
 season = 'winter'  
print("Season is",season)

palindrome

class Solution(object):  
 def isPalindrome(self, s):  
 *"""* ***:type*** *s: str* ***:rtype****: bool  
 """* x = ""  
 diff = ord('a') - ord('A')  
 for i in s:  
 if ord(i)>=ord('a') and ord(i)<=ord('z') or ord(i)>=ord("0") and ord(i)<=ord("9"):  
 x+=i  
 elif ord(i)>=ord('A') and ord(i)<=ord('Z'):  
 i = chr(diff+ord(i))  
 x+=i  
 #print(s)  
 #print(x)  
 return x == x[::-1]  
ob1 = Solution()  
print(ob1.isPalindrome("A Man, a Plan, a Canal: Panama"))