HOMEWORK-1

DILARA ISIKLI

Introduction

Problem1: Say "Hello, World!" With Python

```
if __name__ == '__main__':
    print "Hello, World!"
```

Problem2: Python If-Else

```
#check n for it is odd or even and range of the n because of deciding print werid or not werid

#!/bin/python

import math

import os

import random

import re

import sys

if __name__ == '__main__':

    n = int(raw_input().strip())

    if (n % 2 == 1):

        print("Weird")

    elif (n % 2 == 0 and n <5 and n>=2):

        print("Not Weird")

    elif (n % 2 ==0 and n<=20 and n>=6):

        print("Weird")

    elif(n % 2 ==0 and n >20):

        print("Not Weird")
```

Problem3: Arithmetic Operators

```
if __name__ == '__main__': #take 2 input than use arithmetic oper.

    a = int(raw_input())

    b = int(raw_input())
    print( a + b)

    print(a-b)

    print(a*b)
```

Problem4 : Python Division

```
from __future__ import division
# take two integers and print integer division and float division,
if __name__ == '__main__':
    a = int(raw_input())
    b = int(raw_input())
    print(int(a/b))
    print(float(a/b))
```

Problem5: Loops

```
# take n integer. Print n time value * 2
if __name__ == '__main__':
    n = int(raw_input())
    for value in range(n):
        print(value ** 2)
```

Problem6: Write a function

```
# if year can be divided by 4, is a leap year. İf year can be divided by 100, it is NOT a leap year.
#if year  is divisible by 400, it is a leap year.

def is_leap(year):
    leap = False
    if year % 400 == 0:
        leap = True
    elif year % 100 == 0:
        leap = False
    elif year % 4 == 0:
        leap = True
    return leap
```

Problem7: Print Function

```
# read an integer (value) , print 1,2,3..Value

from __future__ import print_function
if __name__ == '__main__':
    value = int(raw_input())

for value in range(1,value+1):
    print(value, end='')
```

Data types

Problem1: List Comprehensions

```
#x,y,z are dimensions of a cuboid along. print a list of all possible coordinates
# i+j+k not equal to n
if __name__ == '__main__':

    x = int(raw_input())

    y = int(raw_input())

    z = int(raw_input())

    n = int(raw_input())

   value = [[i, j, k] for i in range(x + 1) for j in range(y + 1) for k in range(z + 1) if i + j + k != n]

    print(value)
```

Problem2: Find the Runner-Up Score!

```
# n is list leng. , take n scores and find runner-up (second) print runnerup score

if __name__ == '__main__':

    n = int(raw_input())

    arr = map(int, raw_input().split())

    maxArr= max(arr)

    runnerUp= -101        # i >=-100 so we can take runnerup=-101 as a first value

    for value in range(n):

        if arr[value] != maxArr and arr[value] > runnerUp:

            runnerUp = arr[value]

    print(runnerUp)
```

Problem3: Nested Lists

```
#Take names and grades for each student. Store them in a nested list.
#Print the names of any students  having the second lowest grade.

if __name__ == '__main__':

    nameList=[]

    scoreList=[]

    for _ in range(int(raw_input())):

        name = raw_input()

        score = float(raw_input())

        nameList.append(name)

        scoreList.append(score)



    nestedList = []

    nestedList.extend([list(a) for a in zip(nameList, scoreList)])

    newList=[]

    scoreList2 = sorted(scoreList)

    scoreList3 =[]

 for i in range(len(scoreList2)-1):

        if(scoreList2[i] < scoreList2[i+1]):

            scoreList3.append( scoreList2[i])

    if(len(scoreList3) == 1 or len(scoreList3) == 0):

        scoreList3.append( scoreList2[-1])

    for i in range(len(nestedList)):

        if(nestedList[i][1] == scoreList3[1]):

            newList.append(nestedList[i][0])
    for i in sorted(newList):
        print i
```

Problem4: Finding the percentage

#take n students for names and marks of lessons. Required to save the record in a dictionary data type.

#Output the average percentage marks obtained by that student, correct to two decimal places

```
if __name__ == '__main__':
    n = int(raw_input())
    student_marks = {}
    for _ in range(n):
        line = raw_input().split()
        name, scores = line[0], line[1:]
        scores = map(float, scores)
        student_marks[name] = scores
    query_name = raw_input()




    for key, value in student_marks.items():
        if(query_name == key ):

            print "%.2f" %(sum(value)/len(value))
```

Problem5: Lists

#The first line contains an integer,n , denoting the number of commands.
#Each line i of the n subsequent lines contains one of the commands described above.

```
if __name__ == '__main__':
    N = int(raw_input())
    list= []
    for i in range(N):
        value = raw_input().split()
        if len(value) == 3:
             eval("list." + value[0] + "(" + value[1] + "," + value[2] + ")")
        elif len(value) == 2:
            eval("list." + value[0] + "(" + value[1] + ")")
        elif value[0] == "print":
            print list
        else:
            eval("list." + value[0] + "()")
```

Problem6: Tuples

```
#take n integers, create a tüple of integers. Then compute and print the hash() of tuple
if __name__ == '__main__':
    n = int(raw_input())
    integer_list = map(int, raw_input().split())
    t= tuple(integer_list)
    print hash(t)
```

Strings

Problem1: sWAP cASE

```
# convert all lowercase letters to uppercase letters and vice versa.
def swap_case(s):
    return s.swapcase()

if __name__ == '__main__':
    s = raw_input()
    result = swap_case(s)
    print result
```

Problem2: String Split and Join

```
# given a string. Split the string on a " " (space) delimiter and join using a - hyphen.

def split_and_join(line):

   string = line

   sp = string.split(" ")

   return  "-".join(sp)

   if __name__ == '__main__':

   line = raw_input()
   result = split_and_join(line)  #use function
   print result
```

Problem3: What’s your name?

```
#take the firstname and lastname.

# Print:Hello firstname lastname! You justdelved into python.

def print_full_name(a, b):

   print ("Hello " + a+ " " + b + "! You just delved into python.")
```

Problem4: Mutations

```
# Take a string.

# position : index location

# character is which

def mutate_string(string, position, character):

 string = string[:(position)] + character + string[(position+1):] # replace the character at index

 return(string)

if __name__ == '__main__':

   s = raw_input()

   i, c = raw_input().split()

   s_new = mutate_string(s, int(i), c)

   print s_new
```

Problem5: Find a string

```
def count_substring(string, subString):
    #take string and substring to check string include substring or not
    count = 0

    for i in range(len(string)):
        if subString[0] == string[i]: #if string has first letter of subst. continu.
            if subString == string[i:i+len(subString)]: #if string has all letter of sub. add 1 to count
                count += 1
    return(count)

if __name__ == '__main__':
    string = raw_input().strip()
    sub_string = raw_input().strip()

    count = count_substring(string, sub_string)
    print count
```

Problem6: String Validators

```
#check if a string is composed of alphabetical characters, alphanumeric characters, digits, etc.
if __name__ == '__main__':
    string = raw_input()

print any(value.isalnum() for value in string) # print True if has any alphanumeric characters
print any(value.isalpha() for value in string) # print True if  has any alphabetical characters.
print any(value.isdigit() for value in string) # print True if  has any digits
print any(value.islower() for value in string) # print True if  has any lowercase characters
print any(value.isupper() for value in string) #, print True if  has any uppercase characters
```

Problem7: Tex Alignment

```
fre = int(input())

h = 'H'

for i in range(fre):

    print((h * i).rjust(fre - 1) + h + (h * i).ljust(fre - 1))


for i in range(fre + 1):

    print((h * fre).center(fre * 2) + (h * fre).center(fre * 6))


for i in range((fre + 1) // 2):

    print((h * fre * 5).center(fre * 6))


for i in range(fre + 1):

    print((h * fre).center(fre * 2) + (h * fre).center(fre * 6))


for i in range(fre):

    print(((h*(fre-i-1)).rjust(fre)+h +(h*(fre- i - 1)).ljust(fre)).rjust(fre * 6))
```

Problem8: Text Wrap

```
# wrap the string into a paragraph of width

def wrap(string, max_width):

    text= textwrap.fill(string,max_width) #sperate string to substring which has width is max_width

    return text

if __name__ == '__main__':
    string, max_width = raw_input(), int(raw_input())
    result = wrap(string, max_width)
    print result
```

Problem9: Designer Door Mat

```
# Mat size must be n*m. Have 'WELCOME' written in the center.
#should only use |, . and - characters.
def door ():
   n,m = map(int,raw_input().split())
   welM = (m - 7) / 2
   first = "-"
   second = ".|."
   wel = "WELCOME"
   width=6
   for i in range((n - 1 )/ 2):  #first part of design
      print((first * ((m-(3 *(2 * i+1)))/2))+(second * (2 * i+1))+(first * ((m-(3 *(2 * i+1)))/2)))
   print ((first * welM) + wel + (first * welM)) #center of design
   for i in xrange(n-2, -1, -2):  #last part of design
      print ( str('.|.')*i ).center(m, '-')



if __name__ == '__main__':
   door()
```

Problem10: String Formatting

```
def print_formatted(number):
   # take number of iterator. Print the following values for each integer from 1 to len(num)
  #1.Decimal, 2. Octal, 3. Hexadecimal (capitalized), And 4. Binary

   width =len('{:b}'.format(number))
   for i in range((number)):
      formatS ="{0:{width}d} {0:{width}o} {0:{width}X} {0:{width}b}".format((i+1),width = width)
      print formatS
```

Problem11: Capitalize!

```
#  first and last names of people begin with a capital letter

def solve(s):

     ListS= list(s)

     for i in range(0,len(ListS)):

        if(i==0):

            ListS[i]= ListS[i].capitalize()  #check first word

        if(ListS[i].isspace()):   #check every word after spaces

            ListS[i+1]= ListS[i+1].capitalize()

            i+=1

     newString="".join(ListS)

     return(newString)

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

    s = raw_input()

    result = solve(s)

    fptr.write(result + '\n')

    fptr.close()
```

Problem12: The Minion Game

```
def minion_game(input_string):

    list1=list(input_string)

    s = 0

    k = 0

    for i in range (0,len(list1)):

        if list1[i] in ['a','e','i','o','u','A','E','I','O','U'] : # if element is vowel it means this subset start with
vowel and score is +  len(list1) - (i+1) + 1 and kevin gains tihs score

            k = k +  len(list1) - (i+1) + 1 #artık olan
        else : # if element is slient letter it means this subset start with vowel and score is +  len(list1) -
(i+1) + 1 and draw gains tihs score
            s = s + len(list1) - (i+1) + 1 #artık
    if k > s :
        print "Kevin", k
    elif k == s :
        print "Draw"
    else :
        print "Stuart", s
if __name__ == '__main__':
    s = raw_input()
    minion_game(s)
```

Problem13: Alphabet Rangoli

```
from collections import defaultdict
import string
def print_rangoli(n):


    width = 4 * (n - 1) + 1
    l = string.ascii_lowercase[:n]

    for i in range(n-1):
        s = l[n-1-i:]
        print(('-'.join(s[-1:0:-1] + s)).center(width, '-'))

    for i in range(n-1, -1, -1):
        s = l[n-1-i:]
        print(('-'.join(s[-1:0:-1] + s)).center(width, '-'))

if __name__ == '__main__':
    n = int(raw_input())
    print_rangoli(n)
```

Problem14: Merge the Tools!

```
def merge_the_tools(string, k):
    for i in range(0, (len(string)-k+1), k):
        s = list(string[i:i+k]) #divide by k
        print ''.join([x for i,x in enumerate(s) if s.index(x) == i]) #non-distinct characters
if __name__ == '__main__':
    string, k = raw_input(), int(raw_input())
    merge_the_tools(string, k)
```

Sets

Problem1: Introduction to Sets

```
#take array, turn to set, sum set and divide sum of set by len(set)

def average(array):

    # your code goes here

    total = 0


    for i in range (len(array)):


        total = total + array[i]

        i+=1


    av = total / (len(array))

    sett = set(array)

    sums = sum(sett)

    lenn= len(sett)

    avv = sums / lenn

    return avv

    if __name__ == '__main__':

    n = int(raw_input())
    arr = map(int, raw_input().split())
    result = average(arr)
    print result
```

Problem2: No Idea!

```
# A and B are   disjoint sets, if a integer is in A happiness +1 else if integer is in b happiness -1
mANDn = set(map(int, raw_input().split()))
arrays = (map(int, raw_input().split()))
A=set(map(int, raw_input().split()))
B=set(map(int, raw_input().split()))
result = 0

for i in arrays:
    if i in A:
        result = result +1
    if i in B:
        result = result - 1
print result
```

Problem3: Symmetric Difference

```
# take two string and compare them. Select symmetric difference elements and print it
if __name__ == '__main__':

    int(raw_input())
    n = raw_input().split() #first string len
    ni = set(list(map(int, n))) #convert from string to set

    int(raw_input())
    m = raw_input().split() #second string len
    mi = set(list(map(int, m))) #convert from string to set

    reslist = [] #create new list for add diffrences
    for i in list(ni.difference(mi)):
        reslist.append(i)
    for j in list(mi.difference(ni)):
        reslist.append(j)
    for k in sorted(reslist):
        print k
```

Problem4: Set .add()

```
# create a set to remove duplicate elements from the list and check len of the set
value = int(raw_input())
valueOfset = set()

for i in range(value):
    string = str(raw_input())
    valueOfset.add(string)

print len(valueOfset)
```

Problem5: Set .union() Operation

```
#take to set and compare them for find union elements. Print number of union elements
NUMBEROFVALUE1= input()
value1 = set(map(int,raw_input().split()))
NUMBEROFVALUE2= input()
value2 = set(map(int,raw_input().split()))
a= value1.union(value2)
b= len(a)
print b
```

Problem6: Set .intersection() Operation

```
# take to set and compare them for find intersection elements. Print number of intersection element
```

```
EnglishNewspaper = input()
studentsE = set(map(int, raw_input().split()))
Frenchnewspaper = input()
studentsF = set(map(int, raw_input().split()))

SETT= studentsE.intersection(studentsF)
print len(SETT)
```

Problem7: Set .difference() Operation

```
#find total number of students who are subscribed to the English newspaper only.
EnglishNewspaper = input()
studentsE = set(map(int, raw_input().split()))
Frenchnewspaper = input()
studentsF = set(map(int, raw_input().split()))

SETT= studentsE.difference(studentsF) #select only english new. subscribed
print len(SETT)
```

Problem8: Set .symmetric_difference() Operation

```
# a union b – (a and b)
EnglishNewspaper = input()
studentsE = set(map(int, raw_input().split()))
Frenchnewspaper = input()
studentsF = set(map(int, raw_input().split()))

SETT= studentsE.symmetric_difference(studentsF)
print len(SETT)
```

Problem9: Set Mutations

```
#Take operation name and implement(update) them

numberOfElements = input()

listOfelements = set(map(int, raw_input().split()))

numberOfotherSets = input()

for i in range (numberOfotherSets):

    kind = raw_input().split()

    otherset = set(map(int,raw_input().split()))


    if kind[0] == 'intersection_update':

        listOfelements.intersection_update(otherset)

    if kind[0] == 'update':

        listOfelements.update(otherset)

    if kind[0] == 'symmetric_difference_update':

        listOfelements.symmetric_difference_update(otherset)

    if kind[0] == 'difference_update':

        listOfelements.difference_update(otherset)



print sum(listOfelements)
```

Problem10: Check Subset

```
#check subset if If set A is subset of set B, print True else False

testCases = input()

listOfDes = []

for i in range(testCases):

    elemOfA = input()

    A = set(map(int, raw_input().split()))

    elemOfB = input()

    B = set(map(int, raw_input().split()))

    if (B.issuperset(A) == False): #is super set or not?
```

```
        listOfDes.append(False)
    if (B.issuperset(A) == True):   #is super set or not?
        listOfDes.append(True)

for value in listOfDes:
    print value
```

Problem11: Check Strict Superset

```
#print True if set A is a strict superset of all other sets. Otherwise, print False.
elements = set(map(int, raw_input().split()))
otherSetsNum = input()
listOfDes = []

for i in range(otherSetsNum):
    otherSet = set(map(int, raw_input().split()))
    if (elements.issuperset(otherSet) == False): #is super set or not?
        listOfDes.append(False)
    if (elements.issuperset(otherSet) == True): #is super set or not?
        listOfDes.append(True)
print (all(listOfDes))
```

Problem12: The Captain's Room

```
from collections import Counter
K = int(raw_input()) #size of each group.
RoommNumList = raw_input().split() #unordered elements of the room number list

RoommNumList = Counter(RoommNumList)
print [x for x in RoommNumList.keys() if RoommNumList[x] ==1][0]
```

Problem13: Set .discard(), .remove() & .pop()

Collections

Problem1: collections.Counter()

```
#Have a list containing the size of each shoe.

#There are numbers of customers who are willing to pay price only if they get the shoe of their desired size

#when a shose is sold , remove the shose number and cont. add price of solded shose

numOfSho = input()

shoeSizes=list(map(int, raw_input().split()))

numOfCus = input()

total=[]

money=[]

for i in range(numOfCus):

   cusAndPrice=list(map(int, raw_input().split()))

   total.append( cusAndPrice)


for  value in total:


   if value[0] in shoeSizes:

      money.append(value[1])

      shoeSizes.remove(value[0])





print sum(money)
```

Problem2: Collections.namedtuple()

```
#print the average marks of the list corrected to 2 decimal places.
from collections import namedtuple
n=input() #take student numbers
students= namedtuple('students',raw_input().split()) #create a tuple as students
print("%.2f" %( sum([float(i.MARKS) for i in [students(*raw_input().split()) for j in range(n)]]) / n ))
#sum marks and div them to get aver.
```

Problem3: Collections.OrderedDict()

```
#list of  items together with their prices that consumers bought on a particular day. #print each item_name and net_price in order of its first occurrence.
from collections import OrderedDict
dic = OrderedDict()
n = int(raw_input())
for _ in range(n):
    l =  raw_input().split()
    item = ' '.join(l[:-1])
    price = int(l[-1])
    if item in dic: #item in dic or not ?
        dic[item] += price
    else:
        dic[item] = price #if not item = price
for key in dic:
    print('{} {}'.format(key, dic[key]))
```

Problem4: Word Order

```
# output order should correspond with the input order of appearance of the word
from collections import OrderedDict
n = int(input())

dic = OrderedDict()
for _ in range(n):
    nextWord = raw_input()
    if nextWord in dic:
        dic[nextWord] =dic[nextWord] +1
    else:
        dic[nextWord] = 1

print(len(dic.values())) #output the number of distinct words from the input.
result= (dic.values())
print ' '.join(map(str,result)) #output the number of occurrences for each distinct word according to their appearance in the input.
```

Problem5: Collections.deque()

```
# Perform append, pop, popleft and appendleft methods on an empty deque (d) and print the result

from collections import deque

d = deque()

n=input()


for i in xrange(n):

    methodsAndValues = raw_input().split()

    if methodsAndValues[0] == 'append':

        d.append(int(methodsAndValues[1]))


    if methodsAndValues[0] == 'appendleft':

        d.appendleft(int(methodsAndValues[1]))

    if methodsAndValues[0] == 'pop':

        d.pop()

    if methodsAndValues[0] == 'popleft':

        d.popleft()


print ' '.join(map(str,d))
```

Problem6: Company Logo

```
#Print the three most common characters along with their occurrence count each on a separate line.Sort output in descending order of occurrence count. If the occurrence count is the same, sort the characters in alphabetical order.

from operator import itemgetter

k = raw_input()

list1 = list(k)

list2 = list1  #copy list to compare them

a= set(list2) #create a set for remove duplication

c = list(a) #create a list again for sorted

b = sorted(c)
```

```
maxx = []

sortedL = []

for i in range(len(b)):

    count = 0

    for j in range(len(list2)):


        if(b[i]==list2[j]):

            count = count +1

    maxx.append(count)


y= zip(b,maxx) #union 2 lists


mlll= sorted(y, key=itemgetter(1), reverse=True)


for i in range(3):

    sortedL.append(mlll[i])



for x in sortedL:

    print(x[0] + " " + str(x[1]))
```

Problem7: DefaultDict Tutorial

```
#For each  words, check whether the word has appeared in group A or not. Print the indices of each
occurrence of M in group A . If it does not appear, print -1.
NM = raw_input().split()
N = int(NM[0])
M = int(NM[1])
A = []
B = []
for i in range(N):
    A.append(raw_input())
for i in range(M):
    B.append((raw_input()))
```

```
indice = []
for i in range(M):
   indice.append([])
   for j in range(N):  #Print the indices of each occurrence of M in group A
      if B[i] == A[j]:
         indice[i].append(str(j+1))
   if len(indice[i])==0: #If it does not appear, print -1.
      indice[i].append('-1')
for i in range(len(indice)):
   print(' '.join(indice[i]))
```

Problem8: Filling Up!

```
#The first line contains a single integer , the number of test cases.
#For each test case, there are  lines.
#The first line of each test case contains , the number of cubes.
#The second line contains  space separated integers, denoting the sideLengths of each cube in that
order.
x = int(input())
for i in range(0,x):
   y = int(input())
   L = list(map(int,raw_input().split()))
   while len(L)>=2 :
      k = len(L)
      if (L[k-1]>=L[k-2]):
         L.pop(k-1)

      elif(L[0]>=L[1]):
         L.pop(0)

      else :
         break

   if len(L)<=2:
         print("Yes")
   else : print("No")
```

Date and Time

Problem1: Calendar Module

```
# find what the day is on that date.

import datetime
```

```
from datetime import date

import calendar

day = map(int, raw_input().split())  #take a day

DateofDay = datetime.date(day[2], day[0], day[1])

result= DateofDay.strftime("%A")

resultUpper =result.upper() #make upper string for true result.

print resultUpper
```

Problem2: Time Delta

```
import datetime as dt

MonthList = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
        'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'] #create month list

def Month(month): #turn month to integer value

    for index, item in enumerate(MonthList):
        if month == item:
            return index + 1

def CalMin(zone):  #turn poz
    neg = False
    if (zone < 0):
        neg = True
        zone = - zone
    hr = zone / 100
    mins = zone % 100
    value = hr * 60 + mins
    if neg:
        value = - value  #turn poz
    return dt.timedelta(minutes = value)

def CalDT(_str):  #take date and fit it "datetime"
    split = _str.split(' ')
    time = [int(x) for x in split[4].split(':')]
    dd = int(split[1])
    mm = Month(split[2])
    yyyy = int(split[3])
    result = dt.datetime(yyyy, mm, dd, time[0], time[1], time[2])
    tz = int(split[5])
    delta = CalMin(tz)
    return result - delta

def CalSec(td):
    return td.days * 86400 + td.seconds

N = int(raw_input())
for i in range(N):
    dt1 = CalDT(raw_input())
    dt2 = CalDT(raw_input())
    print abs(CalSec(dt1-dt2))
```

Exceptions

Problem1: Exceptions

```
testcase = input() #number of testcase
for _ in range(testcase):
    valuelist = raw_input().split() #create list and take values into it
    try:
        print int(valuelist[0])/int(valuelist[1]) #try divide
    except ZeroDivisionError as e: #catch the errors with except
        print "Error Code:",e
    except ValueError as e:
        print "Error Code:",e
```

Built-ins

Problem1: Zipped!

```
NX = map(int, raw_input().split()) #take number of students and number of lessons
Marks = [map(float, raw_input().split()) for _ in range(NX[1])] #take marks of students
for i in zip(*Marks): #zip marks of same lessons and aver. them print aver.

    print sum(i)/NX[1]
```

Problem2: Athlete Sort!

```
from operator import itemgetter
```

```
N, M = [int(x) for x in raw_input().strip().split()] #take number of atheletes and number of attributes

L = [0] * N

for i in xrange(N):

    L[i] = [int(x) for x in raw_input().strip().split()]

K = int(raw_input())#index of selected athelets

L.sort(key=itemgetter(K)) #sort list according to K

for A in L:

    print ' '.join(map(str,A))
```

Problem3: GinortS

```
from string import ascii_lowercase, ascii_uppercase

sortRule = ascii_lowercase + ascii_uppercase + "1357902468" #create a sort rule first all lowercaes second uppercases third odd numbers and last one even numbers

unsorted =raw_input() #take a string


print reduce(lambda x,y: x+y, sorted(unsorted,key=sortRule.index)) #use reduce fun. to sorted
```

Python Functionals

Problem1: Map and Lambda Function

```
cube = lambda x:x*x*x  #create lambda for cube of x
def fibonacci(n): #create fib. function
    y=0
    z = 1
    for _ in xrange(n):
        yield y  #first of all it should return 0 ,1 and than when n>2 return n**3
        y, z = z, y + z
if __name__ == '__main__':
    n = int(raw_input())
    print map(cube, fibonacci(n))
```

Regex and Parsing Challenges

Problem1: Detect Floating Point Number

```
#a valid float number must satisfy all of the following requirements:
#Number can start with +, - or . symbol.
#Number must contain at least 1 decimal value.
#Number must have exactly one . symbol.
#Number must not give any exceptions when converted using float(N).
import re
n = input()
for _ in range(n):
    print(bool(re.match(r'^[+-]?\d*\.\d+$', raw_input())))
#TURN TRUE OR FALSE WITH BOOL OPERATION
```

Problem2: Re.split()

```
#It's guaranteed that every comma and every dot in s is preceeded and followed by a digit.
import re
regex_pattern = "[,.]+"

import re
print("\n".join(re.split(regex_pattern, raw_input())))
```

Problem3: Group(), Groups() & Groupdict()

```
# find the first occurrence of an alphanumeric character in S(read from left to right) that has consecutive repetitions.

import re

match  = re.search(r'([a-zA-Z0-9])(\1)', raw_input())



if match  :

   print (match .group(1))

else:

   print ("-1")
```

Problem4: Re.findall() & Re.finditer()

```
# find all the substrings of string that contains 2 or more vowels.
import re
```

```
str= raw_input() #get string
vowels='aeiou' #seperate vowels and cons
cons='qwrtypsdfghjklzxcvbnm'
match = re.findall(r'(?<=[%s])([%s]{2,})[%s]' % (cons, vowels, cons),str, flags = re.I) #string that
contains 2 or more vowels
print('\n'.join(match or ['-1']))
```

Problem5: Validating Roman Numerals

```
#create a regular expression for a valid Roman numeral.
#Output a single line containing True or False according to the instructions above.
regex_pattern = "^M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$"
import re
print(str(bool(re.match(regex_pattern, raw_input()))))
```

Problem6: Validating phone numbers

```
import re

num=input() #number of string
for i in range(0,num): #take numbers and check first element (789) and len s equal 10 or not
   S=raw_input()
   match = re.search(r'[789][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]', S)
   if match and len(S)==10:
      print('YES')
   else:
      print('NO')
```

Problem7: Validating and Parsing Email Addresses

```
# print each valid email address in the same order as it was
received as input.
import re
RE = re.compile(r'^<[a-z][\w.-]+@[a-z]+\.[a-z]{,3}>$', re.I)
n = input() #number of email address.
for _ in range(n):
    mail = raw_input() #contains a name and an email address as two
space-separated values
    if RE.match(mail.split()[-1]):
        print(mail)
```

Problem8: Hex Color Code

```
#  It must start with a '#' symbol.
#It can have 3 or 6 digits.
#Each digit is in the range of 0 to F
#A-F letters can be lower case
# CHECK THESE
import re

n = input()
out = []

for i in range(n):
   out.extend(re.findall(r'(?<=.)(#[0-9A-Fa-f]{6}|#[0-9A-Fa-f]{3}?)',raw_input()))
print '\n'.join(out)
```

Problem9: Validating UID

```
 #It must contain at least 2 uppercase English alphabet characters.

#It must contain at least 3 digits (0 - 9).
#It should only contain alphanumeric characters
#No character should repeat.
#There must be exactly 10 characters in a valid UID.
import re
n=input()
for _ in range(n):
   u = ''.join(sorted(raw_input()))
   try:
      assert re.search(r'[A-Z]{2}', u)
      assert re.search(r'\d\d\d', u)
      assert not re.search(r'[^a-zA-Z0-9]', u)
      assert not re.search(r'(.)\1', u)
      assert len(u) == 10
   except:
      print('Invalid')
   else:
      print('Valid')
```

Problem10: Regex Substitution

```
#modify those symbols to the following: && → and || → or
import re

def Sub(s):
   s = re.sub(" \&{2} ", " and ", s)
   s = re.sub(" \|{2} ", " or ", s)
```

```
    s = re.sub(" \&{2} ", " and ",s)
    return re.sub(" \|{2} ", " or ", s)

n=input() #take input
for _ in range(n):
    print(Sub(raw_input())) #take lines
```

Problem11: Validating Credit Card Numbers

```
# It must start with a 4,5 or 6.
#It must contain exactly 16 digits.
#It must only consist of digits (0-9).
#It may have digits in groups of 4, separated by one hyphen "-".
#It must NOT use any other separator like ' ' , '_', etc.
#It must NOT have 4 or more consecutive repeated digits.

import re
n=input() #number of string
for _ in range(n):
    string = raw_input()
    if not re.match(r'^[456].*', string):
        print('Invalid')
    elif not re.match(r'^\d{16}$|^\d{4}-\d{4}-\d{4}-\d{4}$', string):
        print('Invalid')
    elif not re.match(r'^((\d)(?!\2{3,}))+$', re.sub(r'\D', '', string)):
        print('Invalid')
    else:
        print('Valid')
```

XML

Problem1: Find the Score

```
import sys
import xml.etree.ElementTree as etree
#calculate the score that is equal to the number of attributes it has.

def get_attr_number(node):
    score = len(node.attrib)
    for i in node:
        score = score + get_attr_number(i) #make iterable
    return score

if __name__ == '__main__':
    sys.stdin.readline()
```

```
   xml = sys.stdin.read()
   tree = etree.ElementTree(etree.fromstring(xml))
   root = tree.getroot()
   print get_attr_number(root)
```

Problem2: Find the Maximum Depth

```
import xml.etree.ElementTree as etree
#print the maximum level of nesting in it.

maxdepth = 0 #Take the depth of the root as
def depth(elem, level):
    global maxdepth
    level= level+ 1
    if (maxdepth < level):
        maxdepth = level
    for value in elem:
        depth(value, level) #make iteratiable

if __name__ == '__main__':
    n = int(raw_input())
    xml = ""
    for i in range(n):
        xml =  xml + raw_input() + "\n"
    tree = etree.ElementTree(etree.fromstring(xml))
    depth(tree.getroot(), -1)
    print maxdepth
```

Numpy

Problem1: Arrays

```
import numpy


def arrays(arr):

   NumArr = arr[::-1]

   return numpy.array(NumArr, float)#it starts from the end, towards the first, taking each element.
So it reverses


arr = raw_input().strip().split(' ')
```

```
result = arrays(arr)

print(result)


def arrays(arr):

    NumArr = arr[::-1]

    return numpy.array(NumArr, float)
```

Problem2: Shape and Reshape

```
#change the dimensions of an array.

import numpy

print numpy.reshape(map(int, raw_input().split()), (3, 3)) #print 3X3 NumPy array.
```

Problem3: Transpose and Flatten

```
import numpy
N,M = map(int, raw_input().split(' '))  #number of row and columns
arr = []
for n in range(N):
    arr.append(map(int, raw_input().split())) #put element into array
NumArry = numpy.array(arr)

print numpy.transpose(arr) #create transpose for array
print NumArry.flatten() #create flatten for numpy array
```

Problem4: Concatenate

```
import numpy

N, M, P = map(int, raw_input().strip().split(' ')) #n and m are row p is colu.

arr1 = numpy.empty([N, P], dtype='int') #Create an array for first array

arr2 = numpy.empty([M, P], dtype='int') #Create an array for second array

NumofTotal = len(arr1) +len(arr2)

for i in range(NumofTotal):

    if i <  len(arr1) :
        arr1[i] = numpy.array(map(int, raw_input().split(' '))) #put element into arr1 until end of len. arr1
    else:
        arr2[i - N] = numpy.array(map(int, raw_input().split(' ')))#put element into arr2


print numpy.concatenate((arr1, arr2), axis=0) #concatenate arrays
```

Problem5: Zeros and Ones

```
#print an array of the given shape and integer type using the tools numpy.zeros and numpy.ones.
import numpy
N = map(int, raw_input().split()) # space-separated integers row colum. and iteate time
print numpy.zeros(N, dtype='int_') #print zeros
print numpy.ones(N, dtype='int_') #print ones
```

Problem6: Eye and Identity

```
#print an array of size nXm with its main diagonal elements as 1's and 0's everywhere else
import numpy
N, M = map(int, raw_input().split())

a= str(numpy.eye(N, M, k = 0)).replace("0"," 0")

b = a.replace("1"," 1")
print b
```

Problem7: Array Mathematics

```
# Print the result of each operation
import numpy
n = int(raw_input().split(' ')[0])
def makeArray(n):
    arr = []
    for j in range(n):
        arr.append(raw_input().split(' '))
    return arr
A = numpy.array(makeArray(n), int)
B = numpy.array(makeArray(n), int)
```

```
print numpy.add(A,B)   # + numpy
print numpy.subtract(A,B) # - mod pow
print numpy.multiply(A,B) # * numpy
print numpy.divide(A,B) # / numpy
print numpy.mod(A,B) # mod numpy
print numpy.power(A,B) #pow numpy
```

Problem8: Floor, Ceil and Rint

```
import numpy as np
A = np.array(map(float, raw_input().split()))
print(np.floor(A))
print(np.ceil(A))
print(np.rint(A))
```

Problem9: Sum and Prod

```
import numpy
N, M = map(int,raw_input().split())
Arr = numpy.array( [map(int, raw_input().split()) for i in range(N)]
) #take array and turn this as numpy array

sumOfArr = numpy.sum(Arr, axis=0) #calculate sum of arr
print numpy.prod(sumOfArr) #prod of sum of arr
```

Problem10: Min and Max

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

import numpy

N,M=map(int,raw_input().split()) #take number of dimensions

arr=numpy.array([raw_input().split() for i in range(N)],int)

print numpy.max(numpy.min(arr,axis=1)) #take min of max of arr.numpy
```

Problem11: Mean, Var, and Std

```
#The mean along axis 1
#The var along axis 0
#The std along axis none

import numpy
n,m=map(int, raw_input().split())
arr=numpy.array([map(int, raw_input().split()) for i in range(n)])
print numpy.mean(arr, axis=1)
print numpy.var(arr,axis=0)
print numpy.around(numpy.std(arr),12)
```

Problem12: Dot and Cross

```
import numpy
N = int(raw_input())
A = numpy.array([ map(int, raw_input().split()) for i in range(N) ]) #take first array
B = numpy.array([ map(int, raw_input().split()) for i in range(N) ]) #take second array

print numpy.dot(A, B)  # compute their matrix product.
```

Problem13: Inner and Outer

```
# compute A's and B's inner and outer product.

import numpy
a = list(map(int, raw_input().split()))
b = list(map(int, raw_input().split()))

a =  numpy.array(a)
b = numpy.array(b)

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

Problem14: Polynomials

```
#  task is to find the value of P at point X.
import numpy
coefficients = raw_input().split()
coefficient  = map(float, coeff)
x = float(raw_input())
print numpy.polyval(coeff, x)
```

Problem15: Linear Algebra

```
#find the determinant of matrix A
import numpy as np
N = input()
A = np.array([list(map(float, raw_input().split())) for _ in range(N)])
print(float(numpy.linalg.det(A)))
```

Birthday Cake Candles

```
import math
import os
import random
import re
import sys
def birthdayCakeCandles(ar):
   counter = 0
   max_height = max(ar) #take max value from array
   for i in ar:
      if i == max_height: #check how many there are max_height
         counter += 1
   return counter

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

   ar_count = int(raw_input())

   ar = map(int, raw_input().rstrip().split())

   result = birthdayCakeCandles(ar)
```

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

    fptr.close()
```

Kangaroo

```
# Print YES if they can land on the same location at the same time; otherwise, print NO.
x1,v1,x2,v2 = map(int, raw_input().split()) #take  starting position and jump distance for kangaroos

if v1 == v2 or (x1 - x2) % (v2 - v1) != 0 or (x1 - x2) // (v2 - v1) < 0:
    print('NO')
else:
    print('YES') #meet at the same location after same number of jumps print yes
```

Viral Advertising

```
n = int(input()) #the integer number of days
count = 0
people = 5 # they advertise it to exactly 5 people on social media

for _ in range(n):
    count = count + people // 2
    people = (people // 2) * 3
print(count)
```

Recursive Digit Sum

```
#It must return the calculated super digit as an integer.
#superDigit has the following parameter(s):
#n: a string representation of an integer
#k: an integer, the times to concatenate n to k make

import math
import os
import random
import re
import sys

def digDivten(num):

    if num/10 == 0:
        return num
    else:
        return digDivten(sum(list(map(int, list(str(num))))))


def superDigit(n, k):
    a = digDivten(sum(list(map(int, list(str(n)))))*k)
    return a

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

    nk = raw_input().split()

    n = nk[0]

    k = int(nk[1])

    result = superDigit(n, k)

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

    fptr.close()
```

Insertion Sort - Part 1

```
#Print the array as a row of space-separated integers each time there is a shift or insertion.
import sys
count = None
arr = []
for line in sys.stdin:
    if count is None:
        count = int(line.strip()) #size of the array
        continue
    arr = [int(i) for i in line.strip().split(' ')] #array of integers to sort

val = arr[-1]
for i in range(count - 1):
    if val >= arr[count - 2 - i]:
        arr[count - 2 - i + 1] = val
        print(' '.join([str(j) for j in arr]))
        break
    else:
        arr[count - 2 - i + 1] = arr[count - 2 - i]
        print(' '.join([str(j) for j in arr]))
```

```
if val not in arr:
    arr[0] = val
    print(' '.join([str(i) for i in arr]))
```

Insertion Sort - Part 2

```
def LastOne(n, ar, i): #fine last one to use insertionsort
    count = arr[i]

    while i>0 and ar[i-1]>count:
        arr[i] = arr[i-1]
        i=i-1;
    arr[i] = count;
    return arr

def insertionsort(arr,n):
    for i in range(1, n):
        arr = LastOne(n, arr, i);
        print (' '.join(map(str,arr)))

n = int(input()) #an integer representing the length of the array
arr = list(map(int,raw_input().split(" "))) #take array
insertionsort(arr,n)
```