Say "Hello, World!" With Python

```
if __name__ == '__main__':
    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%2==1): print("Weird")
if((n%2==0)&(n>=2)&(n<=5)): print("Not Weird")
if((n%2==0)&(n>=6)&(n<=20)): print("Weird")
if((n%2==0)&(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)
```

Loops

```
if __name__ == '__main__':
    n = int(input())
    i=0
    while(i<n):
        print(i*i)
        i=i+1
```

List Comprehensions

```
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
    L=[]
    i=0
    while(i<=x):
        j=0
        while(j<=y):
            k=0
            while(k<=z):
                if(i+j+k != n):
                    M=[i, j, k]
                    L.append(M)
                k=k+1
            j=j+1
        i=i+1
    print(L)
```

Print Function

```
if __name__ == '__main__':
    n = int(input())
    if(n<10):
        x=0
        i=1
        while(i<(n+1)):
            x=x*10+i
            i=i+1
        s=str(x)
        print(s)
    if((n>9) & (n<100)):
        x=123456789
        i=10
        while(i<(n+1)):
            x=x*100+i
            i=i+1
        s=str(x)
        print(s)
    if(n>99):
        i=1
        x=0
        while(i<10):
            x=x*10+i
            i=i+1
        while((i>9) & (i<100)):
            x=x*100+i
            i=i+1
        while(i<(n+1)):
            x=x*1000+i
            i=i+1
        s=str(x)
        print(s)
```

Write a function

```
def is_leap(year):
    leap = False

    a=year%4
    b=year%100
    c=year%400
    if(c==0):
        leap= True
    if((a==0) & (b!=0)):
        leap= True
    return leap
```

Find the Runner-Up Score!

```
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
    arr=list(arr)
    f=arr[0]
    i=0
    while(i<len(arr)):
        if(arr[i]>f):
            f=arr[i]
        i=i+1
    i=0
    s=-100
    while(i<len(arr)):
        if((arr[i]<f)&(arr[i]>s)):
            s=arr[i]
        i=i+1
    print(s)
```

Nested Lists

```
if __name__ == '__main__':
    L=[]
    for _ in range(int(input())):
        name = input()
        score = float(input())
        L.append([name, score])
    m=100
    i=0
    while(i<len(L)):
        if(m>L[i][1]):
            m=L[i][1]
        i=i+1
    s=100
    i=0
    while(i<len(L)):
        if((L[i][1]>m)&(L[i][1]<s)):
            s=L[i][1]
        i=i+1
    M=[]
    i=0
    while(i<len(L)):
        if(L[i][1]==s):
            M.append(L[i][0])
        i=i+1
    M.sort()
    i=0
    while(i<len(M)):
        print(M[i])
        i=i+1
```

Tuples

```
if __name__ == '__main__':
    n = int(input())
    integer_list = map(int, input().split())
    T=tuple(integer_list)
    print(hash(T))
```

sWAP cASE

```
def swap_case(s):
    i=0
    S=""
    while(i<len(s)):
        if(s[i].isalpha()==1):
            if(s[i]==s[i].lower()):
                S=S+s[i].upper()
            if(s[i]==s[i].upper()):
                S=S+s[i].lower()
        else:
            S=S+s[i]
        i=i+1
    return S
```

String Split and Join

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

if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)
```

What's Your Name?

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

Mutations

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

Find a string

```
def count_substring(string, sub_string):
    koliko=0
    i=0
    while(i<len(string)):
        k=0
        br=0
        while((k<len(sub_string)) & ((i+k)<len(string))):
            if(string[i+k]==sub_string[k]):
                br=br+1
            k=k+1
        if(br==len(sub_string)):
            koliko=koliko+1
        i=i+1
    return koliko
```

String Validators

```
if __name__ == '__main__':
    s = input()
    i=0
    br=0
    slovo=0
    broj=0
    mal=0
    velik=0
    while(i<len(s)):
        if(s[i].isalnum()==1):
            br=1
        if(s[i].isalpha()==1):
            slovo=1
            if(s[i]==s[i].lower()):
                mal=1
            else:
                velik=1
        if(s[i].isdigit()==1):
            broj=1
        i=i+1
    if(br==1):
        print("True")
    else:
        print("False")
    if(slovo==1):
        print("True")
    else:
        print("False")
    if(broj==1):
        print("True")
    else:
        print("False")
    if(mal==1):
        print("True")
    else:
        print("False")
    if(velik==1):
        print("True")
    else:
        print("False")
```

Text Wrap

```
def wrap(string, max_width):
    vanjski=0
    while(vanjski<(len(string)-1)):
        i=0
        mali=""
        while((i<max_width)&((i+vanjski)<len(string))):
            mali=mali+string[i+vanjski]
            i=i+1
        vanjski=vanjski+max_width
        print(mali)
    mali=""
    i=0
    while((i<max_width)&((i+vanjski)<len(string))):
        mali=mali+string[i+vanjski]
        i=i+1
    return mali
```

Introduction to Sets

```
def average(array):
    # your code goes here
    a=0
    i=0
    s=set(array)
    a=list(s)
    ave=0
    while(i<len(a)):
        ave=ave+a[i]
        i=i+1
    ave=ave/len(a)
    return ave
```

No Idea!

```
n, m = map(int,input().split())
L = list(map(int, input().split()))
A = set(map(int, input().split()))
B = set(map(int, input().split()))
h=0
i=0
while(i<len(L)):
    if(L[i] in A):
        h=h+1
    if(L[i] in B):
        h=h-1
    i=i+1
print(h)
```

Capitalize!

```
def solve(s):

    split_s = s[:].split()

    for string in split_s:
        s = s.replace(string, string.capitalize())

    return s
```

Set .add()

```
N=int(input())
i=0
```

```
S=set()
while(i<N):
    S.add(input())
    i=i+1
print(len(S))
```

Symmetric Difference

```
M=int(input())
i=0
A=set((map(int, input().split())))
N=int(input())
B=set(map(int,input().split()))
U=A.union(B)
I=A.intersection(B)
SR=U.difference(I)
L=list(SR)
L.sort()
for i in (L):
    print(i)
```

Set .union() Operation

```
a=int(input())
E=set(map(int,input().split()))
b=int(input())
F=set(map(int,input().split()))
U=E.union(F)
print(len(U))
```

Set .intersection() Operation

```
n=int(input())
englezi=set(map(int,input().split()))
b=int(input())
francuzi=set(map(int,input().split()))
presjek=englezi.intersection(francuzi)
print(len(presjek))
```

Set .difference() Operation

```
n=int(input())
englezi=set(map(int, input().split()))
b=int(input())
francuzi=set(map(int, input().split()))
razlika=englezi.difference(francuzi)
print(len(razlika))
```

Set .symmetric_difference() Operation

```
e=int(input())
englezi=set(map(int, input().split()))
f=int(input())
francuzi=set(map(int,input().split()))
sr=englezi.symmetric_difference(francuzi)
print(len(sr))
```

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()
    average=0.0
    j=0
    while (j<len((student_marks[query_name]))):
        average=average+student_marks[query_name][j]
        j=j+1
    average=average/len(student_marks[query_name])
    print("{:.2f}".format(average))
```

Designer Door Mat

```
n,m=input().split()
```

```
n=int(n)
m=int(m)
br_crta=((3*n-1)/2)-2
br_emoji=1
i=0
while (i<((n-1)//2)):
    s=""
    j=0
    while (j<=br_crta):
        s=s+"-"
        j=j+1
    j=0
    while(j<br_emoji):
        s=s+'.|.'
        j=j+1
    j=0
    while(j<=br_crta):
        s=s+'-'
        j=j+1
    print(s)
    br_crta=br_crta-3
    br_emoji=br_emoji+2
    i=i+1
j=0
s=""
while(j<((3*n-7)/2)):
    s=s+'-'
    j=j+1
s=s+'WELCOME'
j=j+7
while(j<(3*n)):
    s=s+'-'
    j=j+1
print(s)
i=i+1
s=""
br_crta=br_crta+3
br_emoji=br_emoji-2
while(i<n):
    s=""
    j=0
    while (j<=br_crta):
        s=s+'-'
        j=j+1
    j=0
    while(j<br_emoji):
        s=s+'.|.'
        j=j+1
    j=0
    while(j<=br_crta):
        s=s+'-'
        j=j+1
    print(s)
    br_crta=br_crta+3
    br_emoji=br_emoji-2
    i=i+1
```

Check Subset

```
T=int(input())
i=0
S=''
while(i<T):
    br_ela=int(input())
    A=set(map(int,input().split()))
    br_elb=int(input())
    B=set(map(int,input().split()))
    a=list(A)
    b=list(B)
    if(br_ela>br_elb):
        S=S+"False"
        S=S+"\n"
    else:
        k=0
        BR=0
        while(k<br_ela):
            j=0
            br=0
            while(j<br_elb):
                if(a[k]==b[j]):
                    br=br+1
                j=j+1
            if(br!=0):
                BR=BR+1
            k=k+1
        if(BR==br_ela):
            S=S+"True"
            S=S+"\n"
        else:
            S=S+'False'
            S=S+'\n'
    i=i+1
    s=S
print(s)
```

Check Strict Superset

```
A=set(map(int,input().split()))
n=int(input())
i=0
ukupan=0
while(i<n):
    B=set(map(int,input().split()))
    a=list(A)
    b=list(B)

    if(len(a)<len(b)):
        print('False')
        quit()
    else:
        k=0
        BR=0
        while(k<len(a)):
            j=0
            br=0
            while(j<len(b)):
                if(a[k]==b[j]):
                    br=br+1
                j=j+1
            if(br!=0):
                BR=BR+1
            k=k+1
        if((BR==len(b))& (len(a)!=len(b))):
            ukupan=ukupan+1
        else:
            print("False")
            quit()
    i=i+1
if(ukupan==n):
    print('True')
else:
    print('False')
```

Polar Coordinates

```
import cmath
z=complex(input())
print(abs(z))
print(cmath.phase(z))
```

Mod Divmod

```
import math
import __future__
a=int(input())
b=int(input())
print(a//b)
print(a%b)
print(divmod(a,b))
```

Power - Mod Power

```
import math
a=int(input())
b=int(input())
m=int(input())
print(pow(a,b))
print(pow(a,b,m))
```

Integers Come In All Sizes

```
import math
a=int(input())
b=int(input())
c=int(input())
d=int(input())
print((a**(b))+(c**(d)))
```

itertools.product()

```
from itertools import product
A=list(map(int,input().split()))
B=list(map(int,input().split()))
print(*product(A,B))
```

itertools.permutations()

```
import itertools
S, k=input().split()
perm=list(itertools.permutations(sorted(S),int(k)))
i=0
while(i<len(perm)):
    s=''
    j=0
    while(j<len(perm[i])):
        if(perm[i][j].isalpha()):
            s=s+perm[i][j]
        j=j+1
    print(s)
    i=i+1
```

Iterables and Iterators

```
import itertools
N = int(input())
L = list(input().split())
K = int(input())
perm = itertools.permutations(L, K)
PERM = list(perm)
bra = 0
i = 0
ukupno = 0
while i < len(PERM):
    j = 0
    ukupno = ukupno + 1
    while j < K:
        if PERM[i][j] == 'a':
            bra = bra + 1
            break
        j = j + 1
    i = i + 1

print(round(bra / ukupno, 4))
```

Find Angle MBC

```
import math
ab=int(input())
bc=int(input())
a=u"\u00b0"
mbc = math.degrees(math.atan2(ab, bc))
```

```
print(f'{round(mbc)}{a}')
```

itertools.combinations_with_replacement()

```
from itertools import combinations_with_replacement
S, k=input().split()
C=combinations_with_replacement(sorted(S), int(k))
for i in C:
    print(''.join(i))
```

itertools.combinations()

```
from itertools import combinations
S, k=input().split()
K=1
while(K<=int(k)):
    c=combinations(sorted(S), K)
    for i in c:
        print(''.join(i))
    K=K+1
```

Text Alignment

```
#Replace all ______ with rjust, ljust or center.

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).ljust(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(thickness)+c+(c*(thickness-i-
1)).ljust(thickness)).rjust(thickness*6))
```

The Captain's Room

```
from collections import Counter
K = int(input())
rooms = list(map(int, input().split()))
c=Counter(rooms)
kljuc=list(c.keys())
vrijednost=list(c.values())
i=0
while(i<len(kljuc)):
    if(vrijednost[i]==1):
        kapetan=kljuc[i]
    i=i+1
print(kapetan)
```

Triangle Quest

```
for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank
line also
    print(i*ascii(i))
```

Collections.deque()

```
from collections import deque
N=int(input())
i=0
d=deque()
naredba=''
while(i<N):
    naredba=input().split()
    if(naredba[0]=='append'):
        d.append(int(naredba[1]))
    if(naredba[0]=='pop'):
        d.pop()
    if(naredba[0]=='popleft'):
        d.popleft()
    if(naredba[0]=='appendleft'):
        d.appendleft(int(naredba[1]))
    i=i+1
i=1
s=str(d[0])
while(i<len(d)):
    s=s+' '+str(d[i])
    i=i+1
print(s)
```

Calendar Module

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

Incorrect Regex

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

Lists

```
if __name__ == '__main__':
    N = int(input())
    i=0
    s=[]
    L=[]
    while(i<N):
        narediti=[]
        narediti=narediti+input().split()
        if(narediti[0]=='insert'):
            L.insert(int(narediti[1]), int(narediti[2]))
        elif(narediti[0]=='print'):
            print(L)
        elif((narediti[0])=="remove"):
            L.remove(int(narediti[1]))
        elif((narediti[0])=='append'):
            L.append(int(narediti[1]))
        elif(narediti[0]=='sort'):
            L.sort()
        elif(narediti[0]=='pop'):
            L.pop(-1)
        elif(narediti[0]=='reverse'):
            L.reverse()
        i=i+1
```

Compress the String!

```
S=str(input())
L=list(S)
i=0
k=list()
v=list()
while(i<len(L)):
    br=1
    j=i+1
    while(j<len(L)):
        if(L[j]==L[j-1]):
            br=br+1
            j=j+1
        else:
            break
    k.append(int(L[i]))
    v.append(int(br))
    i=j
s=''
i=0
while(i<len(k)):
    s=s+'('+str(v[i])+', '+str(k[i])+') '
    i=i+1
print(s)
```

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

```
n=int(input())
s=set(map(int, input().split()))
N=int(input())
i=0
while(i<N):
    naredba=[]
    naredba=naredba+(input().split())
    if(naredba[0]=='pop'):
        s.pop()
    if(naredba[0]=='remove'):
        s.remove(int(naredba[1]))
    if(naredba[0]=='discard'):
        s.discard(int(naredba[1]))
    i=i+1
print (sum(s))
```

Python Evaluation

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

Map and Lambda Function

```
cube = lambda x: x*x*x

def fibonacci(n):
    L=list()
    i=2
    if(n>0):
        L.append(0)
    if(n>1):
        L.append(1)
        while(i<n):
            L.append(L[i-1]+L[i-2])
            i=i+1
    return L
```

Zipped!

```
N, X=input().split()
N=int(N)
X=int(X)
i=0
s=[]
L=[]
while(i<X):
    L.append(input().split())
    j=0
    while(j<N):
        L[i][j]=float(L[i][j])
        j=j+1
    i=i+1
i=0
s=[]
while(i<N):
    j=0
    a=0
    while(j<X):
        a=a+float(L[j][i])
        j=j+1
    a=a/X
    s.append(a)
    i=i+1
i=0
while(i<len(s)):
    print(round(s[i], 1))
    i=i+1
```

Set Mutations

```
a=int(input())
A=set(map(int,input().split()))
N=int(input())
i=0
while(i<N):
    M=[]
    M=M+input().split()
    j=0
    broj=0
    B=set(map(int, input().split()))
    if(M[0]=='intersection_update'):
        A.intersection_update(B)
        broj=broj+len(A)
    if(M[0]=='update'):
        A.update(B)
        broj=broj+len(A)
    if(M[0]=='symmetric_difference_update'):
        A.symmetric_difference_update(B)
        broj=broj+len(A)
    if(M[0]=="difference_update"):
        A.difference_update(B)
        broj=broj+len(A)
    i=i+1
i=0
broj=0
while(i<len(A)):
    broj=broj+int(list(A)[i])
    i=i+1
print(broj)
```

Word Order

```
from collections import Counter
n=int(input())
i=0
L=list()
while(i<n):
    L.append(input())
    i=i+1
c=Counter(L)
print(len(c.keys()))
A=list(c.values())
i=1
s=str(A[0])
while(i<len(A)):
    s=s+" "+str(A[i])
    i=i+1
print(s)
```

Any or All

```
N=int(input())
L=list(input().split())
i=0
a='False'
br=0
while(i<N):
    L[i]=int(L[i])
    if((L[i])>=0):
        br=br+1
    i=i+1
if(br==N):
    i=0
    while(i<N):
        j=(len(str(L[i]))//2)
        while(j<len(str(L[i]))):
            if(str(L[i]).find(str(L[i])[j])==-j):
                jel=1
            else:
                jel=0
                break
            j=j+1
        if (jel==1):
            a='True'
        i=i+1
print(a)
```

ginortS

```
S=str(input())
veliki=list()
mali=list()
parni=list()
neparni=list()
for i in S:
    if(i.isalpha()):
        if(i==i.upper()):
            veliki.append(i)
        else:
            mali.append(i)
    else:
        if(int(i)%2==0):
            parni.append(i)
        else:
            neparni.append(i)
veliki.sort()
mali.sort()
parni.sort()
neparni.sort()
i=0
s=''
while(i<len(mali)):
    s=s+mali[i]
    i=i+1
i=0
```

```
while(i<len(veliki)):
    s=s+veliki[i]
    i=i+1
i=0
while(i<len(neparni)):
    s=s+str(neparni[i])
    i=i+1
i=0
while(i<len(parni)):
    s=s+str(parni[i])
    i=i+1
print(s)
```

Reduce Function

```
def product(fracs):
    t = Fraction(reduce(lambda x,y : x * y , fracs))

    return t.numerator, t.denominator
```

Re.split()

```
regex_pattern = r""     # Do not delete 'r'.

regex_pattern = r"[,.]"
```

Validating phone numbers

```
N=int(input())
i=0
while(i<N):
    s=str(input())
    if(any([s.startswith('7'), s.startswith('8'), s.startswith('9')]) and (len(s)==10) and 
s.isnumeric()):
        print('YES')
    else:
        print('NO')
    i=i+1
```

String Formatting

```
def print_formatted(number):
# your code goes here
    i=1
    razmak = len(str(bin(number))[2:])
    red = ''
    while(i<=number):
        prvi=str(i)
        drugi=str(oct(i))[2:]
        treci=str(hex(i).upper())[2:]
        cetvrti=str(bin(i))[2:]
        red += ' ' * (razmak - len(str(prvi))) + str(prvi)
        redak = (drugi, treci, cetvrti) #ntorka
        for r in redak:
            red += ' ' * (razmak - len(str(r)) + 1) + str(r)
        i=i+1
        red += "\n"
    red.rstrip()
    print(red[:-1])
    return
```

Hex Color Code

```
N=int(input())
i=0
while i < N:
    hex = input().split()
    jel=-1
    for e in hex:
        if(e.find('{')!=-1):
            jel=e.find('{')
        if (e.find('#')!=-1):
            e= e[e.find('#')+1:]
            provjera = 'abcdefABCDEF1234567890'
            j=0
            while(j<len(e)):
                k=0
                br=0
                while(k<len(provjera)):
                    if(provjera[k]==e[j]):
                        br=1
                    k=k+1
                if(br==0):
                    e=e[:j]
                    if(((len(e)==3) | (len(e)==6))):
                        print(''.join(['#', e]))
                j=j+1
    i=i+1
```

Validating UID

```
T=int(input())
i=0
while(i<T):
    uid=str(input())
    vs=0
    j=0
    uvjeti=0
    br=0
    al=0
    while(j<len(uid)):
        if(uid[j].isalpha()):
            if(uid[j]==uid[j].upper()):
                vs=vs+1
        if(uid[j].isnumeric()):
            br=br+1
        if(uid[j].isalnum()):
            al=al+1
        j=j+1
    if(vs>=2):
        uvjeti=uvjeti+1
    if(br>=3):
        uvjeti=uvjeti+1
    if(al==len(uid)):
        uvjeti=uvjeti+1
    j=0
    problem=0
    while(j<len(uid)):
        k=0
        while(k<len(uid)):
            if(uid[j]==uid[k] and k!=j):
                problem=1
            k=k+1
        j=j+1
    if(problem==0):
        uvjeti=uvjeti+1
    if(len(uid)==10):
        uvjeti=uvjeti+1
    if(uvjeti==5):
        print('Valid')
    else:
        print('Invalid')
    i=i+1
```

collections.Counter()

```
import collections
X=int(input())
cipele=list(map(int, input().split()))
c=collections.Counter(cipele)
N=int(input())
kljucevi = []
vrijednosti = []
for _ in range(N):
    line =input().split()
    size = int(line[0])
    price = int(line[1])
    kljucevi.append(size)
    vrijednosti.append(price)
zarada=0
velicine=list(c.keys())
kolicine=list(c.values())

i=0
while (i<len(kljucevi)):
    j=0
    while (j<len(velicine)):
        if ((kljucevi[i]==velicine[j]) and (kolicine[j])!=0):
            zarada=zarada+vrijednosti[i]
            kolicine[j]=kolicine[j]-1
        j=j+1
    i=i+1
print(zarada)
```

Detect Floating Point Number

```
T=int(input())
i=0
while(i<T):
    N=str(input())
    uvjet=0
    try:
        N=float(N)
        uvjet=uvjet+1
        if(N==float(N)):
            uvjet=uvjet+1
        if(str(N).find('.')!=-1):
            uvjet=uvjet+1
        if(str(N).startswith('-') and str(N).find('.')==1):
            uvjet=uvjet-1
        if(str(N).startswith('+') and str(N).fin('.')==1):
            uvjet=uvjet+1
        if(N==0):
            uvjet=uvjet-1
        if(uvjet==3):
            print (True)
        else:
            print (False)
    except:
        print(False)
    i=i+1
```

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})$"    #
Do not delete 'r'.
pattern = r''
```

Validating and Parsing Email Addresses

```
n=int(input())
i=0
while(i<n):
    uvjeti=0
    adresa=input().split()
    if((adresa[1][adresa[1].find('@'):].find('.'))!=-1 and adresa[1].find('@')!=-1 and
adresa[1].find('.')!=n-1):
        uvjeti=uvjeti+1
    j=2
    br=0
    if(adresa[1][1].isalpha() and adresa[1][0]=="<" and adresa[1][len (adresa[1])-1]=='>'):
        while(j<adresa[1].find('@')):
            if(adresa[1][j]=='-' or adresa[1][j].isalnum() or adresa[1][j]=='.' or adresa[1][j]=='_'):
                br=br+1
            j=j+1
    if (br==adresa[1].find('@')-2):
        uvjeti=uvjeti+1
    k=adresa[1].find('@')+1
    br=0
    while(k<adresa[1].rfind('.')):
        if(adresa[1][k].isalpha()):
            br=br+1
        k=k+1
    k=k+1
    BR=0
    extension=''
    while(k<len(adresa[1])-1):
        extension=extension+adresa[1][k]
        if(adresa[1][k].isalpha()):
            BR=BR+1
        k=k+1
    if((br==adresa[1].rfind('.')-adresa[1].find('@')-1) and (BR==len(adresa[1])-2-
adresa[1].rfind('.'))):
        uvjeti=uvjeti+1
    if(len(extension)<4):
        uvjeti=uvjeti+1
    if(uvjeti==4):
        print(' '.join(adresa))
    i=i+1
```

Validating Email Addresses With a Filter

```
def fun(s):
    i = s.find("@")
    if i == -1:
        return False
    if not s[0:i].replace("-", "a").replace("_", "a").isalnum():
        return False
    j = s.find(".")
    if j == -1:
        return False
    if not s[i+1:j].isalnum():
        return False
    if j < len(s) - 4:
        return False
    if not s[j+1:].isalpha():
        return False
    return True
```

Merge the Tools!

```
def merge_the_tools(string, k):
    # your code goes here
    i=0
    while(i<=len(string)-k):
        j=i
        s=""
        while((j<i+k) & (i+k<=len(string))):
            if(s.find(string[j])==-1):
                s=s+string[j]
            j=j+1
        i=i+k
        print (s)
    return
```

DefaultDict Tutorial

```
from collections import defaultdict
n, m=map(int,input().split())
A=[]
B=[]
i=0
while(i<n):
    A.append(input())
    i=i+1
i=0
while(i<m):
    B.append(input())
    i=i+1
i=0
d=defaultdict(list)
while(i<n):
    d[A[i]].append(i+1)
    i=i+1
i=0
while(i<m):
    if len(d[B[i]])>0:
        print(*d[B[i]])
    else:
        print(-1)
    i=i+1
```

Exceptions

```
T=int(input())
i=0
while(i<T):
    naredba=[]
    naredba=(input().split())
    try:
        print(int(int(naredba[0])/int(naredba[1])))
    except ValueError as z:
        print('Error Code:', z)
    except ZeroDivisionError:
        print('Error Code: integer division or modulo by zero')
    i=i+1
```

Collections.namedtuple()

```
i=0
podaci=namedtuple("student", 'ID, MARKS, NAME, CLASS')
average=0
unos=input().split()
m=unos.index('MARKS')
id=unos.index('ID')
n=unos.index('NAME')
c=unos.index("CLASS")
while(i<N):
    brojevi=input().split()
    student = podaci(brojevi[id], brojevi[m], brojevi[n], brojevi[c])
    average=average+int(student.MARKS)
    i+=1
average=average/N
print(round(average, 2))
```

The Minion Game

```
def minion_game(string):
    n = len(string)
    Kevin = 0
    Stuart = 0
    for start in range(0, n):
        letter = string[start]
        if letter == "A" or letter == "E" or letter == "I" or letter == "O" or letter == "U":
            Kevin = Kevin + n - start
        else:
            Stuart = Stuart + n - start
    if Kevin > Stuart:
        print("Kevin", Kevin)
    elif Stuart > Kevin:
        print("Stuart", Stuart)
    else:
        print("Draw");
```

Maximize It!

```
maxValue=0
def calc(n, s):
    global maxValue
    global m

    if n == k:
        if s > maxValue:
            maxValue = s
    else:
        for i in numbers[n]:
            calc(n+1, (s + i * i) % m)

line=list(map(int, input().split()))
k = line[0]
m = line[1]
numbers = []
for i in range(0, k):
    numbers.append(list(map(int, input().split()))[1:])

calc(0, 0)
print(maxValue)
```

Collections.OrderedDict()

```
from collections import OrderedDict
N=int(input())
i=0
od=OrderedDict()
for _ in range(0, N):
    s = input()
    priceString = s.split()[-1]
    price = int(priceString)
    item = s[0:len(s)-1-len(priceString)]
    if item in od:
        od[item] += price
    else:
        od[item] = price

for item in od:
    print(item, od[item])
```

Validating Credit Card Numbers

```
N=int(input())
i=0
while(i<N):
    uvjet=0
    broj=input()
    if(broj.startswith('4') or broj.startswith('5') or broj.startswith('6')):
        uvjet=uvjet+1
    if(broj.find('-')!=-1):
        j=0
        BROJ=''
        while(j<len(broj)):
            if((j%5==4) & (broj[j]!='-')):
                uvjet=uvjet-1
            if(j%5!=4):
                BROJ=BROJ+broj[j]
            j=j+1
    else:
        BROJ=broj
    if(BROJ.isnumeric()):
        uvjet=uvjet+3
    if(len(BROJ)==16):
        uvjet=uvjet+1
    j=0
    prati=1
    while(j<len(BROJ)-1):
        if(BROJ[j]==BROJ[j+1]):
            prati=prati+1
        else:
            prati=1
        if(prati==4):
            uvjet=uvjet-1
        j=j+1
    if(uvjet==5):
        print("Valid")
    else:
        print("Invalid")
    i=i+1
```

Zeros and Ones

```
import numpy
brojevi=list(map(int,input().split()))
print(numpy.zeros(brojevi, dtype =int))
print(numpy.ones(brojevi, dtype = int))
```

Arrays

```
def arrays(arr):
    # complete this function
    # use numpy.array
    arr.reverse()
    arr=numpy.array(arr, float)
    return arr
```

Shape and Reshape

```
import numpy
podaci=list(map(int, input().split()))
podaci=numpy.array(podaci)
podaci.shape=(3,3)
print(podaci)
```

Transpose and Flatten

```
import numpy
N,M=map(int, input().split())
podaci=[]
i=0
while(i<N):
    m = list(map(int, input().split()))
    podaci.append(m)
    i=i+1
podaci=numpy.array(podaci)
print (numpy.transpose(podaci))
```

```
print(podaci.flatten())
```

Concatenate

```
import numpy
N,M,P=map(int,input().split())
i=0
prva=list()
while(i<N):
    prva.append(list(map(int, input().split())))
    i=i+1
prva=numpy.array(prva)
i=0
druga=list()
while(i<M):
    druga.append(list(map(int,input().split())))
    i=i+1
druga=numpy.array(druga)
print(numpy.concatenate((prva, druga), axis=0))
```

Eye and Identity

```
import numpy
numpy.set_printoptions(legacy='1.13')
N,M=map(int,input().split())
print(numpy.eye(N,M))
```

Floor, Ceil and Rint

```
import numpy
numpy.set_printoptions(legacy='1.13')
A = numpy.array(input().split(), float)
print(numpy.floor(A))
print(numpy.ceil(A))
print(numpy.rint(A))
```

Sum and Prod

```
import numpy
N,M=map(int,input().split())
podaci=list()
for i in range (N):
    podaci.append(input().split())
i=0
while(i<len(podaci)):
    j=0
    while(j<len(podaci[i])):
        podaci[i][j]=int(podaci[i][j])
        j=j+1
    i=i+1
podaci=numpy.array(podaci)
zbroj=numpy.sum(podaci, axis=0)
print(numpy.prod(zbroj))
```

Min and Max

```
import numpy
N,M=map(int,input().split())
podaci=list()
for i in range(N):
    podaci.append(input().split())
i=0
while(i<len(podaci)):
    j=0
    while(j<len(podaci[i])):
        podaci[i][j]=int(podaci[i][j])
        j=j+1
    i=i+1
podaci=numpy.array(podaci)
mini=numpy.min(podaci, axis=1)
print(numpy.max(mini))
```

Mean, Var, and Std

```
import numpy
N,M=map(int,input().split())
podaci=list()
for i in range(N):
    podaci.append(input().split())
```

```
i=0
while(i<len(podaci)):
    j=0
    while(j<len(podaci[i])):
        podaci[i][j]=int(podaci[i][j])
        j=j+1
    i=i+1
podaci=numpy.array(podaci)
print(numpy.mean(podaci, axis=1))
print(numpy.var(podaci, axis=0))
print(round(numpy.std(podaci),11))
```

Dot and Cross

```
import numpy
N=int(input())
A=list()
B=list()
for i in range(N):
    A.append(input().split())
for i in range(N):
    B.append(input().split())
i=0
while(i<N):
    j=0
    while(j<N):
        A[i][j]=(int(A[i][j]))
        B[i][j]=(int(B[i][j]))
        j=j+1
    i=i+1
print(numpy.dot(A, B))
```

Inner and Outer

```
import numpy
A=numpy.array(list(map(int,input().split())))
B=numpy.array(list(map(int,input().split())))
print(numpy.inner(A,B))
print(numpy.outer(A,B))
```

Polynomials

```
import numpy
A=numpy.array(list(map(float,input().split())))
x=float(input())
print (numpy.polyval(A, x))
```

Linear Algebra

```
import numpy
N=int(input())
A=list()
for i in range(N):
    A.append(list(map(float,input().split())))
A=numpy.array(A)
print(round(numpy.linalg.det(A),2))
```

Group(), Groups() & Groupdict()

```
import re
string=(str(input()))
checker=0
for i in range(len(string)-1):
    if string[i]==string[i+1] and string[i].isalnum():
        print(string[i])
        checker=1
        break
if checker==0:
    print(-1)
```

Company Logo

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

```
if __name__ == '__main__':
    s = input()
kopija=s
vektor=list()
for j in range(3):
    i=0
    max=kopija[0]
    while(i<len(kopija)):
        if((s.count(kopija[i])>s.count(max)) or ((s.count(kopija[i])==s.count(max)) and
((kopija[i])<max))):
            max=kopija[i]
        i=i+1
    vektor.append(max)
    kopija=kopija.replace(max, '', s.count(max))
if(s.count(vektor[0])==s.count(vektor[2])):
    vektor=sorted(vektor)
elif(s.count(vektor[0])==s.count(vektor[1]) and (vektor[0]>vektor[1])):
    temp=vektor[0]
    vektor[0]=vektor[1]
    vektor[1]=temp
elif(s.count(vektor[1])==s.count(vektor[2]) and (vektor[1]>vektor[2])):
    temp=vektor[1]
    vektor[1]=vektor[2]
    vektor[2]=temp
for j in range(3):
    print(vektor[j], s.count(vektor[j]))
```

Standardize Mobile Number Using Decorators

```
def wrapper(f):
    def fun(l):
        result = []
        for s in l:
            if len(s) == 13:
                result.append("+91 " + s[3:8] + " " + s[8:])
            elif len(s) == 12:
                result.append("+91 " + s[2:7] + " " + s[7:])
            elif len(s) == 11:
                result.append("+91 " + s[1:6] + " " + s[6:])
            else:
                result.append("+91 " + s[0:5] + " " + s[5:])
        return f(result)
    return fun
```

Decorators 2 - Name Directory

```
def person_lister(f):
    def inner(people):
        return [f(p) for p in sorted(people, key=lambda x: int(x[2]))]
    return inner
```

Words Score

```
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score_words(words):
    score = 0
    for word in words:
        num_vowels = 0
        for letter in word:
            if is_vowel(letter):
                num_vowels += 1
        if num_vowels % 2 == 0:
            score=score+2
        else:
            score=score+1
    return score
```

Time Delta

```
import math
import os
import random
import re
import sys
from datetime import timedelta, datetime

# Complete the time_delta function below.
def time_delta(t1, t2):
    dt1 = datetime.strptime(t1, "%a %d %b %Y %H:%M:%S %z")
    dt2 = datetime.strptime(t2, "%a %d %b %Y %H:%M:%S %z")
    td = dt1-dt2
    return str(int(abs(td).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()
```

Triangle Quest 2

```
for i in range(1,int(input())+1): #More than 2 lines will result in 0 score. Do not leave a blank line also
    print (((10**i - 1) // 9 )**2)
```

XML 1 - Find the Score

```
def get_attr_number(node):
    # your code goes here
    i=0
    suma=0
    for i in node.iter():
        suma=suma+len(i.attrib)
    return suma
```

Alphabet Rangoli

```
def print_rangoli(size):
    abeceda = [chr(x) for x in range (ord('a'),ord('a')+26)] #popunit ce listu abecedom preko
ascii vrijednosti

    broj_slova_u_redu = 1 #jer u prvom redu je samo jedno slovo
    broj_znakova_u_redu = 2*(2*size-2) + 1

    red = ''
    indeks = size
    i=0
    crte=""
    while(i<(broj_znakova_u_redu-3)/2):
        crte=crte+"-"
        i=i+1
    #ovo radi samo do kraja sirenja piramide
    for k in range(size):   #po broju redova (pola)
        for m in range (broj_slova_u_redu + size - 1, size - 1, - 1): #(pola)
            red=red+"-" +abeceda[m-broj_slova_u_redu]
        broj_slova_u_redu += 1
        red += '\n'
    broj_slova_u_redu -= 2
    #ovo radi suzavanje piramide
    for k in range(size - 1):
        for m in range (broj_slova_u_redu + size - 1, size - 1, - 1): #(pola)
            #print("2")
            #red = abeceda[indeks - m - 1] + red
            #print("dodajem", abeceda[indeks - m - 1])
            red=red+"-" +abeceda[m-broj_slova_u_redu]
        broj_slova_u_redu -= 1
        red += '\n'

    #print(red)
    #print("-------------------------------------------------------")

    #ovo zrcali piramidu
    #ovo ce dodati ovaj suprotni dio kao npr "de" u "edcde"
    redovi = red.strip().split("\n")
    cijeli_red = ''
    for s in redovi:
        cijeli_red += s
        if len(s) != 1:
            dio = ''
            for c in s[:-1]:
                dio = c + dio
            cijeli_red = cijeli_red + dio
        cijeli_red += "\n"
    cijeli_red=cijeli_red.split('\n')
    cijeli_red[size-1]=cijeli_red[size-1][1:-1]
    i=0
    while(i<len(cijeli_red)//2-1):
        cijeli_red[i]=crte+cijeli_red[i]+crte
        crte=crte[:-2]
        i=i+1
    i=i+1
    crte="-"
    while(i<len(cijeli_red)):
        cijeli_red[i]=crte+cijeli_red[i]+crte
        crte=crte+"--"
        i=i+1
    cijeli_red="\n".join(cijeli_red[:-1])

    print(cijeli_red)
    return
```

Piling Up!

```
def check(cubes):
    lastValue = max(cubes)
    first = 0
    last = len(cubes)-1
    while first <= last:
        if cubes[first] < cubes[last]:
            index = last
            last -= 1
        else:
            index = first
            first += 1
        a = cubes[index]
        if a > lastValue:
            break
        lastValue = a
    if first > last:
        print("Yes")
    else:
        print("No")

t = int(input())
for _ in range(0, t):
    input()
    check(list(map(int, input().split())))
```

Array Mathematics

```
import numpy
N,M=map(int,input().split())
A=list()
B=list()
for i in range(N):
    A.append(input().split())
for i in range(N):
    B.append(input().split())
for i in range(N):
    for j in range(M):
        A[i][j]=int(A[i][j])
        B[i][j]=int(B[i][j])
A=numpy.array(A)
B=numpy.array(B)
print(numpy.add(A, B))
print(numpy.subtract(A, B))
print(numpy.multiply(A, B))
print(numpy.floor_divide(A, B))
print(numpy.mod(A, B))
print(numpy.power(A, B))
```

HTML Parser - Part 1

```
from html.parser import HTMLParser

# create a subclass and override the handler methods
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Start :", tag)
        for a in attrs:
            print("->", a[0], ">", a[1])

    def handle_endtag(self, tag):
        print("End   :", tag)

    def handle_startendtag(self, tag, attrs):
        print("Empty :", tag)
        for a in attrs:
            print("->", a[0], ">", a[1])

# instantiate the parser and fed it some HTML
parser = MyHTMLParser()
```

```
n = int(input())
for _ in range(0, n):
    parser.feed(input())
```

HTML Parser - Part 2

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle_comment(self, data):
        if data.find("\n") >= 0:
            print(">>> Multi-line Comment")
        else:
            print(">>> Single-line Comment")
        print(data)

    def handle_data(self, data):
        if data == "\n":
            return
        print(">>> Data")
        print(data)

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

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

Validating Postal Codes

```
regex_integer_in_range = r"^[1-9]\d{5}$"    # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(?P<digit>\d)(?=\d(?P=digit))"    # Do not delete
'r'.
```

Matrix Script

```
#!/bin/python3

import math
import os
import random
import re
import sys




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

n = int(first_multiple_input[0])

m = int(first_multiple_input[1])

matrix = []

for _ in range(n):
    matrix_item = input()
    matrix.append(matrix_item)

text = ""

for col in range(0, m):
    for row in range(0, n):
        text += matrix[row][col]

print(re.sub("(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])", " ", text))
```

Re.findall() & Re.finditer()

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
pattern=r"(?<=[^aeiouAEIOU])[aeiouAEIOU]{2,}(?=[^aeiouAEIOU])"
string=str(input())
found=0
for i in re.findall(pattern,string):
    found=1
    print(i)
if found==0:
    print(-1)
```

Re.start() & Re.end()

Strings:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
string=str(input())
pattern=str(input())
found=0  #boolean if we find solution
for i in range(len(string)):
    try:
        if string[i:i+len(pattern)]==pattern:
            print("("+str(i)+", "+str(i+len(pattern)-1)+")")
            found=1
    except:
        break
if found==0:
    print("(-1, -1)")
```

regex:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
S = str(input())
k = str(input())
found=0
pattern = f"(?=({k}))"
m = re.finditer(pattern, S)

for i in m:
    found=1
    print("("+str(i.start(1))+", "+str(i.end(1)-1)+")")
if found==0:
    print("(-1, -1)")
```

Regex Substitution

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
n=int(input())
string=[]
```

```
for i in range(n):
    line=str(input())
    line=re.sub("(?<=\s)&&(?=\s)","and", line)
    line=re.sub("(?<=\s)\|\|(?=\s)","or", line)
    string.append(line)
for i in string:
    print(i)
```

XML2 - Find the Maximum Depth

```
maxdepth = 0
maxdepth = 0
def depth(elem, level):
    level += 1
    global maxdepth
    if level > maxdepth:
        maxdepth = level
    for x in elem:
        depth(x, level)
```

Detect HTML Tags, Attributes and Attribute Values

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(tag)
        for a in attrs:
            print("->", a[0], ">", a[1])

    def handle_startendtag(self, tag, attrs):
        print(tag)
        for a in attrs:
            print("->", a[0], ">", a[1])

# instantiate the parser and fed it some HTML
parser = MyHTMLParser()
n = int(input())
for _ in range(0, n):
    parser.feed(input())
```