SAPIENZA
UNIVERSITÀ DI ROMA

ADM - Homework 1

First semester 2021-2022

Valentin GUILHEM

Say "Hello, World!" With Python

```
print("Hello, World!")
```

Python If-Else

```
#!/bin/python3

import math
import os
import random
import re
import sys



if __name__ == '__main__':
    n = int(input().strip())

    if n == 0 :
        print("Weird")

    if n%2 == 1 :
        print("Weird")
    else :
        if 2 <= n <= 5 :
            print("Not Weird")
        if 6 <= n <= 20 :
            print("Weird")
        if n > 20 :
           print("Not Weird")
```

Arithmetic Operators

```
if __name__ == '__main__':
    a = int(input())
    b = int(input())

    print(a+b)
    print(a-b)
    print(a*b)
```

Python: Division

```
if __name__ == '__main__':
    a = int(input())
    b = int(input())

    print(a//b)
    print(a/b)
```

sWAP cASE

```
def swap_case(s):
    return s.swapcase()
```

What's Your Name?

```
def print_full_name(first, last):
    return (print("Hello",first,last + "!
You just delved into python."))
```

Find a string

```
def count_substring(string, sub_string):
    num = 0
    for i in range(len(string)):
        if string[i:i+len(sub_string)] ==
sub_string :
            num +=1
    return num
```

Text Alignment

```
thickness = int(input()) #This must be an
odd number
c = 'H'

#Top Cone
for i in range(thickness):
```

```
print((c*i).rjust(thickness-1)+c+(c*i).lj
ust(thickness-1))

#Top Pillars
for i in range(thickness+1):

print((c*thickness).center(thickness*2)+(
c*thickness).center(thickness*6))

#Middle Belt
for i in range((thickness+1)//2):

print((c*thickness*5).center(thickness*6)
)

#Bottom Pillars
for i in range(thickness+1):

print((c*thickness).center(thickness*2)+(
c*thickness).center(thickness*6))

#Bottom Cone
for i in range(thickness):

print(((c*(thickness-i-1)).rjust(thicknes
s)+c+(c*(thickness-i-1)).ljust(thickness)
```

```
).rjust(thickness*6))
```

Text Wrap

```
def wrap(string, max_width):
    new_str = ""
    word_list = textwrap.wrap(string,
max_width)
    for i in word_list :
        new_str += i + '\n'
    return new_str
```

String Validators

```
if __name__ == '__main__':
    s = input()

print (any(char.isalnum() for char in s))
print (any(char.isalpha() for char in s))
print (any(char.isdigit() for char in s))
print (any(char.islower() for char in s))
print (any(char.isupper() for char in s))
```

String Split and Join

```
def split_and_join(line):
    # write your code here
    line = line.split(" ")
    line = "-".join(line)
    return line
```

Designer Door Mat

```
if __name__ == '__main__':
        N, M = map(int, input().split("
"))

for i in range(N) :
    symbol = ".|."
    if i < (N-1)/2 :
        print((symbol*(2*i+1)).center(M,
"-"))
    elif i == (N-1)/2 :
        print("WELCOME".center(M, "-"))
    else :

print((symbol*(2*(N-1-i)+1)).center(M,
"-"))
```

Alphabet Rangoli

```
def print_rangoli(size):
    alp = 'abcdefghijklmnopqrstuvwxyz'
    res = ""
    for i in range (0,n,-1):
        res += alp[i]
        print(res)
    return res
```

Write a function

```
def is_leap(year):
    leap = False
    if year%4 == 0:
        leap = True
    if year%4 == 0 and year%100 == 0 and
not year%400 == 0 :
        leap = False
    if year%4 == 0 and year%100 == 0 and
year%400 == 0 :
        leap = True
    return leap
```

Symmetric Difference

```
a=int(input())
M=set(map(int,input().split()))
b=int(input())
N=set(map(int,input().split()))

C=(M.difference(N)).union(N.difference(M)
)
X=list(C)
X.sort()

for i in X:
    print(i)
```

Introduction to Sets

```
def average(array):
    total =
float(sum(set(array)))/int(len(set(array)
))
    return total
```

Set .add()

```
n = int(input())
h = [ ]

for i in range(0,n):
    k = (input())
    h.append(k)

a = set(h)
b = list(a)
print(len(b))
```

No Idea!

```
n , m = input().split()
arr = input().split()
A = set(input().split())
B = set(input().split())

happy = 0

for i in arr :
    if i in A :
        happy += 1
```

```
    if i in B :
        happy -= 1

print(happy)
```

Set .discard(), .remove() & .pop()

```
n = int(input())
s = set(map(int, input().split()))

def function_set(s) :
    for i in range(int(input())):
        function =  input().split()

        if function[0] == 'pop':
            s.pop()

        if function[0] == 'remove':
            s.remove(int(function[1]))

        if function[0] == 'discard':
            s.discard(int(function[1]))

    return sum(s)
print(function_set(s))
```

Set .union() Operation

```
a = int(input())
b = set(input().split())
c = int(input())
d = set(input().split())

print(len(b.union(d)))
```

Set .intersection() Operation

```
a = int(input())
b = set(input().split())
c = int(input())
d = set(input().split())

print(len(b.intersection(d)))
```

Set .difference() Operation

```
a = int(input())
b = set(input().split())
c = int(input())
d = set(input().split())
```

```
print(len(b.difference(d)))
```

Set .symmetric_difference() Operation

```
a = int(input())
b = set(input().split())
c = int(input())
d = set(input().split())

print(len(b.symmetric_difference(d)))
```

Set Mutations

```
useless = int(input())
A = set(map(int, input().split()))
n = int(input())

def function_set(A) :
    for i in range(n):

        function, ne = input().split()
        B = set(map(int,
input().split()))

        if function ==
```

```
'intersection_update':
            A.intersection_update(B)

        if function == 'update':
            A.update(B)

        if function ==
'difference_update':
            A.difference_update(B)

        if function ==
'symmetric_difference_update':

A.symmetric_difference_update(B)

    return sum(A)

print(function_set(A))
```

The Captain's Room

```
k = int(input())
n  = list(map(int,input().split()))

n.sort()
```

```
for i in range(len(n)):
    if(i == 0 and n[0] != n[1]):
        cap = n[i]
        break
    elif(i == len(n)-1 and n[i]!=
n[i-1]):
        cap = n[i]
        break
    elif(n[i-1]!=n[i] and n[i+1]!= n[i]):
        cap = n[i]
        break
```

Check Subset

```
n = int(input())

for x in range(n):
    A = int(input())
    a = set(input().split())
    B = int(input())
    b = set(input().split())
    print(a.issubset(b))
```

Find the Runner-Up Score!

```
if __name__ == '__main__':

    n = int(input())
    arr = map(int, input().split())


    lst = list(arr)
    m = max(lst)
    lst.remove(m)

    for i in lst :
        if i == m :
            lst.remove(i)

    print(max(lst))
```

Nested Lists

```
marks_s = []
marks_l = []
n = int(input())
```

```
for i in range(n):
    name = input()
    marks = float(input())

    record=[name,marks]
    marks_s.append(record)
    marks_l.append(marks)

marks_s.sort()
s = sorted(set(marks_l))[1]

for i,j in marks_s:
    if(j==s):
        print(i)
```

Finding the percentage

```
if __name__ == '__main__':
    n = int(input())
    student_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        student_marks[name] = scores
    query_name = input()
```

```
avg =
sum(student_marks[query_name])/len(studen
t_marks[query_name])
print(format(avg,".2f"))
```

Check Strict Superset

```
A = set(map(int, input().split()))

for i in range(int(input())):
    B = set(map(int, input().split()))
    bol = A.issuperset(B)
    if(not bol):
        break

print(bol)
```

collections.Counter()

```
from collections import Counter

input()
c = Counter(input().split())
tot = 0
```

```
for i in range(int(input())):
    n,p = input().split()
    if c[n]>0 :
        tot += int(p)
        c[n] -= 1
print(tot)
```

DefaultDict Tutorial

```
from collections import defaultdict

d = defaultdict(list)
N,M = map(int, input().split())
l = []

for i in range(1, N+1) :
    d[input()].append(i)

for i in range(M) :
    l.append(input())

for i in l :
    if i in d :
        print(*d[i])
```

```
    else :
        print(-1)
```

Collections.namedtuple()

```
from collections import namedtuple

N = int(input())
tot = 0
student_tuple = namedtuple('Student',
input().split())

for i in range(N):
    student =
student_tuple(*input().split())
    tot += int(student.MARKS)
print(round(tot/N,2))
```

Collections.OrderedDict()

```
from collections import OrderedDict

dictio = OrderedDict()
```

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

    l = list(input().split())

    if len(l)==3:
        a = l[0]+' '+l[1]
        b = int(l[2])
    else :
        a = l[0]
        b = int(l[1])

    if a not in dictio :
        dictio[a] = b
    else:
        dictio[a] += b

[print(n,p) for n,p in dictio.items()]
```

Collections.deque()

```
from collections import deque

d = deque()

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

    c = input().split()

    if c[0] == "append":
        d.append(c[1])

    if c[0] == "appendleft":
        d.appendleft(c[1])

    if c[0] == "pop":
        d.pop()

    if c[0] == "popleft":
        d.popleft()

print(*d)
```

Piling Up!

```
# Enter your code here. Read input from
STDIN. Print output to STDOUT

T = int(input())

for i in range(T):
    num_cube = int(input())
    cube = list(map(int,
input().split()))

    if max(cube) <= cube[0] or max(cube)
<= cube[-1] :
        print('Yes')
    else :
        print('No')
```

Mutations

```
def mutate_string(string, position,
character):
    s = list(string)
    s[position] = character
    string ="".join(s)
    return string
```

itertools.product()

```
from itertools import product

list1 = list(map(int,input().split()))
list2 = list(map(int,input().split()))
z = product(list1,list2)

for i in z:
    print(i,end=' ')
```

Calendar Module

```
import calendar

m,d,y = map(int,input().split())
print(calendar.day_name[calendar.weekday(
y,m,d)].upper())
```

Merge the Tools!

```
def merge_the_tools(string, k):
    n = 0
    stri = ''
    for i in string:
        n += 1
        if i not in stri:
            stri += i
        if n == k:
            print(stri)
            stri = ''
            n = 0
    return stri
```

The Minion Game

```
def minion_game(string):

    kevin = 0
    stuart = 0

    for i in range(len(string)):
        if string[i] in 'AEIOU':
            kevin += (len(string)-i)
        else:
            stuart += (len(string)-i)

    if kevin>stuart:
        print("Kevin " + str(kevin))

    elif kevin<stuart:
        print("Stuart " + str(stuart))

    else:
        print("Draw")
```

Exceptions

```
for i in range(int(input())):
    a,b = input().split()

    try:
        print(int(a) // int(b))

    except BaseException as e:
        print("Error Code:", e)
```

Incorrect Regex

```
import re
for i in range(int(input())):
    try:
        re.compile(input())
        print(True)
    except:
        print(False)
```

Zipped!

```
n,x = input().split()
x = int(x)
```

```
l=[]

for i in range(x):
    lst =
list(map(float,input().split()))
    l.append(lst)

for i in zip(*l):
    print(sum(i)/x)
```

Athlete Sort

```
import math
import os
import random
import re
import sys

if __name__ == '__main__':
    nm = input().split()

    n = int(nm[0])
    m = int(nm[1])
    arr = []
```

```
    for _ in range(n):
        arr.append(list(map(int,
input().rstrip().split())))

    k = int(input())

    for l in sorted(arr, key= lambda x:
x[k]):
        print(*l)
```

Map and Lambda Function

```
cube = lambda x: pow(x,3)# complete the
lambda function

def fibonacci(l):
    a = 0
    b = 1
    l = []
    if n > 1:
        l.append(a)
        l.append(b)
        for i in range(0, n-2):
            f = a + b
            l.append(f)
```

```
            a = b
            b = f
    elif n == 0:
        return []

    else:
        l.append(a)
    return l
```

Re.split()

```
regex_pattern = r",|\." # Do not delete
'r'.
```

Detect Floating Point Number

```
T = int(input())

for i in range(T):
  try:
    print(bool(float(input())))
  except:
    print('False')
```

Group(), Groups() & Groupdict()

```
st = input()

for i in range(len(st)-1):
    if st[i] == st[i+1]:
        if st[i:i+2].isalnum() == True:
            print(st[i])
            break;
else:
    print('-1')
```

Re.findall() & Re.finditer()

```
import re

st = input()
m =
re.findall(r'(?<=[QWRTYPSDFGHJKLZXCVBNMqw
rtypsdfghjklzxcvbnm])[AEIOUaeiou]{2,}(?=[
QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbn
m])',st)

if m:
    for i in m:
```

```
        print(i)
else:
    print('-1')
```

Re.start() & Re.end()

```
s=input()
k=input()

if s.find(k) == -1:
    print((-1,-1))

else:
 for i in range(len(s)-len(k)+1):
    if s[i:i+len(k)]==k:
        print((i,i+len(k)-1))
```

Regex Substitution

```
import re
text = ""

for i in range(int(input())):
    text += input()+'\n'
text = re.sub(" && ", " and ", text)
```

```
text = re.sub(" && ", " and ", text)

print(text.replace(" || ", " or "))
```

Validating phone numbers

```
for i in range(int(input())):
    n=(input())
    if len(n) == 10:
        try:
            if int(n):
                if n[0] == "9" or n[0] ==
"7" or n[0] == "8":
                    print("YES")
                else:
                    print("NO")
            else:
                print("NO")
        except:
            print("NO")
    else:
        print("NO")
```

Arrays

```
def arrays(arr):
    a = numpy.array(arr, float)
    b = a[::-1]
    return(b)
```

Shape and Reshape

```
import numpy as np

arr=np.array(list(map(int,input().split()
)))
arr.shape=(3,3)
print(arr)
```

Transpose and Flatten

```
import numpy as np


w,x = input().split()
w = int(w)
lst = []
for i in range(w):
```

```
    y = input().split()
    lst.append(y)
arr = np.array(lst,int)

print(np.transpose(arr))
print(arr.flatten())
```

Concatenate

```
import numpy as np

n,m,p = map(int, input().split(" "))
lst=[]

for i in range(0,n+m):

    l = input().split()
    lst.append(l)

print(np.array(lst,int))
```

Zeros and Ones

```
import numpy as np

i = list(map(int, input().split()))
print(np.zeros((i), int))
print(np.ones((i), int))
```

Eye and Identity

```
import numpy as np

np.set_printoptions(legacy='1.13') #Found
on StackOverflow
N,M = map(int, input().split())
print(np.eye(N,M))
```

Sum and Prod

```
import numpy as np

lst=[]
n,m =map(int,input().split())

for i in range(n):
```

```
    lst.append(list(map(int,
input().split())))

print(np.prod(np.sum(lst, axis=0)))
```

Min and Max

```
import numpy as np

n, m = map(int, input().split())
a = np.array([input().split() for i in
range(n)],int)
print(np.max(np.min(a, axis=1), axis=0))
```

Mean, Var, and Std

```
import numpy as np

np.set_printoptions(legacy=False)
arr = np.array([input().split() for x in
range(int(input().split()[0]))], int)

print(np.mean(arr, axis=1))
print(np.var(arr, axis=0))
print(np.around(np.std(arr, axis =
```

```
None),decimals=11))
```

Dot and Cross

```
import numpy as np

n = int(input())
a = np.array([input().split() for i in
range(n)], int)
b = np.array([input().split() for i in
range(n)], int)
print (np.dot(a, b))
```

Inner and Outer

```
import numpy as np

a = np.array(input().split(), int)
b = np.array(input().split(), int)

print(np.inner(a,b))
print(np.outer(a,b))
```

Polynomials

```
import numpy as np

n = list(map(float,input().split()));
m = input();
print(np.polyval(n,int(m)));
```

Array Mathematics

```
import numpy as np

n, m = (int(x) for x in input().split())

a = np.array([[int(x) for x in
input().split()] for i in range(n)])
b = np.array([[int(x) for x in
input().split()] for i in range(n)])


np.set_printoptions(suppress = True)

print(np.round(a + b))
print(np.round(a - b))
print(np.round(a * b))
```

```
print(np.round(a // b))
print(np.round(a % b))
print(np.round(a ** b))
```

Floor, Ceil and Rint

```
import numpy as np

np.set_printoptions(legacy = '1.13')
arr = np.array(input().strip().split(),
float)

print(np.floor(arr))
print(np.ceil(arr))
print(np.rint(arr))
```

Linear Algebra

```
import numpy as np

np.set_printoptions(legacy='1.13')
a = []
n = int(input())
for i in range(n):
    a.append(input().split())
```

```
arr = np.array(a,float)

print(np.linalg.det(arr))
```

Reduce Function

```
def product(fracs):
    t = reduce(lambda x,y: x*y, fracs)
    return t.numerator, t.denominator
```

Validating and Parsing Email Addresses

```
import re

for i in range(int(input())):
    a, b = input().split()
    if
re.match(r'^<[a-z]+[\.a-z0-9_-]*@[a-z]+\.
[a-z]{1,3}>$', b):
        print(a, b)
```

Hex Color Code

```
import re

for i in range(int(input())) :
    n =  input()
    z =
re.findall(r"[\s:](#[0-9A-Fa-f]{6}|#[0-9A
-Fa-f]{3})",n)
    if z :
        print("\n".join(z))
```

Validating UID

```
import re
value = int(input())
reg =
r'^(?=(?:[a-z\d]*[A-Z]){2})(?=(?:\D*\d){3
})(?:([a-zA-Z\d])(?!.*\1)){10}$'
for i in range(value):
    match = re.match(reg,str(input()))
    if match:
        print("Valid")
    else:
        print("Invalid")
```

HTML Parser - Part 1

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, att):
        print ("Start :",tag)
        for i in att:
            print("->",i[0],">",i[1])
    def handle_endtag(self, tag):
        print ("End   :", tag)
    def handle_startendtag(self, tag,
att):
        print ("Empty :", tag)
        for i in att:
            print("->",i[0],">",i[1])

parser = MyHTMLParser()
for i in range(int(input())):
    parser.feed(input())
```

HTML Parser - Part 2

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_comment(self,data):
        if '\n' in data:
            print('>>> Multi-line
Comment')
            print(data)
        else:
            print('>>> Single-line
Comment')
            print(data)

    def handle_data(self,data):
        if data!='\n':
            print('>>> Data')
            print(data)

html = ""
for i in range(int(input())):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
```

```
parser.feed(html)
parser.close()
```

Detect HTML Tags, Attributes and Attribute Values

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag,
attrs):
        print(tag)
        for attr in attrs:
            print("->", attr[0], ">",
attr[1])
parser = MyHTMLParser()

for i in range(int(input())):
    parser.feed(input())
```

XML 1 - Find the Score

```
def get_attr_number(node) :
    num = 0
    num +=  len(node.attrib)
```

```
    for child in node :
        num += get_attr_number(child)
    return num
```

XML2 - Find the Maximum Depth

```
max_d = 0
def depth(elem, lvl):
    global max_d
    if lvl == 0:
        lvl += 1
    if len(elem) <1:
        if lvl>max_d:
            max_d = lvl
    else:
        for i in elem:
            depth(i,lvl+1)
```

Standardize Mobile Number Using Decorators

```
def wrapper(f):
    def fun(l):
        f(f"+91 {i[-10:-5]} {i[-5:]}" for
i in l)
    return fun
```

Decorators 2 - Name Directory

```
def age(x):
    return int(x[2])

def person_lister(f):
    def inner(pple):
        return map(f,sorted(pple,
key=age))
    return inner
```

Word Order

```
from collections import Counter

n = int(input())
chars = [input().strip() for i in
range(n)]
counts = Counter(chars)

print(len(counts))
print(*counts.values())
```

Input()

```
x,k = map(int, input().split())
print (k == eval(input()))
```

Python Evaluation

```
eval(input())
```

Any or All

```
n = int(input())
arr = list(map(int,
input().strip().split()))

print(all(map(lambda x:x>=0 , arr)) and
any(map(lambda x:str(x) ==
str(x)[::-1],arr)))
```

Number Line Jumps

```
#!/bin/python3

import math
import os
import random
import re
import sys

# Complete the 'kangaroo' function below.
#
# The function is expected to return a
STRING.
# The function accepts following
parameters:
#  1. INTEGER x1
#  2. INTEGER v1
#  3. INTEGER x2
#  4. INTEGER v2

def kangaroo(x1, v1, x2, v2):
    if v1 > v2 and (x2-x1) % (v1-v2) ==
0:
        return "YES"
    else :
```

```
        return "NO"

if __name__ == '__main__':
    fptr =
open(os.environ['OUTPUT_PATH'], 'w')

    first_multiple_input =
input().rstrip().split()

    x1 = int(first_multiple_input[0])

    v1 = int(first_multiple_input[1])

    x2 = int(first_multiple_input[2])

    v2 = int(first_multiple_input[3])

    result = kangaroo(x1, v1, x2, v2)

    fptr.write(result + '\n')

    fptr.close()
```

Insertion Sort - Part 2

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'insertionSort2' function
below.
#
# The function accepts following
parameters:
#  1. INTEGER n
#  2. INTEGER_ARRAY arr
#

def insertionSort2(n, arr):
    for i in range(1,n):
        for j in range(i):
            if(arr[i]<arr[j]):

arr[i],arr[j]=arr[j],arr[i]
```

```
        print(*arr)

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int,
input().rstrip().split()))

    insertionSort2(n, arr)
```

Birthday Cake Candles

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'birthdayCakeCandles'
function below.
#
# The function is expected to return an
```

```
INTEGER.
# The function accepts INTEGER_ARRAY
candles as parameter.
#

def birthdayCakeCandles(candles):
    return candles.count(max(candles))

if __name__ == '__main__':
    fptr =
open(os.environ['OUTPUT_PATH'], 'w')

    candles_count = int(input().strip())

    candles = list(map(int,
input().rstrip().split()))

    result = birthdayCakeCandles(candles)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Insertion Sort - Part 1

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'insertionSort1' function
below.
#
# The function accepts following
parameters:
#  1. INTEGER n
#  2. INTEGER_ARRAY arr
#
def insertionSort1(n, arr):
    # Write your code here
    i = n-1
    num = arr[i]
    while(i > 0 and num < arr[i-1]):
        arr[i] = arr[i-1]
        print(*arr)
```

```
        i -= 1
    arr[i] = num
    print(*arr)

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int,
input().rstrip().split()))

    insertionSort1(n, arr)
```

Recursive Digit Sum

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'superDigit' function
below.
```

```
#
# The function is expected to return an
INTEGER.
# The function accepts following
parameters:
#  1. STRING n
#  2. INTEGER k
#

def superDigit(n, k):
    sum = 0
    l = [1,2,3,4,5,6,7,8,9]

    if n in l:
        return n
    else:
        for i in str(n):
            sum += int(i)
        sum *= k
    return superDigit(sum, 1)

if __name__ == '__main__':
    fptr =
open(os.environ['OUTPUT_PATH'], 'w')

    first_multiple_input =
```

```
input().rstrip().split()

    n = first_multiple_input[0]

    k = int(first_multiple_input[1])

    result = superDigit(n, k)

    fptr.write(str(result) + '\n')

    fptr.close()
```

Viral Advertising

```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'viralAdvertising'
function below.
```

```
#
# The function is expected to return an
INTEGER.
# The function accepts INTEGER n as
parameter.
#

def viralAdvertising(n):
    a = 0
    b = 5
    for i in range(n):
        c = int(b/2)
        a += c
        b = c*3
    return a

if __name__ == '__main__':
    fptr =
open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    result = viralAdvertising(n)

    fptr.write(str(result) + '\n')
    fptr.close()
```

Validating Credit Card Numbers

```
import re

for i in range(int(input())) :
    n = input()
    if
re.search(r'^[4-6][\d]{3}[\-]?[\d]{4}[\-]
?[\d]{4}[\-]?[\d]{4}$',n) and not
re.search(r'(\d)(?=\1\1\1+)',n.replace('-
','')) :
        print('Valid')
    else :
        print('Invalid')
```

Validating Roman Numerals

```
regex_pattern =
r"^M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})
(IX|IV|V?I{0,3})$"

#Helped with Stackoverflow
```

ginortS

```
st = input()
a,b,c,d=[],[],[],[]

for i in st:
    if i.islower() == True:
        a.append(i)
    elif i.isupper() == True:
        b.append(i)
    elif int(i)%2!=0:
        c.append(i)
    else:
        d.append(i)

print(''.join(sorted(a))+''.join(sorted(b
))+''.join(sorted(c))+''.join(sorted(d)))
```

Time Delta

```
#!/bin/python3

import math
import os
import random
```

```
import re
import sys
import datetime

# Complete the time_delta function below.

def time_delta(t1, t2):
    t1=datetime.datetime.strptime(t1,'%a
%d %b %Y %H:%M:%S %z')
    t2=datetime.datetime.strptime(t2,'%a
%d %b %Y %H:%M:%S %z')
    return
str(int((abs(t1-t2)).total_seconds()))

if __name__ == '__main__':
    fptr =
open(os.environ['OUTPUT_PATH'], 'w')

    t = int(input())

    for t_itr in range(t):
        t1 = input()

        t2 = input()

        delta = time_delta(t1, t2)
```

```
        fptr.write(delta + '\n')

    fptr.close()
```

Tuples

```
if __name__ == '__main__':

    n = int(input())
    t = tuple(map(int,input().split()))

print(hash(t))
```