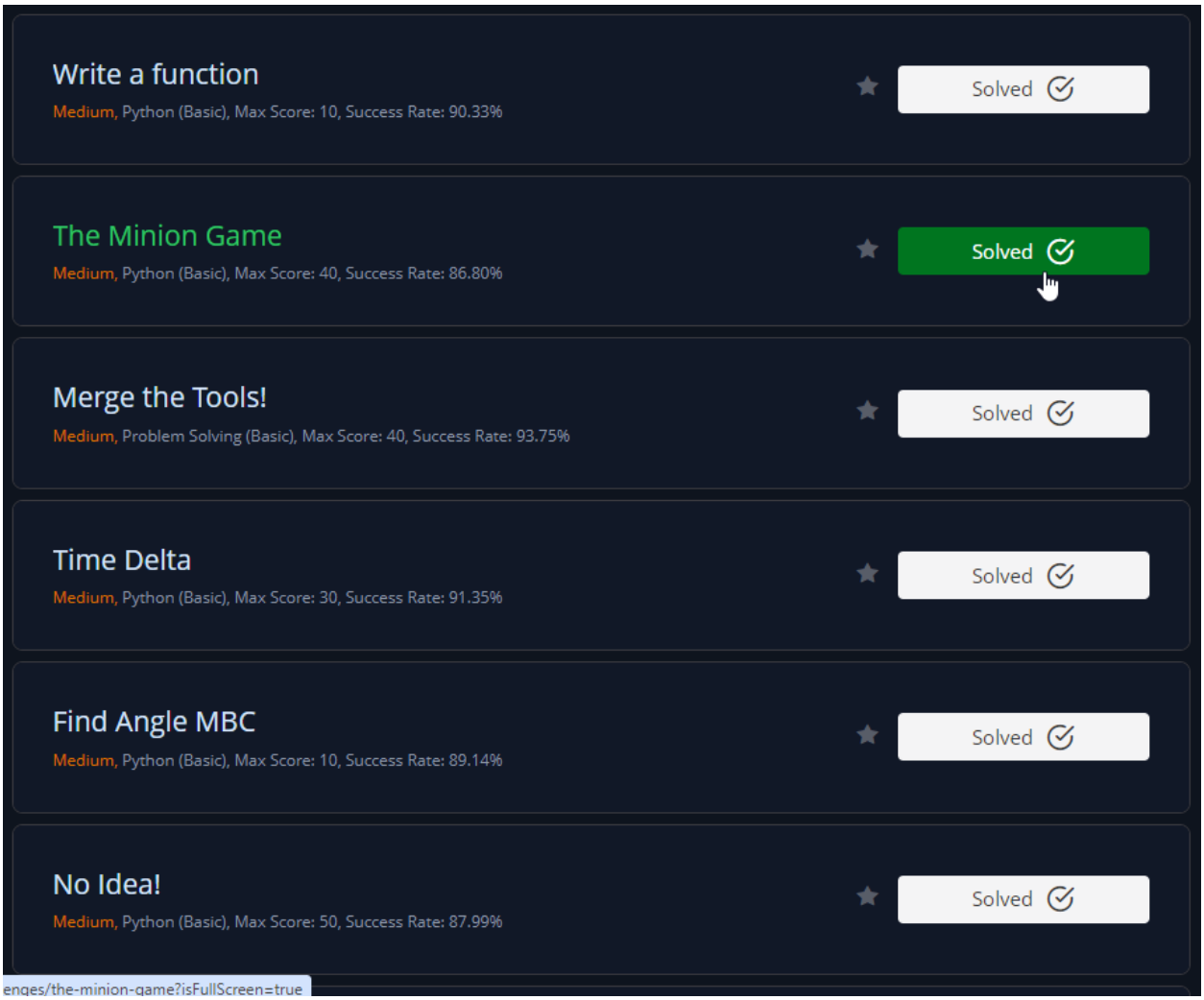
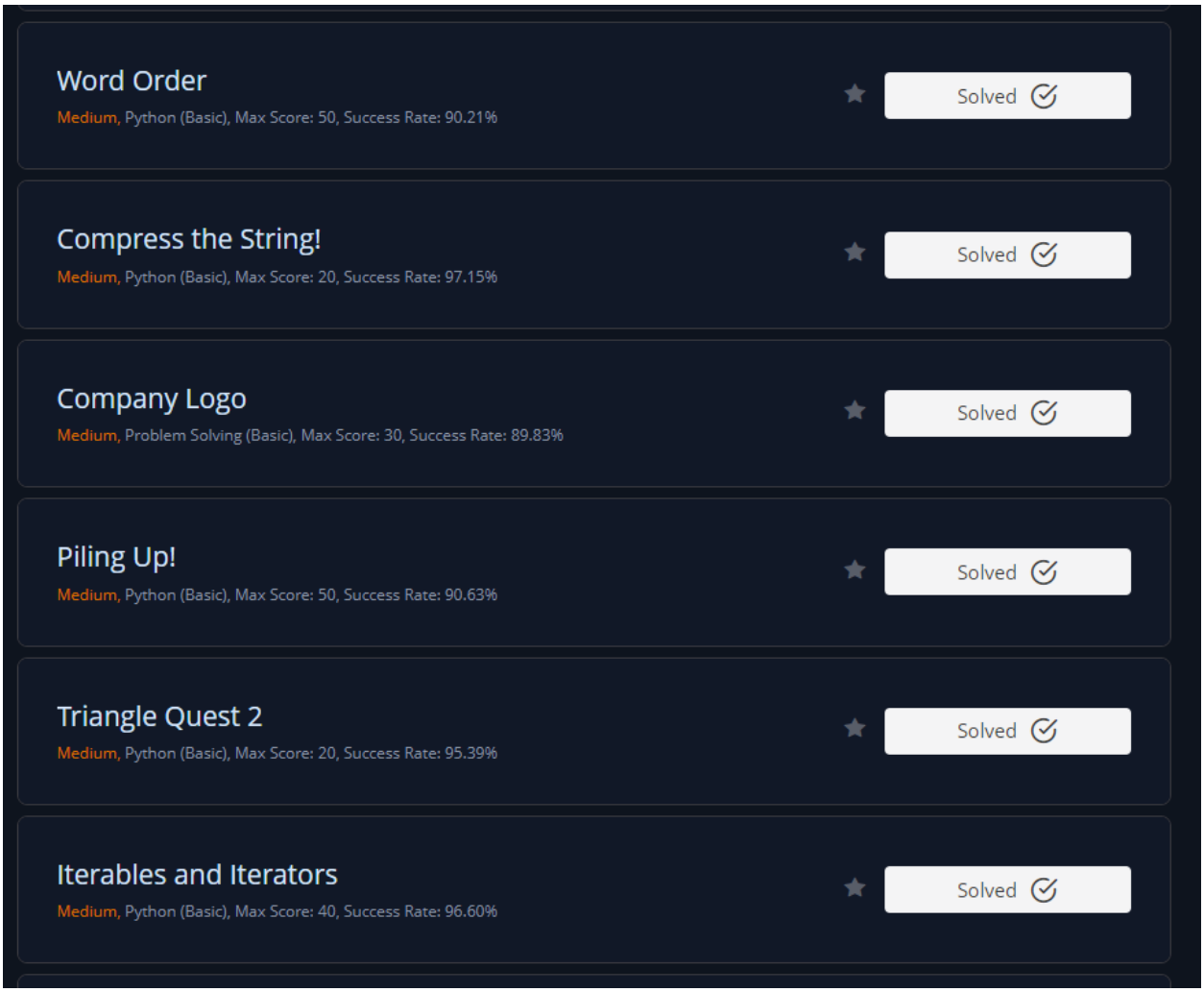
National University of Sciences and Technology

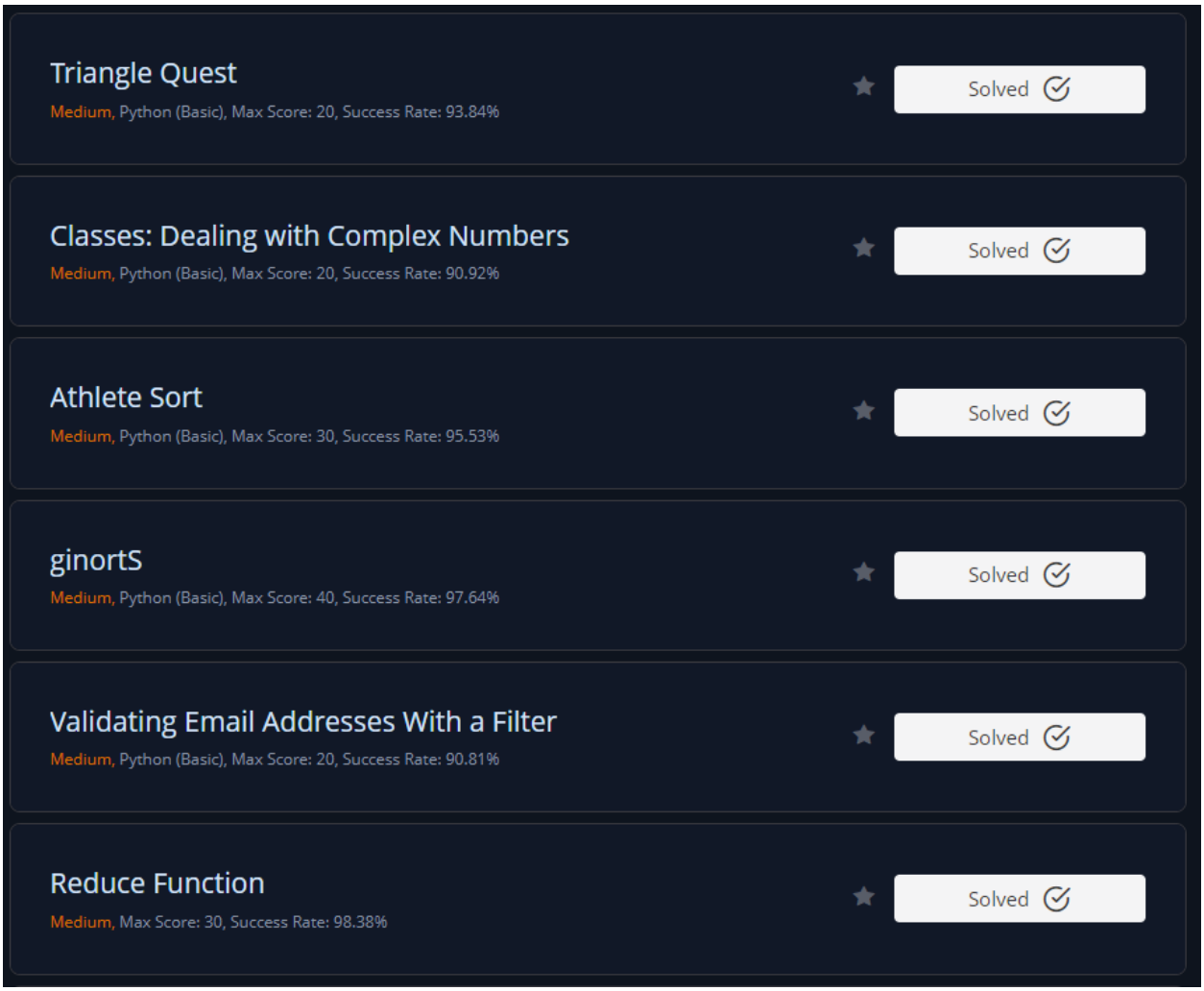
School of Mechanical and Manufacturing Engineering

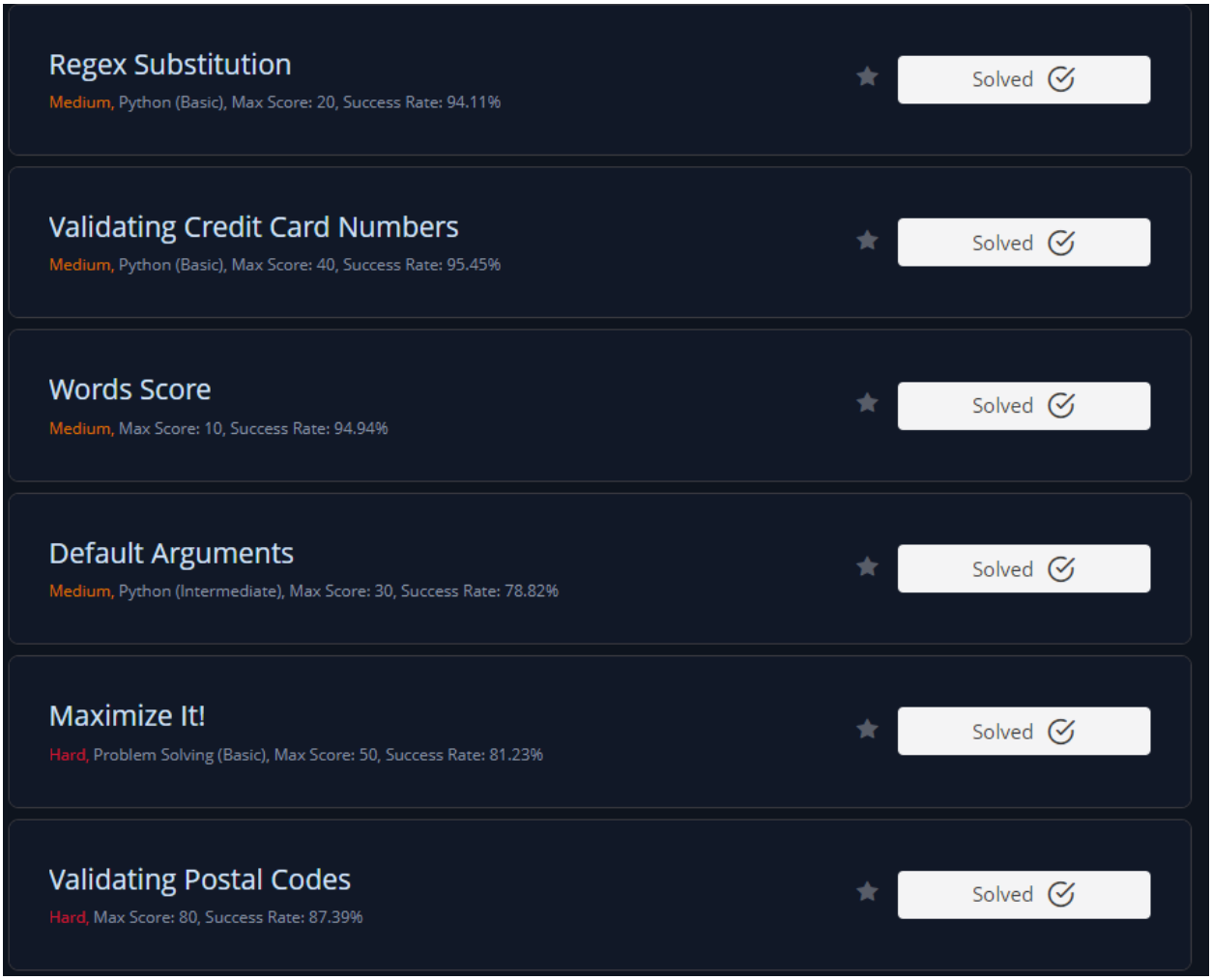
MS Robotics and Intelligent Machine Engineering

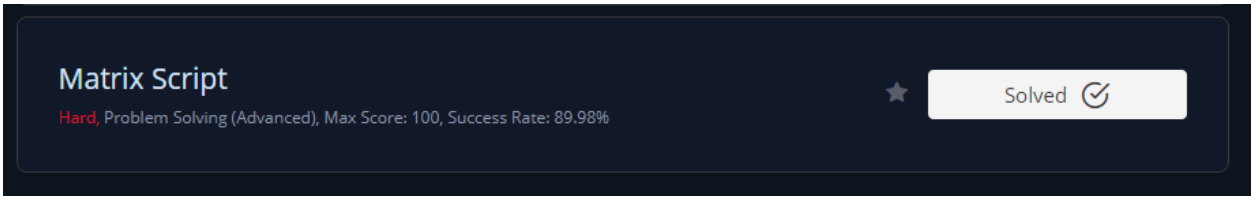
SUBMITTED TO	Dr. Yasar Ayaz
SUBMITTED BY	Muhammad Fazeel
REGISTRATION NO	453385
ASSIGNMENT NO	3
CLASS	Artificial Intelligence (CSE-860)
Date of Submission:	18 September, 2023


Write a function
Medium, Python (Basic), Max Score: 10, Success Rate: 90.33%
Solved
The Minion Game
Medium, Python (Basic), Max Score: 40, Success Rate: 86.80%
Solved
Merge the Tools!
Medium, Problem Solving (Basic), Max Score: 40, Success Rate: 93.75%
Solved
Time Delta
Medium, Python (Basic), Max Score: 30, Success Rate: 91.35%
Solved
Find Angle MBC
Medium, Python (Basic), Max Score: 10, Success Rate: 89.14%
Solved
No Idea!
Medium, Python (Basic), Max Score: 50, Success Rate: 87.99%
Solved
enges/the-minion-game?isFullScreen=true


Word Order
Medium, Python (Basic), Max Score: 50, Success Rate: 90.21%
Solved
Compress the String!
Medium, Python (Basic), Max Score: 20, Success Rate: 97.15%
Solved
Company Logo
Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.83%
Solved
Piling Up!
Medium, Python (Basic), Max Score: 50, Success Rate: 90.63%
Solved
Triangle Quest 2
Medium, Python (Basic), Max Score: 20, Success Rate: 95.39%
Solved
Iterables and Iterators
Medium, Python (Basic), Max Score: 40, Success Rate: 96.60%
Solved


Triangle Quest
Medium, Python (Basic), Max Score: 20, Success Rate: 93.84%
Solved
Classes: Dealing with Complex Numbers
Medium, Python (Basic), Max Score: 20, Success Rate: 90.92%
Solved
Athlete Sort
Medium, Python (Basic), Max Score: 30, Success Rate: 95.53%
Solved
ginortS
Medium, Python (Basic), Max Score: 40, Success Rate: 97.64%
Solved
Validating Email Addresses With a Filter
Medium, Python (Basic), Max Score: 20, Success Rate: 90.81%
Solved
Reduce Function
Medium, Max Score: 30, Success Rate: 98.38%
Solved


Regex Substitution
Medium, Python (Basic), Max Score: 20, Success Rate: 94.11%
Solved
Validating Credit Card Numbers
Medium, Python (Basic), Max Score: 40, Success Rate: 95.45%
Solved
Words Score
Medium, Max Score: 10, Success Rate: 94.94%
Solved
Default Arguments
Medium, Python (Intermediate), Max Score: 30, Success Rate: 78.82%
Solved
Maximize It!
Hard, Problem Solving (Basic), Max Score: 50, Success Rate: 81.23%
Solved
Validating Postal Codes
Hard, Max Score: 80, Success Rate: 87.39%
Solved


Matrix Script
Hard, Problem Solving (Advanced), Max Score: 100, Success Rate: 89.98%
Solved

Write a function

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

    # Write your logic here
    if (year&4)==0:
        leap = "True"
    else:
        leap = "False"

    return leap

year = int(input())
```

The Minion Game

```
def minion_game(string):
    # your code goes here
    vowels = 'AEIOU'
    str_lenght = len(string)
    kevin_score, stuart_score = 0, 0

    for i in range(str_lenght):
        if s[i] in vowels:
            kevin_score += (str_lenght - i)
        else:
            stuart_score += (str_lenght - i)

    if kevin_score > stuart_score:
        print("Kevin", kevin_score)
    elif kevin_score < stuart_score:
        print("Stuart", stuart_score)
    else:
        print("Draw")
if __name__ == '__main__':
```

Merge the Tools!

```
def merge_the_tools(string, k):
 for i in range(0, len(string), k):
   substring = string[i:i+k]
   seen_chars = set()
   result = ""
   for char in substring:
     if char not in seen_chars:
       result += char
       seen_chars.add(char)
   print(result)

if __name__ == '__main__':…
```

Time Delta

```
#!/bin/python3

import math
import os
import random
import re
import sys

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

if __name__ == '__main__':…
```

Find Angle MBC

```
import math

ab = float(input())
bc = float(input())
ac = math.sqrt(ab**2 + bc**2)
bm = ac / 2.0
mc = bm
# let,
b = mc
c = bm
a = bc
# where b=c
angel_b_radian = math.acos(a / (2 * b))
angel_b_degree = int(round((180 * angel_b_radian) / math.pi))

print(angel_b_degree, chr(176), sep = "")
```

No Idea!

```
def calculate_happiness():
    n, m = map(int, input().split())
    array = list(map(int, input().split()))
    set_A = set(map(int, input().split()))
    set_B = set(map(int, input().split()))
    happiness = 0
    for i in array:
        if i in set_A:
            happiness += 1
        elif i in set_B:
            happiness -= 1
    return happiness

print(calculate_happiness())
```

Word Order

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
n = int(input().strip())
words = []
counts = []

for _ in range(n):
    word = input().strip()
    if word not in words:
        words.append(word)
        counts.append(1)
    else:
        counts[words.index(word)] += 1

print(len(words))
for count in counts:
    print(count, end=' ')
```

Compress the String!

```
s = input().strip()
i = 0
while i < len(s):
    count = 1
    while i + 1 < len(s) and s[i] == s[i+1]:
        i += 1
        count += 1
    print((count, int(s[i])), end=' ')
    i += 1
```

Company Logo

```
if __name__ == '__main__':
    s = input().strip()
    frequency = [0]*26

    for i in s:
        frequency[ord(i)-97] += 1

    characters = [chr(i+97) for i in range(26)]
    frequency, characters = zip(*sorted(zip(frequency, characters), reverse=True))

    for i in range(3):
        if frequency[i]:
            print(characters[i], frequency[i])
```

Piling Up!

```
for _ in range(T):
    n = int(input().strip())
    blocks = list(map(int, input().strip().split()))
    left_index = 0
    right_index = n - 1
    if blocks[left_index] > blocks[right_index]:
        current_block = blocks[left_index]
        left_index += 1
    else:
        current_block = blocks[right_index]
        right_index -= 1
    while left_index <= right_index:
        if blocks[left_index] > blocks[right_index]:
            if blocks[left_index] > current_block:
                print('No')
                break
            current_block = blocks[left_index]
            left_index += 1
        else:
            if blocks[right_index] > current_block:
                print('No')
                break
            current_block = blocks[right_index]
            right_index -= 1
    else:
        print('Yes')
```

Triangle Quest 2

```
for i in range(1, int(input()) + 1):
    print(((10**i - 1) // 9)**2)
```

Iterables and Iterators

```
import itertools
n = int(input())
letters = input().split()
k = int(input())
a_count = letters.count('a')

total_combinations = len(list(itertools.combinations(range(n), k)))
combinations_without_a = len(list(itertools.combinations(list(range(n))[a_count:], k)))

probability = 1 - combinations_without_a / total_combinations
print("{:.3f}".format(probability))
```

Triangle Quest

```
for i in range(1,int(input())):
    print
```

Classes: Dealing with Complex Numbers

```python
import math

class Complex:
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary

    def __add__(self, no):
        return Complex(self.real + no.real, self.imaginary + no.imaginary)

    def __sub__(self, no):
        return Complex(self.real - no.real, self.imaginary - no.imaginary)

    def __mul__(self, no):
        return Complex(
            self.real * no.real - self.imaginary * no.imaginary,
            self.real * no.imaginary + self.imaginary * no.real,
        )

    def __truediv__(self, no):
        divider = no.real**2 + no.imaginary**2
        return Complex(
            (self.real * no.real + self.imaginary * no.imaginary) / divider,
            (self.imaginary * no.real - self.real * no.imaginary) / divider,
        )

    def mod(self):
        return Complex(math.sqrt(self.real**2 + self.imaginary**2), 0.00)

    def __str__(self):
        if self.imaginary == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            if self.imaginary >= 0:
                result = "0.00+%.2fi" % (self.imaginary)
            else:
                result = "0.00-%.2fi" % (abs(self.imaginary))
        elif self.imaginary > 0:
            result = "%.2f+%.2fi" % (self.real, self.imaginary)
        else:
            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))
        return result

if __name__ == '__main__':…
```

Athlete Sort

```python
#!/bin/python3

import math
import os
import random
import re
import sys
if __name__ == "__main__":

    n, m = input().strip().split(' ')

    n, m = [int(n), int(m)]
    arr = []

    for arr_i in range(n):

       arr_t = [int(arr_temp) for arr_temp in input().strip().split(' ')]
       arr.append(arr_t)

    k = int(input().strip())

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

    for row in sorted_arr:
        print(' '.join(str(y) for y in row))
```

ginortS

```python
l = []
u = []
o = []
e = []

for i in sorted(input()):

    if i.isalpha():

        x = u if i.isupper() else l

    else:

        x = o if int(i)%2 else e

    x.append(i)
print("".join(l+u+o+e))
```

Validating Email Addresses With a Filter

```
def fun(s):
    def fun(email):
        try:

        username, url = email.split('@')
        website, extension = url.split('.')

    except ValueError:
        return False

    if username.replace('-', '').replace('_', '').isalnum() is False:
        return False

    elif website.isalnum() is False:
        return False

    elif len(extension) > 3:
        return False
    else:
        return True
def filter_mail(emails):...
```

Reduce Function

```
def product(fracs):
    t = reduce(lambda x, y : x * y, fracs)
    return t.numerator, t.denominator
```

Regex Substitution

```
import re

n = int(input().strip())
for _ in range(n):
    s = input()
    s = re.sub(r'(?<= )&&(?= )', 'and', s)
    s = re.sub(r'(?<= )\|\|(?= )', 'or', s)
    print(s)
```

Validating Credit Card Numbers

```
import re

# taking input from user
n = int(input())

for t in range(n):

    credit = input().strip()
    credit_removed_hiphen = credit.replace('-','')

    valid = True

    length_16 = bool(re.match(r'^[4-6]\d{15}$',credit))
    length_19 = bool(re.match(r'^[4-6]\d{3}-\d{4}-\d{4}-\d{4}$',credit))
    consecutive = bool(re.findall(r'(?=(\d)\1\1\1)',credit_removed_hiphen))

    if length_16 == True or length_19 == True:
        if consecutive == True:
            valid=False
    else:
        valid = False
    if valid == True:
        print('Valid')
    else:
        print('Invalid')
```

Words Score

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

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

Default Arguments

```
class EvenStream(object):…

def print_from_stream(n, stream=EvenStream()):
    for _ in range(n):
        print(stream.get_next())
```

Maximize It!

```
from itertools import product

K, M = map(int,input().split())
N = (list(map(int, input().split()))[1:] for _ in range(K))
results = map(lambda x: sum(i**2 for i in x)%M, product(*N))
print(max(results))
```

Validating Postal Codes

```
regex_integer_in_range = r"_________"
regex_alternating_repetitive_digit_pair = r"_________"

import re
P = input()

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

Matrix Script

```python
#!/bin/python3

import math
import os
import random
import re
import sys



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

n = int(first_multiple_input[0])

m = int(first_multiple_input[1])

matrix = []

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