Name: Muhammad Owais

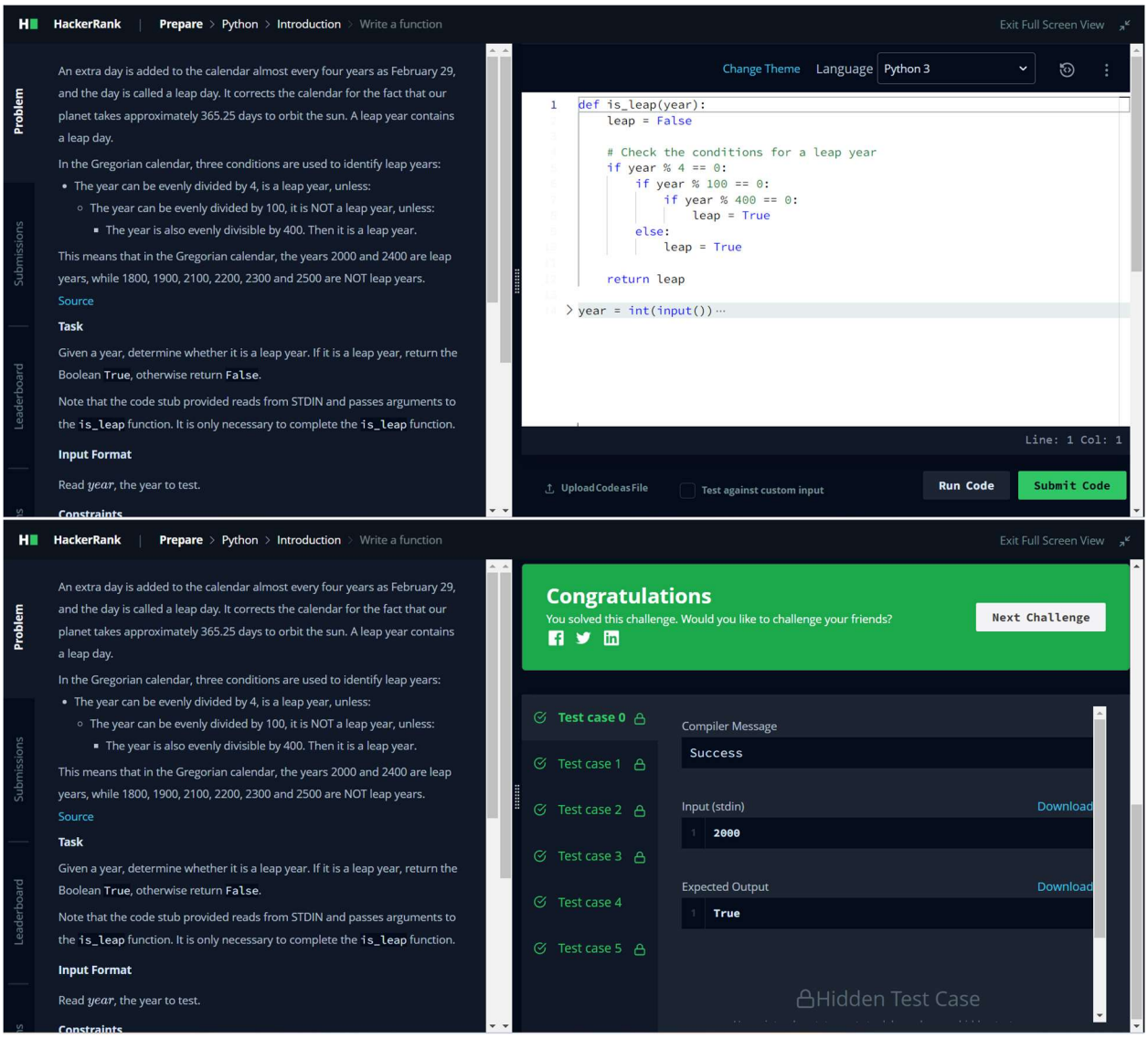
Registration No. 451805

Assignment 3

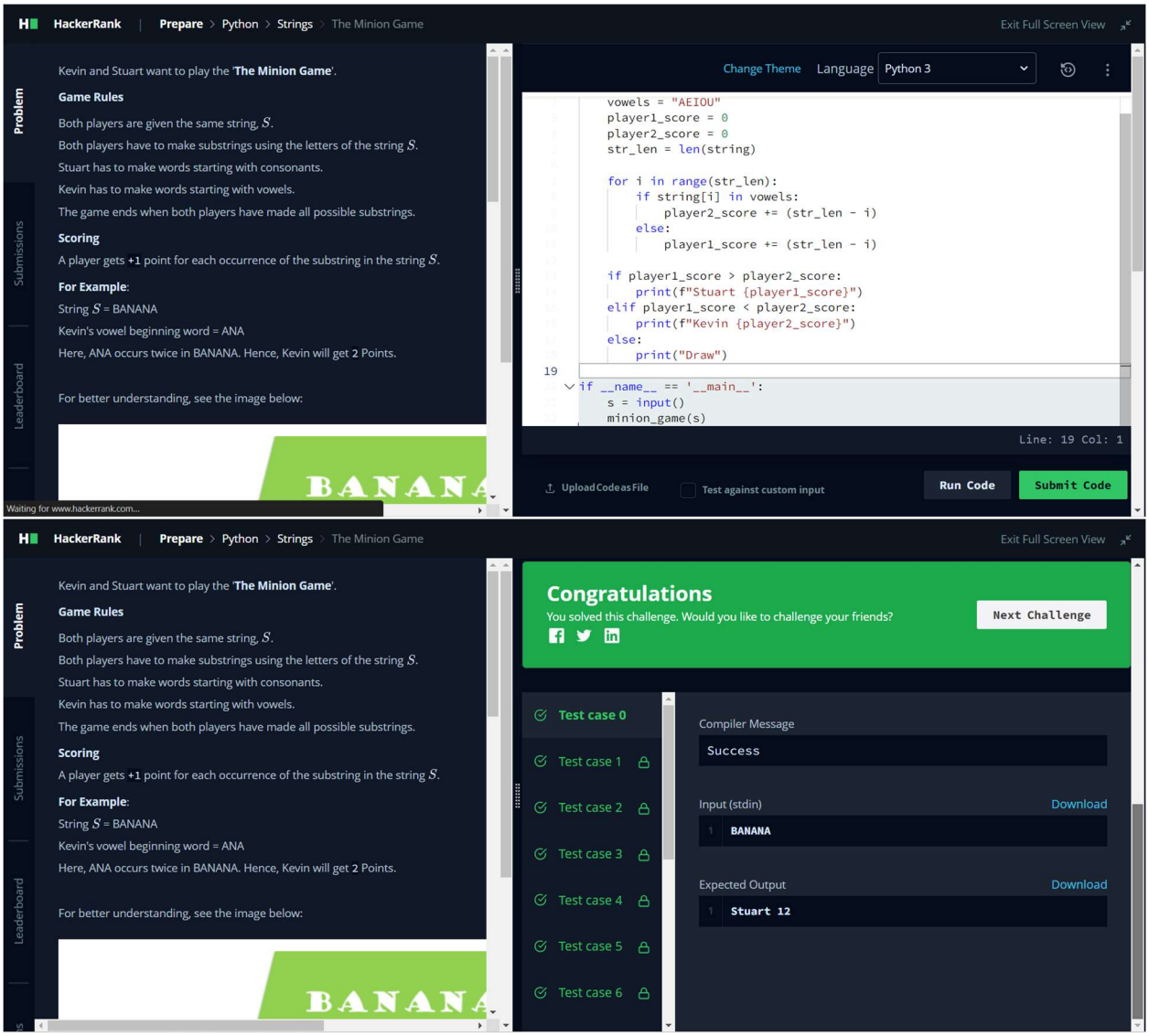
Artificial Intelligence

Medium Level Challenges:

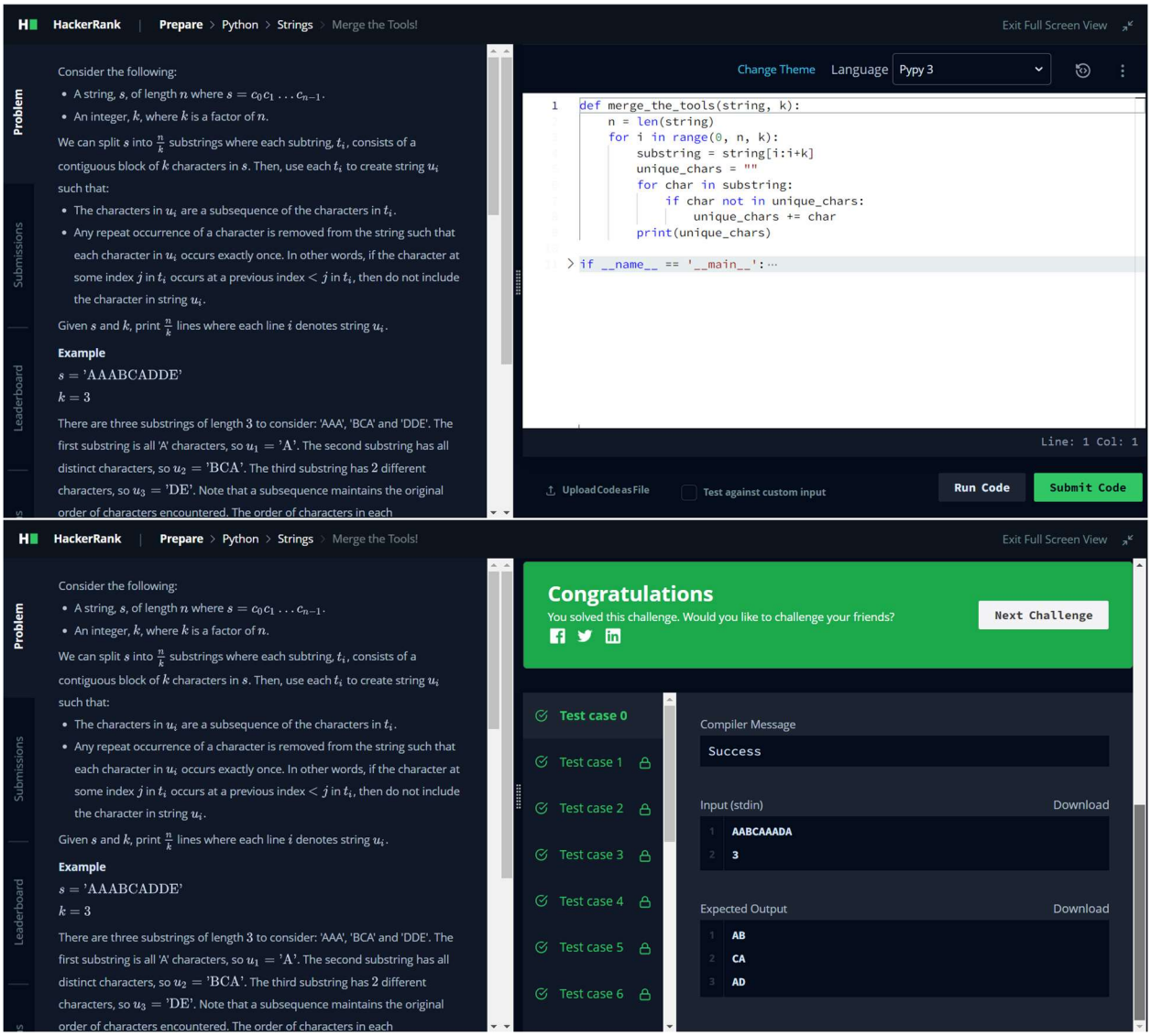
1. Write a Function:



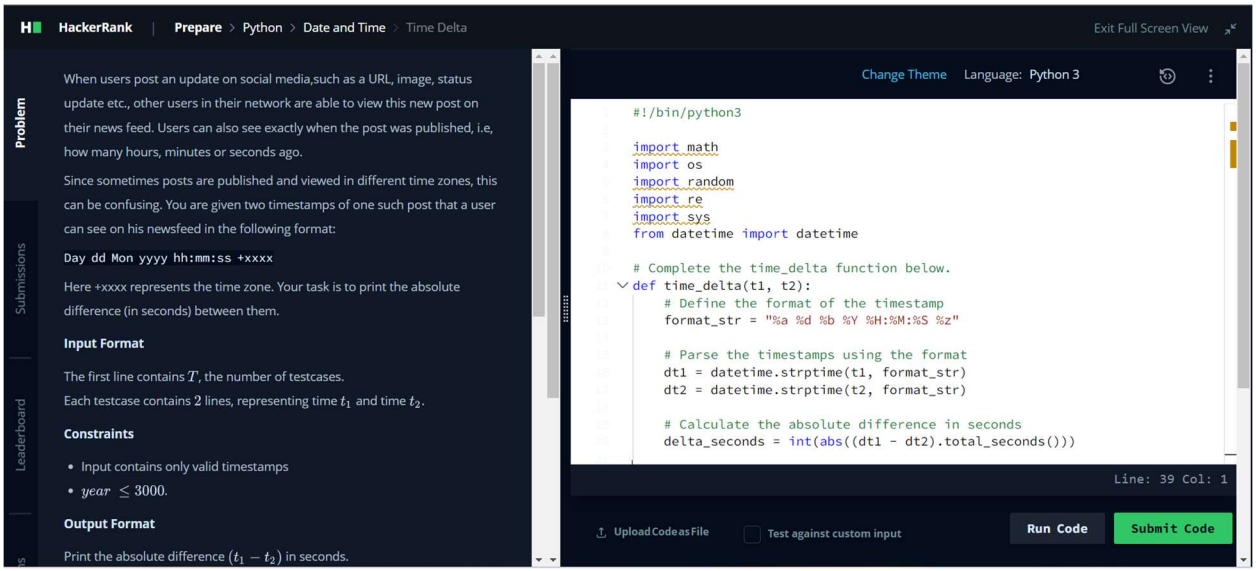
2. The Minion Game:

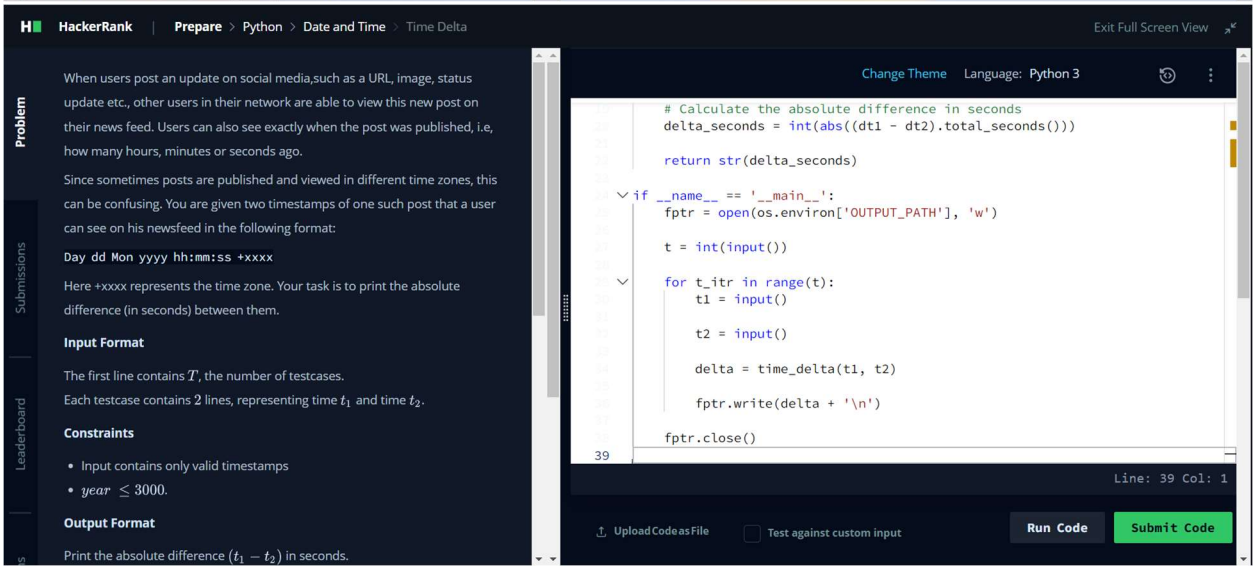


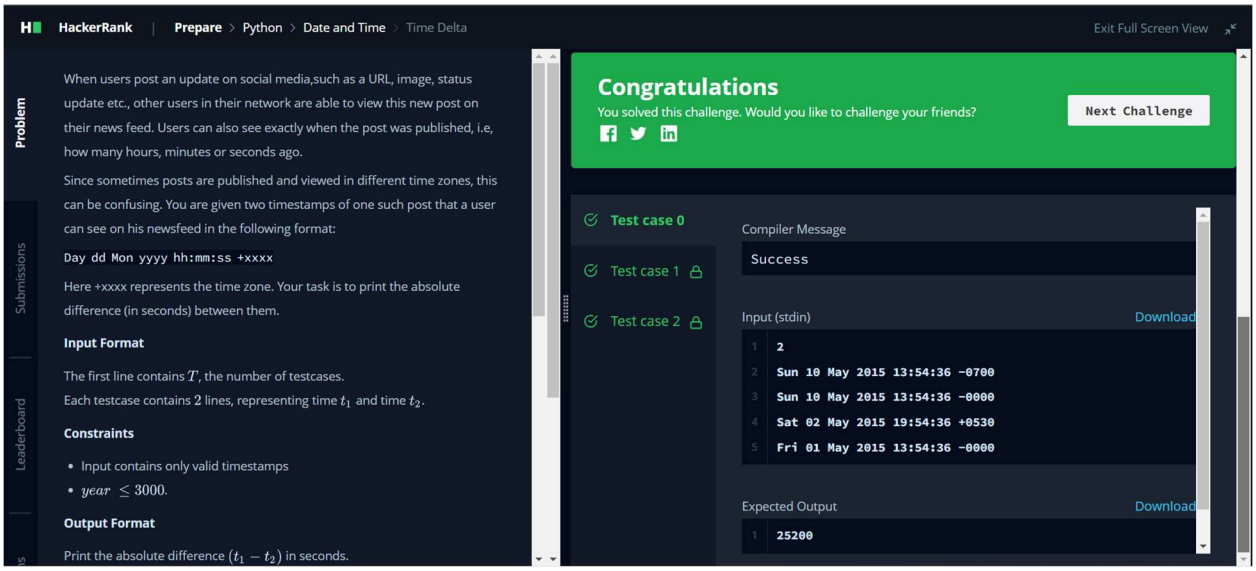
3. Merge the Tools:



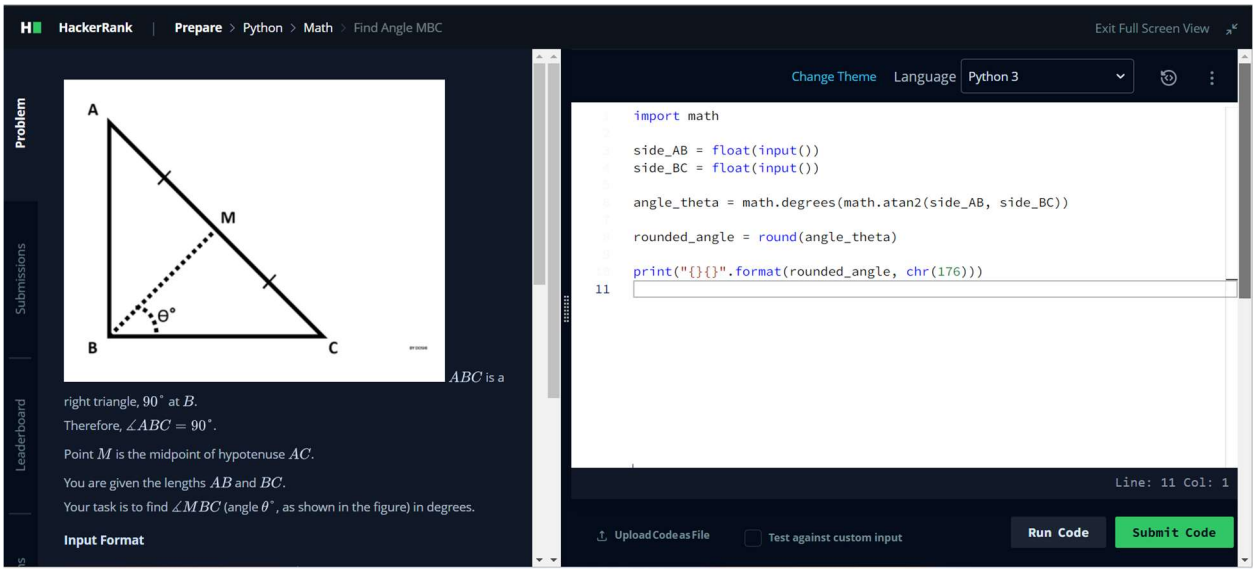
4. Time Delta:

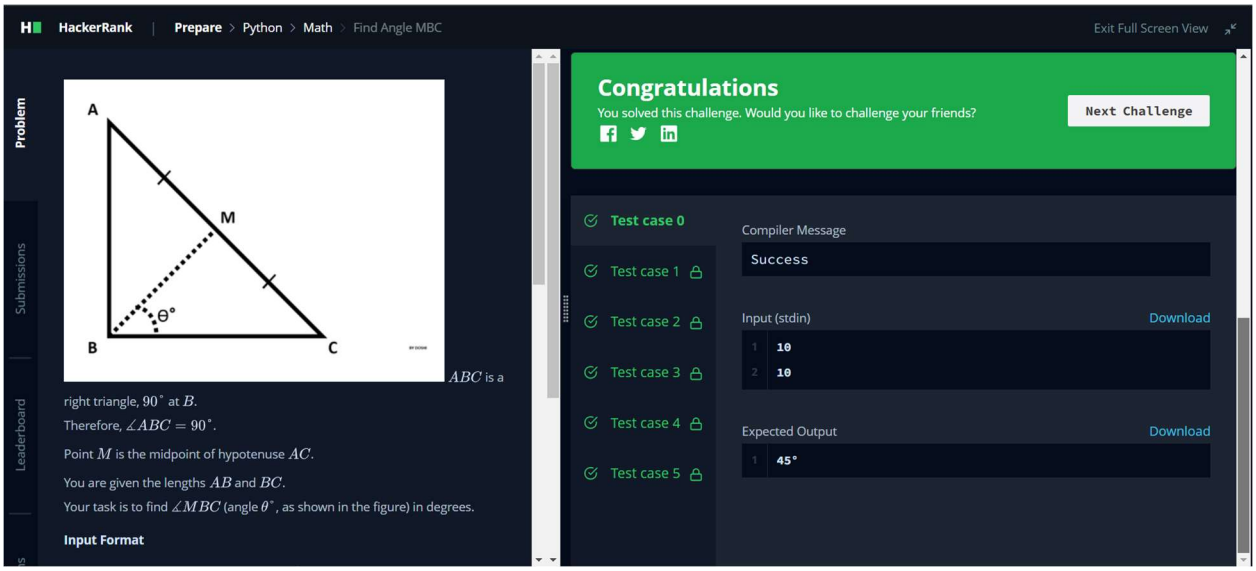

HackerRank | Prepare > Python > Date and Time > Time Delta
Exit Full Screen View
Problem
Submissions
Leaderboard
When users post an update on social media,such as a URL, image, status update etc., other users in their network are able to view this new post on their news feed. Users can also see exactly when the post was published, i.e, how many hours, minutes or seconds ago.
Since sometimes posts are published and viewed in different time zones, this can be confusing. You are given two timestamps of one such post that a user can see on his newsfeed in the following format:
Day dd Mon yyyy hh:mm:ss +xxxx
Here +xxxx represents the time zone. Your task is to print the absolute difference (in seconds) between them.
Input Format
The first line contains T, the number of testcases.
Each testcase contains 2 lines, representing time t_1 and time t_2.
Constraints
Input contains only valid timestamps
$year \leq 3000$.
Output Format
Print the absolute difference $(t_1 - t_2)$ in seconds.
Change Theme
Language: Python 3
#!/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):
# Define the format of the timestamp
format_str = "%a %d %b %Y %H:%M:%S %z"
# Parse the timestamps using the format
dt1 = datetime.strptime(t1, format_str)
dt2 = datetime.strptime(t2, format_str)
# Calculate the absolute difference in seconds
delta_seconds = int(abs((dt1 - dt2).total_seconds()))
Line: 39 Col: 1
Upload Code as File
Test against custom input
Run Code
Submit Code


HackerRank | Prepare > Python > Date and Time > Time Delta
Exit Full Screen View
Problem
Submissions
Leaderboard
When users post an update on social media,such as a URL, image, status update etc., other users in their network are able to view this new post on their news feed. Users can also see exactly when the post was published, i.e, how many hours, minutes or seconds ago.
Since sometimes posts are published and viewed in different time zones, this can be confusing. You are given two timestamps of one such post that a user can see on his newsfeed in the following format:
Day dd Mon yyyy hh:mm:ss +xxxx
Here +xxxx represents the time zone. Your task is to print the absolute difference (in seconds) between them.
Input Format
The first line contains T, the number of testcases.
Each testcase contains 2 lines, representing time t_1 and time t_2.
Constraints
Input contains only valid timestamps
$year \leq 3000$.
Output Format
Print the absolute difference $(t_1 - t_2)$ in seconds.
Change Theme
Language: Python 3
# Calculate the absolute difference in seconds
delta_seconds = int(abs((dt1 - dt2).total_seconds()))
return str(delta_seconds)
if __name__ == '__main__':
fptr = open(os.environ['OUTPUT_PATH'], 'w')
t = int(input())
for t_itr in range(t):
t1 = input()
t2 = input()
delta = time_delta(t1, t2)
fptr.write(delta + '\n')
fptr.close()
39
Line: 39 Col: 1
Upload Code as File
Test against custom input
Run Code
Submit Code

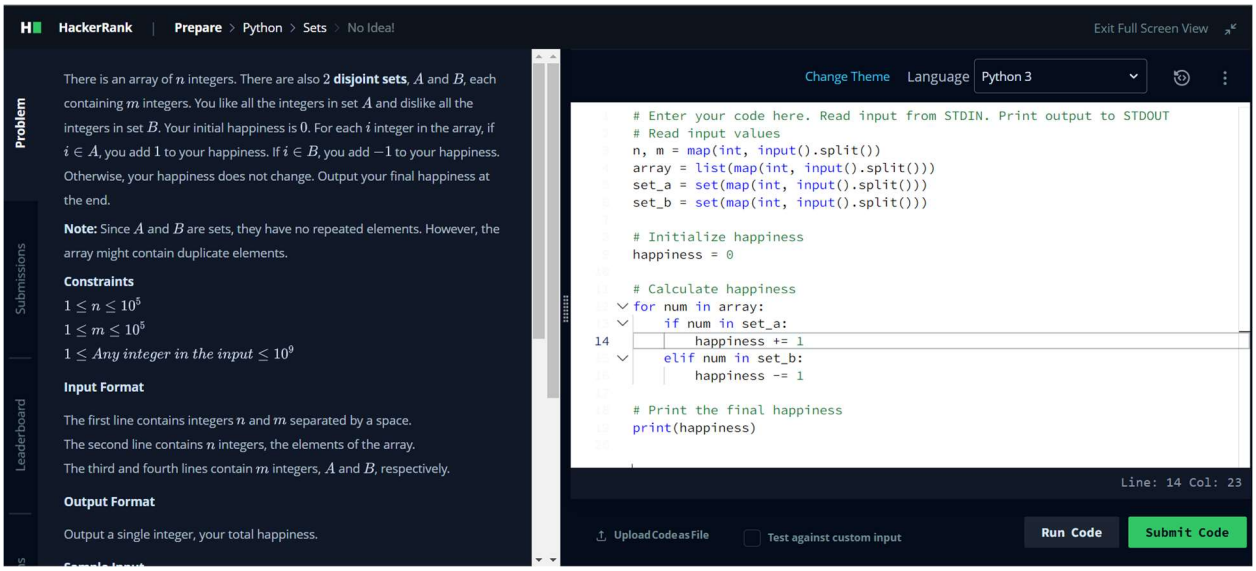


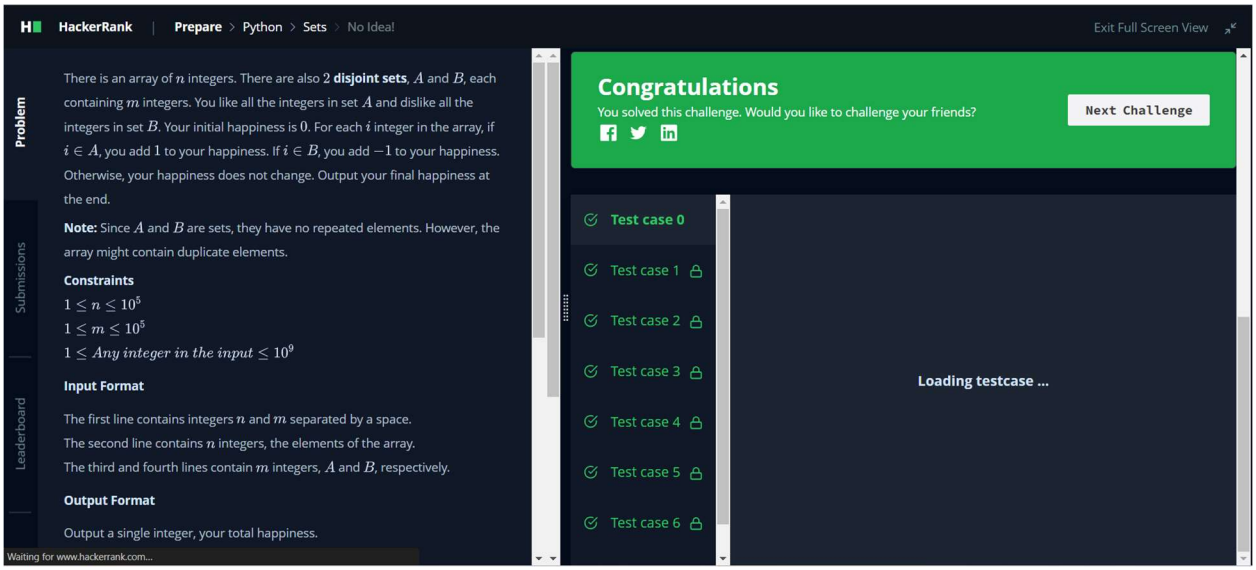
5. Find Angle MBC:



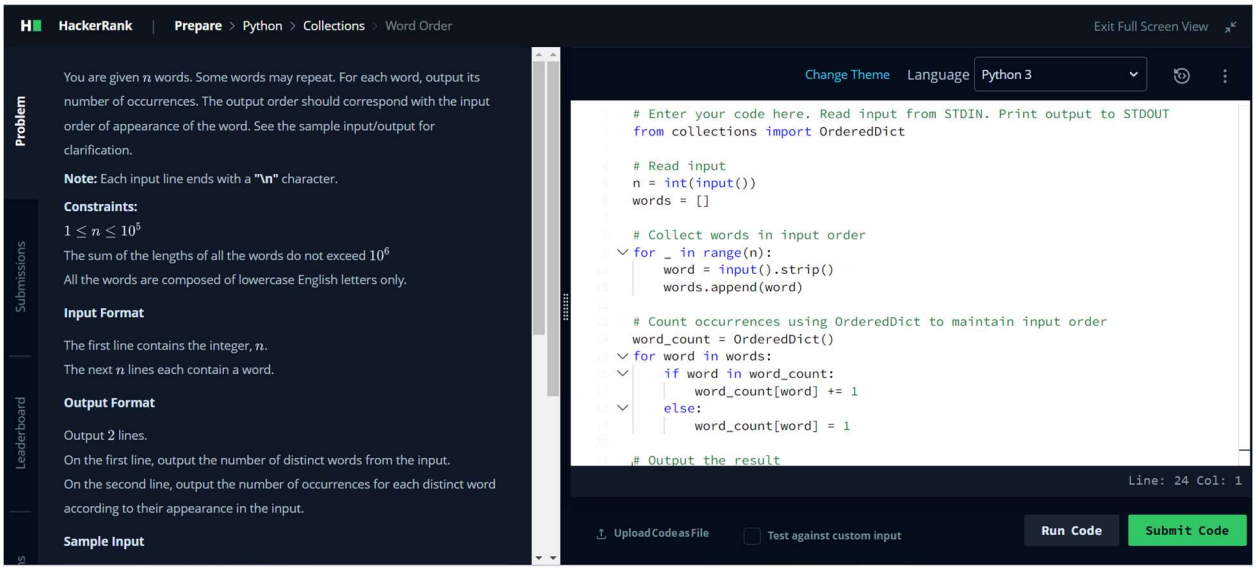

HackerRank | Prepare > Python > Math > Find Angle MBC
Exit Full Screen View
Problem
Submissions
Leaderboard
A
M
θ°
B
C
ABC is a right triangle, 90° at B.
Therefore, $\angle ABC = 90^\circ$.
Point M is the midpoint of hypotenuse AC.
You are given the lengths AB and BC.
Your task is to find $\angle MBC$ (angle θ°, as shown in the figure) in degrees.
Input Format
Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Compiler Message
Success
Input (stdin)
Download
10
10
Expected Output
Download
45°

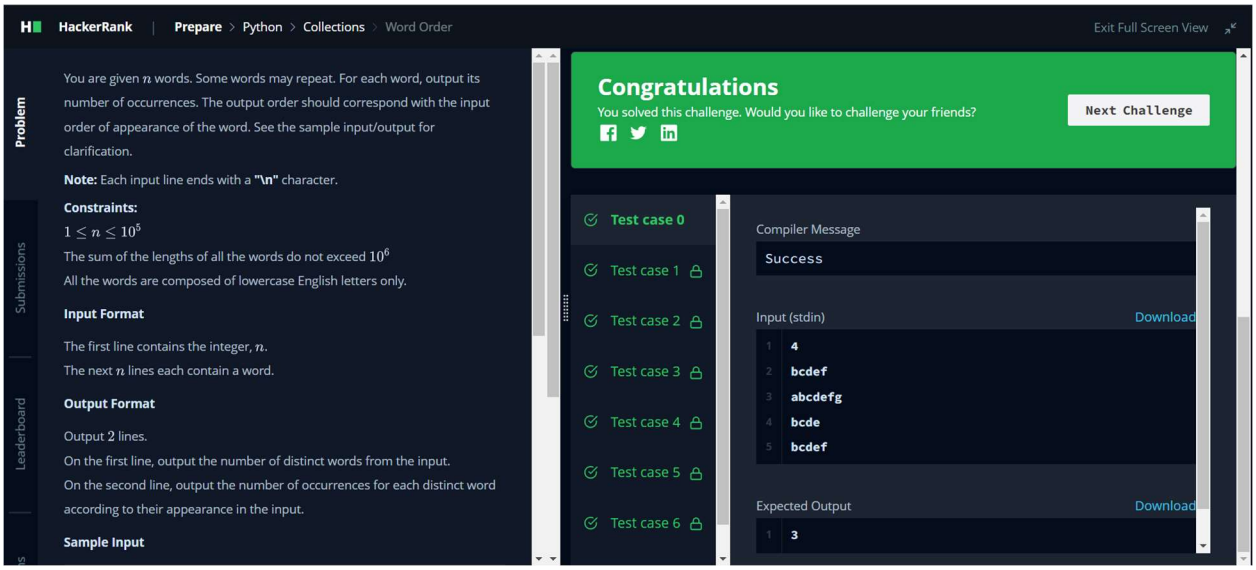
6. No Idea:


HackerRank | Prepare > Python > Sets > No Idea!
Exit Full Screen View
Problem
Submissions
Leaderboard
There is an array of n integers. There are also 2 **disjoint sets**, A and B, each containing m integers. You like all the integers in set A and dislike all the integers in set B. Your initial happiness is 0. For each i integer in the array, if $i \in A$, you add 1 to your happiness. If $i \in B$, you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.
Note: Since A and B are sets, they have no repeated elements. However, the array might contain duplicate elements.
Constraints
$1 \leq n \leq 10^5$
$1 \leq m \leq 10^5$
$1 \leq Any\ integer\ in\ the\ input \leq 10^9$
Input Format
The first line contains integers n and m separated by a space.
The second line contains n integers, the elements of the array.
The third and fourth lines contain m integers, A and B, respectively.
Output Format
Output a single integer, your total happiness.
Change Theme
Language
Python 3
Enter your code here. Read input from STDIN. Print output to STDOUT
Read input values
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()))
# Initialize happiness
happiness = 0
# Calculate happiness
for num in array:
if num in set_a:
happiness += 1
elif num in set_b:
happiness -= 1
# Print the final happiness
print(happiness)
Line: 14 Col: 23
Upload Code as File
Test against custom input
Run Code
Submit Code


HackerRank | Prepare > Python > Sets > No Idea!
Exit Full Screen View
Problem
Submissions
Leaderboard
There is an array of n integers. There are also 2 **disjoint sets**, A and B, each containing m integers. You like all the integers in set A and dislike all the integers in set B. Your initial happiness is 0. For each i integer in the array, if $i \in A$, you add 1 to your happiness. If $i \in B$, you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.
Note: Since A and B are sets, they have no repeated elements. However, the array might contain duplicate elements.
Constraints
$1 \leq n \leq 10^5$
$1 \leq m \leq 10^5$
$1 \leq Any\ integer\ in\ the\ input \leq 10^9$
Input Format
The first line contains integers n and m separated by a space.
The second line contains n integers, the elements of the array.
The third and fourth lines contain m integers, A and B, respectively.
Output Format
Output a single integer, your total happiness.
Waiting for www.hackerrank.com...
Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Loading testcase ...

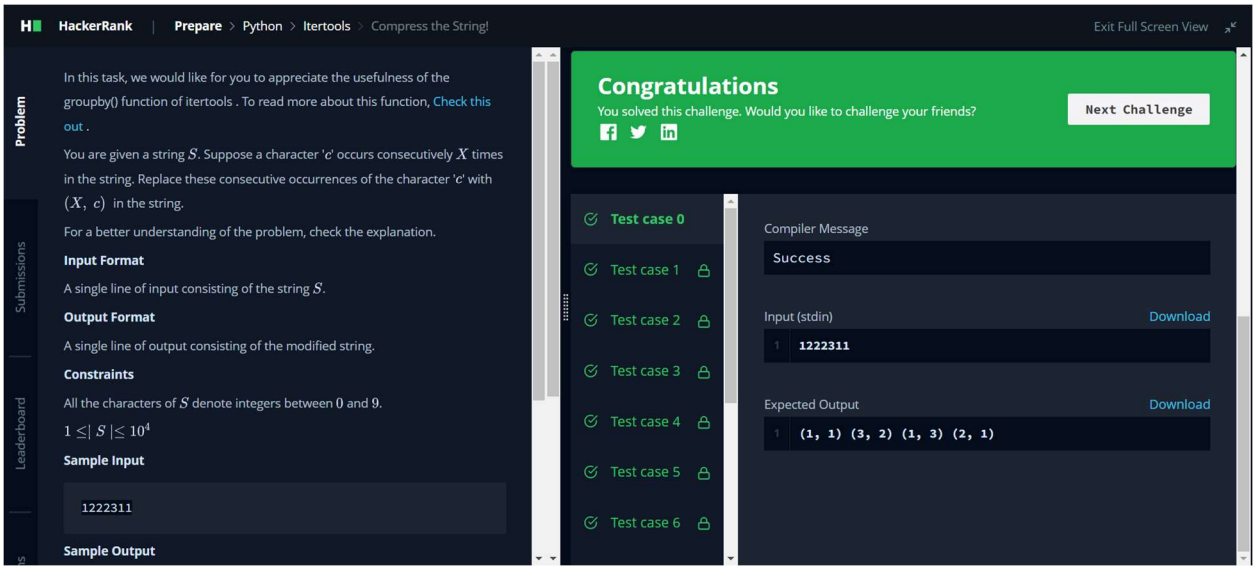
7. Word Order:


HackerRank | Prepare > Python > Collections > Word Order
Exit Full Screen View
Problem
Submissions
Leaderboard
You are given n words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.
Note: Each input line ends with a **"\n"** character.
Constraints:
$1 \leq n \leq 10^5$
The sum of the lengths of all the words do not exceed 10^6
All the words are composed of lowercase English letters only.
Input Format
The first line contains the integer, n.
The next n lines each contain a word.
Output Format
Output 2 lines.
On the first line, output the number of distinct words from the input.
On the second line, output the number of occurrences for each distinct word according to their appearance in the input.
Sample Input
Change Theme
Language
Python 3
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import OrderedDict
# Read input
n = int(input())
words = []
# Collect words in input order
for _ in range(n):
word = input().strip()
words.append(word)
# Count occurrences using OrderedDict to maintain input order
word_count = OrderedDict()
for word in words:
if word in word_count:
word_count[word] += 1
else:
word_count[word] = 1
# Output the result
Line: 24 Col: 1
Upload Code as File
Test against custom input
Run Code
Submit Code



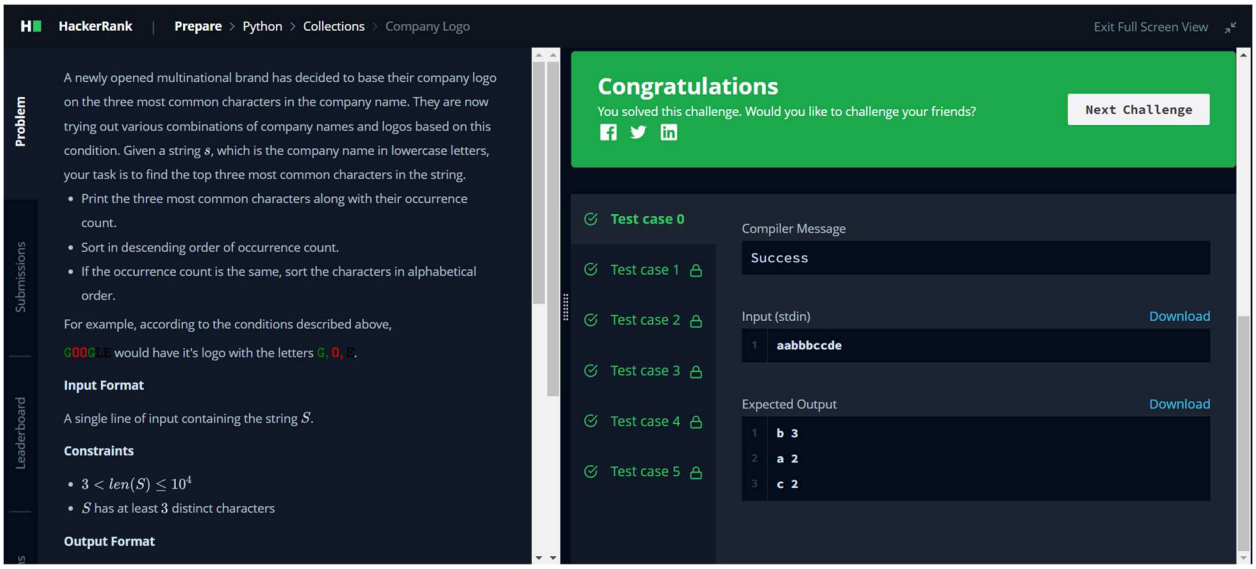
8. Compress the String:

```
from itertools import groupby

def compress_string(s):
    compressed_string = []
    for key, group in groupby(s):
        count = len(list(group))
        compressed_string.append((int(key), count))

    result = ' '.join([f"({count}, {char})" for char, count in com
pressed_string])
    return result

# Read input
s = input().strip()

# Output the modified string
result = compress_string(s)
print(result)
```



9. Company Logo:

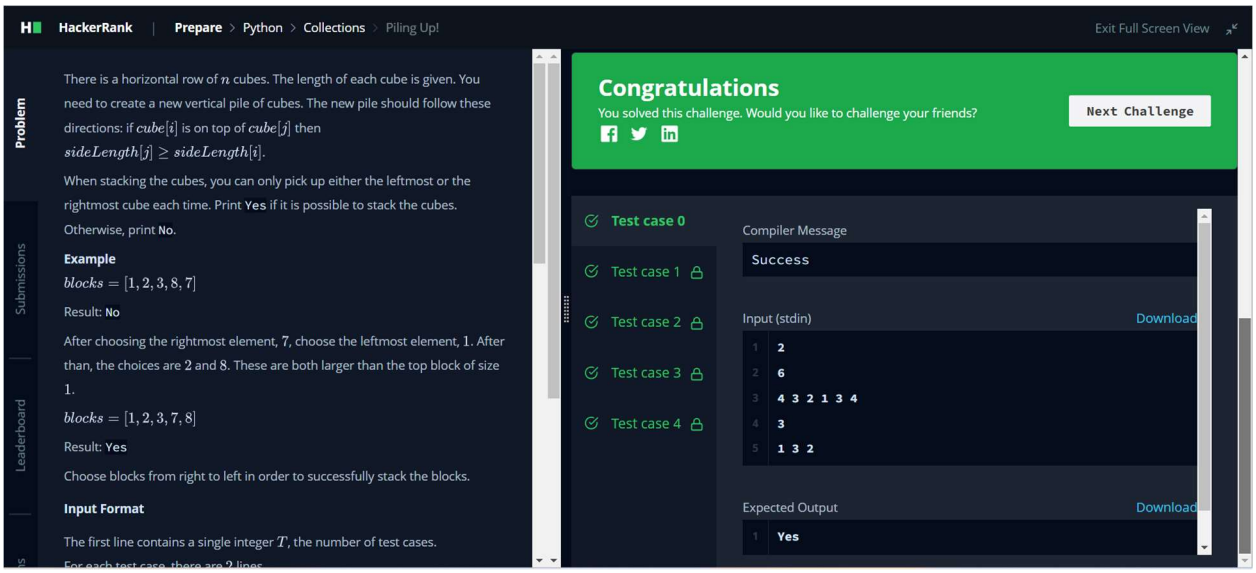
```
#!/bin/python3

from collections import Counter

def top_three_characters(s):
    # Count occurrences of each character
    char_count = Counter(s)

    # Sort by occurrence count (descending) and then by character (ascending)
    sorted_chars = sorted(char_count.items(), key=lambda x: (-x[1], x[0]))

    # Print the top three characters and their occurrence count
    for char, count in sorted_chars[:3]:
        print(f"{char} {count}")

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



10. Piling Up:

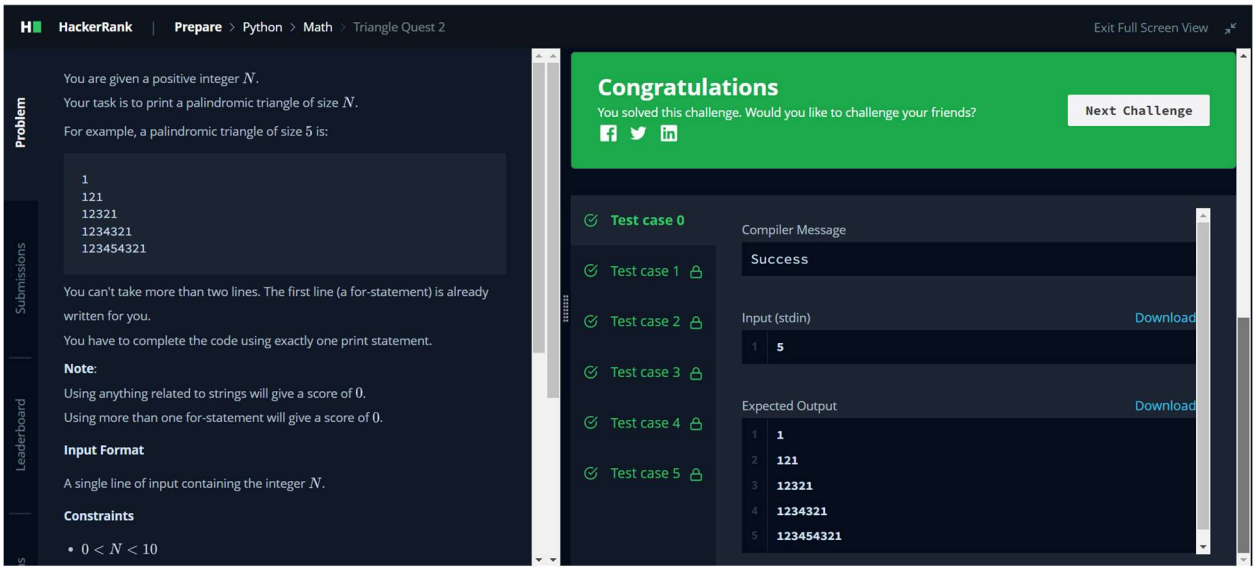
```
def can_stack_cubes(test_cases):
    for cubes in test_cases:
        left, right = 0, len(cubes) - 1
        top_cube = float('inf')

        while left <= right:
            if cubes[left] >= cubes[right] and cubes[left] <= top_
cube:
                top_cube = cubes[left]
                left += 1
            elif cubes[right] >= cubes[left] and cubes[right] <= t
op_cube:
                top_cube = cubes[right]
                right -= 1
            else:
                print("No")
                break
        else:
            print("Yes")

if __name__ == '__main__':
    t = int(input().strip())  # Number of test cases
    test_cases = []

    for _ in range(t):
        _ = int(input().strip())  # Not needed
```

```
        cubes = list(map(int, input().split()))
        test_cases.append(cubes)

    can_stack_cubes(test_cases)
```



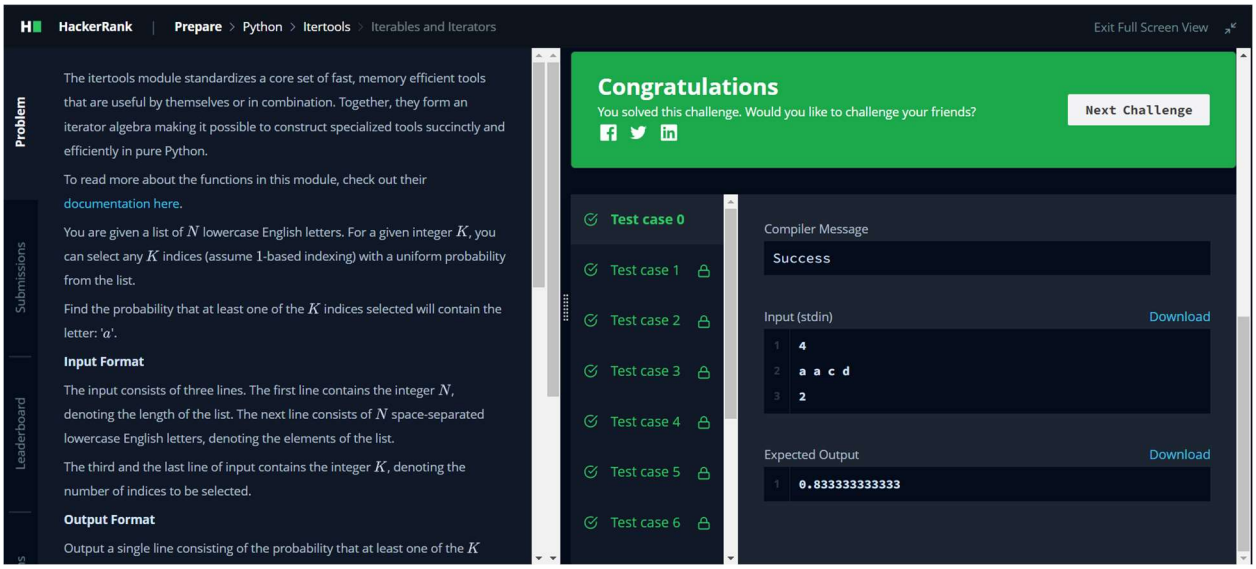
11. Triangle Quest 2:

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



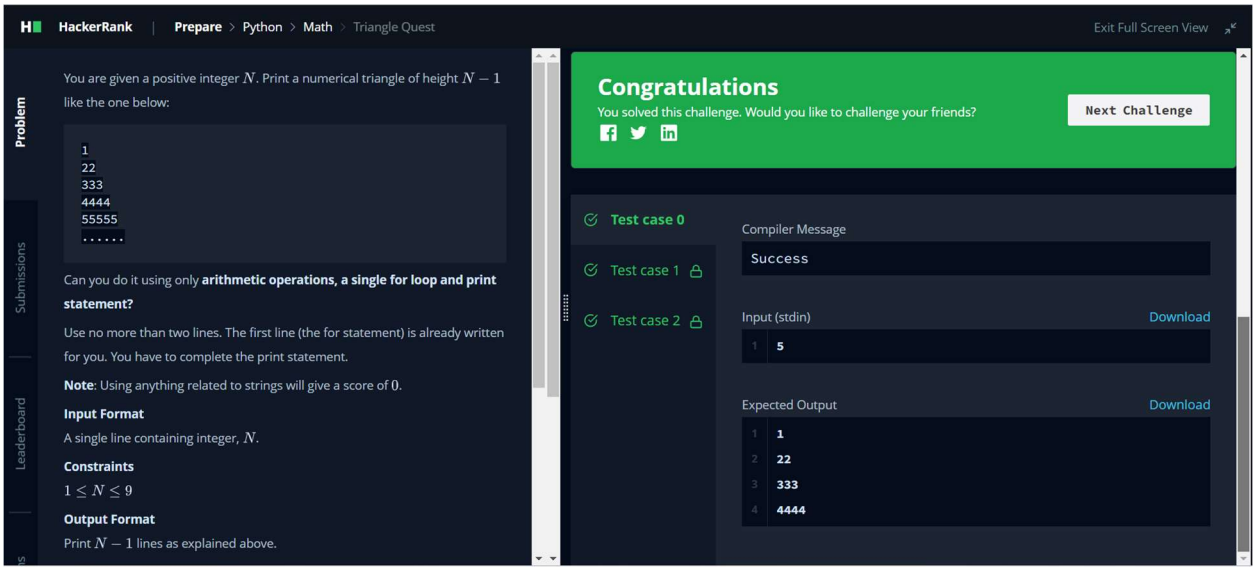
12. Iterables and Iterators

```
from itertools import combinations

def calculate_probability(n, letters, k):
    total_combinations = list(combinations(letters, k))
    combinations_without_a = [comb for comb in total_combinations
if 'a' not in comb]

    probability_no_a = len(combinations_without_a) / len(total_com
binations)
    probability_at_least_one_a = 1 - probability_no_a

    return round(probability_at_least_one_a, 3)

if __name__ == "__main__":
    n = int(input())
    letters = input().split()
    k = int(input())

    result = calculate_probability(n, letters, k)
    print(result)
```



13. Triangle Quest:

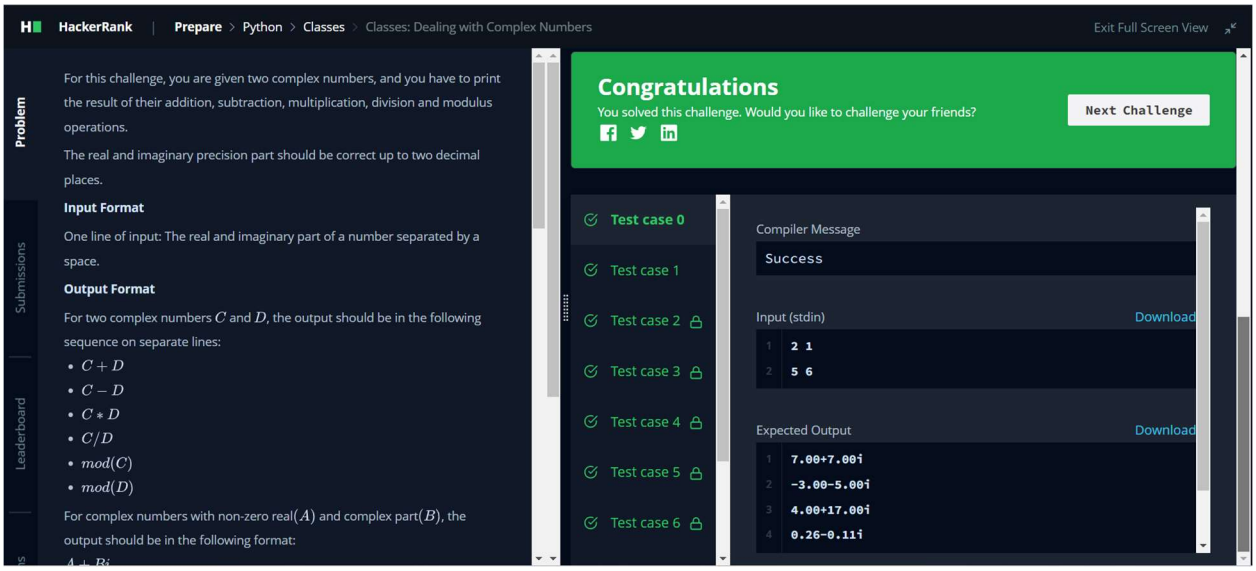
```
for i in range(1, int(input())):
    print((10 ** i - 1) // 9 * i)
```



14. Classes: Dealing with Complex Numbers

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

    def __add__(self, other):
        return Complex(self.real + other.real, self.imaginary + ot
her.imaginary)

    def __sub__(self, other):
        return Complex(self.real - other.real, self.imaginary - ot
her.imaginary)

    def __mul__(self, other):
        real_part = self.real * other.real - self.imaginary * othe
r.imaginary
        imag_part = self.real * other.imaginary + self.imaginary *
 other.real
        return Complex(real_part, imag_part)

    def __truediv__(self, other):
        conjugate = Complex(other.real, -other.imaginary)
        denominator = other * conjugate
        result = self * conjugate
        result.real /= denominator.real
        result.imaginary /= denominator.real
```

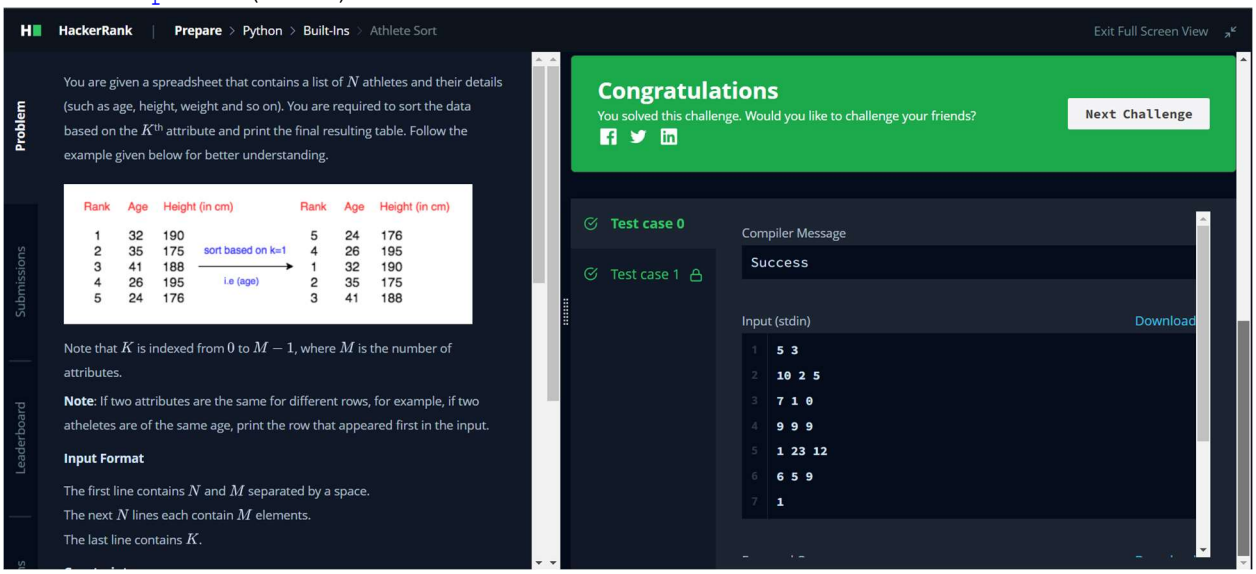
```
        return result

    def mod(self):
        return Complex((self.real**2 + self.imaginary**2)**0.5, 0)

    def __str__(self):
        return "%.2f%+.2fi" % (self.real, self.imaginary)


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')
```



15. Athlete Sort

```
#!/bin/python3

import math
import os
import random
import re
import sys

if __name__ == '__main__':
```

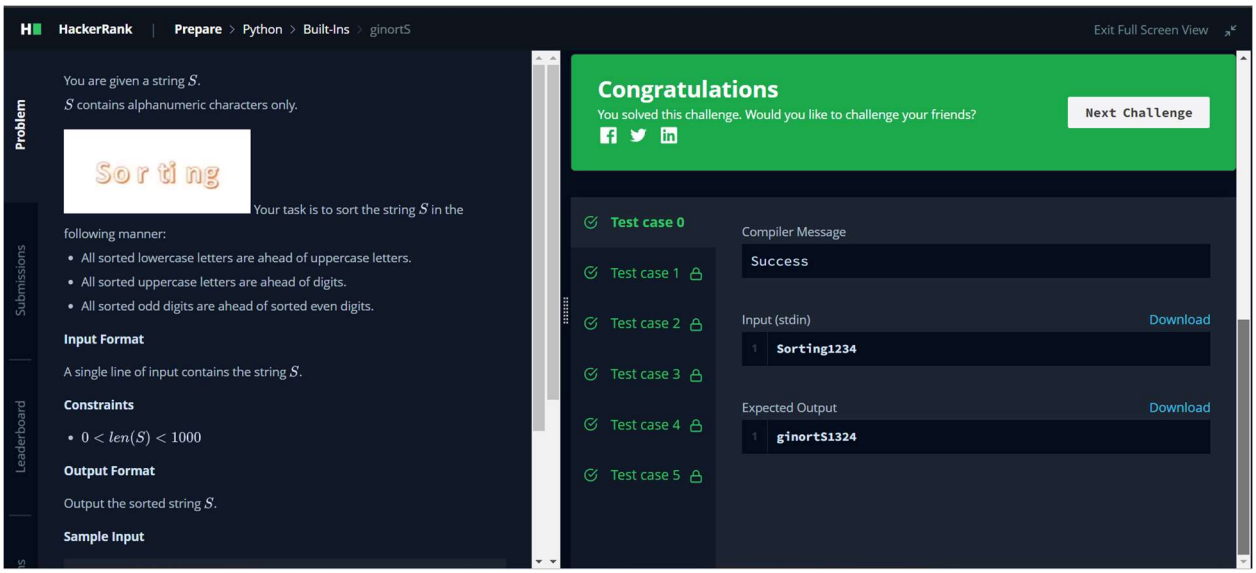
```
nm = input().split()

n = int(nm[0])
m = int(nm[1])

arr = []

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

k = int(input())

# Sort the array based on the k-th attribute
sorted_arr = sorted(arr, key=lambda x: x[k])

# Print the sorted array
for row in sorted_arr:
    print(*row)
```



16. ginortS

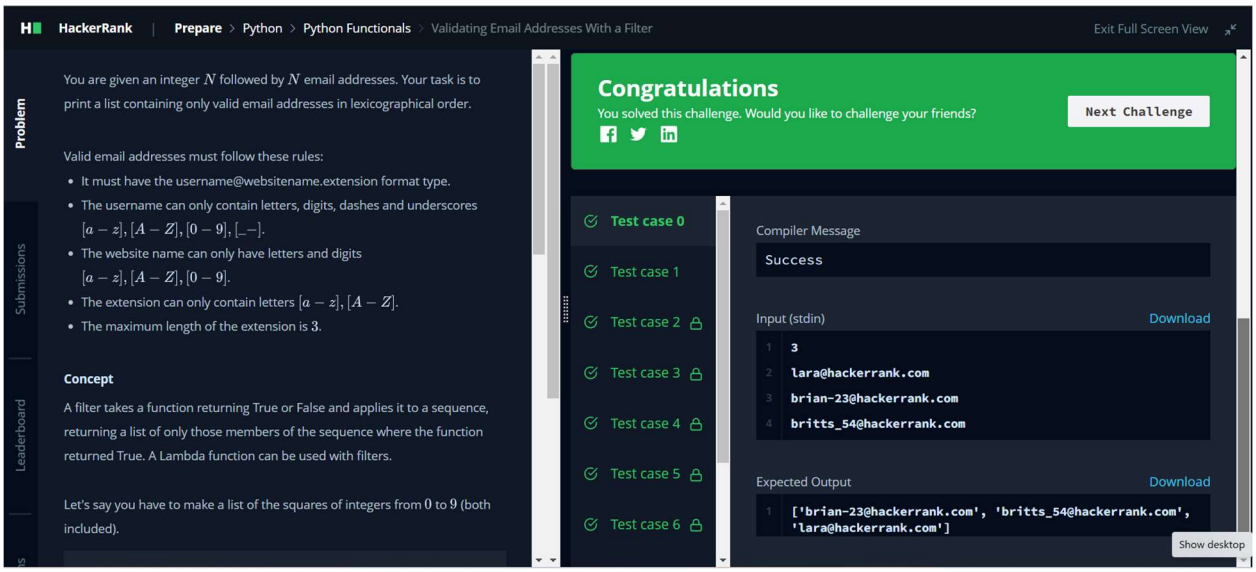
```
# Enter your code here. Read input from STDIN. Print output to STD
OUT
s = input()

# Sort the string based on the specified criteria
sorted_string = sorted(s, key=lambda x: (x.isdigit(), x.isdigit()
and int(x) % 2 == 0, x.isupper(), x))
```

```
# Print the sorted string
print(''.join(sorted_string))
```



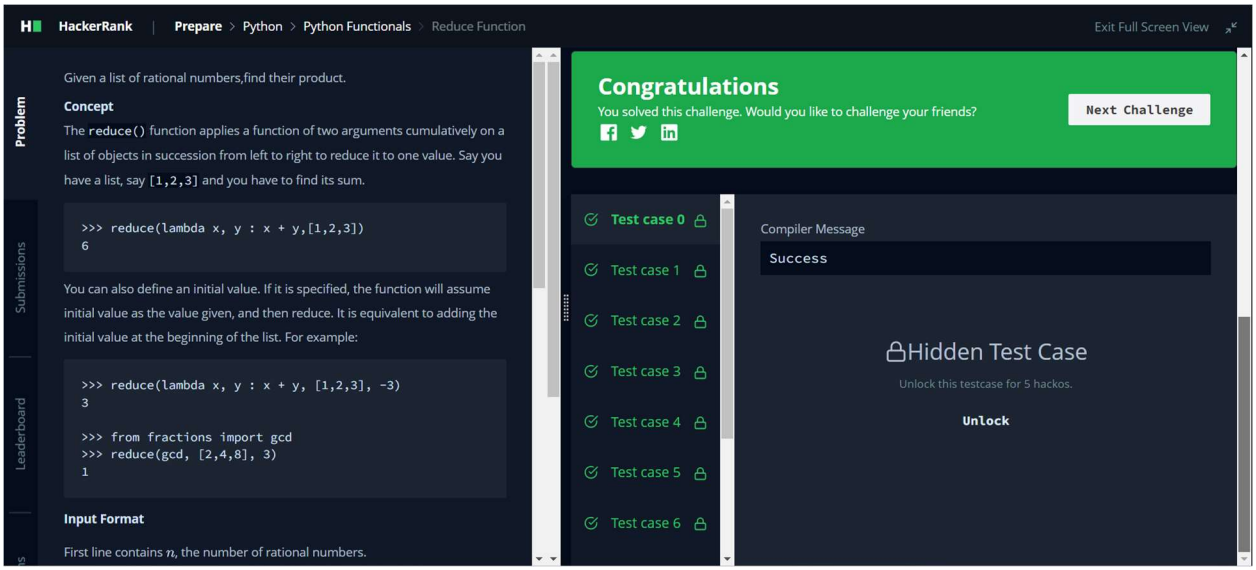
17. Validating Email Addresses With a Filter

```
import re

def fun(s):
    # Define the pattern for a valid email address
    pattern = r'^[a-zA-Z0-9_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'
    return bool(re.match(pattern, s))

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

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)
```



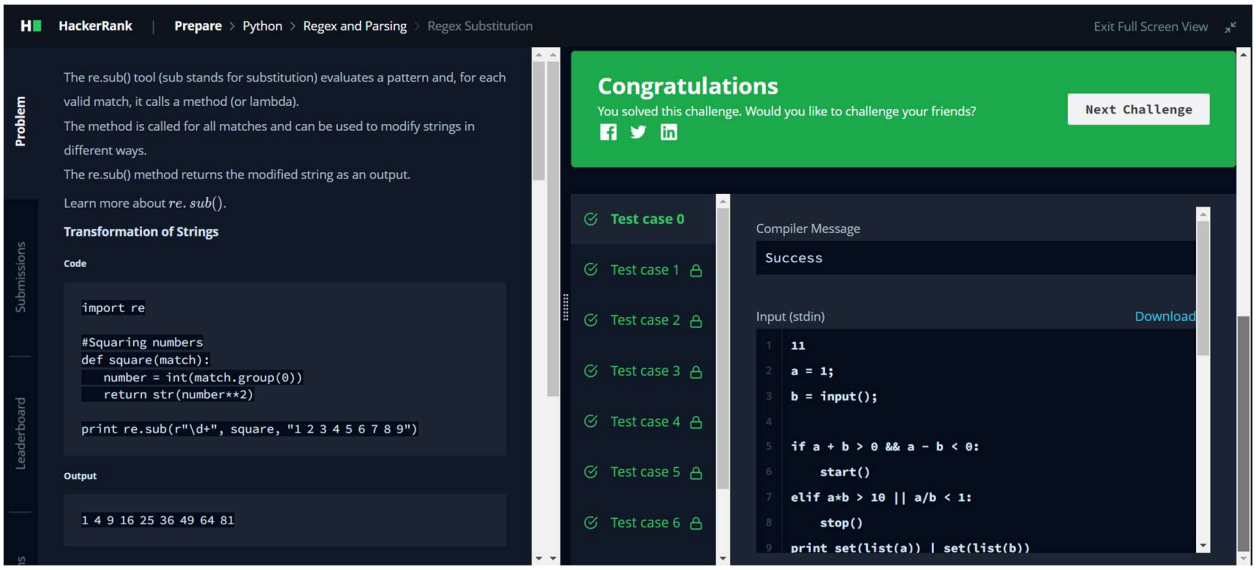
18. Reduce Function

```
from fractions import Fraction
from functools import reduce

def product(fracs):
    t = reduce(lambda x, y: x * y, fracs)
    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)
```



19. Regex Substitution

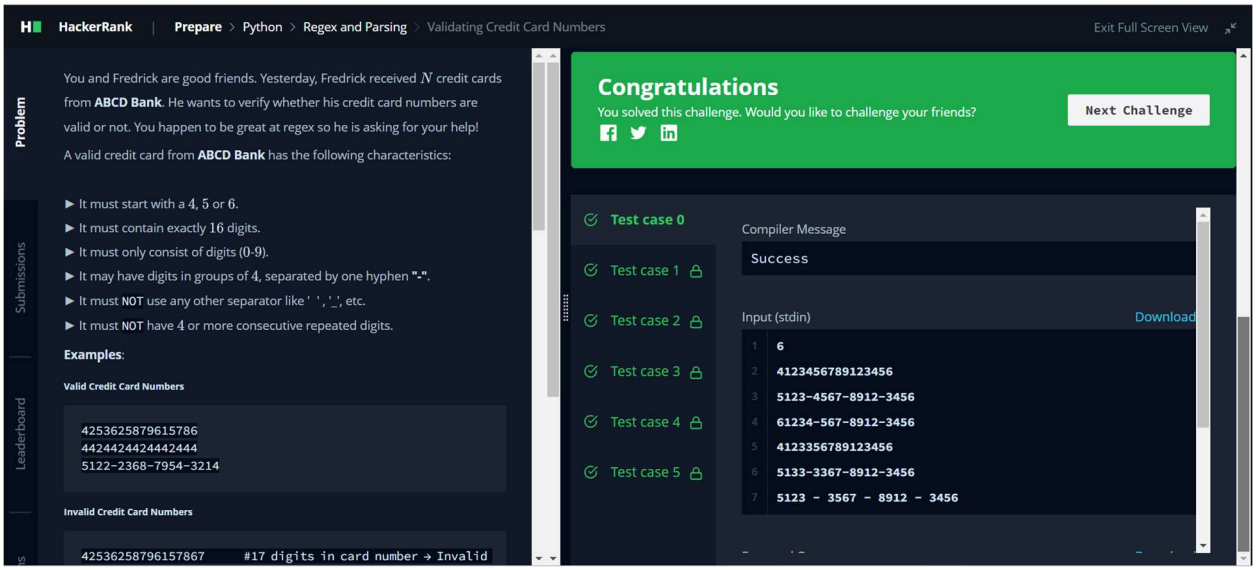
```
import re

n = int(input())
string = '\n'.join([input() for _ in range(n)])
print(re.sub(r"(?<=(\s))\|\|(?=(\s))", 'or', re.sub(r"(?<=(\s))&&(
?=(\s))", 'and', string)))
```



20. Validating Credit Card Numbers

```
import re
```

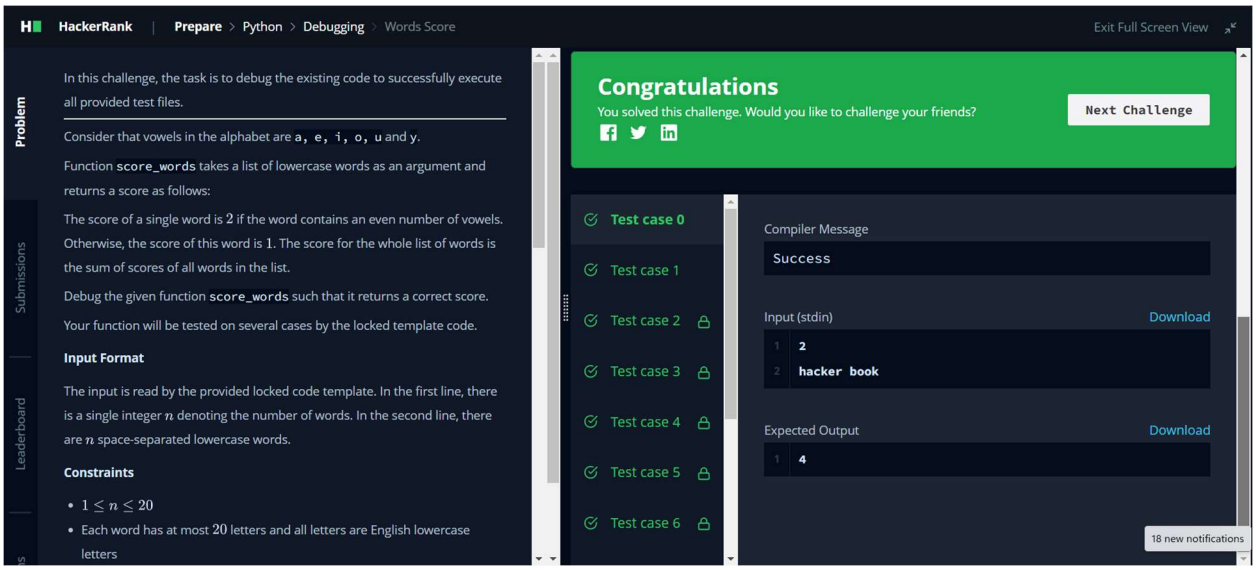
```python
def is_valid_credit_card(card_number):
    pattern = re.compile(
        r'^([4-6]\d{3}-?\d{4}-?\d{4}-?\d{4}|[4-6]\d{15})$'
    )
    if re.match(pattern, card_number):
        card_number = card_number.replace('-', '')
        if re.search(r'(\d)\1{3,}', card_number):
            return 'Invalid'
        else:
            return 'Valid'
    else:
        return 'Invalid'

def main():
    n = int(input())
    for _ in range(n):
        card_number = input().strip()
        result = is_valid_credit_card(card_number)
        print(result)

if __name__ == "__main__":
    main()
```



21. Words Score

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

def score_words(words):
```

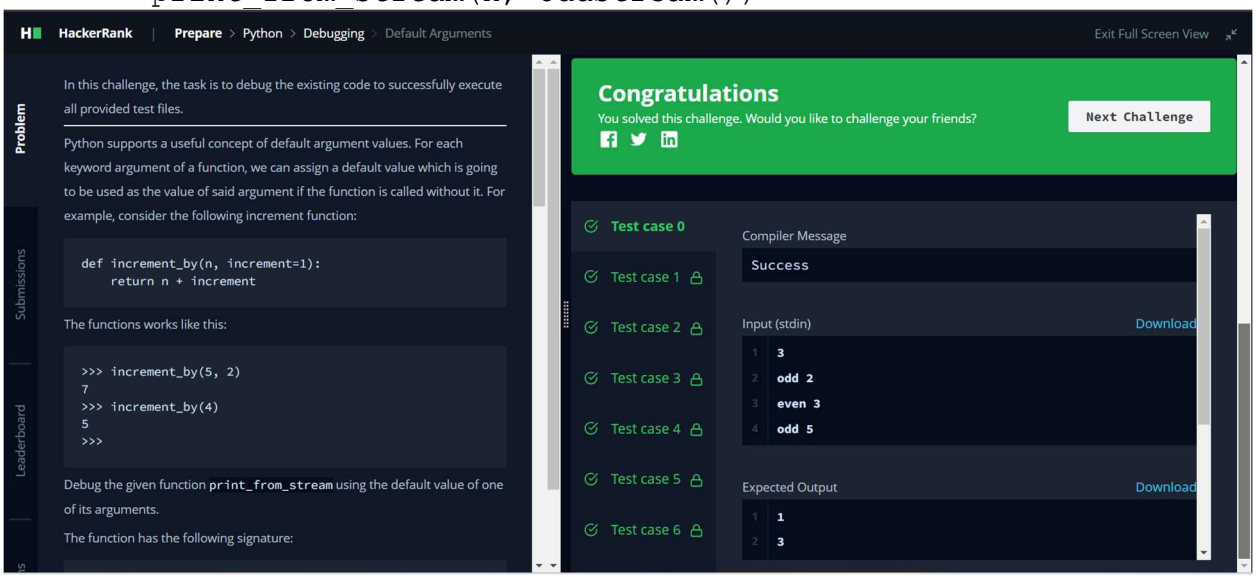
```
    score = 0
    for word in words:
        num_vowels = 0
        for letter in word:
            if is_vowel(letter):
                num_vowels += 1
        if num_vowels % 2 == 0:
            score += 2
        else:
            score += 1
    return score


n = int(input())
words = input().split()
print(score_words(words))
```



22. 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
```

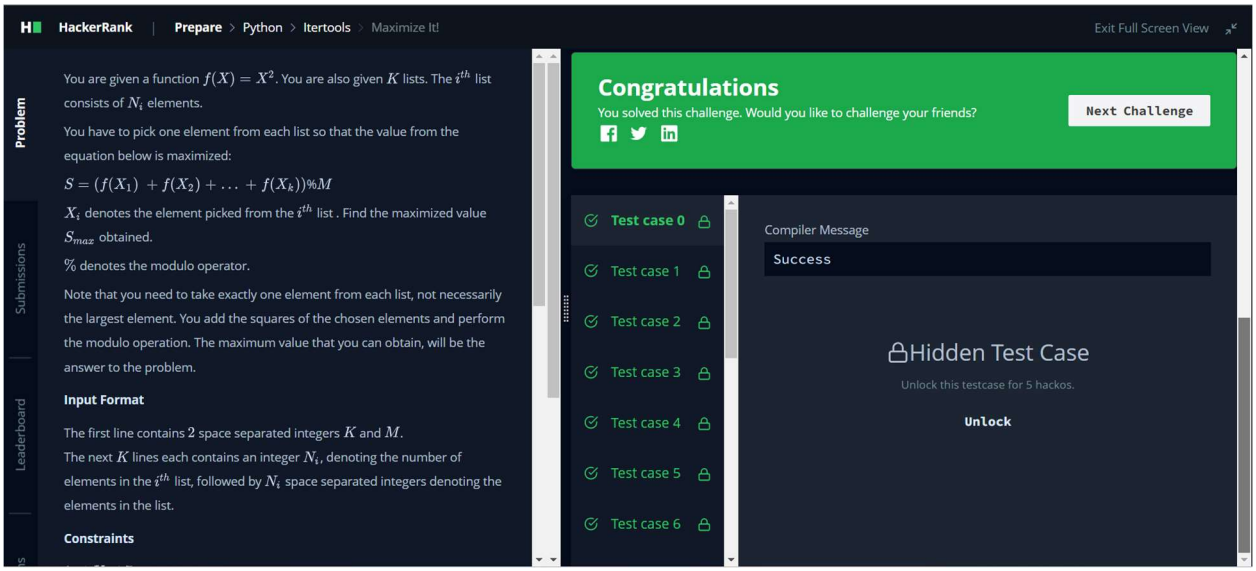
```
class OddStream(object):
    def __init__(self):
        self.current = 1

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

def print_from_stream(n, stream=None):
    if stream is None:
        stream = EvenStream()

    for _ in range(n):
        print(stream.get_next())


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())
```



Hard Difficulty Challenges:

1. Maximize It!

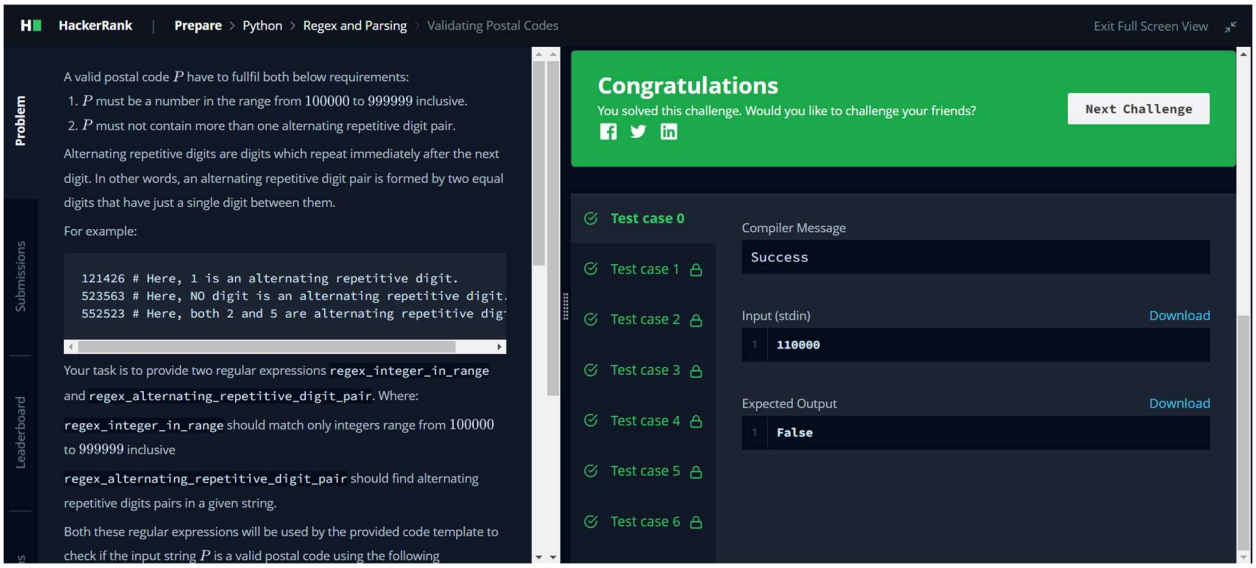
```
from itertools import product

from itertools import product

def find_max(k_lists, M):
    f = lambda x: x*x
    combinations = product(*k_lists)
    results = [(sum([f(a) for a in x]) % M) for x in combinations]
    # Return the maximum value
    return max(results)


K, M = tuple(map(int, input().split(" ")))
k_lists = [list(map(int, input().split(" ")))[1:] for _ in range(K
)]
print(find_max(k_lists, M))
```



2. Validating Postal Codes

```
regex_integer_in_range = r"^[1-9]\d{5}$"
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)"


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

```
#!/bin/python3

import math
import os
import random
import re
import sys

m,n = tuple(map(int, input().split()))
a=[]
for i in range(m):
    a.append(list(input().rstrip('\n')))
s=""
for i in range(n):
    for j in range(m):
        s+=a[j][i]
p = re.compile(r'([a-zA-z0-9][\$\#\!\@\ \%\&]{1,99}[a-zA-Z0-9])')
matches = list(map(lambda x: x[1:-1],re.findall(p, s)))
b=s[:]
for i in matches:
```

```
    b=b.replace(i, " ", 1)

print(b)
```