|  |  |  |
| --- | --- | --- |
| **CHAPTERS** | **SL.NO** | **PROGRAMS** |
| Looping statements | **1.**  **2.**  **3.**  **4.**  **5.**  **6.**  **7.**  **8.**  **9.**  **10.**  **11.**  **12.**  **13.** | **To Calculate the Average of Numbers in a Given List**  n=int(input("Enter the number of elements to be inserted: "))  a=[]  for i in range(0,n):  elem=int(input("Enter element: "))  a.append(elem)  avg=sum(a)/n  print("Average of elements in the list",round(avg,2))  **To Print all Numbers in a Range Divisible by a Given Number**  lower=int(input("Enter lower range limit:"))  upper=int(input("Enter upper range limit:"))  n=int(input("Enter the number to be divided by:"))  for i in range(lower,upper+1):  if(i%n==0):  print(i)  **To Accept Three Digits and Print all Possible Combinations from the Digits**  a=int(input("Enter first number:"))  b=int(input("Enter second number:"))  c=int(input("Enter third number:"))  d=[]  d.append(a)  d.append(b)  d.append(c)  for i in range(0,3):  for j in range(0,3):  for k in range(0,3):  if(i!=j&j!=k&k!=i):  print(d[i],d[j],d[k])  **To Generate all the Divisors of an Integer**  n=int(input("Enter an integer:"))  print("The divisors of the number are:")  for i in range(1,n+1):  if(n%i==0):  print(i,end=’ ’)  **To Find the Smallest Divisor of an Integer**  n=int(input("Enter an integer:"))  a=[]  for i in range(2,n+1):  if(n%i==0):  a.append(i)  a.sort()  print("Smallest divisor is:",a[0])  **To Find the LCM of Two Numbers**  a=int(input("Enter the first number:"))  b=int(input("Enter the second number:"))  if(a>b):  min1=a  else:  min1=b  while(1):  if(min1%a==0 and min1%b==0):  print("LCM is:",min1)  break  min1=min1+1  **To Print the Pascal’s triangle for n number of rows given by the user**  n=int(input("Enter number of rows: "))  a=[]  for i in range(n):  a.append([])  a[i].append(1)  for j in range(1,i):  a[i].append(a[i-1][j-1]+a[i-1][j])  if(n!=0):  a[i].append(1)  for i in range(n):  print(" "\*(n-i),end=" ",sep=" ")  for j in range(0,i+1):  print('{0:6}'.format(a[i][j]),end=" ",sep=" ")  print()  **To Find the Sum of the Series: 1 + x^2/2 + x^3/3 + … x^n/n+++**  n=int(input("Enter the number of terms:"))  x=int(input("Enter the value of x:"))  sum1=1  for i in range(2,n+1):  sum1=sum1+((x\*\*i)/i)  print("The sum of series is",round(sum1,2))  **To Check if a Number is a Palindrome+++**  n=int(input("Enter number:"))  temp=n  rev=0  while(n>0):  dig=n%10  rev=rev\*10+dig  n=n//10  if(temp==rev):  print("The number is a palindrome!")  else:  print("The number isn't a palindrome!")  **To Read a Number n and Print the Natural Numbers Summation Pattern**  n=int(input("Enter a number: "))  for j in range(1,n+1):  a=[]  for i in range(1,j+1):  print(i,sep=" ",end=" ")  if(i<j):  print("+",sep=" ",end=" ")  a.append(i)  print("=",sum(a))    print()  **To Print an Inverted Star Pattern**  n=int(input("Enter number of rows: "))  for i in range (n,0,-1):  print((n-i) \* ' ' + i \* '\*')  **To Print an Identity Matrix**  n=int(input("Enter a number: "))  for i in range(0,n):  for j in range(0,n):  if(i==j):  print("1",sep=" ",end=" ")  else:  print("0",sep=" ",end=" ")  print()  **Write a program to compute 1/2+2/3+3/4+...+n/n+1 with a given n input by console (n>0).**  n=int(input())  sum=0.0  for i in range(1,n+1):  sum += float(float(i)/(i+1))  print(sum) |
| Conditional statements | **14.**  **15.**  **16.** | **To Check if a Number is a Perfect Number**  n = int(input("Enter any number: "))  sum1 = 0  for i in range(1, n):  if(n % i == 0):  sum1 = sum1 + i  if (sum1 == n):  print("The number is a Perfect number!")  else:  print("The number is not a Perfect number!")  **To Print all Integers that Aren’t Divisible by Either 2 or 3 and Lie between 1 and 50**  for i in range(0,51):  if(i%2!=0&i%3!=0):  print(i)  **To Check Whether a Given Year is a Leap Year**  year=int(input("Enter year to be checked:"))  if(year%4==0 and year%100!=0 or year%400==0):  print("The year is a leap year!)  else:  print("The year isn't a leap year!) |
| Lists | 17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48. | **To Find The Largest Number In A List**  a=[]  n=int(input("Enter number of elements:"))  for i in range(1,n+1):  b=int(input("Enter element:"))  a.append(b)  a.sort()  print("Largest element is:",a[n-1])  **To find the second largest number in a list**  a=[]  n=int(input("Enter number of elements:"))  for i in range(1,n+1):  b=int(input("Enter element:"))  a.append(b)  a.sort()  print("Second largest element is:",a[n-2])  **To put even and odd elements in a list into two different lists**  a=[]  n=int(input("Enter number of elements:"))  for i in range(1,n+1):  b=int(input("Enter element:"))  a.append(b)  even=[]  odd=[]  for j in a:  if(j%2==0):  even.append(j)  else:  odd.append(j)  print("The even list",even)  print("The odd list",odd)  **To Merge Two Lists and Sort it**  a=[]  c=[]  n1=int(input("Enter number of elements:"))  for i in range(1,n1+1):  b=int(input("Enter element:"))  a.append(b)  n2=int(input("Enter number of elements:"))  for i in range(1,n2+1):  d=int(input("Enter element:"))  c.append(d)  new=a+c  new.sort()  print("Sorted list is:",new)  **To Sort the List According to the Second Element in Sublist**  a=[['A',34],['B',21],['C',26]]  for i in range(0,len(a)):  for j in range(0,len(a)-i-1):  if(a[j][1]>a[j+1][1]):  temp=a[j]  a[j]=a[j+1]  a[j+1]=temp    print(a)  **To Find the Second Largest Number in a List Using Bubble Sort**  a=[]  n=int(input("Enter number of elements:"))  for i in range(1,n+1):  b=int(input("Enter element:"))  a.append(b)  for i in range(0,len(a)):  for j in range(0,len(a)-i-1):  if(a[j]>a[j+1]):  temp=a[j]  a[j]=a[j+1]  a[j+1]=temp  print('Second largest number is:',a[n-2])  **To Sort a List According to the Length of the Elements**  a=[]  n=int(input("Enter number of elements:"))  for i in range(1,n+1):  b=input("Enter element:")  a.append(b)  a.sort(key=len)  print(a)  **To Find the Union of two Lists**  l1 = []  num1 = int(input('Enter size of list 1: '))  for n in range(num1):  numbers1 = int(input('Enter any number:'))  l1.append(numbers1)    l2 = []  num2 = int(input('Enter size of list 2:'))  for n in range(num2):  numbers2 = int(input('Enter any number:'))  l2.append(numbers2)    union = list(set().union(l1,l2))    print('The Union of two lists is:',union)  **To Find the Intersection of Two Lists**  def intersection(a, b):  return list(set(a) & set(b))    def main():  alist=[]  blist=[]  n1=int(input("Enter number of elements for list1:"))  n2=int(input("Enter number of elements for list2:"))  print("For list1:")  for x in range(0,n1):  element=int(input("Enter element" + str(x+1) + ":"))  alist.append(element)  print("For list2:")  for x in range(0,n2):  element=int(input("Enter element" + str(x+1) + ":"))  blist.append(element)  print("The intersection is :")  print(intersection(alist, blist))  main()  **To Create a List of Tuples with the First Element as the Number and Second Element as the Square of the Number**  l\_range=int(input("Enter the lower range:"))  u\_range=int(input("Enter the upper range:"))  a=[(x,x\*\*2) for x in range(l\_range,u\_range+1)]  print(a)  **To Find all Numbers in a Range which are Perfect Squares and Sum of all Digits in the Number is Less than 10**  l=int(input("Enter lower range: "))  u=int(input("Enter upper range: "))  a=[]  a=[x for x in range(l,u+1) if (int(x\*\*0.5))\*\*2==x and sum(list(map(int,str(x))))<10]  print(a)  **To Find the Cumulative Sum of a List where the ith Element is the Sum of the First i+1 Elements From The Original List**  a=[]  n= int(input("Enter the number of elements in list:"))  for x in range(0,n):  element=int(input("Enter element" + str(x+1) + ":"))  a.append(element)  b=[sum(a[0:x+1]) for x in range(0,len(a))]  print("The original list is: ",a)  print("The new list is: ",b)  **To Sort a List of Tuples in Increasing Order by the Last Element in Each Tuple**  def last(n):  return n[-1]    def sort(tuples):  return sorted(tuples, key=last)  a=input("Enter a list of tuples:")  print("Sorted:")  print(sort(a))  **To Swap the First and Last Value of a List**  a=[]  n= int(input("Enter the number of elements in list:"))  for x in range(0,n):  element=int(input("Enter element" + str(x+1) + ":"))  a.append(element)  temp=a[0]  a[0]=a[n-1]  a[n-1]=temp  print("New list is:")  print(a)  **To Remove the Duplicate Items from a List**  a=[]  n= int(input("Enter the number of elements in list:"))  for x in range(0,n):  element=int(input("Enter element" + str(x+1) + ":"))  a.append(element)  b = set()  unique = []  for x in a:  if x not in b:  unique.append(x)  b.add(x)  print("Non-duplicate items:")  print(unique)  **To Read a List of Words and Return the Length of the Longest One**  a=[]  n= int(input("Enter the number of elements in list:"))  for x in range(0,n):  element=input("Enter element" + str(x+1) + ":")  a.append(element)  max1=len(a[0])  temp=a[0]  for i in a:  if(len(i)>max1):  max1=len(i)  temp=i  print("The word with the longest length is:")  print(temp)  **To Remove the ith Occurrence of the Given Word in a List where Words can Repeat**  a=[]  n= int(input("Enter the number of elements in list:"))  for x in range(0,n):  element=input("Enter element" + str(x+1) + ":")  a.append(element)  print(a)  c=[]  count=0  b=input("Enter word to remove: ")  n=int(input("Enter the occurrence to remove: "))  for i in a:  if(i==b):  count=count+1  if(count!=n):  c.append(i)  else:  c.append(i)  if(count==0):  print("Item not found ")  else:  print("The number of repetitions is: ",count)  print("Updated list is: ",c)  print("The distinct elements are: ",set(a))  **To Remove All Tuples in a List of Tuples with the USN Outside the Given Range**  y=[('a','12CS039'),('b','12CS320'),('c','12CS055'),('d','12CS100')]  low=int(input("Enter lower roll number (starting with 12CS):"))  up=int(input("Enter upper roll number (starting with 12CS):"))  l='12CS0'+str(low)  u='12CS'+str(up)  p=[x for x in y if x[1]>l and x[1]<u]  print(p)  **To Find Element Occurring Odd Number of Times in a List**  def find\_odd\_occurring(alist):  ans = 0  for element in alist:  ans = element  return ans  alist = input('Enter the list: ').split()  alist = [int(i) for i in alist]  ans = find\_odd\_occurring(alist)  print('The element that occurs odd number of times:', ans)  **Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5,**  **between 2000 and 3200 (both included).**  **The numbers obtained should be printed in a comma-separated sequence on a single line.**  l=[]  for i in range(2000, 3201):  if (i%7==0) and (i%5!=0):  l.append(str(i))  print (','.join(l))  **Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.**  **Suppose the following input is supplied to the program:**  **34,67,55,33,12,98**  **Then, the output should be:**  **['34', '67', '55', '33', '12', '98']**  **('34', '67', '55', '33', '12', '98')**  values=raw\_input()  l=values.split(",")  t=tuple(l)  print(l)  print(t)  **Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array. The element value in the i-th row and j-th column of the array should be i\*j.**  **Note: i=0,1.., X-1; j=0,1,¡­Y-1.**  **Example**  **Suppose the following inputs are given to the program:**  **3,5**  **Then, the output of the program should be:**  **[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]**  input\_str = input()  dimensions=[int(x) for x in input\_str.split(',')]  rowNum=dimensions[0]  colNum=dimensions[1]  multilist = [[0 for col in range(colNum)] for row in range(rowNum)]  for row in range(rowNum):  for col in range(colNum):  multilist[row][col]= row\*col  print(multilist)  **Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.**  **Suppose the following input is supplied to the program:**  **without,hello,bag,world**  **Then, the output should be:**  **bag,hello,without,world**  items=[x for x in input().split(',')]  items.sort()  print (','.join(items))  **Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.**  lines = []  while True:  s = input()  if s:  lines.append(s.upper())  else:  break  for sentence in lines:  print(sentence)  **Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically**.  s = input()  words = [word for word in s.split(" ")]  print (" ".join(sorted(list(set(words)))))  **Write a program to generate all sentences where subject is in ["I", "You"] and verb is in ["Play", "Love"] and the object is in ["Hockey","Football"].**  subjects=["I", "You"]  verbs=["Play", "Love"]  objects=["Hockey","Football"]  for i in range(len(subjects)):  for j in range(len(verbs)):  for k in range(len(objects)):  sentence = "%s %s %s." % (subjects[i], verbs[j], objects[k])  print(sentence)  **Write a program to print the list after removing delete even numbers in [5,6,77,45,22,12,24].**  li = [5,6,77,45,22,12,24]  li = [x for x in li if x%2!=0]  print(li)  **By using list comprehension, write a program to print the list after removing delete numbers which are divisible by 5 and 7 in [12,24,35,70,88,120,155].**  li = [12,24,35,70,88,120,155]  li = [x for x in li if x%5!=0 and x%7!=0]  print(li)  **By using list comprehension, write a program to print the list after removing the 0th, 2nd, 4th,6th numbers in [12,24,35,70,88,120,155].**  li = [12,24,35,70,88,120,155]  li = [x for (i,x) in enumerate(li) if i%2!=0]  print(li)  **By using list comprehension, write a program generate a 3\*5\*8 3D array whose each element is 0.**  array = [[ [0 for col in range(8)] for col in range(5)] for row in range(3)]  print(array)  **By using list comprehension, write a program to print the list after removing the 0th,4th,5th numbers in [12,24,35,70,88,120,155].**  li = [12,24,35,70,88,120,155]  li = [x for (i,x) in enumerate(li) if i not in (0,4,5)]  print(li)  **By using list comprehension, write a program to print the list after removing the value 24 in [12,24,35,24,88,120,155].**  li = [12,24,35,24,88,120,155]  li = [x for x in li if x!=24]  print li |
| Strings | 49.  50.  51.  52.  53.  54.  55.  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69. | **Replace all Occurrences of ‘a’ with $ in a String**  string=input("Enter string:")  string=string.replace('a','$')  string=string.replace('A','$')  print("Modified string:")  print(string)  **Write a program which accepts a string from console and print it in reverse order.**  s= input()  s = s[::-1]  print(s)  **Write a program which accepts a string from console and print the characters that have even indexes.**  s=input()  s = s[::2]  print(s)  **Remove the nth Index Character from a Non-Empty String**  def remove(string, n):  first = string[:n]  last = string[n+1:]  return first + last  string=input("Enter the sring:")  n=int(input("Enter the index of the character to remove:"))  print("Modified string:")  print(remove(string, n))  **Detect if Two Strings are Anagrams**  s1=input("Enter first string:")  s2=input("Enter second string:")  if(sorted(s1)==sorted(s2)):  print("The strings are anagrams.")  else:  print("The strings aren't anagrams.")  **Form a New String where the First Character and the Last Character have been Exchanged**  def change(string):  return string[-1:] + string[1:-1] + string[:1]  string=input("Enter string:")  print("Modified string:")  print(change(string))  **Count the Number of Vowels in a String**  string=raw\_input("Enter string:")  vowels=0  for i in string:  if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or i=='U'):  vowels=vowels+1  print("Number of vowels are:")  print(vowels)  **Take in a String and Replace Every Blank Space with Hyphen**  string=input("Enter string:")  string=string.replace(' ','-')  print("Modified string:")  print(string)  **Calculate the Length of a String Without Using a Library Function**  string=input("Enter string:")  count=0  for i in string:  count=count+1  print("Length of the string is:")  print(count)  **Remove the Characters of Odd Index Values in a String**  def modify(string):  final = ""  for i in range(len(string)):  if i % 2 == 0:  final = final + string[i]  return final  string=input("Enter string:")  print("Modified string is:")  print(modify(string))  **Calculate the Number of Words and the Number of Characters Present in a String**  string=input("Enter string:")  char=0  word=1  for i in string:  char=char+1  if(i==' '):  word=word+1  print("Number of words in the string:")  print(word)  print("Number of characters in the string:")  print(char)  **Take in Two Strings and Display the Larger String without Using Built-in Functions**  string1=input("Enter first string:")  string2=input("Enter second string:")  count1=0  count2=0  for i in string1:  count1=count1+1  for j in string2:  count2=count2+1  if(count1<count2):  print("Larger string is:")  print(string2)  elif(count1==count2):  print("Both strings are equal.")  else:  print("Larger string is:")  print(string1)  **Count Number of Lowercase Characters in a String**  string=input("Enter string:")  count=0  for i in string:  if(i.islower()):  count=count+1  print("The number of lowercase characters is:")  print(count)  **Check if a String is a Palindrome or Not**  string=input("Enter string:")  if(string==string[::-1]):  print("The string is a palindrome")  else:  print("The string isn't a palindrome")  **Calculate the Number of Upper Case Letters and Lower Case Letters in a String**  string=input("Enter string:")  count1=0  count2=0  for i in string:  if(i.islower()):  count1=count1+1  elif(i.isupper()):  count2=count2+1  print("The number of lowercase characters is:")  print(count1)  print("The number of uppercase characters is:")  print(count2)  **Check if a String is a Pangram or Not**  from string import ascii\_lowercase as asc\_lower  def check(s):  return set(asc\_lower) - set(s.lower()) == set([])  strng=input("Enter string:")  if(check(strng)==True):  print("The string is a pangram")  else:  print("The string isn't a pangram")  **Accept a Hyphen Separated Sequence of Words as Input and Print the Words in a Hyphen-Separated Sequence after Sorting them Alphabetically**  print("Enter a hyphen separated sequence of words:")  lst=[n for n in input().split('-')]  lst.sort()  print("Sorted:")  print('-'.join(lst))  **Calculate the Number of Digits and Letters in a String**  string=input("Enter string:")  count1=0  count2=0  for i in string:  if(i.isdigit()):  count1=count1+1  count2=count2+1  print("The number of digits is:")  print(count1)  print("The number of characters is:")  print(count2)  **Form a New String Made of the First 2 and Last 2 characters From a Given String**  string=input("Enter string:")  count=0  for i in string:  count=count+1  new=string[0:2]+string[count-2:count]  print("Newly formed string is:")  print(new)  **Count the Occurrences of a Word in a Given String Sentence**  string=input("Enter string:")  word=input("Enter word:")  a=[]  count=0  a=string.split(" ")  for i in range(0,len(a)):  if(word==a[i]):  count=count+1  print("Count of the word is:")  print(count)  **Check if a Substring is Present in a Given String**  string=input("Enter string:")  sub\_str=input("Enter word:")  if(string.find(sub\_str)==-1):  print("Substring not found in string!")  else:  print("Substring in string!") |
| Dictionary | 70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82. | **Add a Key-Value Pair to the Dictionary**  key=int(input("Enter the key (int) to be added:"))  value=int(input("Enter the value for the key to be added:"))  d={}  d.update({key:value})  print("Updated dictionary is:")  print(d)  **Write a program which count and print the numbers of each character in a string input by console.**  dic = {}  s=raw\_input()  for s in s:  dic[s] = dic.get(s,0)+1  print ('\n'.join(['%s,%s' % (k, v) for k, v in dic.items()]))  **With a given integral number n, write a program to generate a dictionary that contains (i, i\*i) such that is an integral number between 1 and n (both included) and then the program should print the dictionary.**  **Suppose the following input is supplied to the program:**  **8**  **Then, the output should be:**  **{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}**  n=int(raw\_input())  d=dict()  for i in range(1,n+1):  d[i]=i\*i  print (d)  **Concatenate Two Dictionaries Into One**  d1={'A':1,'B':2}  d2={'C':3}  d1.update(d2)  print("Concatenated dictionary is:")  print(d1)  **Check if a Given Key Exists in a Dictionary or Not**  d={'A':1,'B':2,'C':3}  key=raw\_input("Enter key to check:")  if key in d.keys():  print("Key is present and value of the key is:")  print(d[key])  else:  print("Key isn't present!")  **Generate a Dictionary that Contains Numbers (between 1 and n) in the Form (x,x\*x)**  n=int(input("Enter a number:"))  d={x:x\*x for x in range(1,n+1)}  print(d)  **Sum All the Items in a Dictionary**  d={'A':100,'B':540,'C':239}  print("Total sum of values in the dictionary:")  print(sum(d.values()))  **Multiply All the Items in a Dictionary**  d={'A':10,'B':10,'C':239}  tot=1  for i in d:  tot=tot\*d[i]  print(tot)  **Remove the Given Key from a Dictionary**  d = {'a':1,'b':2,'c':3,'d':4}  print("Initial dictionary")  print(d)  key=raw\_input("Enter the key to delete(a-d):")  if key in d:  del d[key]  else:  print("Key not found!")  exit(0)  print("Updated dictionary")  print(d)  **Form a Dictionary from an Object of a Class**  class A(object):  def \_\_init\_\_(self):  self.A=1  self.B=2  obj=A()  print(obj.\_\_dict\_\_)  **Count the Frequency of Words Appearing in a String Using a Dictionary**  test\_string=raw\_input("Enter string:")  l=[]  l=test\_string.split()  wordfreq=[l.count(p) for p in l]  print(dict(zip(l,wordfreq)))  **Map Two Lists into a Dictionary**  keys=[]  values=[]  n=int(input("Enter number of elements for dictionary:"))  print("For keys:")  for x in range(0,n):  element=int(input("Enter element" + str(x+1) + ":"))  keys.append(element)  print("For values:")  for x in range(0,n):  element=int(input("Enter element" + str(x+1) + ":"))  values.append(element)  d=dict(zip(keys,values))  print("The dictionary is:")  print(d)  **Dictionary with Key as First Character and Value as Words Starting with that Character**  test\_string=raw\_input("Enter string:")  l=test\_string.split()  d={}  for word in l:  if(word[0] not in d.keys()):  d[word[0]]=[]  d[word[0]].append(word)  else:  if(word not in d[word[0]]):  d[word[0]].append(word)  for k,v in d.items():  print(k,":",v) |
| Sets | 83.  84.  85.  86.  87.  88.  89.  90. | **Count the Number of Vowels Present in a String using Sets**  s=input("Enter string:")  count = 0  vowels = set("aeiou")  for letter in s:  if letter in vowels:  count += 1  print("Count of the vowels is:")  print(count)  **Check Common Letters in Two Input Strings**  s1=input("Enter first string:")  s2=input("Enter second string:")  a=list(set(s1)&set(s2))  print("The common letters are:")  for i in a:  print(i)  **Display which Letters are in the First String but not in the Second**  s1=input("Enter first string:")  s2=input("Enter second string:")  a=list(set(s1)-set(s2))  print("The letters are:")  for i in a:  print(i)  **Display which Letters are Present in Both the Strings**  s1=input("Enter first string:")  s2=input("Enter second string:")  a=list(set(s1)|set(s2))  print("The letters are:")  for i in a:  print(i)  **Display which Letters are in the Two Strings but not in Both**  s1=input("Enter first string:")  s2=input("Enter second string:")  a=list(set(s1)^set(s2))  print("The letters are:")  for i in a:  print(i)  **Write a program which accepts a string as input to print "Yes" if the string is "yes" or "YES" or "Yes", otherwise print "No".**  s= input()  if s=="yes" or s=="YES" or s=="Yes":  print("Yes")  else:  print("No")  **With two given lists [1,3,6,78,35,55] and [12,24,35,24,88,120,155], write a program to make a list whose elements are intersection of the above given lists.**  set1=set([1,3,6,78,35,55])  set2=set([12,24,35,24,88,120,155])  set1 &= set2  li=list(set1)  print(li)  **With a given list [12,24,35,24,88,120,155,88,120,155], write a program to print this list after removing all duplicate values with original order reserved.**  def removeDuplicate( li ):  newli=[]  seen = set()  for item in li:  if item not in seen:  seen.add( item )  newli.append(item)  return newli  li=[12,24,35,24,88,120,155,88,120,155]  print(removeDuplicate(li)) |
| Tuples | 91.  92.  93.  94.  95.  96.  97. | **With a given tuple (1,2,3,4,5,6,7,8,9,10), write a program to print the first half values in one line and the last half values in one line.**  tp=(1,2,3,4,5,6,7,8,9,10)  tp1=tp[:len(tp)//2]  tp2=tp[len(tp)//2:]  print(tp1)  print(tp2)  **Write a program to generate and print another tuple whose values are even numbers in the given tuple (1,2,3,4,5,6,7,8,9,10).**  tp=(1,2,3,4,5,6,7,8,9,10)  li=list()  for i in tp:  if tp[i]%2==0:  li.append(tp[i])  tp2=tuple(li)  print(tp2)  **Reverse the following tuple**  **aTuple = (10, 20, 30, 40, 50)**  aTuple = (10, 20, 30, 40, 50)  aTuple = aTuple[::-1]  print(aTuple)  **Swap the following two tuples**  **tuple1 = (11, 22)**  **tuple2 = (99, 88)**  tuple1 = (11, 22)  tuple2 = (99, 88)  tuple1, tuple2 = tuple2, tuple1  print(tuple2)  print(tuple1)  **Copy element 44 and 55 from the following tuple into a new tuple**  **tuple1 = (11, 22, 33, 44, 55, 66)**  tuple1 = (11, 22, 33, 44, 55, 66)  tuple2 = tuple1[3:-1]  print(tuple2)  **Sort a tuple of tuples by 2nd item**  **tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))**  tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))  tuple1 = tuple(sorted(list(tuple1), key=lambda x: x[1]))  print(tuple1)  **Check if all items in the following tuple are the same**  **tuple1 = (45, 45, 45, 45)**  tuple1 = (45, 45, 45, 45)  print(all(tuple1)) |
| Functions | 98.  99.  100.  101.  102.  103.  104.  105.  106.  107.  108.  109.  110.  111.  112.  113.  114.  115.  116.  117.  118.  119.  120. | **Find the Fibonacci Series without Using Recursion**  a=int(input("Enter the first number of the series "))  b=int(input("Enter the second number of the series "))  n=int(input("Enter the number of terms needed "))  print(a,b,end=" ")  while(n-2):  c=a+b  a=b  b=c  print(c,end=" ")  n=n-1  **Remove the last element from the list**  def display(i):  listdata=[11,22,33,44,56]  listdata.pop(i)  print("updated listdata")  return listdata  i=int(input("enter i value"))  result=display(i)  print(result)  **find the factorial of a number without recursion**  n=int(input("Enter number:"))  fact=1  while(n>0):  fact=fact\*n  n=n-1  print("Factorial of the number is: ")  print(fact)  **Find the Max of three numbers**  def max\_of\_two( x, y ):  if x > y:  return x  return y  def max\_of\_three( x, y, z ):  return max\_of\_two( x, max\_of\_two( y, z ) )  print(max\_of\_three(3, 6, -5))  **Find the sum of all the numbers in a list**  def sum(numbers):  total = 0  for x in numbers:  total += x  return total  print(sum((8, 2, 3, 0, 7)))  **Check whether a number is in a given range**  def test\_range(n):  if n in range(3,9):  print( " %s is in the range"%str(n))  else :  print("The number is outside the given range.")  test\_range(5)  **Generate and print a list where the values are square of numbers between two numbers**  def printValue ():  l = list()  for i in range(1,21):  l.append(i\*\*2)  print(l)    printValue ()  **Define a function which can compute the sum of two numbers.**  def SumFunction(number1, number2):  return number1+number2  SumFunction(1,2)  **Define a function that can convert a integer into a string**  def printValue(n):  print(str(n))  printValue (3)  **Define a function that can convert a integer into a string and print it in console.**  def printValue(n):  print(str(n))  printValue (3)  **Define a function that can receive two integral numbers in string form and compute their sum and then print it in console.**  def printValue(s1,s2):  print(int(s1)+int(s2))  printValue ("3","4") #7  **Define a function that can accept two strings as input and concatenate them and then print it in console.**  def printValue(s1,s2):  print(s1+s2)  printValue ("3","4") #34  **Define a function that can accept two strings as input and print the string with maximum length in console. If two strings have the same length, then the function should print al l strings line by line.**  def printValue(s1,s2):  len1 = len(s1)  len2 = len(s2)  if len1>len2:  print(s1)  elif len2>len1:  print (s2)  else:  print(s1)  print(s2)  printValue ("one","three")  **Define a function that can accept an integer number as input and print the "It is an even number" if the number is even, otherwise print "It is an odd number".**  def checkValue(n):  if n%2 == 0:  print("It is an even number")  else:  print("It is an odd number")  checkValue(7)  **Define a function which can print a dictionary where the keys are numbers between 1 and 3 (both included) and the values are square of keys.**  def printDict():  d=dict()  d[1]=1  d[2]=2\*\*2  d[3]=3\*\*2  print(d)  printDict ()  **Define a function which can print a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys.**  def printDict():  d=dict()  for i in range(1,21):  d[i]=i\*\*2  print(d)  printDict ()  **Define a function which can generate a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys. The function should just print the values only.**  def printDict():  d=dict()  for i in range(1,21):  d[i]=i\*\*2  for (k,v) in d.items():  print(v)  printDict ()  **Define a function which can generate a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys. The function should just print the keys only.**  def printDict():  d=dict()  for i in range(1,21):  d[i]=i\*\*2  for k in d.keys():  print (k)  printDict ()  **Define a function which can generate and print a list where the values are square of numbers between 1 and 20 (both included).**  def printList():  li=list()  for i in range(1,21):  li.append(i\*\*2)  print(li)  printList ()  **Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the first 5 elements in the list.**  def printList():  li=list()  for i in range(1,21):  li.append(i\*\*2)  print(li[:5])  printList ()  **Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the last 5 elements in the list.**  def printList():  li=list()  for i in range(1,21):  li.append(i\*\*2)  print(li[-5:])  printList ()  **Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print all values except the first 5 elements in the list.**  def printList():  li=list()  for i in range(1,21):  li.append(i\*\*2)  print(li[5:])  printList ()  **Define a function which can generate and print a tuple where the value are square of numbers between 1 and 20 (both included).**  def printTuple():  li=list()  for i in range(1,21):  li.append(i\*\*2)  print(tuple(li))  printTuple () |
| Recursion | 121.  122.  123.  124.  125.  126.  127.  128.  129.  130.  131.  132.  133.  134.  135.  136.  137. | **Determine Whether a Given Number is Even or Odd Recursively**  def check(n):  if (n < 2):  return (n % 2 == 0)  return (check(n - 2))  n=int(input("Enter number:"))  if(check(n)==True):  print("Number is even!")  else:  print("Number is odd!")  **Determine How Many Times a Given Letter Occurs in a String Recursively**  def check(string,ch):  if not string:  return 0  elif string[0]==ch:  return 1+check(string[1:],ch)  else:  return check(string[1:],ch)  string=raw\_input("Enter string:")  ch=raw\_input("Enter character to check:")  print("Count is:")  print(check(string,ch))  **Find the Fibonacci Series Using Recursion**  def fibonacci(n):  if(n <= 1):  return n  else:  return(fibonacci(n-1) + fibonacci(n-2))  n = int(input("Enter number of terms:"))  print("Fibonacci sequence:")  for i in range(n):  print fibonacci(i)  **Find the Factorial of a Number Using Recursion**  def factorial(n):  if(n <= 1):  return 1  else:  return(n\*factorial(n-1))  n = int(input("Enter number:"))  print("Factorial:")  print(factorial(n))  **Find the Sum of Elements in a List Recursively**  def sum\_arr(arr,size):  if (size == 0):  return 0  else:  return arr[size-1] + sum\_arr(arr,size-1)  n=int(input("Enter the number of elements for list:"))  a=[]  for i in range(0,n):  element=int(input("Enter element:"))  a.append(element)  print("The list is:")  print(a)  print("Sum of items in list:")  b=sum\_arr(a,n)  print(b)  **Find the Binary Equivalent of a Number Recursively**  l=[]  def convert(b):  if(b==0):  return l  dig=b%2  l.append(dig)  convert(b//2)  a=int(input("Enter a number: "))  convert(a)  l.reverse()  print("Binary equivalent:")  for i in l:  print i  **Find the Sum of the Digits of the Number Recursively**  l=[]  def sum\_digits(b):  if(b==0):  return l  dig=b%10  l.append(dig)  sum\_digits(b//10)  n=int(input("Enter a number: "))  sum\_digits(n)  print(sum(l))  **Find the LCM of Two Numbers Using Recursion**  def lcm(a,b):  lcm.multiple=lcm.multiple+b  if((lcm.multiple % a == 0) and (lcm.multiple % b == 0)):  return lcm.multiple;  else:  lcm(a, b)  return lcm.multiple  lcm.multiple=0  a=int(input("Enter first number:"))  b=int(input("Enter second number:"))  if(a>b):  LCM=lcm(b,a)  else:  LCM=lcm(a,b)  print(LCM)  **Find the GCD of Two Numbers Using Recursion**  def gcd(a,b):  if(b==0):  return a  else:  return gcd(b,a%b)  a=int(input("Enter first number:"))  b=int(input("Enter second number:"))  GCD=gcd(a,b)  print("GCD is: ")  print(GCD)  **Find if a Number is Prime or Not Prime Using Recursion**  def check(n, div = None):  if div is None:  div = n - 1  while div >= 2:  if n % div == 0:  print("Number not prime")  return False  else:  return check(n, div-1)  else:  print("Number is prime")  return 'True'  n=int(input("Enter number: "))  check(n)  **Find the Product of two Numbers Using Recursion**  def product(a,b):  if(a<b):  return product(b,a)  elif(b!=0):  return(a+product(a,b-1))  else:  return 0  a=int(input("Enter first number: "))  b=int(input("Enter second number: "))  print("Product is: ",product(a,b))  **Find the Power of a Number Using Recursion**  def power(base,exp):  if(exp==1):  return(base)  if(exp!=1):  return(base\*power(base,exp-1))  base=int(input("Enter base: "))  exp=int(input("Enter exponential value: "))  print("Result:",power(base,exp))  **Check Whether a String is a Palindrome or not Using Recursion**  def is\_palindrome(s):  if len(s) < 1:  return True  else:  if s[0] == s[-1]:  return is\_palindrome(s[1:-1])  else:  return False  a=str(input("Enter string:"))  if(is\_palindrome(a)==True):  print("String is a palindrome!")  else:  print("String isn't a palindrome!")  **Reverse a String Using Recursion**  def reverse(string):  if len(string) == 0:  return string  else:  return reverse(string[1:]) + string[0]  a = str(input("Enter the string to be reversed: "))  print(reverse(a))  **Flatten a Nested List using Recursion**  def flatten(S):  if S == []:  return S  if isinstance(S[0], list):  return flatten(S[0]) + flatten(S[1:])  return S[:1] + flatten(S[1:])  s=[[1,2],[3,4]]  print("Flattened list is: ",flatten(s))  **Find the Total Sum of a Nested List Using Recursion**  def sum1(lst):  total = 0  for element in lst:  if (type(element) == type([])):  total = total + sum1(element)  else:  total = total + element  return total  print( "Sum is:",sum1([[1,2],[3,4]]))  **Find the Length of a List Using Recursion**  def length(lst):  if not lst:  return 0  return 1 + length(lst[1::2]) + length(lst[2::2])  a=[1,2,3]  print("Length of the string is: ")  print(a) |
| Searching and sorting | 138.  139.  140.  141.  142.  143.  143.  144.  145.  146. | **Binary search**  def binary\_search(item\_list,item):  first = 0  last = len(item\_list)-1  found = False  while( first<=last and not found):  mid = (first + last)//2  if item\_list[mid] == item :  found = True  else:  if item < item\_list[mid]:  last = mid - 1  else:  first = mid + 1  return found    print(binary\_search([1,2,3,5,8], 6))  print(binary\_search([1,2,3,5,8], 5))  **Sequential search**  def Sequential\_Search(dlist, item):  pos = 0  found = False  while pos < len(dlist) and not found:  if dlist[pos] == item:  found = True  else:  pos = pos + 1    return found, pos  print(Sequential\_Search([11,23,58,31,56,77,43,12,65,19],31))  **Search for an ordered list**  def Ordered\_binary\_Search(olist, item):    if len(olist) == 0:  return False  else:  midpoint = len(olist) // 2  if olist[midpoint] == item:  return True  else:  if item < olist[midpoint]:  return binarySearch(olist[:midpoint], item)  else:  return binarySearch(olist[midpoint+1:], item)  def binarySearch(alist, item):  first = 0  last = len(alist) - 1  found = False  while first <= last and not found:  midpoint = (first + last) // 2  if alist[midpoint] == item:  found = True  else:  if item < alist[midpoint]:  last = midpoint - 1  else:  first = midpoint + 1  return found  print(Ordered\_binary\_Search([0, 1, 3, 8, 14, 18, 19, 34, 52], 3))  print(Ordered\_binary\_Search([0, 1, 3, 8, 14, 18, 19, 34, 52], 17))  **Bubble sort**  def bubbleSort(nlist):  for passnum in range(len(nlist)-1,0,-1):  for i in range(passnum):  if nlist[i]>nlist[i+1]:  temp = nlist[i]  nlist[i] = nlist[i+1]  nlist[i+1] = temp  nlist = [14,46,43,27,57,41,45,21,70]  bubbleSort(nlist)  print(nlist)  **Selection sort**  def selectionSort(nlist):  for fillslot in range(len(nlist)-1,0,-1):  maxpos=0  for location in range(1,fillslot+1):  if nlist[location]>nlist[maxpos]:  maxpos = location  temp = nlist[fillslot]  nlist[fillslot] = nlist[maxpos]  nlist[maxpos] = temp  nlist = [14,46,43,27,57,41,45,21,70]  selectionSort(nlist)  print(nlist)  **Insertion sort**  def insertionSort(nlist):  for index in range(1,len(nlist)):  currentvalue = nlist[index]  position = index  while position>0 and nlist[position-1]>currentvalue:  nlist[position]=nlist[position-1]  position = position-1  nlist[position]=currentvalue  nlist = [14,46,43,27,57,41,45,21,70]  insertionSort(nlist)  print(nlist)  **Shell sort**  def shellSort(alist):  sublistcount = len(alist)//2  while sublistcount > 0:  for start\_position in range(sublistcount):  gap\_InsertionSort(alist, start\_position, sublistcount)  print("After increments of size",sublistcount, "The list is",nlist)  sublistcount = sublistcount // 2  def gap\_InsertionSort(nlist,start,gap):  for i in range(start+gap,len(nlist),gap):  current\_value = nlist[i]  position = i  while position>=gap and nlist[position-gap]>current\_value:  nlist[position]=nlist[position-gap]  position = position-gap  nlist[position]=current\_value  nlist = [14,46,43,27,57,41,45,21,70]  shellSort(nlist)  print(nlist)  def shellSort(alist):  sublistcount = len(alist)//2  while sublistcount > 0:  for start\_position in range(sublistcount):  gap\_InsertionSort(alist, start\_position, sublistcount)  print("After increments of size",sublistcount, "The list is",nlist)  sublistcount = sublistcount // 2  **Merge sort**  def mergeSort(nlist):  print("Splitting ",nlist)  if len(nlist)>1:  mid = len(nlist)//2  lefthalf = nlist[:mid]  righthalf = nlist[mid:]  mergeSort(lefthalf)  mergeSort(righthalf)  i=j=k=0  while i < len(lefthalf) and j < len(righthalf):  if lefthalf[i] < righthalf[j]:  nlist[k]=lefthalf[i]  i=i+1  else:  nlist[k]=righthalf[j]  j=j+1  k=k+1  while i < len(lefthalf):  nlist[k]=lefthalf[i]  i=i+1  k=k+1  while j < len(righthalf):  nlist[k]=righthalf[j]  j=j+1  k=k+1  print("Merging ",nlist)  nlist = [14,46,43,27,57,41,45,21,70]  mergeSort(nlist)  print(nlist)  **Quick sort**  def quickSort(data\_list):  quickSortHlp(data\_list,0,len(data\_list)-1)  def quickSortHlp(data\_list,first,last):  if first < last:  splitpoint = partition(data\_list,first,last)  quickSortHlp(data\_list,first,splitpoint-1)  quickSortHlp(data\_list,splitpoint+1,last)  def partition(data\_list,first,last):  pivotvalue = data\_list[first]  leftmark = first+1  rightmark = last  done = False  while not done:  while leftmark <= rightmark and data\_list[leftmark] <= pivotvalue:  leftmark = leftmark + 1  while data\_list[rightmark] >= pivotvalue and rightmark >= leftmark:  rightmark = rightmark -1  if rightmark < leftmark:  done = True  else:  temp = data\_list[leftmark]  data\_list[leftmark] = data\_list[rightmark]  data\_list[rightmark] = temp  temp = data\_list[first]  data\_list[first] = data\_list[rightmark]  data\_list[rightmark] = temp  return rightmark  data\_list = [54,26,93,17,77,31,44,55,20]  quickSort(data\_list)  print(data\_list)  **Counting sort algorithm**  def counting\_sort(array1, max\_val):  m = max\_val + 1  count = [0] \* m    for a in array1:  count[a] += 1  i = 0  for a in range(m):  for c in range(count[a]):  array1[i] = a  i += 1  return array1  print(counting\_sort( [1, 2, 7, 3, 2, 1, 4, 2, 3, 2, 1], 7 )) |
| Class and objects | 147.  148.  149.  150.  151.  152.  153.  154.  155.  156.  157.  158.  159.  160.  161.  162.  163.  164.  165. | **Define a class Person and its two child classes: Male and Female. All classes have a method "getGender" which can print "Male" for Male class and "Female" for Female class.**  class Person(object):  def getGender( self ):  return "Unknown"  class Male( Person ):  def getGender( self ):  return "Male"  class Female( Person ):  def getGender( self ):  return "Female"  aMale = Male()  aFemale= Female()  print(aMale.getGender())  print(aFemale.getGender())  **Define a class which has at least two methods:**  **getString: to get a string from console input**  **printString: to print the string in upper case.**  class InputOutString(object):  def \_\_init\_\_(self):  self.s = ""  def getString(self):  self.s = raw\_input()  def printString(self):  print self.s.upper()  strObj = InputOutString()  strObj.getString()  strObj.printString()  **Define a class with a generator which can iterate the numbers, which are divisible by 7, between a given range 0 and n.**  def putNumbers(n):  i = 0  while i<n:  j=i  i=i+1  if j%7==0:  yield j  for i in reverse(100):  print (i)  **Write a method which can calculate square value of number**  class sample:  def square(num):  return num \*\* 2  obj=sample()  print(obj.square(2))  print (obj.square(3))  **Define a class, which have a class parameter and have a same instance parameter.**  class Person:  name = "Person"  def \_\_init\_\_(self, name = None):  # self.name is the instance parameter  self.name = name  jeffrey = Person("Jeffrey")  print (“{} name is {}”.format (Person.name, Jeffrey.name))  nico = Person()  nico.name = "Nico"  print (“ {} name is {}”.format (Person.name, nico.name))  **Define a class named American which has a static method called printNationality.**  class American(object):  @staticmethod  def printNationality():  print("America")  anAmerican = American()  anAmerican.printNationality()  American.printNationality()  **Define a class named American and its subclass NewYorker.**  class American(object):  print("America")  class NewYorker(American):  print("NewYork")  anAmerican = American()  aNewYorker = NewYorker()  print(anAmerican)  print(aNewYorker)  **Define a class named Circle which can be constructed by a radius. The Circle class has a method which can compute the area.**  class Circle(object):  def \_\_init\_\_(self, r):  self.radius = r  def area(self):  return self.radius\*\*2\*3.14  aCircle = Circle(2)  print(aCircle.area())  **Define a class named Rectangle which can be constructed by a length and width. The Rectangle class has a method which can compute the area.**  class Rectangle(object):  def \_\_init\_\_(self, l, w):  self.length = l  self.width = w  def area(self):  return self.length\*self.width  aRectangle = Rectangle(2,10)  print(aRectangle.area())  **Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have a area function which can print the area of the shape where Shape's area is 0 by default.**  class Shape(object):  def \_\_init\_\_(self):  pass  def area(self):  return 0  class Square(Shape):  def \_\_init\_\_(self, l):  Shape.\_\_init\_\_(self)  self.length = l  def area(self):  return self.length\*self.length  aSquare= Square(3)  print(aSquare.area())  **Write a Python program to convert an integer to a roman numeral.**  class py\_solution:  def int\_to\_Roman(self, num):  val = [  1000, 900, 500, 400,  100, 90, 50, 40,  10, 9, 5, 4,  1  ]  syb = [  "M", "CM", "D", "CD",  "C", "XC", "L", "XL",  "X", "IX", "V", "IV",  "I"  ]  roman\_num = ''  i = 0  while num > 0:  for \_ in range(num // val[i]):  roman\_num += syb[i]  num -= val[i]  i += 1  return roman\_num  print(py\_solution().int\_to\_Roman(1))  print(py\_solution().int\_to\_Roman(4000))  **Write a Python program to convert a roman numeral to an integer.**  class py\_solution:  def roman\_to\_int(self, s):  rom\_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}  int\_val = 0  for i in range(len(s)):  if i > 0 and rom\_val[s[i]] > rom\_val[s[i - 1]]:  int\_val += rom\_val[s[i]] - 2 \* rom\_val[s[i - 1]]  else:  int\_val += rom\_val[s[i]]  return int\_val  print(py\_solution().roman\_to\_int('MMMCMLXXXVI'))  print(py\_solution().roman\_to\_int('MMMM'))  print(py\_solution().roman\_to\_int('C'))  **Write a Python program to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order,**  **for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid.**  class py\_solution:  def is\_valid\_parenthese(self, str1):  stack, pchar = [], {"(": ")", "{": "}", "[": "]"}  for parenthese in str1:  if parenthese in pchar:  stack.append(parenthese)  elif len(stack) == 0 or pchar[stack.pop()] != parenthese:  return False  return len(stack) == 0  print(py\_solution().is\_valid\_parenthese("(){}[]"))  print(py\_solution().is\_valid\_parenthese("()[{)}"))  print(py\_solution().is\_valid\_parenthese("()"))  **Write a Python program to get all possible unique subsets from a set of distinct integers.**  class py\_solution:  def sub\_sets(self, sset):  return self.subsetsRecur([], sorted(sset))    def subsetsRecur(self, current, sset):  if sset:  return self.subsetsRecur(current, sset[1:]) + self.subsetsRecur(current + [sset[0]], sset[1:])  return [current]  print(py\_solution().sub\_sets([4,5,6]))  **Write a Python program to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.**  class py\_solution:  def twoSum(self, nums, target):  lookup = {}  for i, num in enumerate(nums):  if target - num in lookup:  return (lookup[target - num], i )  lookup[num] = i  print("index1=%d, index2=%d" % py\_solution().twoSum((10,20,10,40,50,60,70),50))  **Write a Python program to find the three elements that sum to zero from a set (array) of n real numbers.**  class py\_solution:  def threeSum(self, nums):  nums, result, i = sorted(nums), [], 0  while i < len(nums) - 2:  j, k = i + 1, len(nums) - 1  while j < k:  if nums[i] + nums[j] + nums[k] < 0:  j += 1  elif nums[i] + nums[j] + nums[k] > 0:  k -= 1  else:  result.append([nums[i], nums[j], nums[k]])  j, k = j + 1, k - 1  while j < k and nums[j] == nums[j - 1]:  j += 1  while j < k and nums[k] == nums[k + 1]:  k -= 1  i += 1  while i < len(nums) - 2 and nums[i] == nums[i - 1]:  i += 1  return result  print(py\_solution().threeSum([-25, -10, -7, -3, 2, 4, 8, 10]))  **Write a Python program to implement pow(x, n).**  class py\_solution:  def pow(self, x, n):  if x==0 or x==1 or n==1:  return x  if x==-1:  if n%2 ==0:  return 1  else:  return -1  if n==0:  return 1  if n<0:  return 1/self.pow(x,-n)  val = self.pow(x,n//2)  if n%2 ==0:  return val\*val  return val\*val\*x  print(py\_solution().pow(2, -3))  print(py\_solution().pow(3, 5))  **Write a Python program to reverse a string word by word.**  class py\_solution:  def reverse\_words(self, s):  return ' '.join(reverse(s.split()))  print(py\_solution().reverse\_words('hello .py'))  print(py\_solution().pow(100, 0))  **Write a Python class which has two methods get\_String and print\_String. get\_String accept a string from the user and print\_String print the string in upper case.**  class IOString():  def \_\_init\_\_(self):  self.str1 = ""  def get\_String(self):  self.str1 = input()  def print\_String(self):  print(self.str1.upper())  str1 = IOString()  str1.get\_String()  str1.print\_String() |
| Logical programs | 166.  167.  168.  169.  170.  171.  172.  173.  174.  175.  176.  177.  178.  179.  180.  181.  182.  183.  184.  185.  186. | **Write a program that calculates and prints the value according to the given formula:**  **Q = Square root of [(2 \* C \* D)/H]**  **Following are the fixed values of C and H:**  **C is 50. H is 30.**  **D is the variable whose values should be input to your program in a comma-separated sequence.**  import math  c=50  h=30  value = []  items=[x for x in input().split(',')]  for d in items:  value.append(str(int(round(math.sqrt(2\*c\*float(d)/h)))))  print ','.join(value)  **Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.**  value = []  items=[x for x in raw\_input().split(',')]  for p in items:  intp = int(p, 2)  if not intp%5:  value.append(p)  print ','.join(value)  **Write a program, which will find all such numbers between 1000 and 3000 (both included) such that each digit of the number is an even number.**  values = []  for i in range(1000, 3001):  s = str(i)  if (int(s[0])%2==0) and (int(s[1])%2==0) and (int(s[2])%2==0) and (int(s[3])%2==0):  values.append(s)  print ",".join(values)  **Write a program that accepts a sentence and calculate the number of letters and digits.**  s = input()  d={"DIGITS":0, "LETTERS":0}  for c in s:  if c.isdigit():  d["DIGITS"]+=1  elif c.isalpha():  d["LETTERS"]+=1  else:  pass  print("LETTERS", d["LETTERS"])  print("DIGITS", d["DIGITS"])  **Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters.**  s = input()  d={"UPPER CASE":0, "LOWER CASE":0}  for c in s:  if c.isupper():  d["UPPER CASE"]+=1  elif c.islower():  d["LOWER CASE"]+=1  else:  pass  print "UPPER CASE", d["UPPER CASE"]  print "LOWER CASE", d["LOWER CASE"]  **Write a program that computes the value of a+aa+aaa+aaaa with a given digit as the value of a.**  **Suppose the following input is supplied to the program:**  **9**  **Then, the output should be:**  **11106**  a =input()  n1 = int( "%s" % a )  n2 = int( "%s%s" % (a,a) )  n3 = int( "%s%s%s" % (a,a,a) )  n4 = int( "%s%s%s%s" % (a,a,a,a) )  print(n1+n2+n3+n4)  **Use a list comprehension to square each odd number in a list. The list is input by a sequence of comma-separated numbers.**  values = input()  numbers = [x for x in values.split(",") if int(x)%2!=0]  print (",".join(numbers))  **Write a program that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following:**  **D 100**  **W 200**  **D means deposit while W means withdrawal.**  **Suppose the following input is supplied to the program:**  **D 300**  **D 300**  **W 200**  **D 100**  **Then, the output should be:**  **500**  netAmount = 0  while True:  s =input()  if not s:  break  values = s.split(" ")  operation = values[0]  amount = int(values[1])  if operation=="D":  netAmount+=amount  elif operation=="W":  netAmount-=amount  else:  pass  print(netAmount)  **Write a program to sort the (name, age, height) tuples by ascending order where name is string, age and height are numbers. The tuples are input by console. The sort criteria is:**  **1: Sort based on name;**  **2: Then sort based on age;**  **3: Then sort by score.**  **The priority is that name > age > score.**  **If the following tuples are given as input to the program:**  **Tom,19,80**  **John,20,90**  **Jony,17,91**  **Jony,17,93**  **Json,21,85**  **Then, the output of the program should be:**  **[('John', '20', '90'), ('Jony', '17', '91'), ('Jony', '17', '93'), ('Json', '21', '85'), ('Tom', '19', '80')]**  from operator import itemgetter, attrgetter  l = []  while True:  s = input()  if not s:  break  l.append(tuple(s.split(",")))  print(sorted(l, key=itemgetter(0,1,2)))  **A robot moves in a plane starting from the original point (0,0). The robot can move toward UP, DOWN, LEFT and RIGHT with a given steps. The trace of robot movement is shown as the following:**  **UP 5**  **DOWN 3**  **LEFT 3**  **RIGHT 2**  **¡­**  **The numbers after the direction are steps. Please write a program to compute the distance from current position after a sequence of movement and original point. If the distance is a float, then just print the nearest integer.**  **Example:**  **If the following tuples are given as input to the program:**  **UP 5**  **DOWN 3**  **LEFT 3**  **RIGHT 2**  **Then, the output of the program should be:**  **2**  import math  pos = [0,0]  while True:  s = input()  if not s:  break  movement = s.split(" ")  direction = movement[0]  steps = int(movement[1])  if direction=="UP":  pos[0]+=steps  elif direction=="DOWN":  pos[0]-=steps  elif direction=="LEFT":  pos[1]-=steps  elif direction=="RIGHT":  pos[1]+=steps  else:  pass  print (int(round(math.sqrt(pos[1]\*\*2+pos[0]\*\*2))))  **Write a program to compute the frequency of the words from the input. The output should output after sorting the key alphanumerically.**  freq = {} # frequency of words in text  line = raw\_input()  for word in line.split():  freq[word] = freq.get(word,0)+1  words = freq.keys()  words.sort()  for w in words:  print ("%s:%d" % (w,freq[w]))  **Write a program to compute:**  **f(n)=f(n-1)+100 when n>0**  **and f(0)=1**  **with a given n input by console (n>0).**  def f(n):  if n==0:  return 0  else:  return f(n-1)+100  n=int(input())  print(f(n))  **The Fibonacci Sequence is computed based on the following formula:**  **f(n)=0 if n=0**  **f(n)=1 if n=1**  **f(n)=f(n-1)+f(n-2) if n>1**  **Write a program to compute the value of f(n) with a given n input by console.**  def f(n):  if n == 0: return 0  elif n == 1: return 1  else: return f(n-1)+f(n-2)  n=int(input())  print(f(n))  **The Fibonacci Sequence is computed based on the following formula:**  **f(n)=0 if n=0**  **f(n)=1 if n=1**  **f(n)=f(n-1)+f(n-2) if n>1**  **Write a program using list comprehension to print the Fibonacci Sequence in comma separated form with a given n input by console.**  def f(n):  if n == 0: return 0  elif n == 1: return 1  else: return f(n-1)+f(n-2)  n=int(input())  values = [str(f(x)) for x in range(0, n+1)]  print(",".join(values))  **Write a program using generator to print the even numbers between 0 and n in comma separated form while n is input by console.**  def EvenGenerator(n):  i=0  while i<=n:  if i%2==0:  yield i  i+=1  n=int(input())  values = []  for i in EvenGenerator(n):  values.append(str(i))  print(",".join(values))  **Write a program using generator to print the numbers which can be divisible by 5 and 7 between 0 and n in comma separated form while n is input by console.**  def NumGenerator(n):  for i in range(n+1):  if i%5==0 and i%7==0:  yield i  n=int(input())  values = []  for i in NumGenerator(n):  values.append(str(i))  print (",".join(values))  **Write assert statements to verify that every number in the list [2,4,6,8] is even.**  li = [2,4,6,8]  for i in li:  assert i%2==0  **write a program which prints all permutations of [1,2,3]**  import itertools  print(list(itertools.permutations([1,2,3])))  **Write a program to solve a classic ancient Chinese puzzle:**  **We count 35 heads and 94 legs among the chickens and rabbits in a farm. How many rabbits and how many chickens do we have?**  def solve(numheads,numlegs):  ns='No solutions!'  for i in range(numheads+1):  j=numheads-i  if 2\*i+4\*j==numlegs:  return i,j  return ns,ns  numheads=35  numlegs=94  solutions=solve(numheads,numlegs)  print(solutions)  **Write a program which accepts basic mathematic expression from console and print the evaluation result**.  expression = input()  print(eval(expression))  **Write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.**  import math  def bin\_search(li, element):  bottom = 0  top = len(li)-1  index = -1  while top>=bottom and index==-1:  mid = int(math.floor((top+bottom)/2.0))  if li[mid]==element:  index = mid  elif li[mid]>element:  top = mid-1  else:  bottom = mid+1  return index  li=[2,5,7,9,11,17,222]  print(bin\_search(li,11))  print(bin\_search(li,12)) |
| Regular Expression | 187.  188.  189.  190.  191.  192.  193.  194.  195.  196.  197.  198.  199.  200.  201.  202.  203.  204.  205.  206.  207.  208.  209.  210.  211.  212.  213.  214.  215.  216.  217.  218.  219.  220.  221.  222.  223.  224.  225.  226.  227.  228.  229.  230.  231. | **A website requires the users to input username and password to register. Write a program to check the validity of password input by users.**  **Following are the criteria for checking the password:**  **1. At least 1 letter between [a-z]**  **2. At least 1 number between [0-9]**  **1. At least 1 letter between [A-Z]**  **3. At least 1 character from [$#@]**  **4. Minimum length of transaction password: 6**  **5. Maximum length of transaction password: 12**  **Example**  **If the following passwords are given as input to the program:**  **ABd1234@1,a F1#,2w3E\*,2We3345**  **Then, the output of the program should be:**  **ABd1234@1**  import re  value = []  items=[x for x in raw\_input().split(',')]  for p in items:  if len(p)<6 or len(p)>12:  continue  else:  pass  if not re.search("[a-z]",p):  continue  elif not re.search("[0-9]",p):  continue  elif not re.search("[A-Z]",p):  continue  elif not re.search("[$#@]",p):  continue  elif re.search("\s",p):  continue  else:  pass  value.append(p)  print(",".join(value))  **Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the user name of a given email address. Both user names and company names are composed of letters only.**  import re  emailAddress = raw\_input()  pat2 = "(\w+)@((\w+\.)+(com))"  r2 = re.match(pat2,emailAddress)  print(r2.group(1))  **Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the company name of a given email address. Both user names and company names are composed of letters only.**  import re  emailAddress = raw\_input()  pat2 = "(\w+)@(\w+)\.(com)"  r2 = re.match(pat2,emailAddress)  print(r2.group(2))  **Write a program which accepts a sequence of words separated by whitespace as input to print the words composed of digits only.**  import re  s = raw\_input()  print re.findall("\d+",s)  **Write a Python program that matches a string that has an a followed by zero or more b's.**  import re  def text\_match(text):  patterns = 'ab\*?'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("ac"))  print(text\_match("abc"))  print(text\_match("abbc"))  **Write a Python program that matches a string that has an a followed by one or more b's.**  import re  def text\_match(text):  patterns = 'ab+?'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("ab"))  print(text\_match("abc"))  **Write a Python program that matches a string that has an a followed by three 'b'.**  import re  def text\_match(text):  patterns = 'ab{3}?'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("abbb"))  print(text\_match("aabbbbbc"))  **Write a Python program to find sequences of lowercase letters joined with a underscore.**  import re  def text\_match(text):  patterns = '^[a-z]+\_[a-z]+$'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("aab\_cbbbc"))  print(text\_match("aab\_Abbbc"))  print(text\_match("Aaab\_abbbc"))  **Write a Python program that matches a word at the beginning of a string.**  import re  def text\_match(text):  patterns = '^\w+'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("The quick brown fox jumps over the lazy dog."))  print(text\_match(" The quick brown fox jumps over the lazy dog."))  **Write a Python program that matches a word at the end of a string, with optional punctuation.**  import re  def text\_match(text):  patterns = '\w+\S\*$'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("The quick brown fox jumps over the lazy dog."))  print(text\_match("The quick brown fox jumps over the lazy dog."))  **Write a Python program that matches a word containing 'z', not at the start or end of the word.**  import re  def text\_match(text):  patterns = '\Bz\B'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("The quick brown fox jumps over the lazy dog."))  print(text\_match("Python Exercises."))  print(text\_match("The quick brown fox jumps over the lazy dog "))  **Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.**  import re  def text\_match(text):  patterns = '^[a-zA-Z0-9\_]\*$'  if re.search(patterns, text):  return 'Found a match!'  else:  return('Not matched!')  print(text\_match("The quick brown fox jumps over the lazy dog."))  print(text\_match("Python\_Exercises\_1"))  **Write a Python program to remove leading zeros from an IP address.**  import re  ip = "216.08.094.196"  string = re.sub('\.[0]\*', '.', ip)  print(string)  **Write a Python program to check for a number at the end of a string.**  import re  def end\_num(string):  text = re.compile(r".\*[0-9]$")  if text.match(string):  return True  else:  return False  print(end\_num('abcdef'))  print(end\_num('abcdef6'))  **Write Python program to search the numbers (0-9) of length between 1 to 3 in a given string.**  import re  results = re.finditer(r"([0-9]{1,3})", "Exercises number 1, 12, 13, and 345 are important")  print("Number of length 1 to 3")  for n in results:  print(n.group(0))  **Write a Python program to search some literals strings in a string.**  import re  patterns = [ 'fox', 'dog', 'horse' ]  text = 'The quick brown fox jumps over the lazy dog.'  for pattern in patterns:  print('Searching for "%s" in "%s" ->' % (pattern, text),)  if re.search(pattern, text):  print('Matched!')  else:  print('Not Matched!')  **Write a Python program to search a literals string in a string and also find the location within the original string where the pattern occurs.**  import re  pattern = 'fox'  text = 'The quick brown fox jumps over the lazy dog.'  match = re.search(pattern, text)  s = match.start()  e = match.end()  print('Found "%s" in "%s" from %d to %d ' % \  (match.re.pattern, match.string, s, e))  **Write a Python program to find the substrings within a string.**  import re  text = 'Python exercises, PHP exercises, C# exercises'  pattern = 'exercises'  for match in re.findall(pattern, text):  print('Found "%s"' % match)  **Write a Python program to find the occurrence and position of the substrings within a string.**  import re  text = 'Python exercises, PHP exercises, C# exercises'  pattern = 'exercises'  for match in re.finditer(pattern, text):  s = match.start()  e = match.end()  print('Found "%s" at %d:%d' % (text[s:e], s, e))  **Write a Python program to replace whitespaces with an underscore and vice versa.**  import re  text = 'Python Exercises'  text =text.replace (" ", "\_")  print(text)  text =text.replace ("\_", " ")  print(text)  **Write a Python program to extract year, month and date from an url.**  import re  def extract\_date(url):  return re.findall(r'/(\d{4})/(\d{1,2})/(\d{1,2})/', url)  url1= "https://www.washingtonpost.com/news/football-insider/wp/2016/09/02/odell-beckhams-fame-rests-on-one-stupid-little-ball-josh-norman-tells-author/"  print(extract\_date(url1))  **Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy format.**  import re  def change\_date\_format(dt):  return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)  dt1 = "2026-01-02"  print("Original date in YYY-MM-DD Format: ",dt1)  print("New date in DD-MM-YYYY Format: ",change\_date\_format(dt1))  **Write a Python program to match if two words from a list of words starting with letter 'P'.**  import re  # Sample strings.  words = ["Python PHP", "Java JavaScript", "c c++"]  for w in words:  m = re.match("(P\w+)\W(P\w+)", w)  # Check for success  if m:  print(m.groups())  **Write a Python program to separate and print the numbers of a given string.**  import re  # Sample string.  text = "Ten 10, Twenty 20, Thirty 30"  result = re.split("\D+", text)  # Print results.  for element in result:  print(element)  **Write a Python program to find all words starting with 'a' or 'e' in a given string.**  import re  # Input.  text = "The following example creates an ArrayList with a capacity of 50 elements. Four elements are then added to the ArrayList and the ArrayList is trimmed accordingly."  #find all the words starting with 'a' or 'e'  list = re.findall("[ae]\w+", text)  print(list)  **Write a Python program to separate and print the numbers and their position of a given string.**  import re  text = "The following example creates an ArrayList with a capacity of 50 elements. Four elements are then added to the ArrayList and the ArrayList is trimmed accordingly."  for m in re.finditer("\d+", text):  print(m.group(0))  print("Index position:", m.start())  **Write a Python program to abbreviate 'Road' as 'Rd.' in a given string.**  import re  street = '21 Ramkrishna Road'  print(re.sub('Road$', 'Rd.', street))  **Write a Python program to replace all occurrences of space, comma, or dot with a colon.**  import re  text = 'Python Exercises, PHP exercises.'  print(re.sub("[ ,.]", ":", text))  **Write a Python program to find all five characters long word in a string.**  import re  text = 'The quick brown fox jumps over the lazy dog.'  print(re.findall(r"\b\w{5}\b", text))  **Write a Python program to find all words which are at least 4 characters long in a string.**  import re  text = 'The quick brown fox jumps over the lazy dog.'  print(re.findall(r"\b\w{4,}\b", text))  **Write a python program to convert camel case string to snake case string.**  def camel\_to\_snake(text):  import re  str1 = re.sub('(.)([A-Z][a-z]+)', r'\1\_\2', text)  return re.sub('([a-z0-9])([A-Z])', r'\1\_\2', str1).lower()  print(camel\_to\_snake('PythonExercises'))  **Write a Python program to extract values between quotation marks of a string.**  import re  text1 = '"Python", "PHP", "Java"'  print(re.findall(r'"(.\*?)"', text1))  **Write a Python program to remove multiple spaces in a string.**  import re  text1 = 'Python Exercises'  print("Original string:",text1)  print("Without extra spaces:",re.sub(' +',' ',text1))  **Write a Python program to remove all whitespaces from a string.**  import re  text1 = ' Python Exercises '  print("Original string:",text1)  print("Without extra spaces:",re.sub(r'\s+', '',text1))  **Write a Python program to find urls in a string.**  import re  text = '<p>Contents :</p><a href="https://w3resource.com">Python Examples</a><a href="http://github.com">Even More Examples</a>'  urls = re.findall('http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', text)  print("Original string: ",text)  print("Urls: ",urls)  **Write a Python program to do a case-insensitive string replacement.**  import re  text = "PHP Exercises"  print("Original Text: ",text)  redata = re.compile(re.escape('php'), re.IGNORECASE)  new\_text = redata.sub('php', 'PHP Exercises')  print("Using 'php' replace PHP")  print("New Text: ",new\_text)  **Write a Python program to remove the ANSI escape sequences from a string.**  import re  text = "\t\u001b[0;35mgoogle.com\u001b[0m \u001b[0;36m216.58.218.206\u001b[0m"  print("Original Text: ",text)  reaesc = re.compile(r'\x1b[^m]\*m')  new\_text = reaesc.sub('', text)  print("New Text: ",new\_text)  **Write a Python program to find all adverbs and their positions in a given sentence.**  import re  text = "Clearly, he has no excuse for such behavior."  for m in re.finditer(r"\w+ly", text):  print('%d-%d: %s' % (m.start(), m.end(), m.group(0)))  **Write a Python program to split a string with multiple delimiters.**  import re  text = 'The quick brown\nfox jumps\*over the lazy dog.'  print(re.split('; |, |\\*|\n',text))  **Write a Python program to check a decimal with a precision of 2.**  def is\_decimal(num):  import re  dnumre = re.compile(r"""^[0-9]+(\.[0-9]{1,2})?$""")  result = dnumre.search(num)  return bool(result)  print(is\_decimal('123.11'))  print(is\_decimal('123.1'))  print(is\_decimal('123'))  **Write a Python program to remove words from a string of length between 1 and a given number.**  import re  text = "The quick brown fox jumps over the lazy dog."  # remove words between 1 and 3  shortword = re.compile(r'\W\*\b\w{1,3}\b')  print(shortword.sub('', text))  **Write a Python program to remove the parenthesis area in a string.**  import re  items = ["example (.com)", "w3resource", "github (.com)", "stackoverflow (.com)"]  for item in items:  print(re.sub(r" ?\([^)]+\)", "", item))    **Write a Python program to insert spaces between words starting with capital letters.**  import re  def capital\_words\_spaces(str1):  return re.sub(r"(\w)([A-Z])", r"\1 \2", str1)  print(capital\_words\_spaces("Python"))  print(capital\_words\_spaces("PythonExercises"))  print(capital\_words\_spaces("PythonExercisesPracticeSolution"))  **Write a Python program that reads a given expression and evaluates it.**  **conditions:**  **The expression consists of numerical values, operators and parentheses, and the ends with '='.**  **The operators includes +, -, \*, / where, represents, addition, subtraction, multiplication and division.**  **When two operators have the same precedence, they are applied to left to right.**  **You may assume that there is no division by zero.**  **All calculation is performed as integers, and after the decimal point should be truncated Length of the expression will not exceed 100.**  **-1 × 10 9 ≤ intermediate results of computation ≤ 10 9**  #https://bit.ly/2lxQysi  import re  print("Input number of data sets:")  class c(int):  def \_\_add\_\_(self,n):  return c(int(self)+int(n))  def \_\_sub\_\_(self,n):  return c(int(self)-int(n))  def \_\_mul\_\_(self,n):  return c(int(self)\*int(n))  def \_\_truediv\_\_(self,n):  return c(int(int(self)/int(n)))    for \_ in range(int(input())):  print("Input an expression:")  **Write a Python program to remove lowercase substrings from a given string.**  import re  str1 = 'KDeoALOklOOHserfLoAJSIskdsf'  print("Original string:")  print(str1)  print("After removing lowercase letters, above string becomes:")  remove\_lower = lambda text: re.sub('[a-z]', '', text)  result = remove\_lower(str1)  print(result)  print(eval(re.sub(r'(\d+)',r'c(\1)',input()[:-1]))) |
| Filter,Map and Lambda | 232.  233.  234.  235.  236.  237.  238.  238.  239.  240  241.  242.  243.  244.  245.  246.  247.  248.  249.  250. | **Write a program which can filter even numbers in a list by using filter function. The list is: [1,2,3,4,5,6,7,8,9,10].**  li = [1,2,3,4,5,6,7,8,9,10]  evenNumbers = filter(lambda x: x%2==0, li)  print(evenNumbers)  **Write a program which can map() to make a list whose elements are square of elements in [1,2,3,4,5,6,7,8,9,10].**  li = [1,2,3,4,5,6,7,8,9,10]  squaredNumbers = map(lambda x: x\*\*2, li)  print(squaredNumbers)  **Write a program which can map() and filter() to make a list whose elements are square of even number in [1,2,3,4,5,6,7,8,9,10].**  li = [1,2,3,4,5,6,7,8,9,10]  evenNumbers = map(lambda x: x\*\*2, filter(lambda x: x%2==0, li))  print(evenNumbers)  **Write a program which can filter() to make a list whose elements are even number between 1 and 20 (both included).**  evenNumbers = filter(lambda x: x%2==0, range(1,21))  print(evenNumbers)  **Write a program which can map() to make a list whose elements are square of numbers between 1 and 20 (both included).**  squaredNumbers = map(lambda x: x\*\*2, range(1,21))  print(squaredNumbers)  **Write a Python program to create a lambda function that adds 15 to a given number passed in as an argument, also create a lambda function that multiplies argument x with argument y and print the result.**  r = lambda a : a + 15  print(r(10))  r = lambda x, y : x \* y  print(r(12, 4))  **Write a Python program to create a function that takes one argument, and that argument will be multiplied with an unknown given number.**  def func\_compute(n):  return lambda x : x \* n  result = func\_compute(2)  print("Double the number of 15 =", result(15))  result = func\_compute(3)  print("Triple the number of 15 =", result(15))  **Write a Python program to sort a list of tuples using Lambda.**  subject\_marks = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]  print("Original list of tuples:")  print(subject\_marks)  subject\_marks.sort(key = lambda x: x[1])  print("\nSorting the List of Tuples:")  print(subject\_marks)  **Write a Python program to sort a list of dictionaries using Lambda.**  models = [{'make':'Nokia', 'model':216, 'color':'Black'}, {'make':'Mi Max', 'model':'2', 'color':'Gold'}, {'make':'Samsung', 'model': 7, 'color':'Blue'}]  print("Original list of dictionaries :")  print(models)  sorted\_models = sorted(models, key = lambda x: x['color'])  print("\nSorting the List of dictionaries :")  print(sorted\_models)  **Write a Python program to filter a list of integers using Lambda.**  nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  print("Original list of integers:")  print(nums)  print("\nEven numbers from the said list:")  even\_nums = list(filter(lambda x: x%2 == 0, nums))  print(even\_nums)  print("\nOdd numbers from the said list:")  odd\_nums = list(filter(lambda x: x%2 != 0, nums))  print(odd\_nums)  **Write a Python program to square and cube every number in a given list of integers using Lambda.**  nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  print("Original list of integers:")  print(nums)  print("\nSquare every number of the said list:")  square\_nums = list(map(lambda x: x \*\* 2, nums))  print(square\_nums)  print("\nCube every number of the said list:")  cube\_nums = list(map(lambda x: x \*\* 3, nums))  print(cube\_nums)  **Write a Python program to find whether a given string starts with a given character using Lambda.**  starts\_with = lambda x: True if x.startswith('P') else False  print(starts\_with('Python'))  starts\_with = lambda x: True if x.startswith('P') else False  print(starts\_with('Java'))  **Write a Python program to extract year, month, date and time using Lambda.**  import datetime  now = datetime.datetime.now()  print(now)  year = lambda x: x.year  month = lambda x: x.month  day = lambda x: x.day  t = lambda x: x.time()  print(year(now))  print(month(now))  print(day(now))  print(t(now))  **Write a Python program to check whether a given string is number or not using Lambda.**  is\_num = lambda q: q.replace('.','',1).isdigit()  print(is\_num('26587'))  print(is\_num('4.2365'))  print(is\_num('-12547'))  print("\nPrint checking numbers:")  is\_num1 = lambda r: is\_num(r[1:]) if r[0]=='-' else is\_num(r)  print(is\_num1('-16.4'))  print(is\_num1('-24587.11'))  **Write a Python program to create Fibonacci series upto n using Lambda.**  from functools import reduce    fib\_series = lambda n: reduce(lambda x, \_: x+[x[-1]+x[2]], range(n-2), [0, 1])    print("Fibonacci series upto 2:")  print(fib\_series(2))  print("\nFibonacci series upto 5:")  print(fib\_series(5))  **Write a Python program to find intersection of two given arrays using Lambda.**  array\_nums1 = [1, 2, 3, 5, 7, 8, 9, 10]  array\_nums2 = [1, 2, 4, 8, 9]  print("Original arrays:")  print(array\_nums1)  print(array\_nums2)  result = list(filter(lambda x: x in array\_nums1, array\_nums2))  print ("\nIntersection of the said arrays: ",result)  **Write a Python program to rearrange positive and negative numbers in a given array using Lambda.**  array\_nums = [-1, 2, -3, 5, 7, 8, 9, -10]  print("Original arrays:")  print(array\_nums)  result = [x for x in array\_nums if x < 0] + [x for x in array\_nums if x >= 0]  print("\nRearrange positive and negative numbers of the said array:")  print(result)  **Write a Python program to count the even, odd numbers in a given array of integers using Lambda.**  array\_nums = [1, 2, 3, 5, 7, 8, 9, 10]  print("Original arrays:")  print(array\_nums)  odd\_ctr = len(list(filter(lambda x: (x%2 != 0) , array\_nums)))  even\_ctr = len(list(filter(lambda x: (x%2 == 0) , array\_nums)))  print("\nNumber of even numbers in the above array: ", even\_ctr)  print("\nNumber of odd numbers in the above array: ", odd\_ctr)  **Write a Python program to filter a given list whether the values in the list are having length of 6 using Lambda.**  weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']  days = filter(lambda day: day if len(day)==6 else '', weekdays)  for d in days:  print(d)  **Write a Python program to add two given lists using map and lambda.**  nums1 = [1, 2, 3]  nums2 = [4, 5, 6]  print("Original list:")  print(nums1)  print(nums2)  result = map(lambda x, y: x + y, nums1, nums2)  print("\nResult: after adding two list")  print(list(result)) |
| Exception Handling | 251.  252. | **Raise a RuntimeError exception.**  def sample(n):  for i in range(n):  if i==5:  raise RuntimeError('something wrong')  else:  print(i)  print(sample(10))  **Write a function to compute 5/0 and use try/except to catch the exceptions.**  def throws():  return 5/0  try:  throws()  except ZeroDivisionError:  print("division by zero!")  except Exception:  print('Caught an exception')  finally:  print('In finally block for cleanup') |
| Random | 253.  254.  255.  256.  257.  258.    259.  260.  261.  262.  263.  264.  265.  266. | **Generate a random float where the value is between 10 and 100 using Python math module.**  import random  print(random.random()\*100)  **To Generate Random Numbers from 1 to 20 and Append Them to the List**  import random  a=[]  n=int(input("Enter number of elements:"))  for j in range(n):  a.append(random.randint(1,20))  print('Randomised list is: ',a)  **Generate a random float where the value is between 5 and 95 using Python math module.**  import random  print(random.random()\*100-5)  **Write a program to output a random even number between 0 and 10 inclusive using random module and list comprehension.**  import random  print(random.choice([i for i in range(11) if i%2==0]))  **Write a program to output a random number, which is divisible by 5 and 7, between 0 and 10 inclusive using random module and list comprehension.**  import random  print(random.choice([i for i in range(201) if i%5==0 and i%7==0]))  **Write a program to generate a list with 5 random numbers between 100 and 200 inclusive.**  import random  print(random.sample(range(100), 5))  **Write a program to randomly generate a list with 5 even numbers between 100 and 200 inclusive.**  import random  print(random.sample([i for i in range(100,201) if i%2==0], 5))  **Write a program to randomly generate a list with 5 numbers, which are divisible by 5 and 7 , between 1 and 1000 inclusive.**  import random  print(random.sample([i for i in range(1,1001) if i%5==0 and i%7==0], 5))  **Write a program to randomly print a integer number between 7 and 15 inclusive.**  import random  print(random.randrange(7,16))  **Write a program to compress and decompress the string "hello world!hello world!hello world!hello world!".**  import zlib  s = 'hello world!hello world!hello world!hello world!'  t = zlib.compress(s)  print(t)  print(zlib.decompress(t))  **Write a program to print the running time of execution of "1+1" for 100 times.**  from timeit import Timer  t = Timer("for i in range(100):1+1")  print(t.timeit())  **Write a program to shuffle and print the list [3,6,7,8].**  from random import shuffle  li = [3,6,7,8]  shuffle(li)  print(li)  **Write a program to shuffle and print the list [3,6,7,8].**  from random import shuffle  li = [3,6,7,8]  shuffle(li)  print(li)  **Write a Python program to flip a coin 10000 times and count heads and tails.**  import random  import itertools  results = {  'heads': 0,  'tails': 0,  }  sides = list(results.keys())  for i in range(10000):  results[random.choice(sides)] += 1  print('Heads:', results['heads'])  print('Tails:', results['tails']) |
| JSON | 267.  268.  269.  270.  271.  272.  273.  274.  275. | **Write a Python program to convert JSON data to Python object.**  import json  json\_obj = '{ "Name":"David", "Class":"I", "Age":6 }'  python\_obj = json.loads(json\_obj)  print("\nJSON data:")  print(python\_obj)  print("\nName: ",python\_obj["Name"])  print("Class: ",python\_obj["Class"])  print("Age: ",python\_obj["Age"])  **Write a Python program to convert Python object to JSON data.**  import json  python\_obj = {  "name": "David",  "class":"I",  "age": 6  }  print(type(python\_obj))  j\_data = json.dumps(python\_obj)  print(j\_data)  **Write a Python program to convert Python objects into JSON strings. Print all the values.**  import json  python\_dict = {"name": "David", "age": 6, "class":"I"}  python\_list = ["Red", "Green", "Black"]  python\_str = "Python Json"  python\_int = (1234)  python\_float = (21.34)  python\_T = (True)  python\_F = (False)  python\_N = (None)  json\_dict = json.dumps(python\_dict)  json\_list = json.dumps(python\_list)  json\_str = json.dumps(python\_str)  json\_num1 = json.dumps(python\_int)  json\_num2 = json.dumps(python\_float)  json\_t = json.dumps(python\_T)  json\_f = json.dumps(python\_F)  json\_n = json.dumps(python\_N)  print("json dict : ", json\_dict)  print("jason list : ", json\_list)  print("json string : ", json\_str)  print("json number1 : ", json\_num1)  print("json number2 : ", json\_num2)  print("json true : ", json\_t)  print("json false : ", json\_f)  print("json null ; ", json\_n)  **Write a Python program to convert Python dictionary object (sort by key) to JSON data. Print the object members with indent level 4.**  import json  j\_str = {'4': 5, '6': 7, '1': 3, '2': 4}  print("Original String:")  print(j\_str)  print("\nJSON data:")  print(json.dumps(j\_str, sort\_keys=True, indent=4))  **Write a Python program to convert JSON encoded data into Python objects.**  import json  jobj\_dict = '{"name": "David", "age": 6, "class": "I"}'  jobj\_list = '["Red", "Green", "Black"]'  jobj\_string = '"Python Json"'  jobj\_int = '1234'  jobj\_float = '21.34'  python\_dict = json.loads(jobj\_dict)  python\_list = json.loads(jobj\_list)  python\_str = json.loads(jobj\_string)  python\_int = json.loads(jobj\_int)  python\_float = json.loads(jobj\_float)  print("Python dictionary: ", python\_dict)  print("Python list: ", python\_list)  print("Python string: ", python\_str)  print("Python integer: ", python\_int)  print("Python float: ", python\_float)  **Write a Python program to create a new JSON file from an existing JSON file.**  import json  with open('states.json') as f:  state\_data= json.load(f)  for state in state\_data['states']:  del state['area\_codes']  with open('new\_states.json', 'w') as f:  json.dump(state\_data, f, indent=2)  **Write a Python program to check whether an instance is complex or not.**  import json  def encode\_complex(object):  if isinstance(object, complex):  return [object.real, object.imag]  raise TypeError(repr(object) + " is not JSON serialized")  complex\_obj = json.dumps(2 + 3j, default=encode\_complex)  print(complex\_obj)  **Write a Python program to check whether a JSON string contains complex object or not.**  import json  def is\_complex\_num(objct):  if '\_\_complex\_\_' in objct:  return complex(objct['real'], objct['img'])  return objct  complex\_object =json.loads('{"\_\_complex\_\_": true, "real": 4, "img": 5}', object\_hook = is\_complex\_num)  simple\_object =json.loads('{"real": 4, "img": 3}', object\_hook = is\_complex\_num)  print("Complex\_object: ",complex\_object)  print("Without complex object: ",simple\_object)  **Write a Python program to access only unique key value of a Python object.**  iimport json  python\_obj = '{"a": 1, "a": 2, "a": 3, "a": 4, "b": 1, "b": 2}'  print("Original Python object:")  print(python\_obj)  json\_obj = json.loads(python\_obj)  print("\nUnique Key in a JSON object:")  print(json\_obj) |
| Pattern | 276.  277.  **278**.  279.  280.  281.  282.  283.  284.  285.  286.  287.  288.  289.  290.  291.  292.  293.  294.  295. | **Pyramid Pattern Program**  def pattern(n):  k = 2 \* n - 2  for i in range(0,n):  for j in range(0,k):  print(end=" ")  k = k - 1  for j in range(0, i+1):  print("\*", end=" ")  pattern(5)  **Reverse Pyramid Pattern Program**  def pattern(n):  k = 2\*n -2  for i in range(n,-1,-1):  for j in range(k,0,-1):  print(end=" ")  k = k +1  for j in range(0, i+1):  print("\*", end=" ")  pattern(5)  **Right Start Pattern Program**  def pattern(n):  for i in range(0, n):  for j in range(0, i + 1):  print("\* ", end="")  print("&#92r")  for i in range(n, 0 , -1):  for j in range(0, i + 1):  print("\* ", end="")  pattern(5)  **Left Start Pattern Program**  def pattern(n):  k = 2 \* n - 2  for i in range(0, n-1):  for j in range(0, k):  print(end=" ")  k = k - 2  for j in range(0, i + 1):  print("\* ", end="")    k = -1  for i in range(n-1,-1,-1):  for j in range(k,-1,-1):  print(end=" ")  k = k + 2  for j in range(0, i + 1):  print("\* ", end="")  pattern(5)  **Hour glass pattern program**  def pattern(n):  k = n - 2  for i in range(n, -1 , -1):  for j in range(k , 0 , -1):  print(end=" ")  k = k + 1  for j in range(0, i+1):  print("\* " , end="")    k = 2 \* n - 2  for i in range(0 , n+1):  for j in range(0 , k):  print(end="")  k = k - 1  for j in range(0, i + 1):  print("\* ", end="")  pattern(5)  **Half-Pyramid Pattern Program**  def pattern(n):  for i in range(0,n):  for j in range(0, i+1):  print("\* " , end="")  pattern(5)  **Left Half-Pyramid Pattern Program**  def pattern(n):  k = 2 \* n - 2  for i in range(0, n):  for j in range(0, k):  print(end=" ")  k = k - 2  for j in range(0, i + 1):  print("\* ", end="")  pattern(5)  **Downward Half-Pyramid Pattern Program**  def pattern(n):  for i in range(n, -1, -1):  for j in range(0, i + 1):  print("\* ", end="")  pattern(5)    **Diamond Star Pattern Program**  for i in range(5):  for j in range(5):  if i + j == 2 or i - j == 2 or i + j == 6 or j - i == 2:  print("\*", end="")  else:  print(end=" ")  print()    pattern(5)  **Diamond Shaped Pattern Program**  def pattern(n):  k = 2 \* n - 2  for i in range(0, n):  for j in range(0 , k):  print(end=" ")  k = k - 1  for j in range(0 , i + 1 ):  print("\* ", end="")  k = n - 2  for i in range(n , -1, -1):  for j in range(k , 0 , -1):  print(end=" ")  k = k + 1  for j in range(0 , i + 1):  print("\* ", end="")  pattern(5)  **Simple Numbers Program**  def pattern(n):  x = 0  for i in range(0 , n):  x += 1  for j in range(0, i + 1):  print(x , end=" ")  print("")  pattern(5)  **Pascal's triangle program**  def pascal(n):  for i in range(0, n):  for j in range(0, i + 1):  print(function(i, j)," ", end="")  print()  def function(n, k):  res = 1  if (k &gt; n - k):  k = n - k  for i in range(0, k):  res = res \* (n - i)  res = res // (i + 1)  return res  pascal(7)  **Half-Pyramid Pattern With Numbers**  def pattern(n):  for i in range(1, n):  for j in range(1, i + 1):  print(j, end= " ")    pattern(5)    **Diamond Pattern With Numbers**  def pattern(n):  k = 2 \* n - 2  x = 0  for i in range(0, n):  x += 1  for j in range(0, k):  print(end=" ")  k = k - 1  for j in range(0, i + 1):  print(x, end=" ")    k = n - 2  x = n + 2  for i in range(n, -1, -1):  x -= 1  for j in range(k, 0, -1):  print(end=" ")  k = k + 1  for j in range(0, i + 1):  print(x, end=" ")    pattern(5)  **Descending Order Pattern Program**  def pattern(n):  for i in range(n, 0, -1):  for j in range(1, i + 1):  print(j, end=" ")  pattern(5)  **Binary Numbers Pattern Program**  def pattern(n):  k = 2 \* n - 2  for i in range(0, n):  for j in range(0, k):  print(end=" ")  k = k - 1  for j in range(0, i + 1):  print('10', end="")  pattern(5)  **Character Pattern Program**  def pattern(n):  k = 2 \* n - 2  x = 65  for i in range(0, n):  for j in range(0, k):  print(end=" ")  k = k - 1  for j in range(0, i + 1):  ch = chr(x)  print(ch, end=" ")  x += 1  print("")  pattern(7)  **K Shape Character Program**  for i in range(7):  for j in range(7):  if j == 0 or i - j == 3 or i + j == 3:  print("\*", end="")  else:  print(end=" ")  print()  **Triangle Character Pattern Program**  def pattern(n):  k = 2 \* n - 2  x = 65  for i in range(0, n):  ch = chr(x)  x += 1  for j in range(0, k):  print(end=" ")  k = k - 1  for j in range(0, i + 1):  print(ch, end=" ")  print("")  pattern(5)  **Diamond Shaped Character Pattern Program**  def pattern(n):  k = 2 \* n - 2  for i in range(0, n):  for j in range(0, k):  print(end=" ")  k = k - 1  x = 65  for j in range(0, i + 1):  ch = chr(x)  print(ch, end=" ")  x += 1  print("")  k = n - 2  x = 65  for i in range(n, -1, -1):  for j in range(k, 0, -1):  print(end=" ")  k = k + 1  for j in range(0, i + 1):  ch = chr(x)  print(ch, end=" ")  x += 1  print(")  pattern(5) |
| Numpy | 296.  297.  298.  299.  300.  301.  302.  303.  304.  305.  306. | **Create a 4X2 integer array and Prints its attributes**  import numpy  firstArray = numpy.empty([4,2], dtype = numpy.uint16)  print("Printing Array")  print(firstArray)  print("Printing numpy array Attributes")  print("1> Array Shape is: ", firstArray.shape)  print("2>. Array dimensions are ", firstArray.ndim)  print("3>. Length of each element of array in bytes is ", firstArray.itemsize)  **Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10**  import numpy  print("Creating 5X2 array using numpy.arange")  sampleArray = numpy.arange(100, 200, 10)  sampleArray = sampleArray.reshape(5,2)  print (sampleArray)  **Following is the provided numPy array. return array of items in the third column from all rows**  import numpy  sampleArray = numpy.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])  print("Printing Input Array")  print(sampleArray)  print("\n Printing array of items in the third column from all rows")  newArray = sampleArray[...,1]  print(newArray)  **Following is the given numpy array return array of odd rows and even columns**  **sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24],**  **[27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])**  import numpy  sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24],  [27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])  print("Printing Input Array")  print(sampleArray)  print("\n Printing array of odd rows and even columns")  newArray = sampleArray[::2, 1::2]  print(newArray)  **Add the following two NumPy arrays and Modify a result array by calculating the square root of each element**  **arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])**  **arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])**  import numpy  arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])  arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])  resultArray = arrayOne + arrayTwo  print("addition of two arrays is \n")  print(resultArray)  for num in numpy.nditer(resultArray, op\_flags = ['readwrite']):  num[...] = num\*num  print("\nResult array after calculating the square root of all elements\n")  print(resultArray)  **Split the array into four equal-sized sub-arrays**  import numpy  print("Creating 8X3 array using numpy.arange")  sampleArray = numpy.arange(10, 34, 1)  sampleArray = sampleArray.reshape(8,3)  print (sampleArray)  print("\nDividing 8X3 array into 4 sub array\n")  subArrays = numpy.split(sampleArray, 4)  print(subArrays)  **Sort following NumPy array**  **sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])**  import numpy  print("Printing Original array")  sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])  print (sampleArray)  sortArrayByRow = sampleArray[:,sampleArray[1,:].argsort()]  print("Sorting Original array by secoond row")  print(sortArrayByRow)  print("Sorting Original array by secoond column")  sortArrayByColumn = sampleArray[sampleArray[:,1].argsort()]  print(sortArrayByColumn)  **Following is the 2-D array. Print max from axis 0 and min from axis 1**  **sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])**  import numpy  print("Printing Original array")  sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])  print (sampleArray)  minOfAxisOne = numpy.amin(sampleArray, 1)  print("Printing amin Of Axis 1")  print(minOfAxisOne)  maxOfAxisOne = numpy.amax(sampleArray, 0)  print("Printing amax Of Axis 0")  print(maxOfAxisOne)  **Following is the input NumPy array delete column two and insert following new column in its place.**  **sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])**  **newColumn = numpy.array([[10,10,10]])**  import numpy  print("Printing Original array")  sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])  print (sampleArray)  print("Array after deleting column 2 on axis 1")  sampleArray = numpy.delete(sampleArray , 1, axis = 1)  print (sampleArray)  arr = numpy.array([[10,10,10]])  print("Array after inserting column 2 on axis 1")  sampleArray = numpy.insert(sampleArray , 1, arr, axis = 1)  print (sampleArray)  **Create a two 2-D array and Plot it using matplotlib**  import numpy  print("Printing Original array")  sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])  print (sampleArray)  print("Array after deleting column 2 on axis 1")  sampleArray = numpy.delete(sampleArray , 1, axis = 1)  print (sampleArray)  arr = numpy.array([[10,10,10]])  print("Array after inserting column 2 on axis 1")  sampleArray = numpy.insert(sampleArray , 1, arr, axis = 1)  print (sampleArray)  **Write a programm to convert numpy array into tuples.**  import numpy as np  ini\_array = np.array([['manjeet', 'akshat'], ['nikhil', 'akash']])  result = tuple(map(tuple, ini\_array))    print ("Resultant Array :"+str(result)) |
| Math | 307.  308.  309.  310.  311.  312.  313.  314.  315.  316.  317.  318.  319.  320.  321.  322.  323.  324.  325.  326.  327.  328.  329.  330.  331.  332.  333.  334.  335. | **solving a square equation and returning multiple values**  **ax^2 + bx + c=0**  import math  def solve(a,b,c):  d=math.sqrt(b\*\*2-4\*a\*c)  x1=(-b-d)/(2\*a)  x2=(-b+d)/(2\*a)  return x1,x2  x1,x2=solve(1,3,-4)  print(x1)  print(x2)  **floating point calculation are not always precise**  a=0.1+0.2-0.3  print(a)  from decimal import Decimal  b=Decimal('0.1')+\  Decimal('0.2')-\  Decimal('0.3')  print(b)  **Program to find the minimum element in tuple list**  test\_list = [(2, 4), (6, 7), (5, 1), (6, 10), (8, 7)]  print("The original list : " + str(test\_list))  res = min(int(j) for i in test\_list for j in i)  print("The Minimum element of list is : " + str(res))  **Write a Python program to convert degree to radian.**  pi=22/7  degree = float(input("Input degrees: "))  radian = degree\*(pi/180)  print(radian)  **Write a Python program to convert radian to degree.**  pi=22/7  radian = float(input("Input radians: "))  degree = radian\*(180/pi)  print(degree)  **Write a Python program to calculate the area of a trapezoid.**  base\_1 = 5  base\_2 = 6  height = float(input("Height of trapezoid: "))  base\_1 = float(input('Base one value: '))  base\_2 = float(input('Base two value: '))  area = ((base\_1 + base\_2) / 2) \* height  print("Area is:", area)  **Write a Python program to sum all amicable numbers from 1 to specified numbers.**  **Note: Amicable numbers are two different numbers so related that the sum of the proper divisors of each is equal to the other number. (A proper divisor of a number is a positive factor of that number other than the number itself. For example, the proper divisors of 6 are 1, 2, and 3.)**  def amicable\_numbers\_sum(limit):  if not isinstance(limit, int):  return "Input is not an integer!"  if limit < 1:  return "Input must be bigger than 0!"  amicables = set()  for num in range(2, limit+1):  if num in amicables:  continue  sum\_fact = sum([fact for fact in range(1, num) if num % fact == 0])  sum\_fact2 = sum([fact for fact in range(1, sum\_fact) if sum\_fact % fact == 0])  if num == sum\_fact2 and num != sum\_fact:  amicables.add(num)  amicables.add(sum\_fact2)  return sum(amicables)  print(amicable\_numbers\_sum(9999))  **Write a Python program to calculate the area of a parallelogram.**  base = float(input('Length of base: '))  height = float(input('Measurement of height: '))  area = base \* height  print("Area is: ", area)  **Write a Python program to calculate surface volume and area of a cylinder.**  pi=22/7  height = float(input('Height of cylinder: '))  radian = float(input('Radius of cylinder: '))  volume = pi \* radian \* radian \* height  sur\_area = ((2\*pi\*radian) \* height) + ((pi\*radian\*\*2)\*2)  print("Volume is: ", volume)  print("Surface Area is: ", sur\_area)  **Write a Python program to calculate surface volume and area of a sphere.**  pi=22/7  radian = float(input('Radius of sphere: '))  sur\_area = 4 \* pi \* radian \*\*2  volume = (4/3) \* (pi \* radian \*\* 3)  print("Surface Area is: ", sur\_area)  print("Volume is: ", volume)  **Write a Python program to calculate arc length of an angle.**  def arclength():  pi=22/7  diameter = float(input('Diameter of circle: '))  angle = float(input('angle measure: '))  if angle >= 360:  print("Angle is not possible")  return  arc\_length = (pi\*diameter) \* (angle/360)  print("Arc Length is: ", arc\_length)  arclength()  **Write a Python program to calculate the area of a sector.**  def sectorarea():  pi=22/7  radius = float(input('Radius of Circle: '))  angle = float(input('angle measure: '))  if angle >= 360:  print("Angle is not possible")  return  sur\_area = (pi\*radius\*\*2) \* (angle/360)  print("Sector Area: ", sur\_area)  sectorarea()  **Write a Python program to calculate the discriminant value.**  **Note: The discriminant is the name given to the expression that appears under the square root (radical) sign in the quadratic formula.**  def discriminant():  x\_value = float(input('The x value: '))  y\_value = float(input('The y value: '))  z\_value = float(input('The z value: '))  discriminant = (y\_value\*\*2) - (4\*x\_value\*z\_value)  if discriminant > 0:  print('Two Solutions. Discriminant value is:', discriminant)  elif discriminant == 0:  print('One Solution. Discriminant value is:', discriminant)  elif discriminant < 0:  print('No Real Solutions. Discriminant value is:', discriminant)  discriminant()  **Write a Python program to find the smallest multiple of the first n numbers. Also, display the factors.**  def smallest\_multiple(n):  if (n<=2):  return n  i = n \* 2  factors = [number for number in range(n, 1, -1) if number \* 2 > n]  print(factors)  while True:  for a in factors:  if i % a != 0:  i += n  break  if (a == factors[-1] and i % a == 0):  return i    print(smallest\_multiple(13))  print(smallest\_multiple(11))  **Write a Python program to calculate the difference between the squared sum of first n natural numbers and the sum of squared first n natural numbers.(default value of number=2).**  def sum\_difference(n=2):  sum\_of\_squares = 0  square\_of\_sum = 0  for num in range(1, n+1):  sum\_of\_squares += num \* num  square\_of\_sum += num  square\_of\_sum = square\_of\_sum \*\* 2  return square\_of\_sum - sum\_of\_squares  print(sum\_difference(12))  **Write a Python program to calculate the sum of all digits of the base to the specified power.**  def power\_base\_sum(base, power):  return sum([int(i) for i in str(pow(base, power))])  print(power\_base\_sum(2, 100))  print(power\_base\_sum(8, 10))  **Write a Python program to find out, if the given number is abundant.**  **Note: In number theory, an abundant number or excessive number is a number for which the sum of its proper divisors is greater than the number itself. The integer 12 is the first abundant number. Its proper divisors are 1, 2, 3, 4 and 6 for a total of 16.**  def is\_abundant(n):  fctr\_sum = sum([fctr for fctr in range(1, n) if n % fctr == 0])  return fctr\_sum > n  print(is\_abundant(12))  print(is\_abundant(13))  **Write a Python program to returns sum of all divisors of a number.**  def sum\_div(number):  divisors = [1]  for i in range(2, number):  if (number % i)==0:  divisors.append(i)  return sum(divisors)  print(sum\_div(8))  print(sum\_div(12))  **Write a Python program to print all permutations of a given string (including duplicates).**  def permute\_string(str):  if len(str) == 0:  return ['']  prev\_list = permute\_string(str[1:len(str)])  next\_list = []  for i in range(0,len(prev\_list)):  for j in range(0,len(str)):  new\_str = prev\_list[i][0:j]+str[0]+prev\_list[i][j:len(str)-1]  if new\_str not in next\_list:  next\_list.append(new\_str)  return next\_list  print(permute\_string('ABCD'))  **Write a Python program to multiply two integers without using the \* operator in python.**  def multiply(x, y):  if y < 0:  return -multiply(x, -y)  elif y == 0:  return 0  elif y == 1:  return x  else:  return x + multiply(x, y - 1)  print(multiply(3, 5))  **Write a Python program to calculate magic square.**  **A magic square is an arrangement of distinct numbers (i.e., each number is used once), usually integers, in a square grid, where the numbers in each row, and in each column, and the numbers in the main and secondary diagonals, all add up to the same number, called the "magic constant." A magic square has the same number of rows as it has columns, and in conventional math notation, "n" stands for the number of rows (and columns) it has. Thus, a magic square always contains n2 numbers, and its size (the number of rows [and columns] it has) is described as being "of order n".**  def magic\_square\_test(my\_matrix):  iSize = len(my\_matrix[0])  sum\_list = []    #Horizontal Part:  sum\_list.extend([sum (lines) for lines in my\_matrix])  #Vertical Part:  for col in range(iSize):  sum\_list.append(sum(row[col] for row in my\_matrix))  #Diagonals Part  result1 = 0  for i in range(0,iSize):  result1 +=my\_matrix[i][i]  sum\_list.append(result1)    result2 = 0  for i in range(iSize-1,-1,-1):  result2 +=my\_matrix[i][i]  sum\_list.append(result2)  if len(set(sum\_list))>1:  return False  return True  m=[[7, 12, 1, 14], [2, 13, 8, 11], [16, 3, 10, 5], [9, 6, 15, 4]]  print(magic\_square\_test(m))  m=[[2, 7, 6], [9, 5, 1], [4, 3, 8]]  print(magic\_square\_test(m))  **Write a Python program to print number with commas as thousands separators (from right side)**  print("{:,}".format(1000000))  print("{:,}".format(10000))  **Write a python program to calculate the area of a regular polygon.**  from math import tan, pi  n\_sides = int(input("Input number of sides: "))  s\_length = float(input("Input the length of a side: "))  p\_area = n\_sides \* (s\_length \*\* 2) / (4 \* tan(pi / n\_sides))  print("The area of the polygon is: ",p\_area)  **Write a Python program to convert a binary number to decimal number.**  b\_num = list(input("Input a binary number: "))  value = 0  for i in range(len(b\_num)):  digit = b\_num.pop()  if digit == '1':  value = value + pow(2, i)  print("The decimal value of the number is", value)  **Write a Python program to print a complex number and its real and imaginary parts.**  cn = complex(2,3)  print("Complex Number: ",cn)  print("Complex Number - Real part: ",cn.real)  print("Complex Number - Imaginary part: ",cn.imag)  **Write a Python program to create a Pythagorean theorem calculator.**  **Note : In mathematics, the Pythagorean theorem, also known as Pythagoras' theorem, is a fundamental relation in Euclidean geometry among the three sides of a right triangle. It states that the square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides.**  from math import sqrt  print('Pythagorean theorem calculator! Calculate your triangle sides.')  print('Assume the sides are a, b, c and c is the hypotenuse (the side opposite the right angle')  formula = input('Which side (a, b, c) do you wish to calculate? side> ')  if formula == 'c':  side\_a = int(input('Input the length of side a: '))  side\_b = int(input('Input the length of side b: '))  side\_c = sqrt(side\_a \* side\_a + side\_b \* side\_b)    print('The length of side c is: ' )  print(side\_c)  elif formula == 'a':  side\_b = int(input('Input the length of side b: '))  side\_c = int(input('Input the length of side c: '))    side\_a = sqrt((side\_c \* side\_c) - (side\_b \* side\_b))    print('The length of side a is' )  print(side\_a)  elif formula == 'b':  side\_a = int(input('Input the length of side a: '))  side\_b = int(input('Input the length of side c: '))    side\_c = sqrt(side\_c \* side\_c - side\_a \* side\_a)    print('The length of side b is')  print(side\_c)  else:  print('Please select a side between a, b, c')  **Write a Python program to convert a float to ratio.**  from fractions import Fraction  value = 4.2  print(Fraction(value).limit\_denominator())  **Write a program to generate a simple calculator**  def add(x, y):  return x + y  def subtract(x, y):  return x - y  def multiply(x, y):  return x \* y  def divide(x, y):  return x / y  print("Select operation.")  print("1.Add")  print("2.Subtract")  print("3.Multiply")  print("4.Divide")  choice = input("Enter choice(1/2/3/4): ")  num1 = float(input("Enter first number: "))  num2 = float(input("Enter second number: "))  if choice == '1':  print(num1,"+",num2,"=", add(num1,num2))  elif choice == '2':  print(num1,"-",num2,"=", subtract(num1,num2))  elif choice == '3':  print(num1,"\*",num2,"=", multiply(num1,num2))  elif choice == '4':  print(num1,"/",num2,"=", divide(num1,num2))  else:  print("Invalid input")  **Write a program to shuffle a deck of cards**  import itertools, random  deck=list(itertools.product(range(1,14),['Spade','Heart','Diamond','Club']))  random.shuffle(deck)  print("You got:")  for i in range(5):  print(deck[i][0], "of", deck[i][1]) |
| Date time module | 336.  337.  338.  339.  340.  341.  342.  343.  344.  345.  346.  347.  348.  349. | **Write a Python script to display the various Date Time formats.**  **a) Current date and time**  **b) Current year**  **c) Month of year**  **d) Week number of the year**  **e) Weekday of the week**  **f) Day of year**  **g) Day of the month**  **h) Day of week**  import time  import datetime  print("Current date and time: " , datetime.datetime.now())  print("Current year: ", datetime.date.today().strftime("%Y"))  print("Month of year: ", datetime.date.today().strftime("%B"))  print("Week number of the year: ", datetime.date.today().strftime("%W"))  print("Weekday of the week: ", datetime.date.today().strftime("%w"))  print("Day of year: ", datetime.date.today().strftime("%j"))  print("Day of the month : ", datetime.date.today().strftime("%d"))  print("Day of week: ", datetime.date.today().strftime("%A"))  **Write a Python program to convert a string to datetime.**  from datetime import datetime  date\_object = datetime.strptime('Jul 1 2014 2:43PM', '%b %d %Y %I:%M%p')  print(date\_object)  **Write a Python program to get the current time.**  import datetime  print(datetime.datetime.now().time())  **Write a Python program to subtract five days from current date.**  from datetime import date, timedelta  dt = date.today() - timedelta(5)  print('Current Date :',date.today())  **Write a Python program to print yesterday, today, tomorrow.**  import datetime  today = datetime.date.today()  yesterday = today - datetime.timedelta(days = 1)  tomorrow = today + datetime.timedelta(days = 1)  print('Yesterday : ',yesterday)  print('Today : ',today)  print('Tomorrow : ',tomorrow)  print('5 days before Current Date :',dt)  **Write a Python program to convert the date to datetime (midnight of the date).**  from datetime import date  from datetime import datetime  dt = date.today()  print(datetime.combine(dt, datetime.min.time()))  **Write a Python program to print next 5 days starting from today.**  import datetime  base = datetime.datetime.today()  for x in range(0, 5):  print(base + datetime.timedelta(days=x))  **Write a Python program to convert Year/Month/Day to Day of Year.**  import datetime  today = datetime.datetime.now()  day\_of\_year = (today - datetime.datetime(today.year, 1, 1)).days +1  print(day\_of\_year)  **Write a Python program to get current time in milliseconds.**  import time  milli\_sec = int(round(time.time() \* 1000))  print(milli\_sec)  **Write a Python program to find the date of the first Monday of a given week.**  import time  print(time.asctime(time.strptime('2015 50 1', '%Y %W %w')))  **Write a Python program to select all the Sundays of a specified year.**  from datetime import date, timedelta  def all\_sundays(year):  # January 1st of the given year  dt = date(year, 1, 1)  # First Sunday of the given year  dt += timedelta(days = 6 - dt.weekday())  while dt.year == year:  yield dt  dt += timedelta(days = 7)    for s in all\_sundays(2020):  print(s)  **Write a Python program to get days between two dates.**  from datetime import date  a = date(2000,2,28)  b = date(2001,2,28)  print(b-a)  **Write a Python program to get the date of the last Tuesday.**  from datetime import date  from datetime import timedelta  today = date.today()  offset = (today.weekday() - 1) % 7  last\_tuesday = today - timedelta(days=offset)  print(last\_tuesday)  **Write a Python program to test the third Tuesday of a month.**  from datetime import datetime  def is\_third\_tuesday(s):  d = datetime.strptime(s, '%b %d, %Y')  return d.weekday() == 1 and 14 < d.day < 22  print(is\_third\_tuesday('Jun 23, 2015'))  print(is\_third\_tuesday('Jun 16, 2015'))  print(is\_third\_tuesday('Jul 21, 2015')) |
| Matrix | 350.  351.  352.  353.  354.  355.  356.  357.  358.  359.  360.  361.  362.  363.  364.  365.  366.  367.  368.  369.  370.  371.  372.  373.  374. | **Program to add two matrices using nested loop**  X = [[12,7,3],  [4 ,5,6],  [7 ,8,9]]  Y = [[5,8,1],  [6,7,3],  [4,5,9]]  result = [[0,0,0],  [0,0,0],  [0,0,0]]  for i in range(len(X)):  for j in range(len(X[0])):  result[i][j] = X[i][j] + Y[i][j]  for r in result:  print(r)  **Program to transpose a matrix using a nested loop**  X = [[12,7],  [4 ,5],  [3 ,8]]  result = [[0,0,0],  [0,0,0]]  for i in range(len(X)):  for j in range(len(X[0])):  result[j][i] = X[i][j]  for r in result:  print(r)  **Program to multiply two matrices using nested loops**  X = [[12,7,3],  [4 ,5,6],  [7 ,8,9]]  Y = [[5,8,1,2],  [6,7,3,0],  [4,5,9,1]]  result = [[0,0,0,0],  [0,0,0,0],  [0,0,0,0]]  for i in range(len(X)):  for j in range(len(Y[0])):  for k in range(len(Y)):  result[i][j] += X[i][k] \* Y[k][j]  for r in result:  print(r)  **Program to find the row with minimum sum**  test\_matrix = [[1, 3, 1], [4, 5, 3], [1, 2, 4]]    print ("The original matrix is : " + str(test\_matrix))  res = reduce(lambda i, j: i if sum(i) < sum(j) else j, test\_matrix)  print ("Minimum sum row is : " + str(res))  **Find the Sum of columns of a 2-D Matrix where first element is odd**  def sumColumns(arr, r, c):  sum = 0  for j in range(c):  if (arr[0][j] % 2 != 0):  for i in range(r):  sum += arr[i][j]  return sum    arr = [ [8, 2, 3, 5], [9, 8, 7, 6], [1, 2, 5, 5] ]  r = len(arr)  c = len(arr[0])  print(sumColumns(arr, r, c))  **Check if the rows of a binary matrix can be made unique by removing a single column**  def uniqueRows(s, m, n):  i, j, count = 0, 0, 0    for i in range(n):  for j in range(i):  if (s[i] == s[j]):  count += 1  break    if (count >= 1):  print("No" )  else:  print("Yes")    m = 3  n = 3  s = [[ 1, 0, 1 ],  [ 0, 0, 0 ],  [ 1, 0, 0 ]]    uniqueRows(s, m, n)  **Find the row with minimum element**  test\_matrix = [[1, 3, 1], [4, 5, 3], [7, 2, 4]]    print ("The original matrix is : " + str(test\_matrix))  res = reduce(lambda i, j: i if min(i) < min(j) else j, test\_matrix)  print ("Minimum element sublist is : " + str(res))  **Find the index of minimum element**  test\_list = [2, 4, 6, 8, 2, 2]  print("The original list : " + str(test\_list))    temp = min(test\_list)  res = [i for i, j in enumerate(test\_list) if j == temp]  print("The Positions of minimum element : " + str(res))  **Remove similar element rows in tuple Matrix**  test\_tup = ((1, 3, 5), (2, 2, 2),  (9, 10, 10), (4, 4, 4))  print("The original tuple : " + str(test\_tup))    res = tuple(ele for ele in test\_tup if not all(sub == ele[0] for sub in ele))  print("The tuple after removal of like-element rows : " + str(res))  **Program for Maximum size square sub-matrix with all 1s**  def printMaxSubSquare(M):  R = len(M) # no. of rows in M[][]  C = len(M[0]) # no. of columns in M[][]    S = [[0 for k in range(C)] for l in range(R)]  for i in range(1, R):  for j in range(1, C):  if (M[i][j] == 1):  S[i][j] = min(S[i][j-1], S[i-1][j],  S[i-1][j-1]) + 1  else:  S[i][j] = 0    max\_of\_s = S[0][0]  max\_i = 0  max\_j = 0  for i in range(R):  for j in range(C):  if (max\_of\_s < S[i][j]):  max\_of\_s = S[i][j]  max\_i = i  max\_j = j    print("Maximum size sub-matrix is: ")  for i in range(max\_i, max\_i - max\_of\_s, -1):  for j in range(max\_j, max\_j - max\_of\_s, -1):  print (M[i][j], end = " ")  print("")    M = [[0, 1, 1, 0, 1],  [1, 1, 0, 1, 0],  [0, 1, 1, 1, 0],  [1, 1, 1, 1, 0],  [1, 1, 1, 1, 1],  [0, 0, 0, 0, 0]]    printMaxSubSquare(M)  **Program to extract elements from Matrix in Z form**  arr = [[4, 5, 6, 8],  [1, 2, 3, 1],  [7, 8, 9, 4],  [1, 8, 7, 5]]    n = len(arr[0])    i=0  for j in range(0, n-1):  print(arr[i][j], end =" ")    k = 1  for i in range(0, n):  for j in range(n, 0, -1):  if(j==n-k):  print(arr[i][j], end = " ")  break  k+=1  i=n-1;  for j in range(0, n):  print(arr[i][j], end = " ")  **Program to find Row lengths in Matrix**  test\_list = [[4, 5, 6], [7, 8], [2]]  print("The original list : " + str(test\_list))  res = [sum(len(row) > idx for row in test\_list)  for idx in range(max(map(len, test\_list)))]  print("The row lengths in matrix : " + str(res))  **Find Max value in Nth Column in Matrix**  test\_list = [[5, 6, 7],  [9, 10, 2],  [10, 3, 4]]    print("The original list is : " + str(test\_list))  N = 2  res = [max(i) for i in zip(\*test\_list)][N]    print("Max value of Nth column is : " + str(res))  **Find Min value in Nth Column in Matrix**  test\_list = [[5, 6, 7],  [9, 10, 2],  [10, 3, 4]]    print("The original list is : " + str(test\_list))  N = 2  res = [min(i) for i in zip(\*test\_list)][N]  print("Min value of Nth column is : " + str(res))  **Program to Remove front column from Matrix**  test\_list = [[1, 3, 4], [2, 4, 6], [3, 8, 1]]  print("The original list : " + str(test\_list))  for ele in test\_list: del ele[0]  print("Matrix after removal of front column : " + str(test\_list))  **Program for Matrix Kth column summation**  test\_list = [[5, 6, 7],  [9, 10, 2],  [10, 3, 4]]    print("The original list is : " + str(test\_list))  K = 2  res = [sum(idx) for idx in zip(\*test\_list)][K]  print("Sum of Kth column is : " + str(res))  **Program Non-Repeating value Summation in Matrix**  test\_matrix = [[1, 3, 1], [4, 5, 3], [1, 2, 4]]    print ("The original matrix is : " + str(test\_matrix))  res = sum(list(set(i for j in test\_matrix for i in j)))  print ("Unique values summation in matrix are : " + str(res))  **Program for Consecutive Row summation in Matrix**  test\_list = [[3, 4, 5], [4, 6, 8], [1, 9, 2], [3, 7, 10]]  print("The original list : " + str(test\_list))  res = [sum(abs(i + j) for i, j in zip(\*ele)) for ele in zip(test\_list, test\_list[1:])]  print("The row summation sublist : " + str(res))  **Program to Search in Nth Column of Matrix**  test\_list = [[1, 4, 5], [6, 7, 8], [8, 3, 0]]  print("The original list : " + str(test\_list))  N = 1  ele = 3  res = any(sub[N] == ele for sub in test\_list)  print("Does element exists in particular column : " + str(res))  **Program to Count Negative Numbers in a Column-Wise and Row-Wise Sorted Matrix**  def countNegative(M, n, m):  count = 0  for i in range(n):  for j in range(m):  if M[i][j] < 0:  count += 1  else:  break  return count    M = [  [-3, -2, -1, 1],  [-2, 2, 3, 4],  [ 4, 5, 7, 8]  ]  print(countNegative(M, 3, 4))  **Program to Count all sorted rows in a matrix**  def sortedCount(mat, r, c):  result = 0  for i in range(r):  j = 0  for j in range(c - 1):  if mat[i][j + 1] <= mat[i][j]:  break  if j == c - 2:  result += 1  for i in range(0, r):  j = 0  for j in range(c - 1, 0, -1):  if mat[i][j - 1] <= mat[i][j]:  break  if c > 1 and j == 1:  result += 1  return result    m, n = 4, 5  mat = [[1, 2, 3, 4, 5],  [4, 3, 1, 2, 6],  [8, 7, 6, 5, 4],  [5, 7, 8, 9, 10]]    print(sortedCount(mat, m, n))  **Program to Remove first X rows and columns from the matrix.**  def remove\_X\_Rows\_and\_Columns(a, n, x):    print("Removing First ", x," rows and columns:")  for i in range(x, n):  for j in range(x, n):  print(a[i][j], end = " ")    print()    def prMatrix(a, n):    for i in range(n):  for j in range(n):  print(a[i][j], end = " ")  print()    if \_\_name\_\_ == '\_\_main\_\_':  n = 4  a = [[0 for i in range(n)]  for i in range(n)]  for i in range(n):  for j in range(n):  a[i][j] = (i \* 10 + j)  print("Original Matrix:")  prMatrix(a, n)  x = 2  remove\_X\_Rows\_and\_Columns(a, n, x)  **Program to swap the element of first and last column and display the result**    def interchangeFirstLast(mat, n, m):  rows = n  for i in range(n):  t = mat[i][0];  mat[i][0] = mat[i][n-1]  mat[i][n-1] = t  mat = [[8, 9, 7, 6],  [4, 7, 6, 5],  [3, 2, 1, 8],  [9, 9, 7, 7]]    n = 4  m = 4  interchangeFirstLast(mat, n, m)  for i in range(n):  for j in range(m):  print(mat[i][j], end = " ")  print("\n")  **Program to Count all the columns in a matrix which are sorted in descending**  def countDescCol(A):  countOfCol = 0  for col in zip(\*A):  if all(col[i] >= col[i + 1] for i in range(len(col) - 1)):  countOfCol += 1  return countOfCol  A = [[1, 3], [0, 2]]  print(countDescCol(A))  **Program to interchange the diagonals of matrix**  N = 3  def interchangeDiagonals(array):  for i in range(N):  if (i != N / 2):  temp = array[i][i]  array[i][i] = array[i][N - i - 1]  array[i][N - i - 1] = temp    for i in range(N):  for j in range(N):  print(array[i][j], end = " ")  print()  if \_\_name\_\_ == '\_\_main\_\_':  array = [ 4, 5, 6 ],[ 1, 2, 3 ],[ 7, 8, 9 ]  interchangeDiagonals(array) |
| Puzzles | 375.  376.  377.  378.  379.  380.  381.  382.  383.  384.  385.  386.  387.  389.  390.  391.  392.  393.  394.  395.  396.  397.  398.  399.  400.  401.  402.  403.  404.  405.  406.  407.  408.  409.  410.  411.  412.  413.  414.  415.  416.  417.  418.  419.  420.    421.  422.  423.  424.  425.  426.  427.  428.  429.  430.  431.  432.  433.  434.  435.  436.  437.  438.  439.  440.  441.  442.  443.  444.  445.  446.  447.  448.  449.  450.  451.  452.  453.  454.  455.  456.  457.  458.  459.  460.  461.  462.  463.  464.  465.  466.  467.  468.  469.  470.  471.  472.  473.  474.  475.  476.  477.  478.  479.  480.  481.  482.  483.  484.  485.  486.  487. | **Print first m multiples of a m number without using any loops in Python.**  def multiple(m, n):  a = range(n, (m \* n)+1, n)  print(\*a)    m = 4  n = 3  multiple(m, n)  **Given N number of bottles in which one bottle is poisoned. So the task is to find out minimum number of rats required to identify the poisoned bottle. A rat can drink any number of bottles at a time.**  **Examples:**  **Input: N = 4**  **Output: 2**  **Input: N = 100**  **Output: 7**  import math  def minRats(n):  return math.ceil(math.log2(n))  n = 1025  print("Minimum ", end = "")  print(minRats(n), end = " ")  print("rat(s) are required")  **Find the winner of the Game to Win by erasing any two consecutive similar alphabets**  **Given a string consisting of lower case alphabets.**  **Rules of the Game:**  **A player can choose a pair of similar consecutive characters and erase them.**  **There are two players playing the game, the player who makes the last move wins.**  **The task is to find the winner if A goes first and both play optimally.**  def findWinner(s) :  count = 0  n = len(s)  st = []  for i in range(n) :  if (len(st) == 0 or st[-1] != s[i]) :  st.append(s[i])  else :  count += 1  st.pop()    if (count % 2 == 0) :  print("B")  else :  print("A")    if \_\_name\_\_ == "\_\_main\_\_" :  s = "kaak"  findWinner(s)  **Program to remove the non duplicates in a string**  def count(key,s):  c=0  for i in s:  if key==i:  c+=1  return c  def sample(s):  temp=""  for i in range(len(s)):  if count(s[i],s)!=1:  temp+=s[i]  return temp  print(sample("hello world"))  **WAP to which accepts a list of numbers and returns true if 1,2,3 apears in the sequence**  def sample(s):  for i in range(len(s)):  if s[i]==1 and s[i:i+3]=[1,2,3]  return True  return False  sample([2,3,4,1,2,3])  **WAP to add 'ing' at the end of the string and return new string. If it is already ending with 'ing' then add 'ly'. If len() <3 leave it unchanged**  def sample(s):  n=len(s)  if n<3:  return s  if s[n-3:]=='ing':  s+='ly'  else:  s+='ing'  return s  print(sample('ending'))  **wap wchich accepts a string containing pattern of brackets and return true if the**  **#pattern of brackets is correct ootherwise it returns false**  **#the string brackets is correct if it is satisify the bellow conditions**  **#no of opening brakets should be equal to close bracket**  **#should not start with ) and ahould not end with (**  def sample(s):  if s[0]==')' or s[-1]=='(':  return False  flag=0  for i in s:  if i=='(':  flag+=1  elif i==')':  flag-=1  if flag==0:  return True  return False  print(sample("((( )))"))  **WAP to calculate the number of characters to be suffixed in order to make the the given sting as a palindrome**  def suffix(s):  n=str(int(len(s))    for i in n:  if s[i]==s[i+1]:  count+=1  return s  suffix('abcdabc')  **WAP to receive the input and produce the below output**  **i/p : ABAABCCCDCDA**  **O/P : 4A2B4C2D**  def count(k,s):  c=0  for i in s:  if k==i:  c+=1  return c  def sample2(s):    res=" "  for i in s:  if i not in res:  res=res+str(count(i,s))+i  return res  print(sample2("ABAABCCCDCDA"))  **WAP to remove the duplicates present in a given string**  def sample(s):  res=""  for i in s:  if i not in res:  res+=i  return res  print(sample("helllo world"))  **WAP to count the number of duplicates removed from given string**  def sample(s):  res=""  for i in s:  if i not in res:  res+=i  return len(s)-len(res)  print(sample("helllo world"))  **WAP to count special symbols present in a given string**  def sample(s):  count=0  string=’aeiouAEIOU’  for i in s:  if i not in string:  count+=i  return count  print(sample("helllo world"))  **WAP to generate the dictionary b/w 1 to n(both inclusive) as the key and its square as the value**  **{1:1,2:4,3:9,4:16,5:25,6:36,7:49,8:84,.....}**  def sample(n):  d={}  for i in range(1,n+1):  d[i]=i\*i  return d  print(sample(9))  **A person has the capacity to keep 1/2/3/4/5 steps at a time. If he has to cover ‘x’ distance, then how many steps he has to keep?**  def steps(x):  l=1  while x>5:  x=x-5  l+=1  return l  pri  nt(steps(18))  **An array of length ‘n’ is given which contains the elements that lies between the range of 1 to n. Return good or bad based on the below conitions.**  **Return good if,**   1. **All the elements are unique.** 2. **The elements are not sorted in ascending order.** 3. **The elements lie between the range 1 to n.**   def array\_pro(N,a):  for i in a:  if a.count(i)>1 or i<1 or i>N:  return "Bad"  if a==sorted(a):  return "Bad"  return "Good"  print(array\_pro(4,[6,2,1,4]))  print(array\_pro(4,[3,2,1,4]))  print(array\_pro(4,[2,1,1,3]))  **WAP to add the numbers in a list such that if the specified number is present then the sum should contain the previous and next number without considering the secified number.**  **Example:**  **I/p:[1,2,3,4] key=3 🡪 O/p : 1**  **I/p:[1,2,3,2,1] key=2 🡪 O/p : 0**  **I/p:[1,2,3,] key=1 🡪 O/p : 3**  def pos(l,key):  temp=[]  for i in range(0,len(l)):  if(key==l[i]):  temp.append(i-1)  temp.append(i)  temp.append(i+1)  return temp  def add\_num(ld,key):  sum=0  Pos=pos(ld,key)  for i in range(len(ld)):  if i not in Pos:  sum+=ld[i]  return sum  print(add\_num([1,2,7,4,5,6,2,4],4))  **Given a list of numbers wap to check wether one of the 1st four value having 9 or not,even if the length is less than 4 it should be present**  def sample(l):  n=len(l)  if n<4:  return 9 in l  return 9 in l[:4]  sample([1,2,9,5,3,7,8])  **Write a python function which accepts a sentence and finds the number of letters and digits in the sentence.It should return the list containing the first value as lettercount and the second value should be digit count.**  def sample(s):  temp=[0,0]  for i in s:  if i>='a' and i<='z' or i>='A' and i<='Z':  temp[0]+=1  elif i>='0' and i<='9':  temp[1]+=1  else:  pass  return temp  print(sample('Hi!!!! The numbers are 12345678900985'))  **WAP to which accepts a list of numbers and returns true if 1,2,3 apears in the sequence**  def sample(s):  for i in range(len(s)):  if s[i]==1 and s[i:i+3]=[1,2,3]  return True  return False  **Given two numbers, write a python function which returns True if first number is a seed of second number.If ‘x’ is said to be the seed of ‘y’, then the product of each digits of x and the number x should be equal to y.**    **Example: 123,738 🡪123\*1\*2\*3=738 🡪 True**  def sample(a,b):  product=a  while(a!=0):  r=a%10  a=a//10  product\*=r  return product==b  print(sample(123,738))  **Write a python function which accepts a string made of two character and last 2 characters as the return value. If the length of given string is <2 return -1. If len==2, consider both the characters as first 2 as well as last 2.**  **Example: I/p 🡪 hellohai, O/p 🡪 heai**  **I/p 🡪 ab, O/p🡪 abab**  **I/p 🡪 a, O/p 🡪 -1**  def sample(s):  n=len(s)  if n<2:  return -1  return s[:2]+s[-2:]  print(sample('hellohai'))  **Write a python function to find and display 5 digit whose,**   1. **First digit is 2 more than second digit.** 2. **Third digit is 2 less than second digit.** 3. **Fourth digit is 2 less than third digit.** 4. **Fifth digit is 2 more than fourth digit.** 5. **Sum of all the 5 digits is 19.** 6. **Sum of third, fourth, fifth digits is equal to first digit.**   def sample(a):  sumx=0  sumn=0  sum=19  res=''  for i in a:  if i[0]=='x':  sumx+=1  sumn+=int(i[1:])  x=(sum+sumx)//sumn    for i in a:  res+=str(x+int(i[1:]))  print(res)  sample(['x+2','x+0','x-2','x-4','x-2'])  **Write a python function which accepts list of numbers and returns True, if the list contains 2 next to 2 else return False.**  def sample(l):  for i in range(len(l)-1):  if (l[i]==2 and l[i+1]==2):  return True  return False  print(sample([1,2,2,3,5,6]))  **Write a python function to find out whether the given number is divisible by the sum of the digits or not.**  def func(n):  sum\_d=0  temp=str(n)  for i in range(len(temp)):  sum\_d+=int(temp[i])  if n%sum\_d==0:  return True  return False  print(func(72))  **Given two positive integers, write a python function to return True if one of them is 10 or if their sum is 10 else return False.**  def sample(a,b):  return a==10 or b==10 or a+b==10  print(10,9)  print(2,8)  print(2,5)  **Write a Python program to check the validity of a password (input from users).**  **Validation :**  **At least 1 letter between [a-z] and 1 letter between [A-Z].**  **At least 1 number between [0-9].**  **At least 1 character from [$#@].**  **Minimum length 6 characters.**  **Maximum length 16 characters.**  import re  p= input("Input your password")  x = True  while x:  if (len(p)<6 or len(p)>12):  break  elif not re.search("[a-z]",p):  break  elif not re.search("[0-9]",p):  break  elif not re.search("[A-Z]",p):  break  elif not re.search("[$#@]",p):  break  elif re.search("\s",p):  break  else:  print("Valid Password")  x=False  break  if x:  print("Not a Valid Password")  **Write a Python program to calculate a dog's age in dog's years.**  **Note: For the first two years, a dog year is equal to 10.5 human years. After that, each dog year equals 4 human years.**  h\_age = int(input("Input a dog's age in human years: "))  if h\_age < 0:  print("Age must be positive number.")  exit()  elif h\_age <= 2:  d\_age = h\_age \* 10.5  else:  d\_age = 21 + (h\_age - 2)\*4  print("The dog's age in dog's years is", d\_age)  **Write a Python program to check whether an alphabet is a vowel or consonant.**  l = input("Input a letter of the alphabet: ")  if l in ('a', 'e', 'i', 'o', 'u'):  print("%s is a vowel." % l)  elif l == 'y':  print("Sometimes letter y stand for vowel, sometimes stand for consonant.")  else:  print("%s is a consonant." % l)  **Write a Python program to convert month name to a number of days.**  print("List of months: January, February, March, April, May, June, July, August, September, October, November, December")  month\_name = input("Input the name of Month: ")  if month\_name == "February":  print("No. of days: 28/29 days")  elif month\_name in ("April", "June", "September", "November"):  print("No. of days: 30 days")  elif month\_name in ("January", "March", "May", "July", "August", "October", "December"):  print("No. of days: 31 day")  else:  print("Wrong month name")    **Write a Python program to sum of two given integers. However, if the sum is between 15 to 20 it will return 20.**  def sum(x, y):  sum = x + y  if sum in range(15, 20):  return 20  else:  return sum  print(sum(10, 6))  print(sum(10, 2))  print(sum(10, 12))  **Write a Python program to check a string represent an integer or not?**  text = input("Input a string: ")  text = text.strip()  if len(text) < 1:  print("Input a valid text")  else:  if all(text[i] in "0123456789" for i in range(len(text))):  print("The string is an integer.")  elif (text[0] in "+-") and \  all(text[i] in "0123456789" for i in range(1,len(text))):  print("The string is an integer.")  else:  print("The string is not an integer.")  **Write a Python program to check a triangle is equilateral, isosceles or scalene.**  **Note :**  **An equilateral triangle is a triangle in which all three sides are equal.**  **A scalene triangle is a triangle that has three unequal sides.**  **An isosceles triangle is a triangle with (at least) two equal sides.**  print("Input lengths of the triangle sides: ")  x = int(input("x: "))  y = int(input("y: "))  z = int(input("z: "))  if x == y == z:  print("Equilateral triangle")  elif x==y or y==z or z==x:  print("isosceles triangle")  else:  print("Scalene triangle")  **Write a Python program that reads two integers representing a month and day and prints the season for that month and day.**  month = input("Input the month (e.g. January, February etc.): ")  day = int(input("Input the day: "))  if month in ('January', 'February', 'March'):  season = 'winter'  elif month in ('April', 'May', 'June'):  season = 'spring'  elif month in ('July', 'August', 'September'):  season = 'summer'  else:  season = 'autumn'  if (month == 'March') and (day > 19):  season = 'spring'  elif (month == 'June') and (day > 20):  season = 'summer'  elif (month == 'September') and (day > 21):  season = 'autumn'  elif (month == 'December') and (day > 20):  season = 'winter'  print("Season is",season)  **Write a Python program to display astrological sign for given date of birth.**  day = int(input("Input birthday: "))  month = input("Input month of birth (e.g. march, july etc): ")  if month == 'december':  astro\_sign = 'Sagittarius' if (day < 22) else 'capricorn'  elif month == 'january':  astro\_sign = 'Capricorn' if (day < 20) else 'aquarius'  elif month == 'february':  astro\_sign = 'Aquarius' if (day < 19) else 'pisces'  elif month == 'march':  astro\_sign = 'Pisces' if (day < 21) else 'aries'  elif month == 'april':  astro\_sign = 'Aries' if (day < 20) else 'taurus'  elif month == 'may':  astro\_sign = 'Taurus' if (day < 21) else 'gemini'  elif month == 'june':  astro\_sign = 'Gemini' if (day < 21) else 'cancer'  elif month == 'july':  astro\_sign = 'Cancer' if (day < 23) else 'leo'  elif month == 'august':  astro\_sign = 'Leo' if (day < 23) else 'virgo'  elif month == 'september':  astro\_sign = 'Virgo' if (day < 23) else 'libra'  elif month == 'october':  astro\_sign = 'Libra' if (day < 23) else 'scorpio'  elif month == 'november':  astro\_sign = 'scorpio' if (day < 22) else 'sagittarius'  print("Your Astrological sign is :",astro\_sign)  **Write a Python program to display the sign of the Chinese Zodiac for given year in which you were born.**  year = int(input("Input your birth year: "))  if (year - 2000) % 12 == 0:  sign = 'Dragon'  elif (year - 2000) % 12 == 1:  sign = 'Snake'  elif (year - 2000) % 12 == 2:  sign = 'Horse'  elif (year - 2000) % 12 == 3:  sign = 'sheep'  elif (year - 2000) % 12 == 4:  sign = 'Monkey'  elif (year - 2000) % 12 == 5:  sign = 'Rooster'  elif (year - 2000) % 12 == 6:  sign = 'Dog'  elif (year - 2000) % 12 == 7:  sign = 'Pig'  elif (year - 2000) % 12 == 8:  sign = 'Rat'  elif (year - 2000) % 12 == 9:  sign = 'Ox'  elif (year - 2000) % 12 == 10:  sign = 'Tiger'  else:  sign = 'Hare'  print("Your Zodiac sign :",sign)  **Write a Python program to find the median of three values.**  a = float(input("Input first number: "))  b = float(input("Input second number: "))  c = float(input("Input third number: "))  if a > b:  if a < c:  median = a  elif b > c:  median = b  else:  median = c  else:  if a > c:  median = a  elif b < c:  median = b  else:  median = c  print("The median is", median)  **Write a Python program to calculate the sum and average of n integer numbers (input from the user). Input 0 to finish.**  print("Input some integers to calculate their sum and average. Input 0 to exit.")  count = 0  sum = 0.0  number = 1  while number != 0:  number = int(input(""))  sum = sum + number  count += 1  if count == 0:  print("Input some numbers")  else:  print("Average and Sum of the above numbers are: ", sum / (count-1), sum)  **Write a Python program to count the elements in a list until an element is a tuple.**  num = [10,20,30,(10,20),40]  ctr = 0  for n in num:  if isinstance(n, tuple):  break  ctr += 1  print(ctr)  **Write a Python program to get the difference between a given number and 17, if the number is greater than 17 return double the absolute difference.**  def difference(n):  if n <= 17:  return 17 - n  else:  return (n - 17) \* 2  print(difference(22))  print(difference(14))  **Write a Python program to test whether a number is within 100 of 1000 or 2000.**  def near\_thousand(n):  return ((abs(1000 - n) <= 100) or (abs(2000 - n) <= 100))  print(near\_thousand(1000))  print(near\_thousand(900))  **Write a Python program to calculate the sum of three given numbers, if the values are equal then return thrice of their sum.**  def sum\_thrice(x, y, z):  sum = x + y + z    if x == y == z:  sum = sum \* 3  return sum  print(sum\_thrice(1, 2, 3))  print(sum\_thrice(3, 3, 3))  **Write a Python program to get a new string from a given string where "Is" has been added to the front. If the given string already begins with "Is" then return the string unchanged.**  def new\_string(str):  if len(str) >= 2 and str[:2] == "Is":  return str  return "Is" + str  print(new\_string("Array"))  print(new\_string("IsEmpty"))  **Write a Python program to get a string which is n (non-negative integer) copies of a given string.**  def larger\_string(str, n):  result = ""  for i in range(n):  result = result + str  return result  print(larger\_string('abc', 2))  print(larger\_string('.py', 3))  **Write a Python program to get the n (non-negative integer) copies of the first 2 characters of a given string. Return the n copies of the whole string if the length is less than 2.**  def substring\_copy(str, n):  flen = 2  if flen > len(str):  flen = len(str)  substr = str[:flen]    result = ""  for i in range(n):  result = result + substr  return result  print(substring\_copy('abcdef', 2))  print(substring\_copy('p', 3))  **Write a Python program to check whether a specified value is contained in a group of values.**  def is\_group\_member(group\_data, n):  for value in group\_data:  if n == value:  return True  return False  print(is\_group\_member([1, 5, 8, 3], 3))  print(is\_group\_member([5, 8, 3], -1))  **Write a Python program to create a histogram from a given list of integers.**  def histogram( items ):  for n in items:  output = ''  times = n  while( times > 0 ):  output += '\*'  times = times - 1  print(output)  histogram([2, 3, 6, 5])  **Write a Python program to concatenate all elements in a list into a string and return it.**  def concatenate\_list\_data(list):  result= ''  for element in list:  result += str(element)  return result  print(concatenate\_list\_data([1, 5, 12, 2]))  **Write a Python program to print out a set containing all the colors from color\_list\_1 which are not present in color\_list\_2.**  color\_list\_1 = set(["White", "Black", "Red"])  color\_list\_2 = set(["Red", "Green"])  print(color\_list\_1.difference(color\_list\_2))  **Write a Python program which will return true if the two given integer values are equal or their sum or difference is 5.**  def test\_number5(x, y):  if x == y or abs(x-y) == 5 or (x+y) == 5:  return True  else:  return False  print(test\_number5(7, 2))  print(test\_number5(3, 2))  print(test\_number5(2, 2))  **Write a Python program to add two objects if both objects are an integer type.**  def add\_numbers(a, b):  if not (isinstance(a, int) and isinstance(b, int)):  raise TypeError("Inputs must be integers")  return a + b  print(add\_numbers(10, 20))  **Write a Python program to solve (x + y) \* (x + y).**  x, y = 4, 3  result = x \* x + 2 \* x \* y + y \* y  print("({} + {}) ^ 2) = {}".format(x, y, result))  amt = 10000  int = 3.5  years = 7  future\_value = amt\*((1+(0.01\*int)) \*\* years)  print(round(future\_value,2))  **Write a Python program to compute the distance between the points (x1, y1) and (x2, y2).**  import math  p1 = [4, 0]  p2 = [6, 6]  distance = math.sqrt( ((p1[0]-p2[0])\*\*2)+((p1[1]-p2[1])\*\*2) )  print(distance)  **Write a Python program to convert height (in feet and inches) to centimeters.**  print("Input your height: ")  h\_ft = int(input("Feet: "))  h\_inch = int(input("Inches: "))  h\_inch += h\_ft \* 12  h\_cm = round(h\_inch \* 2.54, 1)  print("Your height is : %d cm." % h\_cm)  **Write a Python program to convert the distance (in feet) to inches, yards, and miles.**  d\_ft = int(input("Input distance in feet: "))  d\_inches = d\_ft \* 12  d\_yards = d\_ft / 3.0  d\_miles = d\_ft / 5280.0  print("The distance in inches is %i inches." % d\_inches)  print("The distance in yards is %.2f yards." % d\_yards)  print("The distance in miles is %.2f miles." % d\_miles)  **Write a Python program to calculate body mass index.**  height = float(input("Input your height in meters: "))  weight = float(input("Input your weight in kilogram: "))  print("Your body mass index is: ", round(weight / (height \* height), 2))  **Write a Python program to sort three integers without using conditional statements and loops.**  x = int(input("Input first number: "))  y = int(input("Input second number: "))  z = int(input("Input third number: "))  a1 = min(x, y, z)  a3 = max(x, y, z)  a2 = (x + y + z) - a1 - a3  print("Numbers in sorted order: ", a1, a2, a3)  **Write a Python program to calculate midpoints of a line.**  print('\nCalculate the midpoint of a line :')  x1 = float(input('The value of x (the first endpoint) '))  y1 = float(input('The value of y (the first endpoint) '))  x2 = float(input('The value of x (the first endpoint) '))  y2 = float(input('The value of y (the first endpoint) '))  x\_m\_point = (x1 + x2)/2  y\_m\_point = (y1 + y2)/2  print();  print("The midpoint of line is :")  print( "The midpoint's x value is: ",x\_m\_point)  print( "The midpoint's y value is: ",y\_m\_point)  print()  **Write a Python program to convert a byte string to a list of integers.**  x = b'Abc'  print()  print(list(x))  print()  **Write a Python program to check if a number is positive, negative or zero.**  num = float(input("Input a number: "))  if num > 0:  print("It is positive number")  elif num == 0:  print("It is Zero")  else:  print("It is a negative number")  **Write a Python program to add leading zeroes to a string.**  str1='122.22'  print("Original String: ",str1)  str1 = str1.ljust(8, '0')  print(str1)  str1 = str1.ljust(10, '0')  print(str1)  **Write a Python program to calculate the time runs (difference between start and current time) of a program.**  from timeit import default\_timer  def timer(n):  start = default\_timer()    for row in range(0,n):  print(row)  print(default\_timer() - start)  timer(5)  timer(15)  **Write a Python program to convert an integer to binary keep leading zeros.**  x = 12  print(format(x, '08b'))  print(format(x, '010b'))  **Write a python program to convert decimal to hexadecimal.**  x = 30  print(format(x, '02x'))  x = 4  print(format(x, '02x'))  **Write a Python function to find a distinct pair of numbers whose product is odd from a sequence of integer values.**  def odd\_product(nums):  for i in range(len(nums)):  for j in range(len(nums)):  if i != j:  product = nums[i] \* nums[j]  if product & 1:  return True  return False    dt1 = [2, 4, 6, 8]  **Write a Python function to find a distinct pair of numbers whose product is odd from a sequence of integer values.**  def odd\_product(nums):  for i in range(len(nums)):  for j in range(len(nums)):  if i != j:  product = nums[i] \* nums[j]  if product & 1:  return True  return False  dt1 = [2, 4, 6, 8]  dt2 = [1, 6, 4, 7, 8]  print(dt1, odd\_product(dt1))  print(dt2, odd\_product(dt2))  **Write a Python program to remove and print every third number from a list of numbers until the list becomes empty.**  def remove\_nums(int\_list):  #list starts with 0 index  position = 3 - 1  idx = 0  len\_list = (len(int\_list))  while len\_list>0:  idx = (position+idx)%len\_list  print(int\_list.pop(idx))  len\_list -= 1  nums = [10,20,30,40,50,60,70,80,90]  remove\_nums(nums)  **Write a Python program to find unique triplets whose three elements gives the sum of zero from an array of n integers.**  def three\_sum(nums):  result = []  nums.sort()  for i in range(len(nums)-2):  if i> 0 and nums[i] == nums[i-1]:  continue  l, r = i+1, len(nums)-1  while l < r:  s = nums[i] + nums[l] + nums[r]  if s > 0:  r -= 1  elif s < 0:  l += 1  else:  # found three sum  result.append((nums[i], nums[l], nums[r]))  # remove duplicates  while l < r and nums[l] == nums[l+1]:  l+=1  while l < r and nums[r] == nums[r-1]:  r -= 1  l += 1  r -= 1  return result  x = [1, -6, 4, 2, -1, 2, 0, -2, 0 ]  print(three\_sum(x))  **Write a Python program to add two positive integers without using the '+' operator.**  def add\_without\_plus\_operator(a, b):  while b != 0:  data = a & b  a = a ^ b  b = data << 1  return a  print(add\_without\_plus\_operator(2, 10))  print(add\_without\_plus\_operator(-20, 10))  print(add\_without\_plus\_operator(-10, -20))  **Write a Python program to check the priority of the four operators (+, -, \*, /).**  from collections import deque  import re  \_\_operators\_\_ = "+-/\*"  \_\_parenthesis\_\_ = "()"  \_\_priority\_\_ = {  '+': 0,  '-': 0,  '\*': 1,  '/': 1,  }  def test\_higher\_priority(operator1, operator2):  return \_\_priority\_\_[operator1] >= \_\_priority\_\_[operator2]  print(test\_higher\_priority('\*','-'))  print(test\_higher\_priority('+','-'))  print(test\_higher\_priority('+','\*'))  print(test\_higher\_priority('+','/'))  print(test\_higher\_priority('\*','/'))  **Write a Python program to find the value of n where n degrees of number 2 are written sequentially in a line without spaces.**  def ndegrees(num):  ans = True  n, tempn, i = 2, 2, 2  while ans:  if str(tempn) in num:  i += 1  tempn = pow(n, i)  else:  ans = False  return i-1;  print(ndegrees("2481632"))  print(ndegrees("248163264"))  **Write a Python program to find the number of zeros at the end of a factorial of a given positive number.**  Range of the number(n): (1 ≤ n ≤ 2\*109)  def factendzero(n):  x = n // 5  y = x  while x > 0:  x /= 5  y += int(x)  return y    print(factendzero(5))  print(factendzero(12))  print(factendzero(100))  **Write a Python program to create a sequence where the first four members of the sequence are equal to one, and each successive term of the sequence is equal to the sum of the four previous ones. Find the Nth member of the sequence.**  def new\_seq(n):  if n==1 or n==2 or n==3 or n==4:  return 1  return new\_seq(n-1) + new\_seq(n-2) + new\_seq(n-3) + new\_seq(n-4)  print(new\_seq(5))  print(new\_seq(6))  print(new\_seq(7))  **Write a Python program to find common divisors between two numbers in a given pair.**  def ngcd(x, y):  i=1  while(i<=x and i<=y):  if(x%i==0 and y%i == 0):  gcd=i;  i+=1  return gcd;  def num\_comm\_div(x, y):  n = ngcd(x, y)  result = 0  z = int(n\*\*0.5)  i = 1  while( i <= z ):  if(n % i == 0):  result += 2  if(i == n/i):  result-=1  i+=1  return result  print("Number of common divisors: ",num\_comm\_div(2, 4))  print("Number of common divisors: ",num\_comm\_div(2, 8))  print("Number of common divisors: ",num\_comm\_div(12, 24))  **Write a python program to find heights of the top three building in descending order from eight given buildings.**  Input:  0 ≤ height of building (integer) ≤ 10,000  print("Input the heights of eight buildings:")  l = [int(input()) for i in range(8)]  print("Heights of the top three buildings:")  l = sorted(l)  print(\*l[:4:-1], sep='\n')  **Write a Python program to check whether three given lengths (integers) of three sides form a right triangle. Print "Yes" if the given sides form a right triangle otherwise print "No".**  print("Input three integers(sides of a triangle)")  int\_num = list(map(int,input().split()))  x,y,z = sorted(int\_num)  if x\*\*2+y\*\*2==z\*\*2:  print('Yes')  else:  print('No')  **Write a Python program which solve the equation:**  **ax+by=c**  **dx+ey=f**  **Print the values of x, y where a, b, c, d, e and f are given.**  print("Input the value of a, b, c, d, e, f:")  a, b, c, d, e, f = map(float, input().split())  n = a\*e - b\*d  print("Values of x and y:")  if n != 0:  x = (c\*e - b\*f) / n  y = (a\*f - c\*d) / n  print('{:.3f} {:.3f}'.format(x+0, y+0))  **Write a Python program to compute the amount of the debt in n months. The borrowing amount is $100,000 and the loan adds 5% interest of the debt and rounds it to the nearest 1,000 above month by month.**  def round\_n(n):  if n%1000:  return (1+n//1000)\*1000  else:  return n    def compute\_debt(n):  if n==0: return 100000  return int(round\_n(compute\_debt(n-1)\*1.05))  print("Input number of months:")  result = compute\_debt(int(input()))  print("Amount of debt: ","$"+str(result).strip())  **Write a Python program which reads an integer n and find the number of combinations of a,b,c and d (0 ≤ a,b,c,d ≤ 9) where (a + b + c + d) will be equal to n.**  import itertools  print("Input the number(n):")  n=int(input())  result=0  for (i,j,k) in itertools.product(range(10),range(10),range(10)):  result+=(0<=n-(i+j+k)<=9)  print("Number of combinations:",result)  **Write a Python program to check whether a point (x,y) is in a triangle or not. There is a triangle formed by three points.**  print("Input x1,y1,x2,y2,x3,y3,xp,yp:")  x1,y1,x2,y2,x3,y3,xp,yp = map(float, input().split())  c1 = (x2-x1)\*(yp-y1)-(y2-y1)\*(xp-x1)  c2 = (x3-x2)\*(yp-y2)-(y3-y2)\*(xp-x2)  c3 = (x1-x3)\*(yp-y3)-(y1-y3)\*(xp-x3)  if (c1<0 and c2<0 and c3<0) or (c1>0 and c2>0 and c3>0):  print("The point is in the triangle.")  else:  print("The point is outside the triangle.")  **Write a Python program to test whether two lines PQ and RS are parallel. The four points are P(x1, y1), Q(x2, y2), R(x3, y3), S(x4, y4).**  print("Input x1,y1,x2,y2,x3,y3,xp,yp:")  x1, y1,x2, y2, x3, y3, x4, y4 = map(float, input().split())  print('PQ and RS are parallel.' if abs((x2 - x1)\*(y4 - y3) - (x4 - x3)\*(y2 - y1)) < 1e-10 else 'PQ and RS are not parallel')  **If you draw a straight line on a plane, the plane is divided into two regions. For example, if you pull two straight lines in parallel, you get three areas, and if you draw vertically one to the other you get 4 areas.**  **Write a Python program to create maximum number of regions obtained by drawing n given straight lines.**  while True:  print("Input number of straight lines (o to exit): ")  n=int(input())  if n<=0:  break  print("Number of regions:")  print((n\*n+n+2)//2)  **Write a Python program to find the number of combinations that satisfy p + q + r + s = n where n is a given number <= 4000 and p, q, r, s in the range of 0 to 1000.**  from collections import Counter  print("Input a positive integer: (ctrl+d to exit)")  pair\_dict = Counter()  for i in range(2001):  pair\_dict[i] = min(i, 2000 - i) + 1    while True:  try:  n = int(input())  ans = 0  for i in range(n + 1):  ans += pair\_dict[i] \* pair\_dict[n - i]  print("Number of combinations of a,b,c,d:",ans)  except EOFError:  break  **Write a Python program which adds up columns and rows of given table as shown in the following figure:**  while True:  print("Input number of rows/columns (0 to exit)")  n = int(input())  if n == 0:  break  print("Input cell value:")  x = []  for i in range(n):  x.append([int(num) for num in input().split()])  for i in range(n):  sum = 0  for j in range(n):  sum += x[i][j]  x[i].append(sum)  x.append([])  for i in range(n + 1):  sum = 0  for j in range(n):  sum += x[j][i]  x[n].append(sum)  print("Result:")  for i in range(n + 1):  for j in range(n + 1):  print('{0:>5}'.format(x[i][j]), end="")  print()  **Write a Python program to find the longest common prefix string amongst a given array of strings. Return false If there is no common prefix.**  **For Example, longest common prefix of "abcdefgh" and "abcefgh" is "abc".**  def longest\_Common\_Prefix(str1):    if not str1:  return ""  short\_str = min(str1,key=len)  for i, char in enumerate(short\_str):  for other in str1:  if other[i] != char:  return short\_str[:i]  return short\_str  print(longest\_Common\_Prefix(["abcdefgh","abcefgh"]))  print(longest\_Common\_Prefix(["Python","PHP", "Java"]))  **Write a Python program to reverse only the vowels of a given string.**  def reverse\_vowels(str1):  vowels = ""  for char in str1:  if char in "aeiouAEIOU":  vowels += char  result\_string = ""  for char in str1:  if char in "aeiouAEIOU":  result\_string += vowels[-1]  vowels = vowels[:-1]  else:  result\_string += char  return result\_string  print(reverse\_vowels("w3resource"))  print(reverse\_vowels("Python"))  print(reverse\_vowels("Perl"))  print(reverse\_vowels("USA"))  **Write a Python program to calculate the maximum profit from selling and buying values of stock. An array of numbers represent the stock prices in chronological order.**  **For example, given [8, 10, 7, 5, 7, 15], the function will return 10, since the buying value of the stock is 5 dollars and sell value is 15 dollars.**  def buy\_and\_sell(stock\_price):  max\_profit\_val, current\_max\_val = 0, 0  for price in reversed(stock\_price):  current\_max\_val = max(current\_max\_val, price)  potential\_profit = current\_max\_val - price  max\_profit\_val = max(potential\_profit, max\_profit\_val)  return max\_profit\_val  print(buy\_and\_sell([8, 10, 7, 5, 7, 15]))  print(buy\_and\_sell([1, 2, 8, 1])) #7  print(buy\_and\_sell([]))  **The price of a given stock on each day is stored in an array.**  **Write a Python program to find the maximum profit in one transaction i.e., buy one and sell one share of the stock from the given price value of the said array. You cannot sell a stock before you buy one.**  **Input (Stock price of each day): [224, 236, 247, 258, 259, 225]**  **Output: 35**  def max\_profit(stock\_price):  max\_profit\_amt = 0  for i in range(len(stock\_price)):  profit\_amt = 0  for j in range(i+1, len(stock\_price)):  profit\_amt = stock\_price[j] - stock\_price[i]  if profit\_amt > max\_profit\_amt:  max\_profit\_amt = profit\_amt  return max\_profit\_amt  print(max\_profit([224, 236, 247, 258, 259, 225]))  **Write a Python program to test whether a given number is symmetrical or not.**  **A number is symmetrical when it is equal of its reverse.**  def is\_symmetrical\_num(n):  return str(n) == str(n)[::-1]  print(is\_symmetrical\_num(121))  print(is\_symmetrical\_num(0))  print(is\_symmetrical\_num(122))  **Write a Python program to check whether a given string is an "isogram" or not.**  def check\_isogram(str1):  return len(str1) == len(set(str1.lower()))  print(check\_isogram("w3resource"))  print(check\_isogram("w3r"))  **Write a Python program to replace all but the last five characters of a given string into "\*" and returns the new masked string.**  def mask\_string(str1):  return '\*'\*(len(str1)-5) + str1[-5:]  print(mask\_string("kdi39323swe"))  **Write a Python program that accept two strings and test if the letters in the second string are present in the first string.**  def string\_letter\_check(list\_data):  return all([char in list\_data[0].lower() for char in list\_data[1].lower()])  print(string\_letter\_check(["python", "ypth"]))  print(string\_letter\_check(["python", "ypths"]))  **Write a Python program to find the middle character(s) of a given string. If the length of the string is odd return the middle character and return the middle two characters if the string length is even.**  def middle\_char(txt):  return txt[(len(txt)-1)//2:(len(txt)+2)//2]  print(middle\_char("Python"))  print(middle\_char("PHP"))  print(middle\_char("Java"))  **Write a Python program to check whether every even index contains an even number and every odd index contains odd number of a given list.**  def odd\_even\_position(nums):  return all(nums[i]%2==i%2 for i in range(len(nums)))  print(odd\_even\_position([2, 1, 4, 3, 6, 7, 6, 3]))  **Write a Python program to find the position of the second occurrence of a given string in another given string. If there is no such string return -1.**  def find\_string(txt, str1):  return txt.find(str1, txt.find(str1)+1)  print(find\_string("The quick brown fox jumps over the lazy dog", "the"))  **Write a Python program that takes three integers and check whether the last digit of first number \* the last digit of second number = the last digit of third number.**  def check\_last\_digit(x, y, z):  return str(x\*y)[-1] == str(z)[-1]  print(check\_last\_digit(12, 22, 44))  print(check\_last\_digit(145, 122, 1010))  **Write a Python program find the indices of all occurrences of a given item in a given list.**  def indices\_in\_list(nums\_list, n):  return [idx for idx, i in enumerate(nums\_list) if i == n]  print(indices\_in\_list([1,2,3,4,5,2], 2))  print(indices\_in\_list([3,1,2,3,4,5,6,3,3], 3))  **Write a Python program to reverse all the words which have even length.**  def reverse\_even(txt):  return ' '.join(i[::-1] if not len(i)%2 else i for i in txt.split())  print(reverse\_even("The quick brown fox jumps over the lazy dog"))  print(reverse\_even("Python Exercises"))  **Write a Python program to print letters from the English alphabet from a-z and A-Z.**  import string  print("Alphabet from a-z:")  for letter in string.ascii\_lowercase:  print(letter, end =" ")  print("\nAlphabet from A-Z:")  for letter in string.ascii\_uppercase:  print(letter, end =" ")  **Write a Python program to check if a given positive integer is a power of two.**  def is\_Power\_of\_two(n):  return n > 0 and (n & (n - 1)) == 0  print(is\_Power\_of\_two(4))  print(is\_Power\_of\_two(36))  print(is\_Power\_of\_two(16))  **Write a Python program to check if an integer is the power of another integer.**  def is\_Power(x, y):  while (x%y == 0):  x = x / y  return x == 1  print(is\_Power(16, 2))  print(is\_Power(12, 2))  **Write a Python program to find a missing number from a list.**  def missing\_number(num\_list):  return sum(range(num\_list[0],num\_list[-1]+1)) - sum(num\_list)  print(missing\_number([1,2,3,4,6,7,8]))  print(missing\_number([10,11,12,14,15,16,17]))  **Write a Python program to find the single element in a list where every element appears three times except for one.**  def single\_number(arr):  ones, twos = 0, 0  for x in arr:  ones, twos = (ones ^ x) & ~twos, (ones & x) | (twos & ~x)  assert twos == 0  return ones  arr1 = [5, 3, 4, 3, 5, 5, 3]  arr2 = [-1, 1, 1, -1, -1, 1, 0]  print(single\_number(arr1))  print(single\_number(arr2))  **Write a Python program to add the digits of a positive integer repeatedly until the result has a single digit.**  def add\_digits(num):  return (num - 1) % 9 + 1 if num > 0 else 0  print(add\_digits(48))  **Write a Python program to check a sequence of numbers is an arithmetic progression or not.**  def is\_arithmetic(l):  delta = l[1] - l[0]  for index in range(len(l) - 1):  if not (l[index + 1] - l[index] == delta):  return False  return True  print(is\_arithmetic([5, 7, 9, 11]))  **Write a Python program to check a sequence of numbers is a geometric progression or not.**  def is\_geometric(li):  if len(li) <= 1:  return True  # Calculate ratio  ratio = li[1]/float(li[0])  # Check the ratio of the remaining  for i in range(1, len(li)):  if li[i]/float(li[i-1]) != ratio:  return False  return True  print(is\_geometric([2, 6, 18, 54]))  **Write a Python program where you take any positive integer n, if n is even, divide it by 2 to get n / 2. If n is odd, multiply it by 3 and add 1 to obtain 3n + 1. Repeat the process until you reach 1.**  def collatz\_sequence(x):  num\_seq = [x]  if x < 1:  return []  while x > 1:  if x % 2 == 0:  x = x / 2  else:  x = 3 \* x + 1  # Added line  num\_seq.append(x)  return num\_seq  print(collatz\_sequence(12))  **Write a Python program to push all zeros to the end of a list.**  def move\_zero(num\_list):  a = [0 for i in range(num\_list.count(0))]  x = [ i for i in num\_list if i != 0]  x.extend(a)  return(x)  print(move\_zero([10,0,11,12,0,14,17]))  **Write a Python program to compute the sum of the even-valued terms in the Fibonacci sequence whose values do not exceed one million.**  cache = {}  def fiba(n):  cache[n] = cache.get(n, 0) or (n <= 1 and 1 or fiba(n-1) + fiba(n-2))  return cache[n]  n = 0  x = 0  while fiba(x) <= 1000000:  if not fiba(x) % 2: n = n + fiba(x)  x=x+1  print(n)  **Write a Python program to find the largest prime factor of a given number.**  def Largest\_Prime\_Factor(n):  prime\_factor = 1  i = 2  while i <= n / i:  if n % i == 0:  prime\_factor = i  n /= i  else:  i += 1  if prime\_factor < n:  prime\_factor = n  return prime\_factor    print(Largest\_Prime\_Factor(200))  **Write a Python program to find the largest palindrome made from the product of two 4-digit numbers.**  n = 0  for a in range(9999, 100, -1):  for b in range(a, 100, -1):  x = a \* b  if x > n:  s = str(a \* b)  if s == s[::-1]:  n = a \* b  print(n)  **Write a Python program to compute s the sum of the digits of the number 220.**  def digits\_sum():  n = 2\*\*20  ans = sum(int(c) for c in str(n))  return str(ans)  print(digits\_sum())  **Write a Python program that takes a string of numbers and letters and return string which consists of letters.**  def decode(str1):  ints = "1234567890"  num = ""  letters = ""  result\_string = ""  i = 0  while i < len(str1):  if str1[i] in ints:  num += str1[i]  else:  letters += str1[i]  i += 1  for i, char in enumerate(num):  result\_string += int(char) \* letters[i]  return result\_string    print(decode("4A3B2C1D2A")) |
| Pattern printing | 488.  489.  490.  491.  492.  493.  494.  495.  496.  497.  498.  499.  500.  501.  502.  503.  504.  505.  506.  507.  508.  509.  510.  511.  512.  513.  514.  515.  516.  517.  518.  519.  520.  521.  522.  523.  524.  525.  526.  527.  528.  529.  530.  531. | **Write a Python program to print alphabet pattern 'E'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 1 or ((row == 0 or row == 6) and (column > 1 and column < 6)) or (row == 3 and column > 1 and column < 5)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'A'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (((column == 1 or column == 5) and row != 0) or ((row == 0 or row == 3) and (column > 1 and column < 5))):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'D'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 1 or ((row == 0 or row == 6) and (column > 1 and column < 5)) or (column == 5 and row != 0 and row != 6)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'G'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if ((column == 1 and row != 0 and row != 6) or ((row == 0 or row == 6) and column > 1 and column < 5) or (row == 3 and column > 2 and column < 6) or (column == 5 and row != 0 and row != 2 and row != 6)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'L'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 1 or (row == 6 and column != 0 and column < 6)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'M'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 1 or column == 5 or (row == 2 and (column == 2 or column == 4)) or (row == 3 and column == 3)):  result\_str=result\_str+"\* "  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'O'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (((column == 1 or column == 5) and row != 0 and row != 6) or ((row == 0 or row == 6) and column > 1 and column < 5)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'P'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 1 or ((row == 0 or row == 3) and column > 0 and column < 5) or ((column == 5 or column == 1) and (row == 1 or row == 2))):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'R'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 1 or ((row == 0 or row == 3) and column > 1 and column < 5) or (column == 5 and row != 0 and row < 3) or (column == row - 1 and row > 2)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print the following pattern 'S'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (((row == 0 or row == 3 or row == 6) and column > 1 and column < 5) or (column == 1 and (row == 1 or row == 2 or row == 6)) or (column == 5 and (row == 0 or row == 4 or row == 5))):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'T'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (column == 3 or (row == 0 and column > 0 and column <6)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'U'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (((column == 1 or column == 5) and row != 6) or (row == 6 and column > 1 and column < 5)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'X'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (((column == 1 or column == 5) and (row > 4 or row < 2)) or row == column and column > 0 and column < 6 or (column == 2 and row == 4) or (column == 4 and row == 2)):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print alphabet pattern 'Z'.**  result\_str=""  for row in range(0,7):  for column in range(0,7):  if (((row == 0 or row == 6) and column >= 0 and column <= 6) or row+column==6):  result\_str=result\_str+"\*"  else:  result\_str=result\_str+" "  result\_str=result\_str+"\n"  print(result\_str)  **Write a Python program to print given pattern.**  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  def starPattern1\_A():  n=int(input("enter a number\n"))  for i in range(0,n+1):  for j in range(0,n+1):  if i>j:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  starPattern1\_A()  **Write a Python program to print given pattern.**  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \*  def starPattern2\_A():  n=int(input("enter a number\n"))  for i in range(0,n+1):  for j in range(0,n+1):  if i<j:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  starPattern2\_A()  **Write a Python program to print given pattern.**  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \*  def starPattern3\_A():  n=int(input("enter a number\n"))  for i in range(0,n):  for j in range(0,n):  if i+j<n:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  starPattern3\_A()  **Write a Python program to print given pattern.**  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  def starPattern4\_A():  n=int(input("enter a number\n"))  for i in range(0,n+1):  for j in range(0,n+1):  if i+j>n:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  starPattern4\_A()  **Write a Python program to print given pattern.**  \*  \* \* \*  \* \* \* \* \*  \* \* \* \* \* \* \*  def pattern5\_1(n):  for i in range(n):  for j in range(n):  if i>=j and i+j>=n-1:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern5\_1(7)  **Write a Python program to print given pattern.**  \* \* \* \* \* \* \*  \* \* \* \* \*  \* \* \*  \*  def pattern6(n):  for i in range(n):  for j in range(n):  if i<=j and i+j<=n-1:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern6(7)  **Write a Python program to print given pattern.**  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \*  \* \*  \*  def pattern7(n):  for i in range(n):  for j in range(n):  if i>=j and i+j<=n-1:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern7(7)  **Write a Python program to print given pattern.**  \* \*  \* \* \* \*  \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \*  \* \* \* \*  \* \*  def pattern9(n):  for i in range(n):  for j in range(n):  if (i>=j and i+j<=n-1) or (i<=j and i+j>=n-1):  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern9(7)  **Write a Python program to print given pattern.**  \* \* \* \* \* \* \*  \* \* \* \* \*  \* \* \*  \*  \* \* \*  \* \* \* \* \*  \* \* \* \* \* \* \*  def pattern10(n):  for i in range(n):  for j in range(n):  if (i<=j and i+j<=n-1) or (i>=j and i+j>=n-1):  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern10(7)  **Write a Python program to print given pattern.**  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \*  \* \*  \*  def pattern11(n):  for i in range(n):  for j in range(n):  if i+j>=n-1 and i<=j:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern11(7)  **Write a Python program to print given pattern.**  **1 2 3 4 5 6**  **2 3 4 5 6**  **3 4 5 6**  **4 5 6**  **5 6**  **6**  def pattern1(n):  for i in range(1,n+1):  for j in range(1,n+1):  if i<=j:  print(j,end=" ")  else:  print(" ",end=" ")  print()  pattern1(6)  **Write a Python program to print given pattern.**  **1**  **1 2**  **1 2 3**  **1 2 3 4**  **1 2 3 4 5**  **1 2 3 4 5 6**  def pattern2(n):  for i in range(n):  for j in range(n):  if i>=j:  print(j+1,end=" ")  print()  pattern2(6)  **Write a Python program to print given pattern.**  **1 2 3 4 5**  **1 2 3 4**  **1 2 3**  **1 2**  **1**  def pattern3(n):  for i in range(n):  for j in range(n):  if i+j<n-1:  print(j+1,end=" ")  print()  pattern3(6)  **Write a Python program to print given pattern.**  **6**  **5 6**  **4 5 6**  **3 4 5 6**  **2 3 4 5 6**  def pattern4(n):  for i in range(n):  for j in range(n):  if i+j>n-1:  print(j+1,end=" ")  print()  pattern4(6)  **Write a Python program to print given pattern.**  **0000000**  **0111110**  **0111110**  **0110110**  **0111110**  **0111110**  **0000000**  def pattern(n):  for i in range(n):  k=1  for j in range(n):  if i==0 or i==n-1 or j==0 or j==n-1 or (i==(n-1)/2 and j==(n-1)/2) :  print("0",end="")  #k=k+1  else:  print("1",end="")  print()  print()  pattern(7)  **Write a Python program to print given pattern.**  **1234567**  **1 2**  **1 2**  **1 2 3**  **1 2**  **1 2**  **1234567**  def pattern1(n):  for i in range(n):  k=1  for j in range(n):  if i==0 or i==n-1 or j==0 or j==n-1 or (i==(n-1)/2 and j==(n-1)/2) :  print(k,end="")  k=k+1  else:  print(" ",end="")  print()  print()  pattern1(7)  **Write a Python program to print given pattern.**  **1**  **1 2 3**  **1 2 3 4 5**  **1 2 3 4 5 6 7**  def numberPattern1(n):  for i in range(n):  k=1  for j in range(n):  if i>=j and i+j>=n-1:  print(k,end=" ")  k=k+1  else:  print(" ",end=" ")  print()  print()  numberPattern1(7)  **Write a Python program to print given pattern.**  **6**  **6 5 4**  **6 5 4 3 2**  **6 5 4 3 2 1 0**  def numberPattern2(n):  for i in range(n):  k=6  for j in range(n):  if i>=j and i+j>=n-1:  print(k,end=" ")  k=k-1  else:  print(" ",end=" ")  print()  print()  numberPattern2(7)  **Write a Python program to print given pattern.**  **1 2 3 4 5 6 7**  **1 2 3 4 5**  **1 2 3**  **1**  def numberPattern3(n):  for i in range(n):  k=1  for j in range(n):  if (i<=j and i+j<=n-1):  print(k,end=" ")  k=k+1  else:  print(" ",end=" ")  print()  print()  numberPattern3(7)  **Write a Python program to print given pattern.**  **1 2 3 4 3 2 1**  **1 2 3 2 1**  **1 2 1**  **1**  def numberPattern4(n):  for i in range(n):  k=1  for j in range(n):  if (i<=j and i+j<=n-1):  if j<(n-1)/2:  print(k,end=" ")  k=k+1  else:  print(k,end=" ")  k=k-1  else:  print(" ",end=" ")  print()  print()  numberPattern4(7)  **Write a Python program to print given pattern.**  **1 2 1 2 1 2 1**  **1 0 1 0 1**  **1 2 1**  **1**  def numberPattern5(n):  for i in range(n):  k=1  for j in range(n):  if (i<=j and i+j<=n-1):  if j%2==0:  print(k,end=" ")  k=k+1  else:  print(k,end=" ")  k=k-1  else:  print(" ",end=" ")  print()  print()  numberPattern5(7)  **Write a Python program to print given pattern.**  \*  \* \* \*  \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \*  \* \* \*  \*  n=8  st=2  sp=4  for i in range(1,n):  for j in range(1,sp):  print(' ',end=" ")  for k in range(1,st):  print("\*",end=' ')  print()  if i<n/2:  sp=sp-1  st+=2  else:  sp=sp+1  st-=2  **Write a Python program to print given pattern.**  **1**  **2 5**  **3 6 8**  **4 7 9 10**  def func():  n=5  for i in range(1,n):  num=i  for j in range(1,i+1):  print(num,end=" ")  num=num+n-j-1  print(end=" ")  print()  func()  **Write a Python program to print given pattern.**  **A**  **A B A**  **A B C B A**  **A B C D C B A**  **A B C D E D C B A**  n = 5  for i in range(1,n+1):  ch=ord("A")  for j in range(1,n+1):  if i+j>=n+1:  print(chr(ch),end=" ")  ch+=1  else:  print(" ",end=" ")  ch-=2  for j in range(1,n+1):  if i>j:  print(chr(ch),end=" ")  ch-=1  else:  print(" ",end=" ")  print()  **Write a Python program to print given pattern.**  **A**  **B C D**  **E F G H I**  **J K L M N O P**  **Q R S T U V W X Y**  n=5  ch = ord("A")  for i in range(1,n+1):  for j in range(1,n+1):  if i+j>=n+1:  print(chr(ch), end=" ")  ch+=1  else:  print(" ",end=" ")  for j in range(1,n+1):  if i>j:  print(chr(ch), end=" ")  ch+=1  else:  print(" ",end="")  print()  **Write a Python program to print given pattern.**  \*  \*  \*  \*  \*  \*  \*  \*  \*  \*  \*  n=int(input("enter number of rows: \n"))  for i in range(0,n):  for j in range(0,n):  if i == j:  print("\*",end = "")  else:  print(' ',end = "")  print()  for i in range(0,n):  for j in range(0,n):  if i+j == n-1:  print("\*",end = "")  else:  print(' ',end = "")  print()  **Write a Python program to print given pattern.**  \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \*  \* \* \* \*  \* \*  def pattern5(n):  for i in range(n):  for j in range(n):  if i<=j or i+j<=n-1:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern5(7)  **Write a Python program to print given pattern.**  \* \*  \* \* \* \*  \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  \* \* \* \* \* \* \*  def pattern61(n):  for i in range(n):  for j in range(n):  if i>=j or i+j>=n-1:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern61(7)  **Write a Python program to print given pattern.**  \* \* \* \* \* \* \*  \* \* \* \* \* \*  \* \* \* \* \*  \* \* \* \*  \* \* \* \* \*  \* \* \* \* \* \*  \* \* \* \* \* \* \*  def pattern8(n):  for i in range(n):  for j in range(n):  if i>=j or i+j<=n-1:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern8(7)  **Write a Python program to print given pattern.**  \* \* \* \* \* \* \*  \* \* \* \* \* \*  \* \* \* \* \*  \* \* \* \*  \* \* \* \* \*  \* \* \* \* \* \*  \* \* \* \* \* \* \*  def pattern12(n):  for i in range(n):  for j in range(n):  if i+j>=n-1 or i<=j:  print("\*",end=" ")  else:  print(" ",end=" ")  print()  print()  pattern12 (7) |
| File Handling | 532.  533.  534.  535.  536.  537.  538.    539.  540.  541.  542.  543.  544.  545. | **Write a Python Program to Count the Number of Words in a Text File**  fname = input("Enter file name: ")  num\_words = 0  with open(fname, 'r') as f:  for line in f:  words = line.split()  num\_words += len(words)  print("Number of words:")  print(num\_words)  **Write a Python Program to Count the Number of Lines in a Text File**  fname = input("Enter file name: ")  num\_lines = 0  with open(fname, 'r') as f:  for line in f:  num\_lines += 1  print("Number of lines:")  print(num\_lines)  **Write a Python Program to Read a Text File and Print all the Numbers Present in the Text File.**  fname = input("Enter file name: ")  with open(fname, 'r') as f:  for line in f:  words = line.split()  for i in words:  for letter in i:  if (letter.isdigit()):  print(letter)  **Write a Python Program to Read the Contents of a File in Reverse Order**  filename=input("Enter file name: ")  for line in reversed(list(open(filename))):  print(line.rstrip())  **Write a Python Program to Count the Number of Blank Spaces in a Text File**  fname = input("Enter file name: ")  k = 0  with open(fname, 'r') as f:  for line in f:  words = line.split()  for i in words:  for letter in i:  if(letter.isspace):  k=k+1  print("Occurrences of blank spaces:")  print(k)  **Write a Python Program that Reads a Text File and Counts the Number of Times a Certain Letter Appears in the Text File**  fname = input("Enter file name: ")  l=input("Enter letter to be searched:")  k = 0  with open(fname, 'r') as f:  for line in f:  words = line.split()  for i in words:  for letter in i:  if(letter==l):  k=k+1  print("Occurrences of the letter:")  print(k)  **Write a Python Program to Append the Contents of One File to Another File**  name1 = input("Enter file to be read from: ")  name2 = input("Enter file to be appended to: ")  fin = open(name1, "r")  data2 = fin.read()  fin.close()  fout = open(name2, "a")  fout.write(data2)  fout.close()  **Write a Python program to copy odd lines of one file to other**  fn = open('bcd.txt', 'r')  fn1 = open('nfile.txt', 'w')  cont = fn.readlines()  type(cont)  for i in range(0, len(cont)):  if(i % 2 ! = 0):  fn1.write(cont[i])  else:  pass  fn1.close()  fn1 = open('nfile.txt', 'r')  cont1 = fn1.read()    print(cont1)  fn.close()  fn1.close()  **Write a function called showNumbers that takes a parameter called limit. It should print all the numbers between 0 and limit with a label to identify the even and odd numbers. For example, if the limit is 3, it should print:**  **0 EVEN**  **1 ODD**  **2 EVEN**  **3 ODD**  def shownumber(st=0,end=None):  if end==0:  end=10  for i in range(st,end+1):  if i%2==0:  print(i,"even")  else:  print(i,"odd")  shownumber(0,20)  **Python Program for Find sum of odd factors of a number.**  def odd\_factor(num):  sum\_of\_factor=0  for i in range(1,num+1):  if num%i==0:  sum\_of\_factor+=i  return oddlist  print(odd\_factor(120))  **Write Python program to implement the operation of ATM**  **1.Withdraw**  **2.Deposite**  **3.Balance**  **4. change PIN**  import json as j  import random as r  class ATM():  def verify(self):  fileobj = open("customer\_details.json", 'r')  c\_data = j.load(fileobj)  fileobj.close()  c\_name=input("Enter the Username\n")  if c\_name in c\_data:  pin=int(input("Enter the PIN \n"))  if pin==c\_data[c\_name]["PIN"]:  self.choice(c\_name)  else:  print("Incorrect PIN")  return  else:  print ("user not found")  return  def withdraw(self,name):  fileobj = open("customer\_details.json", 'r')  c\_data = j.load(fileobj)  fileobj.close()  withdarw\_ammount=int(input("Enter the Ammount to withdraw\n"))  if c\_data[name]["Balance"]-withdarw\_ammount>0:  c\_data[name]["Balance"]=c\_data[name]["Balance"]-withdarw\_ammount  fileobj=open("customer\_details.json",'w')  j.dump(c\_data,fileobj,indent=4)  fileobj.close()  self.balance(name)  return  else:  print ("Insufficent Ammount")  return  def deposit(self,name):  fileobj = open("customer\_details.json", 'r')  c\_data = j.load(fileobj)  fileobj.close()  deposite\_ammount = int(input("Enter the Ammount to deposite\n"))  c\_data[name]["Balance"] = c\_data[name]["Balance"] + deposite\_ammount  fileobj = open("customer\_details.json", 'w')  j.dump(c\_data, fileobj,indent=4)  fileobj.close()  self.balance(name)  return  def balance(self,name):  fileobj = open("customer\_details.json", 'r')  c\_data = j.load(fileobj)  fileobj.close()  print("Balance Ammount : ",c\_data[name]["Balance"])  return  def changePIN(self,name):  fileobj = open("customer\_details.json", 'r')  c\_data = j.load(fileobj)  fileobj.close()  listdata = [1234, 2341, 4321, 9871, 8712, 8421, 8234, 1987, 2876, 3456, 4455, 3362, 6371, 7371, 9389, 8829,  2536, 8236, 4368, 9371]  new\_pin=r.sample(listdata, k=1)  c\_data[name]["PIN"]=new\_pin[0]  print("Your New PIN : ",c\_data[name]["PIN"])  fileobj=open("customer\_details.json",'w')  j.dump(c\_data,fileobj,indent=4)  fileobj.close()  return  def choice(self,name):  print ("Select the choice\n 1.Withdraw\n 2.Deposit\n 3.Check Balance \n 4.Change PIN")  choice=int(input("Enter the choice \n"))  if choice==1:  self.withdraw(name)  elif choice==2:  self.deposit(name)  elif choice==3:  self.balance(name)  elif choice==4:  self.changePIN(name)  else:  return  c1=ATM()  c1.verify()  **Write Python program to implement the operation of online Food Order Management**  import json as j  def adduser(valid):  fileobj = open("userdata1.json", 'r')  userdata = j.load(fileobj)  fileobj.close()  print("Username doesnot Exists")  choice = int(input("To create a new User select 1\n"))  if choice == 1:  username = input("Enter the Username\n")  userdata.setdefault(username, {})  listdata=["password","address","phone\_no"]  userdata[username]=userdata[username].fromkeys(listdata)  for i in userdata[username]:  print("Enter the ",i)  userdata[username][i]=input()  fileobj = open("userdata1.json", "w")  j.dump(userdata, fileobj,indent=4)  fileobj.close()  print("User Successfully Created")  valid(intput("Enter the username"))  else:  print("Thank you")  return  def bill(username,orderedfood):  foodobj = open("fooditems.json", "r")  fooditems = j.load(foodobj)  foodobj.close()  totalammount=0  billdetails={}  billdetails["Name"]=username  billdetails.setdefault("fooditems",orderedfood)  for i in orderedfood:  ammount=orderedfood[i]\*fooditems[i]  totalammount+=ammount  billdetails["Total\_ammount"]=totalammount  print(billdetails)  with open(username+".json",'w') as fob:  j.dump(billdetails,fob)  def takeorder(username):  totalfooditem=int(input("enter number food items\n"))  orderedfood={}  for i in range(totalfooditem):  food=input("enter the food from the list\n")  quantity=int(input("enter the no quantity\n"))  orderedfood[food]=quantity  bill(username,orderedfood)  def displayfood(username):  foodobj=open("fooditems.json", "r")  fooditems=j.load(foodobj)  foodobj.close()  for i in fooditems:  print(i,"- Rs",fooditems[i])  takeorder(username)  def valid(username):  fileobj = open("userdata1.json", 'r')  userdict = j.load(fileobj)  fileobj.close()  if username in userdict:  password = input("Enter the password\n")  if password == userdict[username]["password"]:  print("welcome", username,)  displayfood(username)  else:  print("Incorrect password\n")  else:  adduser(valid)  valid(input("Enter the Username\n"))  **Write a Python program to validate the login user**  def valid():  import json as j  fileobj=open("userdata1.json","r")  userdict=j.load(fileobj)  fileobj.close()  username=input("enter the user name\n")  if username in userdict:  password=input("enter the password\n")  if password==userdict[username]["password"]:  print("welcome",username)  else:  print ("entered password is incorrect")  else:  print("Entered username does not exists")  choice=int(input("Do you want to create an user if yes select 1\n"))  if choice==1:  fileobj = open("userdata1.json", 'w')  username=input("enter a Username\n" )  userdict.setdefault(username,{})  password=input("Enter the password\n")  userdict[username]["password"]=password  address=input("Enter the address\n")  userdict[username]["address"]=address  phno=int(input("Enter the Phone no\n"))  userdict[username]["phone\_no"]=phno  print(userdict)  j.dump(userdict,fileobj,indent=4)  fileobj.close()  valid()  else:  return  valid()  **Write a Python program to print the Prime numbers in the given boundry.**  def generateprime(start,end):  primelist=[]  for i in range (start,end):  x=0  for j in range (1,end+1):  if i%j==0:  x+=1  if x<=2 and x>0:  primelist.append(i)  print (primelist)  generateprime(1,20) |