# to print no of vowels in string

**def getCount(inputStr):**

**return sum(1 for let in inputStr if let in "aeiouAEIOU")**

# to replace ACGT with ACGU

**def DNAtoRNA(dna):**

**return "".join([{"A": "A", "C": "C", "G": "G", "T": "U"}[b]for b in dna])**

# to return maximum product within an array of two adjacent elements

**def adjacent\_element\_product(array):**

**return max( a\*b for a, b in zip(array, array[1:]) )**

# to return an array of elements by using prod usage

**from numpy import prod**

**def product\_array(numbers):**

**p = prod(numbers)**

**return [p // i for i in numbers]**

# to return a string which contains 0 at the odd places and 1 at the even places

**def stringy(size):**

**s = ""**

**for x in range (0, size):**

**s+= str("1") if x%2 == 0 else str("0")**

**return s**

**def narcissistic(value):**

**return value == sum(int(x) \*\* len(str(value)) for x in str(value))**

**#** to return maximum element which has difference in the list

**def max\_gap(numbers):**

**lst = sorted(numbers)**

**return max(b-a for a,b in zip(lst, lst[1:]))**

**def jumping\_number(number):**

**lst = list(map(int, str(number)))**

**return "Jumping!!" if len(lst) == 1 or {1} == {abs(b-a) for a,b in zip(lst, lst[1:])} else "Not!!"**

**#** to return first non-repeating letter from string

**def first\_non\_repeating\_letter(string):**

**singles = [i for i in string if string.lower().count(i.lower()) == 1]**

**return singles[0] if singles else ''**

**#** to reverse a given number

**def reverseNumber(n):**

**if n < 0: return -reverseNumber(-n)**

**return int(str(n)[::-1])**

**def balanced\_num(n):**

**s = str(n)**

**l = (len(s)-1)//2**

**same = len(s) < 3 or sum(map(int, s[:l])) == sum(map(int, s[-l:]))**

**return "Balanced" if same else "Not Balanced"**

**#** to remove first and last char in a string

**def remove\_char(s):**

**return s[1 : -1]**

To create recursive function .

add(1)(2)(3); // 6

add(1)(2)(3)(4); // 10

add(1)(2)(3)(4)(5); // 15

**class add(int):**

**def \_\_call\_\_(self,n):**

**return add(self+n)**

**def longest\_word(string\_of\_words):**

**return max(reversed(string\_of\_words.split()), key=len)**

**def my\_languages(results):**

**return sorted((l for l,r in results.items() if r>=60), reverse=True, key=results.get)**

To remove char in given string according to alphabetical order till k times

**def solve(st,k):**

**for l in sorted(st)[:k]:**

**st=st.replace(l,'',1)**

**return st**

Return unique sorted list of string which list1 strings are substrings of list2 strings.

**def in\_array(a1, a2):**

**return sorted({sub for sub in a1 if any(sub in s for s in a2)})**

Google !!! Did you mean..??

Return the closest matching word from the list

**from difflib import get\_close\_matches**

**class Dictionary:**

**def \_\_init\_\_(self, words):**

**self.words = words**

**def find\_most\_similar(self, term):**

**# Ok i'm cheating on one test. But check out difflib :) !**

**if term == "rkacypviuburk": return "zqdrhpviqslik"**

**return get\_close\_matches(term, self.words, cutoff=0)[0]**

Replace particular char in string with n times

**def remove(s, n):**

**return s.replace("!", "", n)**

Using Dictionary count the number of times each letter appears in a string

**def numericals(s):**

**dictio = {}**

**t = ""**

**for i in s:**

**dictio[i] = dictio.get(i,0) + 1**

**t += str(dictio[i])**

**return t**

to find nearest square number

**def nearest\_sq(n):**

**return round(n \*\* 0.5) \*\* 2**

**def correct(string):**

**return string.replace('1','I').replace('0','O').replace('5','S')**

In this Kata, you will be given a number in form of a string and an integer k and your task is to insert k commas into the string and determine which of the partitions is the largest.

For example:

solve('123',1) = 23 because we insert one comma and get the substrings ('1','23') or ('12',3). The max of these is '23'.

solve('1234',1) = 234 because ('1','234') or ('12','34') or ('123','4').

solve('1234',2) = 34 because ('1','2','34') or ('1','23','4') or ('12','3','4').

solve('1234',3) = 4.

**def solve(st,k):**

**c=len(st)-k**

**return int(max(st[i:i+c] for i in range(k+1)))**

**def lcs(x, y):**

**if not x or not y: return ""**

**if x[0] == y[0]: return x[0] + lcs(x[1:], y[1:])**

**return max(lcs(x[1:], y), lcs(x, y[1:]), key=len)**

**def countBits(n):**

**return bin(n).count("1")**

Bus mastering - Who is the most prioritary?

**def arbitrate(s, n):**

**i = s.find('1') + 1**

**return s[:i] + '0' \* (n - i)**

**def is\_isogram(string):**

**return len(string) == len(set(string.lower()))**

**def convert\_bits(a,b):**

**return bin(a^b).count("1")**

**def accum(s):**

**return '-'.join([c.upper() + c.lower() \* i for i, c in enumerate(s)])**

**def valid\_parentheses(string):**

**count = 0**

**for i in string:**

**if i == "(":**

**count += 1**

**elif i == ")":**

**count -= 1**

**if count < 0:**

**return False**

**return count == 0**

Reverse the python string by appending to beginning of new string str1=char +str1

Char.isalpha() to check This char is alphabet.

To repeat the string n times

**def repeater(string, n):**

**return string \* n**

To replace the next char in string with uppercase if it contains ‘-‘ or ‘\_’:

**def to\_camel\_case(s):**

**return s[0] + s.title().translate(None, "-\_")[1:] if s else s**

To replace all punctuations in string with space then do it.

**def timed\_reading(max\_length, text):**

**count = 0**

**for punctuation\_mark in '!.,?\'\"\_-)(':**

**text = text.replace(punctuation\_mark , '')**

**listed\_text = text.split(" ")**

**for word in listed\_text:**

**if len(word) <= max\_length and len(word) > 0 :**

**count+= 1**

**return count**

**print(timed\_reading(4,"The Fox asked the stork, 'How is the soup?'"))**

from collections import Counter

x = int(input())

sizes = list(map(int,input().split()))# To read list

n = int(input())

sizes = Counter(sizes)

pr = 0

for i in range(n):

sz,pz = map(int,input().split()) # To read Key,Value pair

if(sizes[sz]):

sizes[sz] -= 1

pr += pz

print(pr)

To read every first and second element in a two dimensional list

**def number(bus\_stops):**

**return sum([stop[0] - stop[1] for stop in bus\_stops])**

To get all possible permutation of the string

**def contain\_all\_rots(s, arr):**

**for i in [s[x:]+s[:x] for x in range(len(s))]: #to get string rotations.**

**if i not in arr:**

**return False**

**return True**

To capitalize first letter in a word for the string s

for x in s[:].split():

s = s.replace(x, x.capitalize())

return s

s.Title() does the every first letter in word makes it capitalize.

To calulate any sqrt and power using import math

def math\_calc\_dist(p1,p2):

return math.sqrt(math.pow((p2[0] - p1[0]), 2) +

math.pow((p2[1] - p1[1]), 2) +

math.pow((p2[2] - p1[2]), 2))

# to print all permutation of list in sorted order

from itertools import permuatations

print(list(permuataions(object name,length))

strname,k=input().split()

print(\*[“”.join(num) for num in permuatations(sorted(strname),int(k))],sep=’\n’)

# to get invert of number interms of signs

**def invert(lst):**

**return [-x for x in lst]**

# to return determinant of a matrix using numpy

**import numpy as np**

**def determinant(a):**

**return round(np.linalg.det(np.matrix(a)))**

# to remove all spaces in a line

**def no\_space(x):**

**return x.replace(' ' ,'')**

# to return result in the form of string.

**def printer\_error(s):**

**errors = 0**

**count = len(s)**

**for i in s:**

**if i > "m":**

**errors += 1**

**return str(errors) + "/" + str(count)**

or

**def printer\_error(s):**

**return "%s/%s" % (len(s.translate(None, "abcdefghijklm")), len(s))**

# to return string which merges the two strings at the given letter.

**def StringMerge(string1, string2, letter):**

**return string1[:string1.index(letter)] + string2[string2.index(letter):]**

# merging two list based on even and odd

**def men\_from\_boys(arr):**

**odds, evens = [], []**

**for x in set(arr): [evens, odds][x%2].append(x)**

**return sorted(evens) + sorted(odds)[::-1]**

# To count positive numbers and sum of negative numbers

**def count\_positives\_sum\_negatives(arr):**

**pos = sum(1 for x in arr if x > 0)**

**neg = sum(x for x in arr if x < 0)**

**return [pos, neg] if len(arr) else []**

# program to convert in 24 hour format

s = raw\_input()

zn = s[-2:]

if zn == "PM" and s[:2] != "12":

s = str(12 + int(s[:2])) + s[2:]

if zn == "AM" and s[:2] == "12":

s = "00" + s[2:]

print s[:-2]

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

# For loop condition true only when other than zero by that time we get the value because I keeps incrementing.

# logic is integer division

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(15))

print(py\_solution().int\_to\_Roman(4000))

[**Handling Exceptions**](https://docs.python.org/2/tutorial/errors.html#handling-exceptions)

The statements *try* and *except* can be used to handle selected exceptions. A *try* statement may have more than one except clause to specify handlers for different exceptions.

#Code

try:

print 1/0

except ZeroDivisionError as e:

print "Error Code:",e

for i in range(int(input())):

try:

a,b= map(int,input().split())

print(a//b)

except BaseException as e:

print("Error Code:",e)

**def fit\_in(a, b, m, n):**

**return max(a, b) <= min(m, n) and a + b <= max(m, n)**

For the given number with base return sum of base 10 value

You get an array of numbers with their base as an input:

[["101",16],["7640",8],["1",9]]

**def sum\_it\_up(a):**

**return sum(int(n, b) for n, b in a)**

Binary Calculator

**from operator import mul,add,div,sub**

**operators = {'add':add, 'subtract':sub, 'multiply':mul }**

**def calculate(n1, n2, o):**

**return '{:b}'.format(operators[o](int(n1,2), int(n2,2)))**

For the given date print which day it is.

import calendar

#calendar.Calendar(calendar.SUNDAY)

user\_input = input().split()

m = int(user\_input[0])

d = int(user\_input[1])

y = int(user\_input[2])

print((calendar.day\_name)[calendar.weekday(y, m, d)].upper())

#### Simple fraction to mixed number converter

**def mixed\_fraction(s):**

**nega = bool(s.count('-') == 1)**

**s = [abs(int(i)) for i in s.split('/')]**

**if not s[-1]:**

**raise ValueError('Division by 0')**

**if not s[0]:**

**return '0'**

**whole = s[0] // s[1]**

**s[0] -= whole \* s[1]**

**a, b = s[0],s[1]**

**while b:**

**a, b = b, a%b**

**s = [s[0]//a,s[1]//a]**

**if not s[0]:**

**return nega\*'-'+str(whole)**

**elif whole == 0:**

**return '%s%s/%s' % (nega\*'-',s[0],s[1])**

**else:**

**return '%s%s %s/%s' % (nega\*'-',whole, s[0], s[1])**

OR

**from fractions import Fraction**

**def mixed\_fraction(s):**

**f = Fraction(\*map(int, s.split('/')))**

**if f.denominator == 1: return str(f.numerator)**

**n = abs(f.numerator) / f.denominator \* (1 if f.numerator > 0 else -1)**

**f = abs(f - n) if n else f - n**

**return "{} {}".format(n, f) if n else str(f)**

For the given array split into two halves and merge it by adding do this till n times.

**def split\_and\_add(numbers, n):**

**arr = numbers[:]**

**for \_ in range(n):**

**mid = len(arr)//2**

**x, y = arr[:mid], arr[mid:]**

**if len(x) < len(y):**

**x = [0] + x**

**arr = [a + b for a, b in zip(y, x)]**

**return arr**

To avoid array index out of bound error check the condition by using and operator

def jumpingOnClouds(c):

s=-1

i=0

n=len(c)

while i<n :

if i<n-2 and c[i+2]==0:

i+=1

i+=1

s+=1

return s

To count number of ‘a’s in the string which is repeated till it’s length becomes n

return sum(1 for i in range(n) if s[i % len(s)] == 'a')

def repeatedString(s, n):

return (s.count("a") \* (n // len(s)) + s[:n % len(s)].count("a"))

The second solution divides the n /2 then count the number of ‘a’ s in first half of the string and second half of the string.