ADM Assignment 1

Personal Information

Name: Hazem Aboulfotouh

Matricola: 2105193

E-mail: Aboulfotouh.2105193@studenti.uniroma1.it

Problem 1

Hello World

```
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 != 0:
        print('Weird')

    elif n >= 2 and n <= 5:
        print('Not Weird')

    elif n >= 6 and n <= 20:
        print('Weird')

    elif 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())

    numbers = [x for x in range(0,n)]

    for x in numbers:
        print(x*x)
```

Write a Function

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

    # Write your logic here
    if year % 4 == 0:
        leap = True
        if year % 100 == 0:
            leap = False
            if year % 400 == 0:
                leap = True


    return leap
```

Print Function

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

    numbers = ''

    for x in range(1, n+1):
        numbers = numbers + (str(x))

    print(numbers)
```

List Comprehensions

```
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())

Possible_permutations = [[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(Possible_permutations)
```

Find the Runner-Up Score!

```
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
import heapq
largest_two = heapq.nlargest(2,set(arr))

print(largest_two[1])
```

Nested Lists

```
if __name__ == '__main__':

    dict1 = []
    scores = []
    import heapq

    for _ in range(int(input())):
        name = input()
        score = float(input())
        scores.append(score)
        dict1.append([name,score])

    scores = set(scores)
    smallest = heapq.nsmallest(2,scores)
    names = []
    for i in range(len(dict1)):
        if dict1[i][1] == smallest[1]:
            names.append(str(dict1[i][0]))

    names.sort()
    for i in names:
        print(i)
```

Finding the Precentage

```
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 = (sum(student_marks[query_name])/len(student_marks[query_name]))

    formatted_avg = f"{average:.2f}"
    print(formatted_avg)
```

Lists

```
if __name__ == '__main__':
    my_list = []

    N = int(input())
    my_list = []


    for _ in range(N):
        command = input().split()
        action = command[0]

        if action == "insert":
            position = int(command[1])
            element = int(command[2])
            my_list.insert(position, element)

        elif action == "print":
            print(my_list)

        elif action == "remove":
            element = int(command[1])
            my_list.remove(element)

        elif action == "append":
```

```
            element = int(command[1])
            my_list.append(element)

        elif action == "sort":
            my_list.sort()

        elif action == "pop":
            my_list.pop()

        elif action == "reverse":
            my_list.reverse()
```

Tuples

```
if __name__ == '__main__':
    n = int(input())
    integer_list = map(int, input().split())

    t = tuple(integer_list)
    print(hash(t))
```

String Split and Join

```
def split_and_join(line):
    # write your code here

    line = line.split(' ')
    return "-".join(line)

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

Whats Yuur Name?

```
def print_full_name(first, last):
    # Write your code here
    print('Hello',first, last+'! You just delved into python.')
```

Mutations

```
def mutate_string(string, position, character):
    new_string = string[:position] + character + string[position + 1:]
    return new_string
```

Find a String

```
def count_substring(string, sub_string):

    c = 0
    for i in range(len(string) - len(sub_string) + 1):
        if string[i:i + len(sub_string)] == sub_string:
            c += 1
    return c
```

Swap Case

```
def swap_case(s):

    new_s = []

    for i in s:
        if i.isupper():
            i = i.lower()
            new_s.append(i)

        elif i.islower():
            i = i.upper()
            new_s.append(i)

        else:
            new_s.append(i)
    return ''.join(new_s)
```

String Validators

```
if __name__ == '__main__':

  s = input()

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

Text Wrap

```
def wrap(string, max_width):

    words = []
    for i in range(0, len(string), max_width):
         words.append(string[i:i+max_width])

    strings = '\n'.join(words)

    return strings
```

Designer Door Mat

```
def door_mat(rows, columns):

    pattern = [('.|.' * (2 * i + 1)).center(columns, '-') for i in range(rows // 2)]
    welcome = 'WELCOME'.center(columns, '-')
    mat = '\n'.join(pattern + [welcome] + pattern[::-1])
    return mat
```

```
if __name__ == "__main__":

    i, j = map(int, input().split())
    mat = door_mat(i, j)
    print(mat)
```

String Formatting

```
def print_formatted(number):


    width = len(bin(number)[2:])

    for i in range(1, number + 1):

        decimal = str(i).rjust(width)
        octal = oct(i)[2:].rjust(width)
        hexadecimal = hex(i)[2:].upper().rjust(width)
        binary = bin(i)[2:].rjust(width)

        print(decimal, octal, hexadecimal, binary)
```

Capitalize

```
def solve(s):

    names = s.split(' ')
    capitalized_names = []

    for name in names:
        if name.isalpha():
            capitalized_name = name[0].upper() + name[1:].lower()
            capitalized_names.append(capitalized_name)
        else:
            capitalized_names.append(name)

    return ' '.join(capitalized_names)
```

Introduction to sets

```
def average(array):
    # your code goes here
    distinct = []
    distinct = set(array)
    return sum(distinct)/len(distinct)
```

Symmetric Difference

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

diff = sorted(a.symmetric_difference(b))

for element in diff:
    print(element)
```

Set_add

```
m = int(input())
mylist = []
for i in range(m):
    mylist.append(input())

set_a = set(mylist)

print(len(set_a))
```

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

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

for _ in range(commands):
    command = input().split()

    if s and command[0] == "pop" :
        s.pop()

    elif command[0] == "remove":
        if command[1] in s:
            s.remove(int(command[1]))

    elif command[0] == "discard":
        s.discard(int(command[1]))

print(sum(s))
```

Set .union() Operation

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
n = int(input())
set1 = set(map(int, input().split()))

b = int(input())
set2 = set(map(int, input().split()))

print(len(set(set1.union(set2))))
```

Set .intersection() Operation

```
m = int(input())
s1 = set(map(int, input().split()))
n = int(input())
s2 = set(map(int, input().split()))

inter1 = s1.intersection(s2)
inter2 = s2.intersection(s1)

print(len(inter1.union(inter2)))
```

Set .difference() Operation

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

m = int(input())
s1 = set(map(int, input().split()))
n = int(input())
s2 = set(map(int, input().split()))
```

```
print(len(s1.difference(s2)))
```

Set .symmetric_difference() Operation

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

m = int(input())
s1 = set(map(int, input().split()))
n = int(input())
s2 = set(map(int, input().split()))

print(len(s1.symmetric_difference(s2)))
```

Set Mutations

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

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

for i in range(N):
    operation = input().split()
    operation_set = set(map(int, input().split()))

    if operation[0] == 'intersection_update':
        A.intersection_update(operation_set)

    elif operation[0] == 'update':
        A.update(operation_set)

    elif operation[0] == 'symmetric_difference_update':
        A.symmetric_difference_update(operation_set)

    elif operation[0] == 'difference_update':
        A.difference_update(operation_set)

print(sum(A))
```

The Captain's Room

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

from collections import Counter

k = int(input())
rooms = list(map(int, input().split()))

frequency = Counter(rooms)

for key, value in frequency.items():
    if value == 1:
        print(key)
```

Check Subset

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

T = int(input())

for i in range(T):
```

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

    if A.issubset(B):
        print('True')

    else:
        print('False')
```

Check Strict Superset

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

A = set(map(int, input().split()))
n = int(input())
boolean = False


for i in range(n):
    B = set(map(int, input().split()))
    if not B.issubset(A) or B == A :
        boolean = False
        break
    else:
        boolean = True

print(boolean)
```

No Idea!

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
n, m = list(map(int, input().split()))
arr = list(map(int, input().split()))
A = set(map(int, input().split()))
B = set(map(int, input().split()))
H = 0

for i in arr:
    if i in A:
        H += 1

    elif i in B:
        H -= 1

print(H)
```

Calender Module

```
import datetime

mylist = list(map(int, input().split()))
date_obj = datetime.date(mylist[2], mylist[0], mylist[1])
print(date_obj.strftime("%A").upper())
```

Time Delta

```
#!/bin/python3

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

```
from datetime import datetime

# Complete the time_delta function below.
def time_delta(t1, t2):

    t1_obj = datetime.strptime(t1, '%a %d %b %Y %H:%M:%S %z')
    t2_obj = datetime.strptime(t2, '%a %d %b %Y %H:%M:%S %z')

    t1_seconds = int(t1_obj.timestamp())
    t2_seconds = int(t2_obj.timestamp())

    return abs(t1_seconds - t2_seconds)

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

    t = int(input().strip())

    for _ in range(t):
        t1 = input().strip()
        t2 = input().strip()

        delta = time_delta(t1, t2)

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

    fptr.close()
```

collections.Counter()

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

x = int(input())
shoes = list(map(int, input().split()))
n = int(input())
profit = 0

for i in range(n):
    size, price = input().split()
    if int(size) in shoes:
        profit = profit + int(price)
        shoes.remove(int(size))

print(profit)
```

Default Dict Tutorial

```
# Enter your code here. Read input from STDIN. Print output to STDOUT5
from collections import defaultdict
n, m = input().split()
A = defaultdict(list)
B = []

for i in range(int(n)):
    A[input()].append(i+1)

for i in range(int(m)):

    indicies = A[input()]

    if indicies:
        print(*indicies)
    else:
        print('-1')
```

Collections.namedtuple()

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

```
N = int(input())
columns = list(map(str, input().split()))
for i in range(len(columns)):
    if columns[i] == 'MARKS':
        index = i

grades = []
avg_score = 0

for i in range(N):
    grades.append(list(map(str, input().split())))
    avg_score = avg_score + int(grades[i][index])

avg_score = avg_score/N

print(avg_score)
```

Collections.OrderedDict()

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
N = int(input())
items = {}

for i in range(N):
    item = input().split()
    name = ' '.join(item[:-1])
    price = int(item[-1])

    if name in items:
        items[name] += price
    else:
        items[name] = price

for name, total_price in items.items():
    print(name, total_price)
```

Collections.deque()

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import deque
d = deque()
N = int(input())

for i in range(N):

    command = input().split()

    if command[0] == 'append':
        d.append(int(command[1]))

    elif command[0] == 'appendleft':
        d.appendleft(int(command[1]))

    elif command[0] == 'pop':
        d.pop()

    elif command[0] == 'popleft':
        d.popleft()

print(*d)
```

Word Order

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

N = int(input())
words = []
```

```
for i in range(N):
    words.append(input())

freq = Counter(words)

print(len(set(words)))

values = []
for key, value in freq.items():
    values.append(value)

print(*values)
```

Company Logo

```
#!/bin/python3

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

if __name__ == '__main__':
    s = input()
    chars = list(s)
    freq = Counter(chars)
    sorted_dict = dict(sorted(freq.items(), key=lambda item: (-item[1], item[0])))

    c = 0

    for key, value in sorted_dict.items():
        print(key, value)
        c+=1
        if c == 3:
            break
```

Piling Up!

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

from collections import deque
T = int(input())

for i in range(T):

    n = int(input())
    cubes = deque(map(int, input().split()))
    pile = []


    for i in range(n):

        if cubes[0] >= cubes[-1]:

            pile.append(cubes[0])
            cubes.popleft()

        elif cubes[0] < cubes[-1]:

            pile.append(cubes[-1])
            cubes.pop()

    if pile == sorted(pile, reverse=True):
        print('Yes')

    else:
        print('No')
```

Exceptions

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

T = int(input())

for i in range(T):

    numbers = input().split()
    try:
        A = int(numbers[0])
        B = int(numbers[1])
        result = A // B
        print(result)
    except Exception as e:
        print('Error Code:', e)
```

Zipped

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

N, X = input().split()
grades = []

for i in range(int(X)):

    grade = list(map(float, input().split()))
    grades.append(grade)

for i in range(int(N)):
    student_grade = 0.0
    for j in range(int(X)):
        student_grade += grades[j][i]

    print(student_grade / int(X))
```

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())

    arr_sorted = sorted(arr, key=lambda x: x[k])

    for i in range(len(arr_sorted)):
        print(*arr_sorted[i])
```

ginortS

```
string = input()

sorted_string = sorted(string, key=lambda x: (x.isdigit(), x.isdigit() and int(x) % 2 == 0, x.isupper(), x))

output = ''.join(sorted_string)
print(output)
```

Map and Lambda Function

```
cube = lambda x:x*x*x # complete the lambda function

def fibonacci(n):

    if n == 0:
        return []

    elif n == 1:
        return [0]

    elif n == 2:
        return [0, 1]

    fib = [0, 1]
    for i in range(2, n):
        num = fib[-1] + fib[-2]
        fib.append(num)

    return(fib)
```

Standardize Mobile Number Using Decorators

```
def wrapper(f):
    def fun(l):

        # complete the function
        numbers_reformated = []

        for number in l:

            if len(number) == 10:
                numbers_reformated.append("+91 " + number[:5] + " " + number[5:])

            elif len(number) == 11 and number[0] == '0':

                numbers_reformated.append("+91 " + number[1:6] + " " + number[6:])

            elif len(number) == 12 and number[:2] == "91":
                numbers_reformated.append("+91 " + number[2:7] + " " + number[7:])

            elif len(number) == 13 and number[:2] == "+9":
                numbers_reformated.append("+91 " + number[3:8] + " " + number[8:])



        sorted_numbers = sorted(numbers_reformated)

        f(sorted_numbers)

    return fun
```

Decorators 2 - Name Directory

```
def person_lister(f):
    def inner(people):

        sorted_people = sorted(people, key=lambda x: int(x[2]))
```

```
        formatted = []

        for person in sorted_people:
            formatted.append(f(person))

        return formatted
    return inner
```

Arrays

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

Shape and Reshape

```
import numpy as np

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

Transpose and Flatten

```
import numpy as np

N, M = input().split()
arr = []
for i in range(int(N)):
    row = list(map(int, input().split()))
    arr.append(row)

np_arr = np.array(arr)

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

Concatenate

```
import numpy as np

n, m, p = map(int, input().split())


mat1 = []
mat2 = []

for i in range(n):
    row = list(map(int, input().split()))
    mat1.append(row)

for i in range(m):
    row = list(map(int, input().split()))
    mat2.append(row)

arr1 = np.array(mat1)
arr2 = np.array(mat2)

print (np.concatenate((arr1, arr2), axis = 0))
```

Zeros and Ones

```
import numpy

size = tuple(map(int,input().strip().split()))
print( numpy.zeros(size, int) )
print( numpy.ones(size, int) )
```

Eye and Identity

```
import numpy as np
np.set_printoptions(legacy='1.13')

N, M = map(int, input().split())
print(np.eye(N,M, dtype=float))
```

Array Mathematics

```
import numpy as np

N, M = map(int, input().split())

A = []
B = []

for i in range(N):
    row = list(map(int, input().split()))
    A.append(row)

for i in range(N):
    row = list(map(int, input().split()))
    B.append(row)

A = np.array(A)
B = np.array(B)

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))
```

FLoor, Ceil, and Rint

```
import numpy as np
np.set_printoptions(legacy='1.13')

A = list(map(float, input().split()))
A = np.array(A)

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

Sum and Prod

```
import numpy as np
```

```
N, M = map(int, input().split())
A = []


for i in range(N):
    row = list(map(int, input().split()))
    A.append(row)

A = np.array(A)
column_sum = np.sum(A, axis=0)
product_result = np.prod(column_sum)

print(product_result)
```

Min and Max

```
import numpy as np

N,M = map(int, input().split())
A = []

for i in range(N):
    row = list(map(int, input().split()))
    A.append(row)

A = np.array(A)

output = np.min(A, axis=1)

print(np.max(output))
```

Mean, Var, and STD

```
import numpy as np

N, M = map(int, input().split())
A = []

for i in range(N):

    row = list(map(int, input().split()))
    A.append(row)

A = np.array(A)

print(np.mean(A, axis = 1))
print(np.var(A, axis = 0))
print(round(np.std(A),11))
```

Dot and Cross

```
import numpy as np

N = int(input())
A = []
B = []

for i in range(N):
    row = list(map(int, input().split()))
    A.append(row)

for i in range(N):
    row = list(map(int, input().split()))
    B.append(row)

A = np.array(A)
B = np.array(B)
```

```
print(np.dot(A, B))
```

Inner and Outter

```
import numpy as np
A = list(map(int, input().split()))
B = list(map(int, input().split()))

print(np.inner(A, B))
print(np.outer(A, B))
```

Polynomials

```
import numpy as np

P = list(map(float, input().split()))
X = int(input())

print(np.polyval(P, X))
```

Linear Algebra

```
import numpy as np

N = int(input())
A = []

for i in range(N):
    row = list(map(float, input().split()))
    A.append(row)

print(round(np.linalg.det(A), 2))
```

XML 1 - Find the Score

```
def get_attr_number(node):
    xml = len(node.attrib)
    for i in node:
        xml += get_attr_number(i)
    return xml
```

XML2 - Find the Maximum Dept

```
maxdepth = 0

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

Detect Floating Point Number

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

import re


if __name__ == "__main__":

    T = int(input())
    for _ in range(T):
        s = input().strip()
        if re.match(r'^[+-]?\d*\.\d+$', s):
            try:
                float(s)
                print(True)
            except ValueError:
                print(False)
        else:
            print(False)
```

Re.split()

```
regex_pattern = r'[.,](?=\d)'   # Do not delete 'r'.
```

Group(), Groups() & Groupdict()

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

s = input()
output = -1

for i in range(len(s) - 1):
    if s[i].isalnum() and s[i] == s[i + 1]:
        output = s[i]
        break

print(output)
```

Re.findall() & Re.finditer()

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

import re

s = input()
pattern = r'(?<=[qwrtypsdfghjklzxcvbnmQWRTYPSDFGHJKLZXCVBNM])[aeiouAEIOU]{2,}(?=[qwrtypsdfghjklzxcvbnmQWRTYPSDFGHJKLZXCVBNM])'

matches = re.findall(pattern, s)


if matches:
    for match in matches:
        print(match)
else:
    print(-1)
```

Re.start() & Re.end()

```
import re

text = input().strip()
pattern = input().strip()
```

```
pattern = r'(?=('+pattern+'))'
matches = list(re.finditer(pattern, text))

if matches:
    for match in matches:
        start_index = match.start(1)
        end_index = match.end(1) - 1
        print((start_index, end_index))
else:
    print((-1, -1))
```

Regex Substitution

```
n = int(input())

for _ in range(n):
    line = input()
    while ' && ' in line or ' || ' in line:
        line = line.replace(' && ', ' and ').replace(' || ', ' or ')
    print(line)
```

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})|M{0,3}(CM|CD|D?C{0,3})|M{0,3}(XC|XL|L?X{0,3})|M{0,3}(IX|IV|V?I{0,3
```

Validating phone numbers

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

import re

n = int(input())
pattern = r'^[789]\d{9}$'
for _ in range(n):
    number = input().strip()
    if bool(re.match(pattern, number)):
        print("YES")
    else:
        print("NO")
```

Validating and Parsing Email Addresses

```
import re

n = int(input())
for _ in range(n):
    name, email = input().split()
    pattern = r'^[a-zA-Z][\w\.-]*@[a-zA-Z]+\.[a-zA-Z]{1,3}$'
    match = re.match(pattern, email[1:-1])
    if match:
        print(name, email)
```

Hex Color Code

```
import re

N = int(input())
code = []

for i in range(N):
```

```
    line = input()
    code.append(line)

for i in range(N):
    row = re.findall('(?!\A)(#[a-fA-f_0-9]{6}|#[a-fA-f_0-9]{3})', code[i])
    if row != []:
        for j in row:
            print(j)
```

HTML Parser - Part 1

```
from html.parser import HTMLParser

def starttag(tag, attrs):
    print("Start :", tag)
    for attr in attrs:
        print(f"-> {attr[0]} > {attr[1] if attr[1] else 'None'}")

def endtag(tag):
    print("End   :", tag)

def startendtag(tag, attrs):
    print("Empty :", tag)
    for attr in attrs:
        print(f"-> {attr[0]} > {attr[1] if attr[1] else 'None'}")

def parse_html(code):
    parser = HTMLParser()
    parser.handle_starttag = starttag
    parser.handle_endtag = endtag
    parser.handle_startendtag = startendtag
    parser.feed(code)

n = int(input())
code = ""
for _ in range(n):
    code += input()

parse_html(code)
```

HTML Parser - Part 2

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle_comment(self, data):
        if "\n" in data and data != "\n":
            print(">>> Multi-line Comment \n{}".format(data.strip()))

        elif data != "\n":
            print(">>> Single-line Comment \n{}".format(data.strip()))

    def handle_data(self, data):
        if data != "\n":
            print(">>> Data\n{}".format(data))


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

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

Validating UID

```
import re
n = int(input())

for _ in range(n):
    UID = input().strip()

    if len(re.findall(r'[A-Z]', UID)) < 2:
        print("Invalid")
        continue

    if len(re.findall(r'\d', UID)) < 3:
        print("Invalid")
        continue

    if not re.match(r'^[A-Za-z0-9-]+$', UID):
        print("Invalid")
        continue

    if len(set(UID)) != len(UID):
        print("Invalid")
        continue

    if len(UID) != 10:
        print("Invalid")
    else:
        print("Valid")
```

Detect HTML Tags, Attributes and Attribute Values

```
from html.parser import HTMLParser

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

    def handle_endtag(self, tag):
        pass

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

n = int(input())
code = ""
for _ in range(n):
    code += input() + "\n"

parser = MyHTMLParser()
parser.feed(code)
```

Validating Credit Card Numbers

```
import re

n = int(input())
cards = []

for i in range(n):
    card = input()
    cards.append(card)

for card in cards:
    if re.match(r'^(4|5|6)\d{3}-?\d{4}-?\d{4}-?\d{4}$', card):
        card = card.replace("-", "")
        if re.search(r'(\d)(\1{3,})', card):
            print("Invalid")
        else:
```

```
            print("Valid")
    else:
        print("Invalid")
```

Problem 2

Birthday Cake Candles

```
#!/bin/python3

import math
import os
import random
import re
import sys

def birthdayCakeCandles(candles):
    # Write your code here
    highest = max(candles)
    c = 0
    for candle in candles:
     if candle == highest:
         c +=1
    return c

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()
```

Number Line Jumps

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


def kangaroo(x1, v1, x2, v2):
    # Write your code here
    if v1 == v2:
        if x1 == x2:
            return 'YES'
        else:
            return 'NO'
    elif (x2 - x1) % (v1 - v2) == 0 and (x2 - x1) // (v1 - v2) >= 0:
        return 'YES'
    else:
        return 'NO'

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

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

    x1 = int(first_multiple_input[0])
    v1 = int(first_multiple_input[1])
    x2 = int(first_multiple_input[2])
    v2 = int(first_multiple_input[3])

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

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

Viral Advertising

```
#!/bin/python3

import math
import os
import random
import re
import sys



def viralAdvertising(n):
    # Write your code here
    days = []
    day1 = [1,5,2,2]
    days.append(day1)

    for i in range(1,n):
        day = [i+1, (math.floor(days[i-1][1]/2))*3,((math.floor(days[i-1][1]/2))*3)//2, (((math.floor(days[i-1][1]/2))*3)//2)+ days[i-1][3] ]
        days.append(day)
    return days[n-1][3]

if __name__ == '__main__':

    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    n = int(input().strip())
    result = viralAdvertising(n)
    fptr.write(str(result) + '\n')
    fptr.close()
```

Recursive Digit sum

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

def superDigit(n, k):
    # Write your code here

    summ = 0

    for char in n:
        summ =  summ + int(char)

    super_digit = [str(summ) for i in range(k)]
    super_digit = "".join(super_digit)

    while True:
        if len(str(super_digit)) == 1:
            return super_digit

        else:
            x = 0
            for char in str(super_digit):
                x = x + int(char)
            super_digit = x


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()
```

Insertion Sort Part 1

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

def insertionSort1(n, arr):

    element = arr[n - 1]
    i = n - 2

    while i >= 0 and arr[i] > element:
        arr[i + 1] = arr[i]
        print(" ".join(map(str, arr)))
        i = i - 1

    arr[i + 1] = element
    print(" ".join(map(str, arr)))

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

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

    insertionSort1(n, arr)
```

Insertion Sort Part 2

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

def insertionSort2(n, arr):
    # Write your code here

     for i in range(1, n):
        current = arr[i]
        j = i-1

        while j >= 0 and arr[j] > current:
            arr[j+1] = arr[j]
            j = j-1
        arr[j+1] = current

        print(" ".join(map(str, arr)))



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

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

    insertionSort2(n, arr)
```