NUST School of Mechanical & Manufacturing Engineering (SMME)

MS Robotics & Intelligent Machine Engineering

Assignment No. 3

Submitted To:

Instructor Name: Dr. Yasar Ayaz

Course Name: Artificial Intelligence

Dated: 01-01-2024

Submitted By:

Student Name: Maryam Javed

Registration Number: 450870

HackerRank Challenges

Medium Tasks:

1. 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

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

2. Compress the String

```
# Enter your code here. Read input from STDIN. Print output to S
TDOUT
from itertools import groupby

def compress_string(s):
    compressed_string = [(char, len(list(group))) for char, grou
p in groupby(s)]
    result = ' '.join(f"({count}, {char})" for char, count in co
mpressed_string)
    return result

if __name__ == '__main__':
    s = input().strip()
    result = compress_string(s)
    print(result)
```

3. ginortS

```
def custom_sort_key(char):
    if char.islower():
        return (0, char)
    elif char.isupper():
        return (1, char)
    elif char.isdigit():
        return (2, int(char)%2!=1, char)
    else:
        return (3, char)

def custom_sort(s):
    return ''.join(sorted(s, key=custom_sort_key))

# Example usage:
input_string = input()
result = custom_sort(input_string)
print(result)
```

4. Validating the email addresses with a Filter

```
import re
def fun(s):
    reg_pattern = r'^[a-zA-Z0-9_-]+@[a-zA-Z0-9]+\.[a-zA-
Z]{1,3}$'
    return(bool(re.match(reg_pattern,s)))
    # return True if s is a valid email, else return False

def filter_mail(emails):
    return list(filter(fun, emails))

if __name__ == '__main__':
    n = int(input())
    emails = []
    for _ in range(n):
        emails.append(input())

filtered_emails = filter_mail(emails)
filtered_emails.sort()
print(filtered_emails)
```

5. Reduce Function

```
from fractions import Fraction
from functools import reduce

def product(fracs):
    t = reduce(lambda x,y: x*y,fracs,Fraction(1,1))# complete th
is line with a reduce statement
    return t.numerator, t.denominator

if __name__ == '__main__':
    fracs = []
    for _ in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
    print(*result)
```

6. Triangle Quest

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

7. Triangle Quest 2

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

8. Find Angle MBC

```
import math
AB = float(input())
BC = float(input())
theta_rad = math.atan2(AB, BC)
theta_deg = round(theta_rad * (180 / math.pi))
print(theta_deg,end="\u00b0")
```

9. No Idea

```
n, m = map(int, input().split()) #n elements of array, m element
s in each set
array = list(map(int, input().split()))
set_a = set(map(int, input().split()))
set_b = set(map(int, input().split()))
```

```
happiness = 0
for num in array:
    if num in set_a:
        happiness += 1
    elif num in set_b:
        happiness -= 1

print(happiness)
```

10. Time Delta

```
import math
import os
import random
import re
import sys
from datetime import datetime

def time_delta(t1, t2):
    # Convert timestamps to datetime objects
    string = '%a %d %b %Y %H:%M:%S %z' # general format of the s
tring provided as input
    time1 = datetime.strptime(t1, string) #breakdown the given s
tring according to the format
    time2 = datetime.strptime(t2, string)

    # Calculate the absolute difference in seconds
    delta_sec = int(abs((time1 - time2).total_seconds()))

    return str(delta_sec)


if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w') #opens a file to
 write

    t = int(input()) #number of test cases

    for t_itr in range(t):
        t1 = input()

        t2 = input()
```

```
        delta = time_delta(t1, t2)

        fptr.write(delta + '\n')

    fptr.close()
```

11. Word Order

```
from collections import OrderedDict
n = int(input())
word_count = OrderedDict() #to keep track of the occurrences of
each word while preserving the order of insertion
for _ in range(n):
    words = input().strip().split()

    #check whether the word is already inserted and counted

    for check_word in words:
        if check_word in word_count:
            word_count[check_word] += 1
        else:
            word_count[check_word] = 1

non_rep_words = list(word_count.keys()) #keys of each non-
repeated word
print(len(non_rep_words))

print(*word_count.values())
```

12. Merge the Tools

```
def merge_the_tools(string, k):
    n = len(string)
    num_substrings = n // k

    for i in range(0, n, k):
        # Get the current substring of length k
        substring = string[i:i + k]

        # Use a set to store unique characters in the substring
        unique_chars = set()

        # Build the result string without repeated characters
```

```
        result = ''
        for char in substring:
            if char not in unique_chars:
                result += char
                unique_chars.add(char)

        # Print the result string for the current substring
        print(result)


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

13. Company Logo

```
from collections import Counter

def company_logo(s):
    # Count occurrences of each character in the string
    char_counts = Counter(s)

     # Get the unique characters in the string
    unique_chars = list(char_counts.keys())

    # Sort the unique characters based on count first and lexico
graphical order
    sorted_chars = sorted(unique_chars, key=lambda x: (-
char_counts[x], x))

    # Output the result for the top three characters
    for char in sorted_chars[:3]:
        print(f"{char} {char_counts[char]}")

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

14. Piling Up

```
# Input: Number of test cases
T = int(input().strip())
test_cases = []
for k in range(T):
    n = int(input().strip()) #number of cubes
    side_lengths = list(map(int, input().split())) #space separa
ted integers
    test_cases.append((n, side_lengths))

results = []

for case in test_cases:
    num_cubes = case[0]
    side_lengths = case[1]

    i = 0
    j = num_cubes - 1

    while i < j and side_lengths[i] >= side_lengths[i + 1]:
        i += 1

    while i < j and side_lengths[j] >= side_lengths[j - 1]:
        j -= 1

    if i == j:
        results.append("Yes")
    else:
        results.append("No")

for result in results:
    print(result)
```

15. Athlete Sort

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

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())

# Sort the athletes based on the kth attribute
arr.sort(key=lambda x: x[k])

# Output: Sorted list of athletes
for athlete in arr:
    print(*athlete)
```

16. Regex Substitution

```
import re

N = int(input())
for i in range(0,N):
   text = input()
   text = re.sub(r"\ \&\&\ "," and ",text)
   text = re.sub(r"\ \|\|\ "," or ",text)
   text = re.sub(r"\ \&\&\ "," and ",text)
   text = re.sub(r"\ \|\|\ "," or ",text)
   print(text)
```

17. Iterables and Iterators

```
from itertools import combinations

length = int(input())
list = [i for i in input().split()]
k = int(input())
a_idx = []
for i in range(1,length+1):
    if list[i-1]=='a':
        a_idx.append(i)

a_num = 0
comb_len = 0
for comb in combinations(range(1, length + 1), k):
```

```
    comb_len += 1
    if set(comb) & set(a_idx):
        a_num += 1
print(a_num / comb_len)
```

18. Classes: Dealing with complex numbers

```
import math

class Complex(object):
    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):
        prod = complex(self.real , self.imaginary)*complex(no.re
al , no.imaginary)
        return Complex(prod.real , prod.imag)

    def __truediv__(self, no):
        div = complex(self.real , self.imaginary)/complex(no.rea
l , no.imaginary)
        return Complex(div.real , div.imag)


    def mod(self):
        m = math.sqrt(self.real**2 + self.imaginary**2)
        return Complex(m,0)

    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__':
    c = map(float, input().split())
    d = map(float, input().split())
    x = Complex(*c)
    y = Complex(*d)
    print(*map(str, [x+y, x-
y, x*y, x/y, x.mod(), y.mod()]), sep='\n')
```

19. Validating Credit Card Numbers

```
import re

start_with_456 = lambda x: x[0]=='4' or x[0]=='5' or x[0]=='6'

contain_16_digits = lambda x: bool(re.fullmatch(r'\d{16}', ''.jo
in(x.split('-'))))

groups_of_4 = lambda x: all([len(i)==4 for i in x.split('-
') if '-' in x])

repeating_characters = lambda x: not(bool(re.search(r'(\d)\1{3,}
', ''.join(x.split('-')))))

tests = [start_with_456, contain_16_digits, groups_of_4, repeati
ng_characters]

N = int(input())
for _ in range(N):
    s = input()
    if all(map(lambda x: x(s), tests)):
        print('Valid')
    else:
        print('Invalid')
```

20. Word Score

```
def score_words(word) :
    score = 0
    for i in range(len(word)):
        num_vo = 0
        for j in word[i] :
            if j in "aeiouy" :
                num_vo += 1
        if num_vo % 2 == 0 :
            score += 2
        else :
            score += 1
    return score
if __name__=="__main__":
    n = int(input().strip())
    words = input().strip().split()
    result = score_words(words)
    print(result)
```

21. Default Arguments

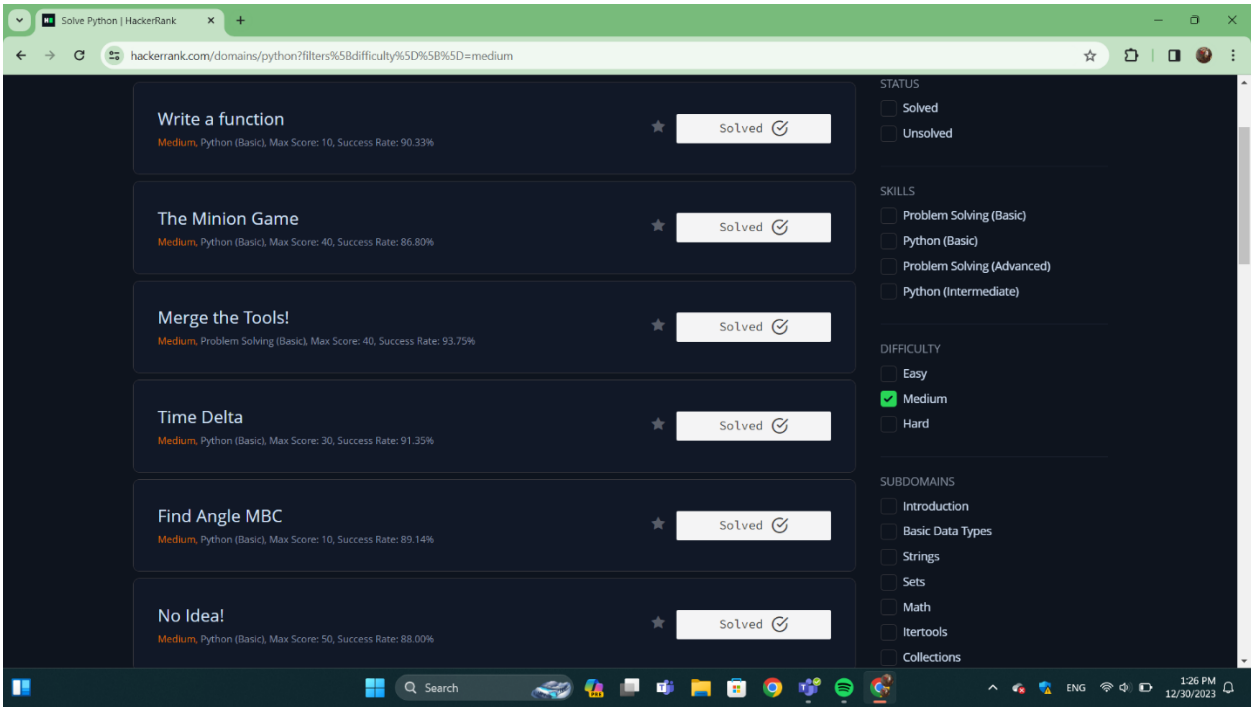
```
class EvenStream(object):
    def __init__(self):
        self.current = 0

    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

class OddStream(object):
    def __init__(self):
        self.current = 1

    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

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

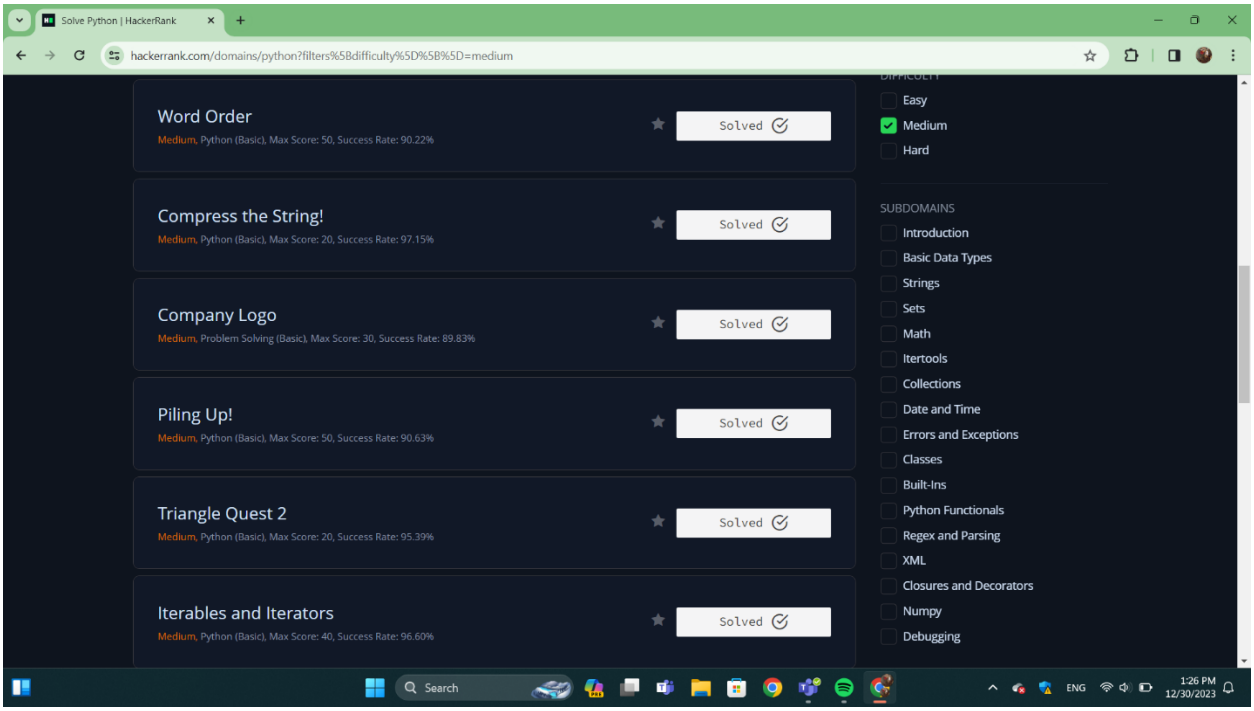
```
        print(stream.get_next())
    stream.current =temp


queries = int(input())
for _ in range(queries):
    stream_name, n = input().split()
    n = int(n)
    if stream_name == "even":
        print_from_stream(n)
    else:
        print_from_stream(n, OddStream())
```

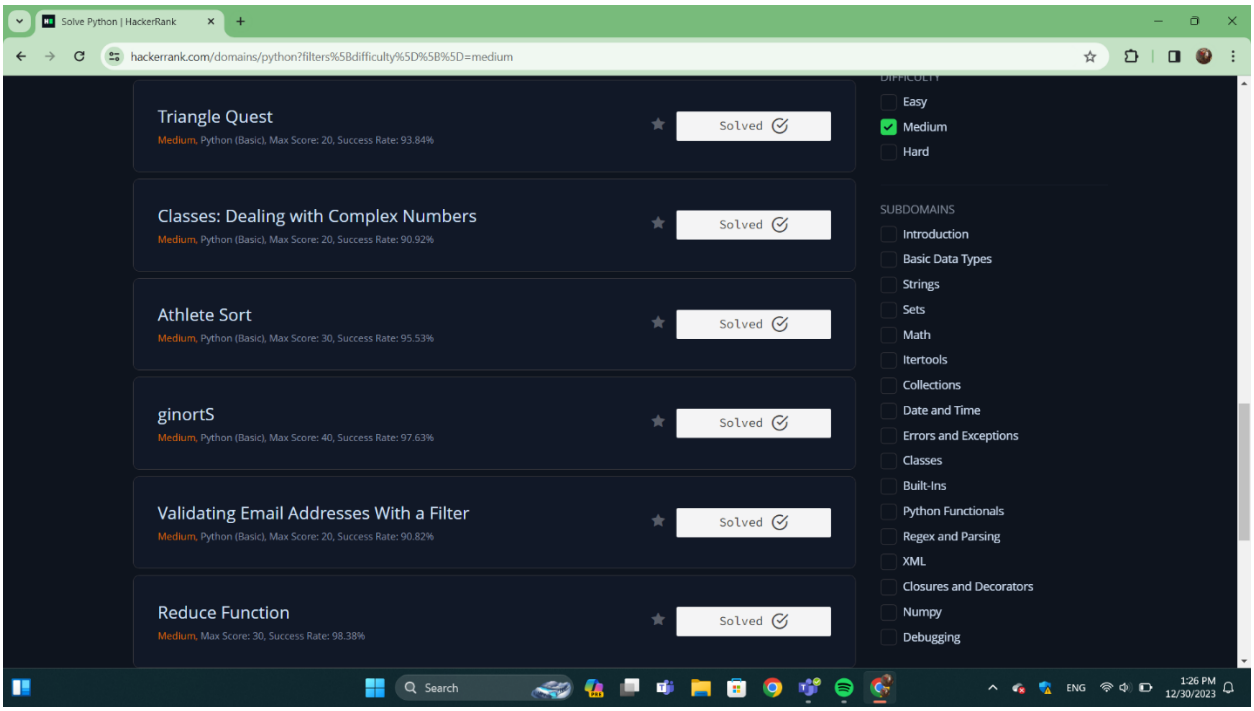
22. The Minion Game

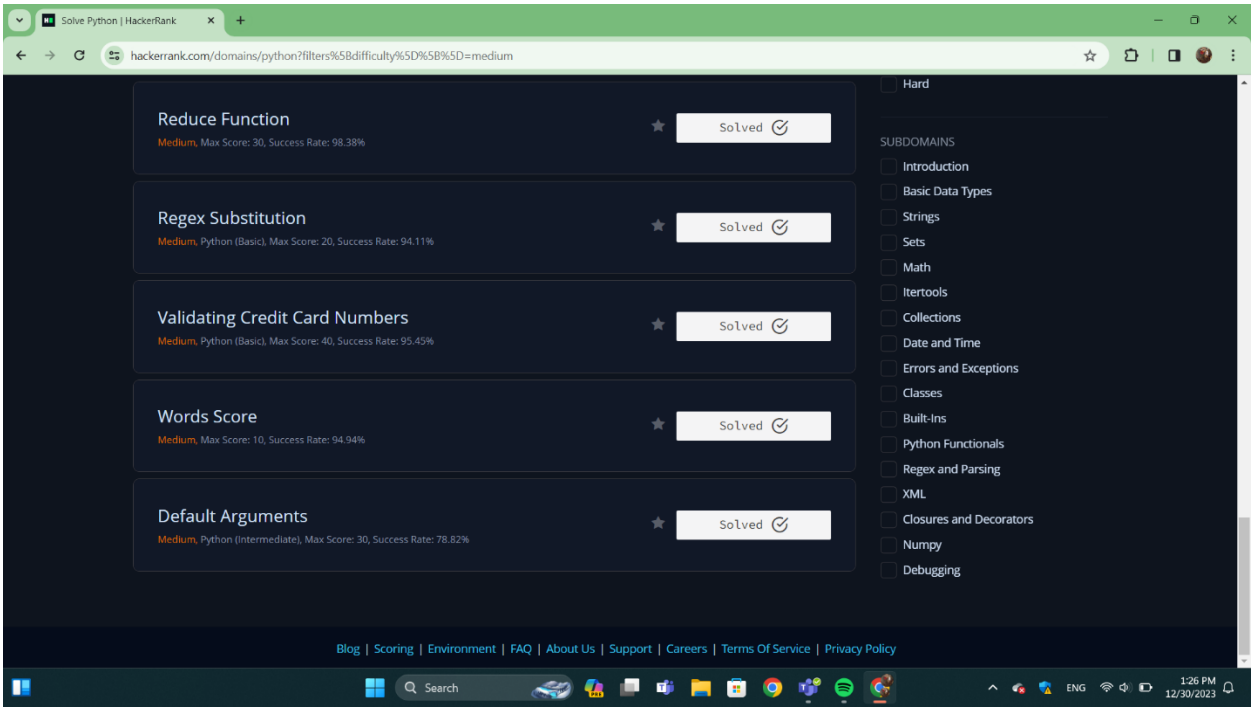
```
def minion_game(string):
    vowels = "AEIOU"
    stuart_score = 0
    kevin_score = 0

    length = len(string)

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

    if kevin_score > stuart_score:
        print(f"Kevin {kevin_score}")
    elif kevin_score < stuart_score:
        print(f"Stuart {stuart_score}")
    else:
        print("Draw")

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

Completed Tasks:






Solve Python | HackerRank
hackerrank.com/domains/python?filters%5Bdifficulty%5D%5B%5D=medium
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: 97.63%
Solved
Validating Email Addresses With a Filter
Medium, Python (Basic), Max Score: 20, Success Rate: 90.82%
Solved
Reduce Function
Medium, Max Score: 30, Success Rate: 98.38%
Solved
Easy
Medium
Hard
SUBDOMAINS
Introduction
Basic Data Types
Strings
Sets
Math
Itertools
Collections
Date and Time
Errors and Exceptions
Classes
Built-Ins
Python Functionals
Regex and Parsing
XML
Closures and Decorators
Numpy
Debugging
Search
ENG
1:26 PM
12/30/2023


Solve Python | HackerRank
hackerrank.com/domains/python?filters%5Bdifficulty%5D%5B%5D=medium
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
Hard
SUBDOMAINS
Introduction
Basic Data Types
Strings
Sets
Math
Itertools
Collections
Date and Time
Errors and Exceptions
Classes
Built-Ins
Python Functionals
Regex and Parsing
XML
Closures and Decorators
Numpy
Debugging
Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy
Search
ENG
1:26 PM
12/30/2023

Hard Challenges:

1. Maximize it

```
from itertools import product

def maximize_it(arrays, m):
    # Generate all possible combinations of elements from arrays
    combinations = product(*arrays)

    # Calculate the maximum value
    max_result = 0
    for combination in combinations:
        current_sum = sum(x ** 2 for x in combination) % m
        max_result = max(max_result, current_sum)

    return max_result

# Input
k, m = map(int, input().split())
arrays=[]
for _ in range(k):
    array = list(map(int, input().split()[1:]))
    arrays.append(array)

# Call the maximize_it function and print the result
result = maximize_it(arrays, m)
print(result)
```

2. 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)"   # Do
 not delete '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)
```

3. 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)

ac_string = ""
for i in range(m):
    for j in range(n):
        ac_string += matrix[j][i]

pat = r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])' #for non-
alphanumeric characters

print(re.sub(pat,' ',ac_string))
```

Completed Tasks: