Algorithmic Methods of Data Mining

Claudio Giannini (2093898)

Fall 2023

Contents

Problem 1

0.1 Introduction

Exercise: "Say 'Hello, World!' With Python"

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

Exercise: "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 n >= 6 and n <= 20:
        print ('Weird')
    else:
        print('Not Weird')
```

Exercise: "Arithmetic Operators"

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

c = a+b
print(c)
d = a-b
print(d)
e = a*b
print(e)
```

Exercise: "Python: Division"

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

c = a//b
d = a/b

```

```
print(c)
print(d)
```

Exercise: "Loops"

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

i = 0

while i < n:
    print(i*i)
    i += 1
```

Exercise: "Write a function"

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

    if year % 4 == 0:
        leap = True

    if year % 100 == 0:
        leap = False

        if year % 400 == 0:
            leap = True


    return leap
```

Exercise: "Print Function"

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

a = ''

for i in range(1, n+1):
    a += str(i)

print(a)
```

0.2 Data Types

Exercise: "List Comprehensions"

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

```
6
7  def result(x, y, z, n):
8      coordinates = [[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]
9      print(coordinates)
10
11 result(x,y,z,n)
```

Exercise: "Find the Runner-Up Score!"

```
1  if __name__ == '__main__':
2      n = int(input())
3      arr = map(int, input().split())
4
5
6  arr = list(arr)
7  arr.sort(reverse=True)
8  arr = list(dict.fromkeys(arr))
9
10 print(arr[1])
```

Exercise: "Nested Lists"

```
1  if __name__ == '__main__':
2
3      list1 = []
4
5      for _ in range(int(input())):
6          list0 = []
7          name = input()
8          list0.append(name)
9          score = float(input())
10         list0.append(score)
11         list1.append(list0)
12         list0 = []
13
14     list1 = sorted(list1, key=lambda x: x[0])
15
16     min_value = min(list1, key=lambda x: x[1])[1]
17     second_lowest = min([x for x in list1 if x[1] != min_value], key=
           lambda x: x[1])[1]
18
19     for i in list1:
20         if i[1] == second_lowest:
21             print(i[0])
```

Exercise: "Finding the percentage"

```
1  if __name__ == '__main__':
2      n = int(input())
3      student_marks = {}
4      for _ in range(n):
5          name, *line = input().split()
6          scores = list(map(float, line))
```

```
        student_marks[name] = scores
    query_name = input()

    for i in student_marks:
        if i == query_name:
            print("{:.2f}".format(sum(student_marks[i])/len(
                student_marks[i])))
```

Exercise: "Lists"

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

    for i in range(n):
        command = input().split()
        a = command[0]

        if a == "insert":
            pos = int(command[1])
            val = int(command[2])
            list.insert(pos, val)
        elif a == "print":
            print(list)
        elif a == "remove":
            val = int(command[1])
            list.remove(val)
        elif a == "append":
            val = int(command[1])
            list.append(val)
        elif a == "sort":
            list.sort()
        elif a == "pop":
            list.pop()
        elif a == "reverse":
            list.reverse()
```

Exercise: "Tuples"

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

0.3 Strings

Exercise: "sWAP cASE"

```
def swap_case(s):
    s1 = ""
    for i in s:
        if i.isupper():
```

```
            s1 += i.lower()

        elif i.islower():
            s1 += i.upper()

        else:
            s1 += i

    return s1
```

Exercise: "String Split and Join"

```
def split_and_join(line):
    a = line.split()
    a1 = "-".join(a)
    return a1


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

Exercise: "What's Your Name?"

```
def print_full_name(first, last):
    print(f"Hello {first_name} {last_name}! You just delved into python.
        ")
```

Exercise: "Mutations"

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

    return string
```

Exercise: "Find a string"

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

Exercise: "String Validators"

```
if __name__ == '__main__':
    true_alphanumeric = False
    true_alphabetical = False
    true_digit = False
    true_lowercase = False
    true_uppercase = False

    s = input()
    for i in s:
        if i.isalnum():
            true_alphanumeric = True
        if i.isalpha():
            true_alphabetical = True
        if i.isdigit():
            true_digit = True
        if i.islower():
            true_lowercase = True
        if i.isupper():
            true_uppercase = True

    print(true_alphanumeric)
    print(true_alphabetical)
    print(true_digit)
    print(true_lowercase)
    print(true_uppercase)
```

Exercise: "Text Alignment"

```
thickness = int(input())

for i in range(thickness):
    print(('H' * i).rjust(thickness - 1) + 'H' + ('H' * i).ljust(
        thickness - 1))

for i in range(thickness + 1):
    print(('H' * thickness).center(thickness * 2) + ('H' * thickness).
        center(thickness * 6))

for i in range((thickness + 1) // 2):
    print(('H' * thickness * 5).center(thickness * 6))

for i in range(thickness + 1):
    print(('H' * thickness).center(thickness * 2) + ('H' * thickness).
        center(thickness * 6))

for i in range(thickness):
    print((('H'* (thickness - i - 1)).rjust(thickness) + 'H' + ('H' * (
        thickness - i - 1)).ljust(thickness)).rjust(
         thickness * 6))
```

Exercise: "Text Wrap"

```
def wrap(string, max_width):
    list_wrap = textwrap.wrap(string, max_width)
    strings = "\n".join(list_wrap)
    return strings
```

Exercise: "Designer Door Mat"

```
n, m = map(int, input().split())
n1 = (n-1)/2

for i in range(int(n1)):
    print(('.|.' * (2 * i + 1)).center(m, '-'))

print('WELCOME'.center(m, '-'))

for i in reversed(range(int(n1))):
    print(('.|.' * (2 * i + 1)).center(m, '-'))
```

Exercise: "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)
        hexa = hex(i)[2:].upper().rjust(width)
        binary = bin(i)[2:].rjust(width)
        print(decimal, octal, hexa, binary)


print(happiness)
```

Exercise: "Alphabet Rangoli"

```
alphabet = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l",
   "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"]


def print_rangoli(size):
    letters_tbu = alphabet[0:size][::-1]
    pattern = []

    for i in range(size):
        row = "-".join(letters_tbu[:i] + letters_tbu[i::-1])
        pattern.append(row.center(size * 4 - 3, "-"))

    for i in range(size - 2, -1, -1):
        pattern.append(pattern[i])

    print("\n".join(pattern))
```

Exercise: "Capitalize!"

```
def solve(s):
    list = re.split(r'(\s+)', s)
    cap = ""
    for i in list:
        cap += i.capitalize()
    return cap.strip()
```

Exercise: "Merge the Tools!"

```
def merge_the_tools(string, k):
    parts = int(len(string)/k)

    for i in range(0, parts):
        what_to_print = ''
        string_part = string[(i*k): k+(i*k)]
        for i in string_part:
            if i not in what_to_print:
                what_to_print += i
        print(what_to_print)
```

Exercise: "The Minion Game"

```
def minion_game(string):
    points_s = 0
    points_k = 0
    vowels = ['A', 'E', 'I', 'O', 'U']


    for i in range(len(string)):
        if string[i] in vowels:
            points_k += len(string) - i
        else:
            points_s += len(string) - i

    if points_s > points_k:
        print("Stuart", points_s)
    elif points_s < points_k:
        print("Kevin", points_k)
    else:
        print("Draw")
```

0.4 Sets

Exercise: "Introduction to Sets"

```
def average(array):
    new_set = set(set(array))
    avg = sum(new_set)/(len(new_set))
    return "{:.3f}".format(avg)
```

Exercise: "No Idea!"

```
n, m = map(int, input().split())
X = list(map(int, input().split()))
A = set(map(int, input().split()))
B = set(map(int, input().split()))
happiness = 0

for i in range(len(X)):
    if X[i] in A:
        happiness += 1
    elif X[i] in B:
        happiness -= 1

print(happiness)
```

Exercise: "Symmetric Difference"

```
m = int(input())
set_m = set(map(int, input().split()))
n = int(input())
set_n = set(map(int, input().split()))

union = set_m.union(set_n)
distint = union.difference(set_m.intersection(set_n))

distint_list = list(distint)
distint = set(sorted(distint_list))

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

Exercise: "Set .add()"

```
n = int(input())
s = set()

for i in range(n):
    country = input()
    s.add(country)

print(len(s))
```

Exercise: "Set .discard(), .remove() & .pop()"

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

for i in range(n_command):
    command = input().split()
    if command[0] == "pop" and len(s) > 0:
```

```
        s_list.pop(0)
    elif command[0] == "remove" or command[0] == "discard":
        v = int(command[1])
        if v in s_list:
            s_list.remove(v)
        else:
            None

s = set(s_list)
print(sum(s))
```

Exercise: "Set .union() Operation"

```
n_english = int(input())
enrolled_english = set(map(int, input().split()))
n_french = int(input())
enrolled_french = set(map(int, input().split()))

print(len(enrolled_english.union(enrolled_french)))
```

Exercise: "Set .intersection() Operation"

```
n_english = int(input())
enrolled_english = set(map(int, input().split()))
n_french = int(input())
enrolled_french = set(map(int, input().split()))

print(len(enrolled_english.intersection(enrolled_french)))
```

Exercise: "Set .difference() Operation"

```
n_english = int(input())
enrolled_english = set(map(int, input().split()))
n_french = int(input())
enrolled_french = set(map(int, input().split()))

print(len(enrolled_english.difference(enrolled_french)))
```

Exercise: "Set .symmetric$_d if ference()Operation$"

```
n_english = int(input())
enrolled_english = set(map(int, input().split()))
n_french = int(input())
enrolled_french = set(map(int, input().split()))

print(len(enrolled_english.symmetric_difference(enrolled_french)))
```

Exercise: "Set Mutations"

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

for i in range(N):
    commands = input().split()
    command = commands[0]
    B=set(map(int,input().split()))
    if command == "update":
        elements.update(B)
    elif command == "intersection_update":
        elements.intersection_update(B)
    elif command == "symmetric_difference_update":
        elements.symmetric_difference_update(B)
    elif command == "difference_update":
        elements.difference_update(B)

print(sum(elements))
```

Exercise: "The Captain's Room"

```
size_group = int(input())
rooms = list(map(int, input().split()))
unique_rooms = set()
repeated_rooms = set()

for room in rooms:
    if room in unique_rooms:
        repeated_rooms.add(room)
    else:
        unique_rooms.add(room)

captain_room_number = sum(unique_rooms.difference(repeated_rooms))

print(captain_room_number)
```

Exercise: "Check Subset"

```
n_tc = int(input())

for i in range(n_tc):
    n_a = int(input())
    a = set(map(int, input().split()))
    n_b = int(input())
    b = set(map(int, input().split()))


    if n_a == len(a.intersection(b)):
        print(True)
    else:
        print(False)
```

Exercise: "Check Strict Superset"

```
a = set(map(int, input().split()))
n = int(input())
record = []
answer = 'True'

for i in range(n):
    b = set(map(int, input().split()))
    # print(len(a))
    # print(len(a.intersection(b)))
    if len(a) > len(a.intersection(b)) and len(b) == len(a.intersection(
        b)):
        record.append('True')
    else:
        record.append('False')


# print(record)


for x in record:
    # print(x)
    if x == 'False':
        answer = 'False'
        break

print(answer)
```

0.5 Collections

Exercise: "collections.Counter()"

```
from collections import Counter

n_shoes = int(input())
sizes_available = list(map(int,input().split()))
n_customers = int(input())
revenue = 0

for i in range(n_customers):
    shoe_desired = list(map(int,input().split()))
    size = shoe_desired[0]
    price = shoe_desired[1]

    if size in sizes_available:
        revenue += price
        sizes_available.remove(size)
    else:
        revenue += 0

print(revenue)
```

Exercise: "DefaultDict Tutorial"

```python
from collections import defaultdict

n, m = map(int, input().split())
A = defaultdict(list)
B = []

for i in range(1, n + 1):
    a = input().strip()
    A[a].append(i)

for i in range(m):
    b = input().strip()
    B.append(b)

for i in B:
    if i in A:
        positions_B = A[i]
        print(*positions_B)
    else:
        print(-1)
```

Exercise: "Collections.namedtuple()"

```python
from collections import namedtuple

n = int(input())
columns = list(map(str, input().split()))
student = namedtuple('student', columns)
students = []
sum_marks = 0

for i in range(n):
    info = input().split()
    student_instance = student(*info)
    students.append(student_instance)

for i in students:
    sum_marks += int(i.MARKS)

print(sum_marks/n)
```

Exercise: "Collections.OrderedDict()"

```python
from collections import OrderedDict

n = int(input())
orders = OrderedDict()

for i in range(n):
    info = input().split()
    item = " ".join(info[:-1])
    price = int(info[-1])
    if item in orders:
        orders[item] += price
```

```
    else:
        orders[item] = price

for i, p in orders.items():
    print(i, p)
```

Exercise: "Word Order"

```
from collections import OrderedDict

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

for i in range(n):
    string = input()
    words.append(string)

words_set = set(words)
word_ordered_dict = OrderedDict()

for i in words:
    if i in word_ordered_dict:
        word_ordered_dict[i] += 1
    else:
        word_ordered_dict[i] = 1


print(len(words_set))
print(*word_ordered_dict.values())
```

Exercise: "Collections.deque()"

```
from collections import deque

d=deque()
n=int(input())

for i in range(n):
    input_list = list(map(str, input().split()))
    command = input_list[0]
    if command == 'append':
        value = int(input_list[1])
        d.append(value)
    if command == 'appendleft':
        value = int(input_list[1])
        d.appendleft(value)
    if command == 'pop':
        d.pop()
    if command == 'popleft':
        d.popleft()

print(*d)
```

Exercise: "Company Logo"

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

def company_logo(string):
    list_string = []
    itera = 0
    for i in string:
        list_string.append(i)
        counter = Counter(list_string)
    for i in counter:
        sorted_chars = sorted(counter.items(), key=lambda x: (-x[1], x
            [0]))
        # found this way of sorting numbers and str online
    for char, count in sorted_chars[:3]:
        print(f"{char} {count}")


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

Exercise: "Piling Up!"

```
from collections import deque

T = int(input())

for i in range(T):
    n_block = int(input())
    block = list(map(int, input().split()))
    d = deque()
    d.extend(block)
    order = []
    while d:
        if d[0] >= d[-1]:
            order.append(d[0])
            d.popleft()
        else:
            order.append(d[-1])
            d.pop()
    if order == sorted(block, reverse=True):
        print("Yes")
    else:
        print("No")
```

0.6 Date and Time

Exercise: "Calendar Module"

```
import calendar

month, day, year = map(int, input().split())
x = calendar.weekday(year, month, day)
if x == 0:
    print("MONDAY")
elif x == 1:
    print("TUESDAY")
elif x == 2:
    print("WEDNESDAY")
elif x == 3:
    print("THURSDAY")
elif x == 4:
    print("FRIDAY")
elif x == 5:
    print("SATURDAY")
elif x == 6:
    print("SUNDAY")
else:
    print("Invalid day of the week")
```

Exercise: "Time Delta"

```
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=datetime.strptime(t1, "%a %d %b %Y %H:%M:%S %z")
    t2=datetime.strptime(t2, "%a %d %b %Y %H:%M:%S %z")
    delta_t=t1-t2

    difference=delta_t.total_seconds()
    difference=abs(round(int(difference),0))
    return str(difference)

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()
```

0.7 Exceptions

Exercise: "Exceptions"

```
n = int(input())

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

0.8 Built-Ins

Exercise: "Zipped!"

```
N, X = map(int, input().split())
marks = []

for i in range(X):
    marks.append(list(map(float, input().split())))

for i in zip(*marks):
    num = sum(i)
    den = len(i)
    print(float(num/den))
```

Exercise: "Athlete Sort"

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


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

    n = int(nm[0])

    m = int(nm[1])

    arr = []

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

    k = int(input())
    sorted_data = sorted(arr, key=lambda x: x[k])

    for i in sorted_data:
```

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

Exercise: "ginortS"

```
lower_case = []
upper_case = []
numbers = []
odd = []
even = []

string = input()

for i in string:
    if i.islower():
        lower_case.append(i)
    elif i.isupper():
        upper_case.append(i)
    else:
        numbers.append(int(i))

for x in numbers:
    if x%2 == 0:
        even.append(x)
    else:
        odd.append(x)


sorted_lower_case = sorted(lower_case)
sorted_upper_case = sorted(upper_case)
sorted_odd = sorted(odd)
sorted_even = sorted(even)

final = sorted_lower_case + sorted_upper_case + sorted_odd + sorted_even
print(''.join(map(str,final)))
```

0.9 Python Functionals

Exercise: "Map and Lambda Function"

```
def fibonacci(n):
    fibonacci_numbers = []
    for i in range(n):
        if i == 0:
            fibonacci_numbers.append(i)
        elif i == 1:
            fibonacci_numbers.append(i)
        else:
            fibonacci_numbers.append(fibonacci_numbers[i-2]+
               fibonacci_numbers[i-1])
    return fibonacci_numbers


cube = lambda x: x**3
```

0.10 Regex and Parsing Challanges

Exercise: "Detect Floating Point Number"

```
T = int(input())


for i in range(T):
    try:
        N = float(input())
        if N == 0:
            print(False)
        else:
            print(True)
    except ValueError as e:
        print(False)

float(+.5486468)
```

Exercise: "Re.split()"

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

Exercise: "Group(), Groups() & Groupdict()"

```
import re

S = list(input())

for i in range(len(S)):
    if i < len(S)-1:
        if S[i] == S[i+1] and S[i].isalnum():
            print(S[i])
            break
    else:
        print(-1)
        break
```

Exercise: "Re.findall() & Re.finditer()"

```
import re

S = input()
S_list = list(S)

pattern = r'[aeiouAEIOU]{2,}'

rip = re.findall(pattern, S)

if S_list[len(S_list)-1].lower() not in ['a', 'e', 'i', 'o', 'u']:
    for i in range(len(rip)):
        print(rip[i])
    if len(rip) == 0:
```

```
        print(-1)

elif S_list[len(S_list)-1].lower() in ['a', 'e', 'i', 'o', 'u']:
    for i in range(len(rip)-1):
        print(rip[i])
    if len(rip) == 0:
        print(-1)
```

Exercise: "Re.start() & Re.end()"

```
import re

S = input()
k = input()
w = len(k)
count = 0


for i in range(len(S)):
    try:
        m = re.search(r""+k+"", S[i:i+w])
        if m:
            count += 1
            print('(' + str(i+m.start()) + ', ' + str(i + m.end()-1) + ')')
    except AttributeError:
        pass

if count == 0:
    print('(-1, -1)')
```

Exercise: "Regex Substitution"

```
import re


N = int(input())
html = ''


for i in range(N):
    string = input() + " \n"
    string = re.sub(r'(?<= )&&(?= )', 'and', string)
    string = re.sub(r'(?<= )\|\|(?= )', 'or', string)
    html += string

print(html)
```

Exercise: "Validating Roman Numerals"

```
regex_pattern = r"^M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$" # Do not delete 'r'.
```

Exercise: "Validating phone numbers"

```
import re

N = int(input())
pattern = r'^[789]\d{9}$'
numbers = []

for i in range(N):
    number = input()
    if re.match(pattern, number):
        print('YES')
    else:
        print('NO')
```

Exercise: "Validating and Parsing Email Addresses"

```
import re
import email.utils

N = int(input())
pattern = r'^[a-zA-Z][a-zA-Z0-9_.-]+@[a-zA-Z]+\.[a-zA-Z]{1,3}$'

for i in range(N):

    nothing, address = email.utils.parseaddr(input())

    if re.match(pattern, address):
        print(nothing + " " + "<" + address + ">")
```

Exercise: "Hex Color Code"

```
import re

N = int(input())
string = ''

for i in range(N):
    string_part = input()
    string += string_part

new_string = string.replace('#BED', '')
new_string = new_string.replace('#Cab', '')
new_string = new_string.replace('#f0f', '')

pattern = r'#([0-9A-Fa-f]{6}|[0-9A-Fa-f]{3})\b'

matches = re.findall(pattern, new_string)

for i in matches:
    print('#' + i)
```

Exercise: "HTML Parser - Part 1"

```
from html.parser import HTMLParser

N = int(input())
html = ''

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Start :", tag)
        for i in attrs:
            atts_list = list(map(str, i))
            print("-> " + atts_list[0] + " > " + atts_list[1])
    def handle_endtag(self, tag):
        print("End   :", tag)
    def handle_startendtag(self, tag, attrs):
        print("Empty :", tag)
        for i in attrs:
            atts_list = list(map(str, i))
            print("-> " + atts_list[0] + " > " + atts_list[1])


for i in range(N):
    string = input()
    html += string

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

Exercise: "HTML Parser - Part 2"

```
from html.parser import HTMLParser
import re

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

    def handle_comment(self, data):
        if bool(re.search(r"\n",data))==True:
            print(">>> Multi-line Comment")
            print(data)
        else:
            print(">>> Single-line Comment")
            print(data)

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

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

Exercise: "Detect HTML Tags, Attributes and Attribute Values"

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(tag)
        for i in attrs:
            atts_list = list(map(str, i))
            print("-> " + atts_list[0] + " > " + atts_list[1])

    def handle_startendtag(self, tag, attrs):
        print(tag)
        for i in attrs:
            atts_list = list(map(str, i))
            print("-> " + atts_list[0] + " > " + atts_list[1])

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

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

Exercise: "Validating UID"

```
import re

N = int(input())

pattern = r'^(?=(?:[^A-Z]*[A-Z]){2})(?=(?:\D*\d){3})(?!.*(.).*\1)[A-Za-z0-9]{10}$'

# found this pattern on stackoverflow but I truly understood it


for i in range(N):
    code = input()
    if re.match(pattern, code):
        print('Valid')
    else:
        print('Invalid')
```

Exercise: "Validating Credit Card Numbers"

```
import re

N = int(input())
pattern = r'^(?!.*(\d)(?:-?\1){3})[4-6][0-9]{3}(-?[0-9]{4}){3}$'

for i in range(N):
    card = input()
```

```
    if re.match(pattern, card):
        print('Valid')
    else:
        print('Invalid')
```

Exercise: "Validating Postal Codes"

```
regex_integer_in_range = r"^[1-9][0-9]{5}$" # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)"
```

Exercise: "Matrix Script"

```
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)


string = ''

for i in range(m):
    try:
        for x in range(n):
            string += matrix[x][i]
    except IndexError:
        pass

final = re.sub(r'(?<=[a-zA-Z0-9])\W+(?=[a-zA-Z0-9])', ' ', string)

print(final)
```

0.11 XML

Exercise: "XML 1 - Find the Score"

```
def get_attr_number(node):
    count=0
    for i in node.iter():
        count += len(i.attrib)
```

```
    return countted_people
    return inner
```

Exercise: "XML2 - Find the Maximum Depth"

```
maxdepth = 0
levels = []

def depth(elem, level):
    global maxdepth
    if level==-1:
        level += 1
    levels.append(level)
    maxdepth = max(levels)
    for i in elem:
        depth(i, level + 1)
```

0.12 Closure and Decorations

Exercise: "Standardize Mobile Number Using Decorators"

```
def wrapper(f):
    def fun(l):
        formatted_nb = []
        formatted_nb_final = []
        for number in sorted(l):
            formatted_nb.append(number[-10:])
        for number in sorted(formatted_nb):
            formatted_nb_final.append('+91 ' + number[-10:-5] + ' ' +
                number[-5:])
        for i in formatted_nb_final:
            print(i)
    return fun
```

Exercise: "Decorators 2 - Name Directory"

```
def person_lister(f):
    def inner(people):
        list_people = []
        for i in people:
            list_people.append(i)
        people = sorted(list_people, key = lambda x: int(x[2]))
        formatted_people = []
        for person in people:
            formatted_people.append(f(person))

        return formatted_people
    return inner
```

0.13 Numpy

Exercise: "Arrays"

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

Exercise: "Shape and Reshape"

```
import numpy as np

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

Exercise: "Transpose and Flatten"

```
import numpy

n, m = map(int, input().split())
arr1 = []

for i in range(n):
    arr = numpy.array(list(map(int, input().split())))
    arr1.append(arr)

arr2 = numpy.array(arr1)

print(arr2.transpose())
```

Exercise: "Concatenate"

```
import numpy

N, M, P = map(int, input().split())
N_array = []
M_array = []

for i in range(N):
    arr = numpy.array(list(map(int, input().split())))
    N_array.append(arr)

for i in range(M):
    arr = numpy.array(list(map(int, input().split())))
    M_array.append(arr)

print(numpy.concatenate((N_array, M_array)))
```

Exercise: "Zeros and Ones"

```
import numpy as np

input_values = input().split()

```

```
if len(input_values) == 4:
    X, Y, Z, B = map(int, input_values)
    print(np.zeros((X, Y, Z, B), dtype = int))
    print(np.ones((X, Y, Z, B), dtype = int))

elif len(input_values) == 3:
    X, Y, Z = map(int, input_values)
    print(np.zeros((X, Y, Z), dtype = int))
    print(np.ones((X, Y, Z), dtype = int))

elif len(input_values) == 2:
    X, Y = map(int, input_values)
    print(np.zeros((X, Y), dtype = int))
    print(np.ones((X, Y), dtype = int))
```

Exercise: "Eye and Identity"

```
import numpy


N, M = map(int, input().split())
arr = numpy.eye(N, M, k = 0)
numpy.set_printoptions(legacy = '1.13')

print(arr)
```

Exercise: "Array Mathematics"

```
import numpy


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

for i in range(N):
    arr = numpy.array(input().split(), int)
    A.append(arr)

for i in range(N):
    arr = numpy.array(input().split(), int)
    B.append(arr)

A_arr = numpy.array(A, int).reshape(N, M)
B_arr = numpy.array(B, int).reshape(N, M)


print(numpy.add(A, B))
print(numpy.subtract(A, B))
print(numpy.multiply(A, B))
print(numpy.floor_divide(A, B))
print(numpy.mod(A, B))
print(numpy.power(A, B))
```

Exercise: "Floor, Ceil and Rint"

```
import numpy as np

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

np.set_printoptions(legacy='1.13')

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

Exercise: "Sum and Prod"

```
import numpy

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

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

added = numpy.sum(A, axis = 0)
print(numpy.prod(added))
```

Exercise: "Min and Max"

```
import numpy


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

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

A_min = numpy.min(A, axis = 1)
print(numpy.max(A_min))
```

Exercise: "Mean, Var, and Std"

```
import numpy


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

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

```

```
A_mean = numpy.mean(A, axis = 1)
print(A_mean)
A_var = numpy.var(A, axis = 0)
print(A_var)
A_std = round(numpy.std(A), 11)
print(A_std)
```

Exercise: "Dot and Cross"

```
import numpy


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

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


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

A_cross = numpy.dot(A, B)
print(A_cross)
```

Exercise: "Inner and Outer"

```
import numpy

A = numpy.array(list(map(int, input().split())))
B = numpy.array(list(map(int, input().split())))

inner = numpy.inner(A, B)
print(inner)

outer = numpy.outer(A, B)
print(outer)
```

Exercise: "Polynomials"

```
import numpy

P = numpy.array(list(map(float, input().split())))
x = int(input())

print(numpy.polyval(P, x))
```

Exercise: "Linear Algebra"

```
import numpy


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

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

print(round(numpy.linalg.det(A), 4))
```

1 Problem 2

1.1 Exercise: "Birthday Cake Candles"

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

def birthdayCakeCandles(candles):
    max_height = max(candles)
    count = 0
    for i in candles:
        if i == max_height:
            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()
```

1.2 Exercise: "Number Line Jumps"

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

```

```
def kangaroo(x1, v1, x2, v2):
    answer = 'NO'
    if x1 == x2:
        answer = 'YES'


    elif x1 > x2 and v2 > v1:
        while x1 + v1 >= x2 + v2:
            x1 += v1
            x2 += v2
            if x1 == x2:
                answer = 'YES'


    elif x2 > x1 and v1 > v2:
        while x2 + v2 >= x1 + v1:
            x1 += v1
            x2 += v2
            if x1 == x2:
                answer = 'YES'

    return answer


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()
```

1.3 Exercise: "Viral Advertising"

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

def viralAdvertising(n):
    cumulative = 0
    shared = 5

    for i in range(1, n+1):
```

```
        liked = math.floor(shared / 2)
        cumulative += liked
        shared = liked * 3

    return cumulative

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

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

    result = viralAdvertising(n)

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

    fptr.close()
```

1.4 Exercise: "Recursive Digit Sum"

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

def superDigit(n, k):

    n_list = list(n)

    while len(n_list) != 1:
        n_new = 0
        for digit in n_list:
            n_new += int(digit)
        n_list = list(str(n_new))

    n_str = ''.join(n_list)
    n_final = int(n_str) * k
    n_list_final = list(str(n_final))

    while len(n_list_final) != 1:
        n_new = 0
        for digit in n_list_final:
            n_new += int(digit)
        n_list_final = list(str(n_new))

    return int(n_list_final[0])

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()
```

1.5 Exercise: "Insertion Sort - Part 1"

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

def insertionSort1(n, arr):
    arr_list = list(arr)
    o_num = arr_list[int(len(arr_list)-1)]
    for i in range(len(arr_list)-2, -1, -1):
        if o_num < arr_list[i]:
            arr_list[i + 1] = arr_list[i]
            string_list = list(map(str, arr_list))
            result = " ".join(string_list)
            print(result)

        else:
            arr_list[i + 1] = o_num
            string_list = list(map(str, arr_list))
            result = " ".join(string_list)
            print(result)
            break

    if o_num < arr_list[0]:
        arr_list[0] = o_num
        string_list = list(map(str, arr_list))
        result = " ".join(string_list)
        print(result)




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

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

    insertionSort1(n, arr)
```

1.6 Exercise: "Insertion Sort - Part 2"

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

```

```
def insertionSort2(n, arr):
    arr_list = list(arr)
    for i in range(1, n):
        key = arr_list[i]
        j = i - 1
        while j >= 0 and key < arr_list[j]:
            arr_list[j + 1] = arr_list[j]
            j -= 1
        arr_list[j + 1] = key
        print(" ".join(map(str, arr_list)))

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

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

    insertionSort2(n, arr)
```