Homework 1 Alessandro Sottile

October 4, 2022

1 Problem 1

2 Introduction

2.0.1 Say "Hello, World!" With Python

[ ]:
```
print("Hello, World!")
```

2.0.2 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 != 0:
    print("Weird")
elif n % 2 == 0 and 2 <= n <= 5:
    print("Not Weird")
elif n % 2 == 0 and 6 <= n <= 20:
    print("Weird")
else:
    print("Not Weird")
```

2.0.3 Arithmetic Operators

[ ]:
```
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print(a+b)
    print(a-b)
    print(a*b)
```

2.0.4 Python Division

[ ]:
```
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print(a//b)
    print(a/b)
```

2.0.5 Loops

[ ]:
```
if __name__ == '__main__':
    n = int(input())
    for i in range(n):
      print(i*i)
```

2.0.6 Write a function

[ ]:
```
def is_leap(year):
    l = False

    # Write your logic here

    if year%4==0:
        if year%100==0:
            if year%400==0:
                l=True
            else:
                l=False
        else:
              l=True
    else:
        l=False

    return l

year = int(input())
print(is_leap(year))
```

2.0.7 Print Function

```
[ ]: if __name__ == '__main__':
         n = int(input())

     for i in range(n):
             print(i + 1 , end = "")
```

3 Data Types

3.0.1 List Comprehensions

```
[ ]: if __name__ == '__main__':
         x = int(input())
         y = int(input())
         z = int(input())
         n = int(input())
         Ok = [[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] # helped with the solutions
     print(Ok)
```

3.0.2 Find the Runner-Up Sore!

```
[ ]: if __name__ == '__main__':
         n = int(input())
         arr = map(int, input().split())
         s=sorted(set(arr), reverse = True)
         s=list(s)
         print(s[1])
```

3.0.3 Nested Lists

```
[ ]: a=list()
     s=list()
     f=list()
     if __name__ == '__main__':
         for _ in range(int(input())):
             name = input()
             score = float(input())
             a.append([name,score])
             s.append(score)
         s=set(s)
         s=list(sorted(set(s)))
         vote_x=s[1]
         for i in range(len(a)):
             if(a[i][1] == vote_x):
```

```
            f.append(a[i][0])
    f.sort()
    for j in range(len(f)):
        print(f[j])
```

3.0.4 Finding the percenatge

[ ]:
```
if __name__ == '__main__':
    n = int(input())
    s_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        s_marks[name] = scores
    query_name = input()
    a=sum(s_marks[query_name])/len(s_marks[query_name])
    format_a = "{:.2f}".format(a)
    print(format_a)
```

3.0.5 Lists

[ ]:
```
if __name__ == '__main__':
    N = int(input())
    operations=list()
    s=list()
    for i in range(N):
     colects=str(input())
     operations.append(colects)


for j in range(N):

    if(operations[j].split()[0] == 'pop'):
       s.pop()

    elif(operations[j].split()[0] == 'remove'):
       s.remove(int(operations[j].split()[1]))

    elif(operations[j].split()[0] == 'reverse'):
       s.reverse()

    elif(operations[j].split()[0] == 'append'):
       s.append(int(operations[j].split()[1]))

    elif(operations[j].split()[0] == 'print'):
       print(s)
    elif(operations[j].split()[0] == 'insert'):
       s.insert(int(operations[j].split()[1]),int(operations[j].split()[2]))

    elif(operations[j].split()[0] == 'sort'):
       s.sort()


    else:
       print('Error')
```

3.0.6 Tuples

[ ]:
```
if __name__ == '__main__':
    n = int(input())
    t=tuple(map(int, input().split()))
    th=hash(t)
    print(th)
```

4 Strings

4.0.1 sWAP cASE

[ ]:
```
def swap_case(s):
    s=list(s)
    s_new=list()
    for i in range(len(s)):

        if(s[i].islower() == True):
            s[i]=s[i].upper()
        else:
            s[i]=s[i].lower()

    return(''.join(s))



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

4.0.2 String Split and Join

[ ]:
```python
def split_and_join(line):
    # write your code here
    line=line.split(" ")
    line="-".join(line)
    return line
if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)
```

4.0.3 What's Your Name?

[ ]:
```python
def print_full_name(first, last):
    # Write your code here
    print('Hello '+ first +' ' + last +'!' + ' '+ 'You just delved into python.
 ↪' )
if __name__ == '__main__':
    first_name = input()
    last_name = input()
    print_full_name(first_name, last_name)
```

4.0.4 Mutations

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

if __name__ == '__main__':
    s = input()
    i, c = input().split()
    s_new = mutate_string(s, int(i), c)
    print(s_new)
```

4.0.5 Find a string

[ ]:
```python
def count_substring(string, sub_string):
    counter=0
    n=len(sub_string)
    for i in range(0, len(string)):
        if(i<len(string)-(n-2)):

            if(string[i:i+n] == sub_string):
              counter = counter+1
              else:
               counter=counter
         else:
           counter=counter
    return(counter)



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

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

4.0.6 String Validators

[ ]:
```
if __name__ == '__main__':
    s = input()
    a=list()
    b=list()
    c=list()
    d=list()
    e=list()
    counter=list()
    kl=0

    for i in range(0,len(s)):
     a.append(s[i].isalnum())
     b.append(s[i].isalpha())
     c.append(s[i].isdigit())
     d.append(s[i].islower())
     e.append(s[i].isupper())
    letters_vect=[a,b,c,d,e]


    for y in range(len(letters_vect)):
        kl=0
        for g in range(len(s)):
         kl += int(letters_vect[y][g])
        if(kl == 0):
          counter.append(False)
        else:
         counter.append(True)
        print(counter[y])
```

4.0.7 Text Alignment

[ ]:
```python
#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))
```

4.0.8 Text Wrap

[ ]:
```python
import textwrap

s=list()
def wrap(string, max_width):
   w = textwrap.TextWrapper(width=max_width)

   w_list = w.wrap(text=string)

   return("\n".join(w_list))

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

4.0.9 Designer Door Mat

[ ]:
```
N, M = map(int,input().split())
for i in range(N//2):
    t = int((2*i)+1)
    print(('.|.'*t).center(M, '-'))
print('WELCOME'.center(M,'-'))
for i in reversed(range(N//2)):
    t = int((2*i)+1)
    print(('.|.'*t).center(M, '-'))
```

4.0.10 String Formatting

[ ]:
```
def print_formatted(number):
    # your code goes here
    l = len(str(bin(number))) - 2
    for i in range(1, number+1):

        print(str(i).rjust(l, " "),str(oct(i))[2:].rjust(l, " "),
        str(hex(i).upper())[2:].rjust(l, " "),str(bin(i))[2:].rjust(l," "))
if __name__ == '__main__':
    n = int(input())
    print_formatted(n)
```

4.0.11 Alphabet Rangoli

[ ]:
```
import string
alphabet = list(string.ascii_lowercase)

def print_rangoli(size):
 size=size
 lettere_sx=str()
 lettere_dx=str()
 for i in range(size-1):
    lettere_sx=str()
    lettere_dx=str()
    t=2*(size-i-1)
    stringa='-'*t
    cc=str(alphabet[size-1-i])
    if(i==0):
     lettere_sx=str()
     lettere_dx=str()
    else:

     for i in range(i):
      lettere_sx = (lettere_sx + str(alphabet[size-1-i])+'-')
     # lettere_dx = (lettere_dx +'-'+ str(alphabet[size-1-i]))
     #print(lettere_sx)
     #print(lettere_dx)
     for i in reversed(range(i+1)):
      lettere_dx = (lettere_dx +'-'+ str(alphabet[size-1-i]))
     #print(lettere_dx)




    stringa_f=stringa + lettere_sx + cc + lettere_dx + stringa
    print(stringa_f)



#printing middle line

 for i in range(size-1):
  print(str(alphabet[size-1-i])+'-', end='')
 print('a', end='')
 for i in reversed(range(size-1)):
  print('-'+str(alphabet[size-1-i]), end='')
 print()

#printing last half



 for i in reversed(range(size-1)):
    lettere_sx=str()
    lettere_dx=str()
    t=2*(size-i-1)
    stringa='-'*t
    cc=str(alphabet[size-1-i])
    if(i==0):
     lettere_sx=str()
     lettere_dx=str()
    else:

     for i in range(i):
      lettere_sx = (lettere_sx + str(alphabet[size-1-i])+'-')
     # lettere_dx = (lettere_dx +'-'+ str(alphabet[size-1-i]))

     #print(lettere_sx)
     #print(lettere_dx)
     for i in reversed(range(i+1)):
      lettere_dx = (lettere_dx +'-'+ str(alphabet[size-1-i]))
     #print(lettere_dx)




    stringa_f=stringa + lettere_sx + cc + lettere_dx + stringa
    print(stringa_f)


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

4.0.12 Capitalize!

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

def solve(s):
    input_n=list(map(str,s.split(' ')))
    print(input_n)
    for i in range(len(input_n)):
        input_n[i]=input_n[i].capitalize()
    s=' '.join(input_n)
    return s



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

    s = input()

    result = solve(s)

    fptr.write(result + '\n')

    fptr.close()
```

4.0.13 Merge the tools

[ ]:
```python
def merge_the_tools(string, k):
    # your code goes here
    n = len(string)
    h = n // k
    s = list()
    for i in range(1,n+1):
        if(string[i-1] not in s):
            s.append(string[i-1])
        if i % k == 0:
            print("".join(s))
            s=list()






if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

4.0.14 The Minion Game

(Helped a bit with the solutions)

[ ]:
```python
def minion_game(string):
    s =0
    k =0
    n=len(string)
    for i in range(n):
        if string[i] in ["A","E","I","O","U"]:
            k =k+n-i
        else:
            s =s+n-i
    if(s>k):
        print("Stuart"+" "+str(s))
    elif(k==s):
        print("Draw")
    else:
        print("Kevin"+" "+str(k))

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

5 Sets

5.0.1 Introduction to Sets

[ ]:
```python
def average(array):
    # your code goes here
    N=len(set(array))
    z=sum(set(array))/N
    return z
if __name__ == '__main__':
    n = int(input())
    arr = list(map(int, input().split()))
    result = average(arr)
    print(result)
```

5.0.2 No Idea!

[ ]:
```python
h = 0
n,m= map(int, input().split())
array=list(map(int,input().split()))
a=set(map(int,input().split()))
b=set(map(int,input().split()))

for j in array:
    if j in a:
        h = h+1
    elif j in b:
        h = h-1
    else:
        continue

print(h)
```

5.0.3 Symmetric Difference

[ ]:
```python
M=int(input())
a=input()
list_a=a.split()
int_a= list(map(int,list_a))
set_a= set(int_a)
N=int(input())
b=input()
list_b=b.split()
int_b= list(map(int,list_b))
set_b= set(int_b)
bf=set_b.difference(set_a)
af=set_a.difference(set_b)
```

```
f=af.union(bf)
f=list(f)
f.sort()
for i in range(len(f)):
 print(f[i])
```

5.0.4 Set.add()

[ ]:
```
N=int(input())
country=set()
for i in range(N):
 collect=str(input())
 country.add(collect)
print(len(country))
```

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

[ ]:
```
operations=list()
n = int(input())
s = set(map(int, input().split()))
m=int(input())
#r1=str(input())
#r2=str(input())
#print(r1.split()[0])

for i in range(m):
 collects=str(input())
 operations.append(collects)

for j in range(m):
 if(operations[j].split()[0] == 'pop'):
  s.pop()
 elif(operations[j].split()[0] == 'remove'):
  s.remove(int(operations[j].split()[1]))

 elif(operations[j].split()[0] == 'discard'):
  s.discard(int(operations[j].split()[1]))

 else:
  print('Error')
print(sum(s))
```

5.0.6 Set.union() Operation

[ ]:
```
n=int(input())
a=set(map(int,input().split()))
m=int(input())
b=set(map(int,input().split()))

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

5.0.7 Set.difference() Operation

[ ]:
```
n=int(input())
a=set(map(int,input().split()))
m=int(input())
b=set(map(int,input().split()))

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

5.0.8 Set.symmetric_difference() Operation

[ ]:
```
n=int(input())
a=set(map(int,input().split()))
m=int(input())
b=set(map(int,input().split()))

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

5.0.9 Set Mutations

[ ]:
```
n = int(input())
a = set(map(int, input().split()))
N = int(input())
for _ in range(N):
    action,nu=input().split()
    new_set=set(map(int, input().split()))
    if action=="update":
        a.update(new_set)
    elif action=="intersection_update":
        a.intersection_update(new_set)
    elif action=="difference_update":
        a.difference_update(new_set)
    elif action=="symmetric_difference_update":
        a.symmetric_difference_update(new_set)
print(sum(a))
```

5.0.10 The Captain's Room

[ ]:
```
from collections import Counter
k=int(input())
q=list(map(int,input().split()))

#lista=set(q)
#lista.remove(' ')
#lista=list(lista)
a=Counter(q)

for j in a:
    if(a[j]!=k):
        print(j)
```

5.0.11 Check Subset

[ ]:
```
for _ in range(int(input())):
    n, a = (int(input()),set(map(int, input().split())))
    m, b = (int(input()),set(map(int, input().split())))
    print(a.intersection(b)==a)
```

5.0.12 Check Strict Subset

[ ]:
```
a = set(map(int, input().split()))
counter=list()
for _ in range(int(input())):
    b=set(map(int,input().split()))
    z=b.union(a)
    counter.append(int(z==a))
print(sum(counter)==len(counter))
```

6 Collections

6.0.1 collections.Counter()

[ ]:
```
from collections import Counter
x= int(input())
y= list(input().split())
N= int(input())
#Counter(y)
set1=Counter(y)

earned=0
for _ in range(N):
     size,money=input().split()
     if(set1[size]>0):
        set1[size]-=1
        earned= earned+int(money)
print(earned)
```

6.0.2 DefaultDict Tutorial

[ ]:
```
from collections import defaultdict
n,m=input().split()
a=list()
b=list()
positions=list()
for _ in range(int(n)):
 a.append(input())
for _ in range(int(m)):
 b.append(input())

for i in b:
    if i in a:
        positions=[]
        for j in range(len(a)):
            if a[j]==i:
                positions.append(str(j+1))
        print(' '.join(positions))
        l=[]
    else:
        print(-1)
```

6.0.3 Collections.namedtuple()

[ ]:
```
from collections import namedtuple
N = int(input())
col_names = input().split()
position=col_names.index('MARKS')
marks=list()


for _ in range(N):
    student_i=input().split()
    m=int(student_i[position])
    marks.append(m)




print(sum(marks)/len(marks))
```

6.0.4 Collections.deque()

[ ]:
```python
from collections import deque
n=int(input())
comands=list()
d=deque()
for i in range(n):
    comands.append(input().split())
for i in range(n):
    if(comands[i][0]=='append'):
        d.append(comands[i][1])
    elif(comands[i][0]=='appendleft'):
        d.appendleft(comands[i][1])
    elif(comands[i][0]=='pop'):
        d.pop()
    elif(comands[i][0]=='popleft'):
        d.popleft()
    else:
        print('error')
for i in range(len(d)):
 print(d[i],end=' ')
```

6.0.5 Collections.OrderedDict()

[ ]:
```python
from collections import OrderedDict
ordinary_dictionary = {}
n=int(input())
items=list()
prices=list()
for i in range(n):
    lista = list(map(str,input().split()))

    if(len(lista)>2):
        items = str(lista[0]+' '+lista[1])
        prices = int(lista[2])
    else:
        items = lista[0]
        prices = int(lista[1])

    if(items in ordinary_dictionary):
        ordinary_dictionary[items] =int(prices) + ordinary_dictionary[items]
    else:
        ordinary_dictionary[items] = int(prices)
for items, prices in ordinary_dictionary.items():
    print(items,prices)
```

6.0.6 Word Order

[ ]:
```
from collections import OrderedDict
d = OrderedDict()
n=int(input())
for i in range(n):
    a = input()
    if a in d:
        d[a] = d[a] + 1
    else:
        d[a] = 1
h=len(d.items())
print(h)
print(*d.values())
```

6.0.7 Compay Logo

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys
from collections import Counter


if __name__ == '__main__':
    s = input()
    s = sorted(s)
    c= Counter(s)
    a=c.most_common(3)
    for i in a:
     print(i[0],i[1])
```

6.0.8 Piling Up!

[ ]:
```
from collections import deque
T=int(input())
for _ in range(T):
    n=int(input())
    blocks=deque(map(int, input().split()))
    max_ = max(blocks)
    if (max_ == blocks[0] or max_== blocks[-1]):
        print('Yes')
    else:
        print('No')
```

7 Date and Time

7.0.1 Calendar Module

[ ]:
```
import calendar
x=str(input())
m=int(x[0:2])
d=int(x[3:5])
y=int(x[6:10])
stamp=calendar.day_name[calendar.weekday(y,m,d)].upper()
print(stamp)
```

7.0.2 Time Delta

[ ]:
```
import math
import os
import random
import re
import sys
import dateutil.parser
# Complete the time_delta function below.
def time_delta(t1, t2):
 date1 = dateutil.parser.parse(t1,fuzzy=True)
 date2 = dateutil.parser.parse(t2,fuzzy=True)
 diff = abs(date2 - date1)
 return(str(int(diff.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()
```

8 Exceptions

8.0.1 Exceptions

[ ]:
```
n=int(input())
a=int()
b=int()
for _ in range(n):
    try:
     a,b=input().split()
     a=int(a)
     b=int(b)
     print(int(a/b))
    except ZeroDivisionError as e:
        print("Error Code: integer division or modulo by zero")
    except ValueError as e:
        print("Error Code:", e)
```

9 Built-Ins

9.0.1 Zipped!

[ ]:
```
N,x= map(int,input().split())
sub=list()
for _ in range(int(x)):
    a = list(map(float, input().split()))
    sub.append(a)

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

9.0.2 ginortS

[ ]:
```
s=list(input())
l,u = '',''
o,e='',''

for i in range(len(s)):
    if(s[i].islower()):
        l=s[i]+l
    elif(s[i].isupper()):
        u=s[i]+u
    elif(s[i].isdigit() and int(s[i])%2 != 0):
        o= s[i]+o
    else:
        e = s[i]+e
```

```
print(''.join(sorted(l))+''.join(sorted(u))+''.join(sorted(o))+''.
 ↪join(sorted(e)))
```

9.0.3 Athlete Sort

[ ]:
```
#!/bin/python3

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)
```

10 Python Functionals

10.0.1 Map and Lambda Function

[ ]:
```
cube = lambda x: pow(x,3)
def fibonacci(n):

 f0=0
 f1=1
 fi = list()
 for _ in range(n):
     fi.append(f0)
     f0, f1 = f1, f0+f1
 return(fi)
```

```
if __name__ == '__main__':
    n = int(input())
    print(list(map(cube, fibonacci(n))))
```

11 Regex and Parsing challenges

11.0.1 Re.split()

[ ]:
```
regex_pattern = r"[,.]"        # Do not delete 'r'.
import re
print("\n".join(re.split(regex_pattern, input())))
```

11.0.2 Group(), Groups() & Groupdict()

[ ]:
```
import re
s=input()
verify=re.search(r"([a-z0-9A-Z])\1",s)
if verify != None:
    print(verify.groups()[0])
else:
    print(-1)
```

11.0.3 Validating phone numbers

[ ]:
```
import re
n=int(input())
for _ in range(n):
    number=input()
    if(re.match(r'[0-9]\d{9}$', number)):
     if (int(number[0]) == 7 and len(number)== 10):
        print('YES')
     elif (int(number[0]) ==  8  and len(number)== 10):
        print('YES')
     elif (int(number[0]) ==  9  and len(number)== 10):
        print('YES')
     else:
        print('NO')
    else:
        print('NO')
```

11.0.4 Validating Roman Numerals

[ ]:
```
regex_pattern = r"(M{0,3})(C[DM]|D?C{0,4})(X[LC]|L?X{0,3})(I[VX]|V?
↪I{0,3})$"        # Copied from stackoverflow.com
```

```
import re
print(str(bool(re.match(regex_pattern, input()))))
```

11.0.5 Validating and Parsing Email Addresses

[ ]:
```
import re
n=int(input())
pattern = r"<[a-z][a-zA-Z0-9\-\.\_]+\@[a-zA-Z]+\.[a-zA-Z]{1,3}>" # Copied from␣
 ↪stackoverflow.com
for _ in range(n):
    s=input()
    name, mail = list(map(str, s.split()))
    if(re.match(pattern, mail)):
        print(s)
```

11.0.6 Hex Color Code

[ ]:
```
import re
pattern = re.compile(r"[\s:](#[0-9A-Fa-f]{3,6})") # Copied from stackoverflow.
 ↪com
n = int(input())

for _ in range(n):
    a = input()
    m = pattern.findall(a)
    if m :
        print(*m, sep='\n')
```

11.0.7 Validating UID

[ ]:
```
import re
n=int(input())
for _ in range(n):
    s=input()
    m = re.search(r"^(?=(?:[a-z\d]*[A-Z]){2})(?=(?:\D*\d){3})(?:([a-zA-Z\d])(?!.
 ↪*\1)){10}$", s) # Copied from stackoverflow.com
    if m:
        print('Valid')
    else:
        print('Invalid')
```

11.0.8 Re.start() & Re.end()

```
[ ]: import re
     s=input()
     k=input()
     n=len(k)-1
     m = list(re.finditer(r'(?={})'.format(k), s))
     if not m:
         print((-1,-1))
     else:
         for i in m:
                 print((i.start(), i.end() + n))
```

11.0.9 Detect Floating Point Nunmber

```
[ ]: import re
     n = int(input())
     path=r'^[-+]?[0-9]*\.[0-9]+$'
     for i in range(n):
         prov=input()
         if re.match(path, prov):
             print('True')
         else:
             print('False')
```

11.0.10 Re.findall() & Re.finditer()

```
[ ]: import re

     s = input()
     m = re.findall("(?
       ↪<=[QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm])[aeiouAEIOU]{2,}(?
       ↪=[QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm])", s)
     if m:
         for i in range(len(m)):
           print(m[i])
     else:
      print(-1)
```

11.0.11 Validating Credit Card

```
[ ]: import re
     n=int(input())
     for _ in range(n):
         number=input()
```

```
    if(re.match(r'^[456]\d{3}(-?\d{4}){3}$', number)): # Copied from␣
↪stackoverflow.com
        if(re.search(r'([0-9])(-?\1){3}', number)):
            print('Invalid')
        else:
            print('Valid')
    else:
        print('Invalid')
```

11.0.12 Regex Substitution

(Helped with the solutions)

[ ]:
```
import re
n = int(input())
p1 = r'(?<=\ )\|\|(?=\ )'
p2 = r'(?<=\ )\&\&(?=\ )'


for i in range(n):
    t = input()
    t = re.sub(p1, 'or', t)
    t = re.sub(p2, 'and', t)
    print(t)
```

11.0.13 Validating Postal Codes

[ ]:
```
regex_integer_in_range = r"^[1-9]\d{5}$"        # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)" # Copied from␣
↪stackoverflow.com


import re
P = input()

print (bool(re.match(regex_integer_in_range, P))
and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

11.0.14 HTML Parser-Part 1

[ ]:
```
from html.parser import HTMLParser
from html.entities import name2codepoint

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Start :", tag)
        for name,value in attrs:
            print(f"-> {name} > {value}")
    def handle_endtag(self, tag):
        print("End   :", tag)

    def handle_startendtag(self, tag, attrs):
        print("Empty :", tag)
        for name,value in attrs:
            print(f"-> {name} > {value}")



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

11.0.15 HTML Parser-Part 2

(helped with solutions)

[ ]:
```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
  def handle_comment(self, data):
          if data != '\n':
            if "\n" in data:
                print(">>> Multi-line Comment")
                print(data)
            else:
                print(">>> Single-line Comment")
                print(data)
  def handle_data(self, data):
        if len(data) > 1:
            print(">>> Data")
            print(data)

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

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

11.0.16 Detect HTML Tags, Attributes and Attribute Values

[ ]:
```
from html.parser import HTMLParser
from html.entities import name2codepoint

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print( tag)
        for name,value in attrs:
            print(f"-> {name} > {value}")


    def handle_startendtag(self, tag, attrs):
        print( tag)
        for name,value in attrs:
            print(f"-> {name} > {value}")



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

11.0.17 Matrix Script

(Helped with Solutions)

[ ]:
```
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)


p=r'(?<=[0-9a-zA-Z])+[^0-9a-zA-Z]+(?=[0-9a-zA-Z])'
iniz = ''
for j in range(0,m):
    for i in range(0,n):
            iniz = iniz + matrix[i][j]

print(re.sub(p,' ',iniz))
```

12 XML

12.0.1

12.0.2 XML 1 - Find the Score

[ ]:
```
import sys
import xml.etree.ElementTree as etree

def get_attr_number(node):
    # your code goes here
    n=len(node.attrib)
    a=list()
    for i in node:
        a.append(get_attr_number(i))

    f=sum(a)
    r=n+f
    return(r)

if __name__ == '__main__':
    sys.stdin.readline()
    xml = sys.stdin.read()
    tree = etree.ElementTree(etree.fromstring(xml))
    root = tree.getroot()
    print(get_attr_number(root))
```

12.0.3 XML 2 - Find the Maximum Depth

[ ]:
```
import xml.etree.ElementTree as etree

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

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

13 Closures and Decorators

13.0.1 Standardize Mobile Number Using Decorators

[ ]:
```
def wrapper(f):
    def fun(l):
        for j in range(len(l)):
            n=len(l[j])
            if(l[j][0] == "0" and n == 11) :
                l[j] = "+91 " + l[j][1:6] + " " + l[j][6:11]
            if(l[j][0:2] == "91" and n == 12) :
                l[j] = "+91 " + l[j][2:7] + " " + l[j][7:12]
            if(n == 10) :
                l[j] = "+91 " + l[j][0:5] + " " + l[j][5:10]
            if(l[j][0:3] == "+91" and n == 13) :
                l[j] = l[j][0:3] + " " + l[j][3:8] + " " + l[j][8:13]
        f(l)
    return fun

@wrapper
def sort_phone(l):
    print(*sorted(l), sep='\n')

if __name__ == '__main__':
    l = [input() for _ in range(int(input()))]
    sort_phone(l)
```

13.0.2 Decorators 2 - Name Directory

[ ]:
```
import operator

def person_lister(f):
    def inner(people):
        for j in range(len(people)):
           people[j][2] = int(people[j][2])

        people.sort(key=operator.itemgetter(2))

        lista = list()
        for i in people:
            lista.append(f(i))

        return lista
    return inner

@person_lister
def name_format(person):
    return ("Mr. " if person[3] == "M" else "Ms. ") + person[0] + " " +␣
 ↪person[1]

if __name__ == '__main__':
    people = [input().split() for i in range(int(input()))]
    print(*name_format(people), sep='\n')
```

14 Numpy

14.0.1 Arrays

[ ]:
```
import numpy

def arrays(arr):
    # complete this function
    # use numpy.array
 s=numpy.array(arr,float)
 s=numpy.flip(s)
 return(s)
arr = input().strip().split(' ')
result = arrays(arr)
print(result)
```

14.0.2 Min and Max

[ ]:
```
import numpy as np
n, m = np.array(input().split(), int)
a=list()
for _ in range(n):
    a.append(input().split())
a = np.max(np.min(np.array(a, int), axis =1))
print(a)
```

14.0.3 Inner and Outer

[ ]:
```
import numpy
a=numpy.array(input().split(),int)
b=numpy.array(input().split(),int)
print (numpy.inner(a, b))
print (numpy.outer(a, b))
```

14.0.4 Polynomials

[ ]:
```
import numpy

s=input().split()
x=int(input())
s=numpy.array(s,float)
print(numpy.polyval(s,x))
```

14.0.5 Dot and Cross

[ ]:
```
import numpy
n=int(input())
a=[]
b=[]
for i in range(n):
    a.append(list(map(int, input().split())))
for i in range(n):
    b.append(list(map(int, input().split())))
a=numpy.array(a)
b=numpy.array(b)
print(numpy.dot(a, b))
```

14.0.6 Shape and Reshape

[ ]:
```
import numpy
s=input().split()
s=numpy.array(s,int)
s.shape=(3,3)
print(s)
```

14.0.7 Sum and Prod

[ ]:
```
import numpy as np
n, m = np.array(input().split(), int)
a=list()
for _ in range(n):
    a.append(input().split())
a=np.sum(np.array(a, int), axis =0)
```

```
print(np.prod(a))
```

14.0.8 Concatenate

[ ]:
```
import numpy as np
n, m,p = list(map(int, input().split()))
a=list()
b=list()
for _ in range(n):
    a.append(input().split())
for _ in range(m):
     b.append(input().split())
a = np.array(a, int)
b = np.array(b, int)

print(np.concatenate((a, b), axis = 0))
```

14.0.9 Floor, Ceil and Rint

[ ]:
```
import numpy as np
np.set_printoptions(legacy='1.13')
a= list(map(float, input().split()))
a=np.array(a,float)
print(np.floor(a))
print(np.ceil(a))
print(np.rint(a))
```

14.0.10 Array Mathematics

[ ]:
```
import numpy as np
n, m = list(map(int, input().split()))
a=list()
b=list()
for _ in range(n):
    a.append(input().split())
for _ in range(n):
    b.append(input().split())
a=np.array(a,int)
b=np.array(b,int)
print(np.add(a, b))
print(np.subtract(a, b))
print(np.multiply(a, b))
print(np.floor_divide(a, b))
print(np.mod(a, b))
print(np.power(a, b))
```

14.0.11 Zeros and Ones

[ ]:
```
import numpy as np
n = list(map(int, input().split()))
a = np.zeros(n, int)
b = np.ones(n, int)
print(a)
print(b)
```

14.0.12 Eye and Identity

[ ]:
```
import numpy as np
np.set_printoptions(legacy='1.13')
n,m = list(map(int, input().split()))
print(np.eye(n,m))
```

14.0.13 Linear Algebra

[ ]:
```
import numpy as np
n = int(input())
a = []
for i in range(n):
    a.append(list(map(float,input().split())))
print(round(np.linalg.det(a),2))
```

14.0.14 Transpose and Flatten

[ ]:
```
import numpy as np
n,m=list(map(int, input().split()))
a=list()
for _ in range(n):
    a.append(list(map(int, input().split())))
a = np.array(a)
print(np.transpose(a))
print(a.flatten())
```

14.0.15 Mean, Var and Std

[ ]:
```
import numpy as np
n,m=list(map(int, input().split()))
a=list()
for _ in range(n):
    a.append(list(map(int, input().split())))
a = np.array(a)
print(np.mean(a,axis=1))
print(np.var(a,axis=0))
print(round(np.std(a,axis=None),11))
```

15 Problem 2

15.0.1 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):
    # Write your code here
    n=len(candles)
    tallest=0
    count=0
    for i in range(n):
        if(int(candles[i])>tallest):
            tallest=candles[i]
    for j in range(n):
        if(int(candles[j])==tallest):
            count=count+1
    return(count)


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()
```

15.0.2 Number Line Jumps

[ ]:
```python
#!/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):
    # Write your code here
    counter = 0
    while x1 != x2:
        x1 = v1 +x1
        x2 = v2 +x2
        counter = counter + 1
        if x1 == x2 :
            return('YES')
        if counter > 10000000 :
            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()
```

15.0.3 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):
    # Write your code here
    likes=list()
    floor=0
    for i in range(n):
        if(i==0):
            floor=math.floor(5/2)
            likes.append(int(floor))
        else:
         recipients=floor*3
         floor=math.floor(int(recipients)/2)
         likes.append(int(floor))
    return(sum(likes))

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

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

    result = viralAdvertising(n)

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

15.0.4 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_=sum1(n)
    string=str(int(sum_)*int(k))
    return sum2(string)

def sum2(s):
    if len(s) > 1:

     return sum2(sum1(s))
    else:
     return s

def sum1(n):
    sum_=0
    for i in list(n):
        sum_=sum_+ int(i)
    sum_ = str(sum_)
    return(sum_)


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()
```

15.0.5 Insertion Sort-Part1

[ ]:
```
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):
    save=()
    for j in range((n-1),0,-1):
        if arr[j] < arr[j-1]:
            save= arr[j]
            arr[j] = arr[j-1]
            print(*arr)
            arr[j-1] = save
        else:
            arr[j]=arr[j]

    print(*arr)



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

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

    insertionSort1(n, arr)
```

15.0.6 Insertion Sort-Part2

[ ]:
```
#!/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):
     save=()
     for i in range(1,n):
        for j in range(0, i+1):
            if (arr[j] > arr[i]):
                save = arr[j]
                arr[j] = arr[i]
                arr[i] = save
            else:
             arr[j]=arr[j]
             arr[i]=arr[i]
        print(* arr)

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

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

    insertionSort2(n, arr)
```