Extra credit - Hacker Rank

EASY

Simplistic prompt: Write a python function solve(s) ensure that the first and last names of people begin with a capital letter in their passports. For example, alison heck should be capitalised correctly as Alison Heck.

```python
#!/bin/python3 ...

# Complete the solve function below.
def solve(s):
    """
    Capitalizes the first letter of each word in a string, ensuring proper
    capitalization for names, as required for a passport.

    Args:
        s: The input string containing the full name.

    Returns:
        The string with each word capitalized.
    """
    # A more robust approach than str.title() to handle cases with numbers
    # and multiple spaces correctly.
    capitalized_words = []

    # Split the string by spaces. This handles multiple spaces between words
    # by creating empty strings in the list, which will be filtered out.
    words = s.split(' ')

    for word in words:
        if word:  # Check if the word is not an empty string
            capitalized_words.append(word.capitalize())
        else:
            capitalized_words.append('')

    # Join the capitalized words back together with a single space.
    return ' '.join(capitalized_words)
if __name__ == '__main__': ...
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
hello world
```

Your Output (stdout)

```
Hello World
```

Expected Output Download

```
Hello World
```

Prompt Engineering techniques: This solution is too slow; can you suggest a more efficient approach with a time complexity of O(n)?

Response: a slightly more "Pythonic" and often faster way to write the same logic is by using a list comprehension.

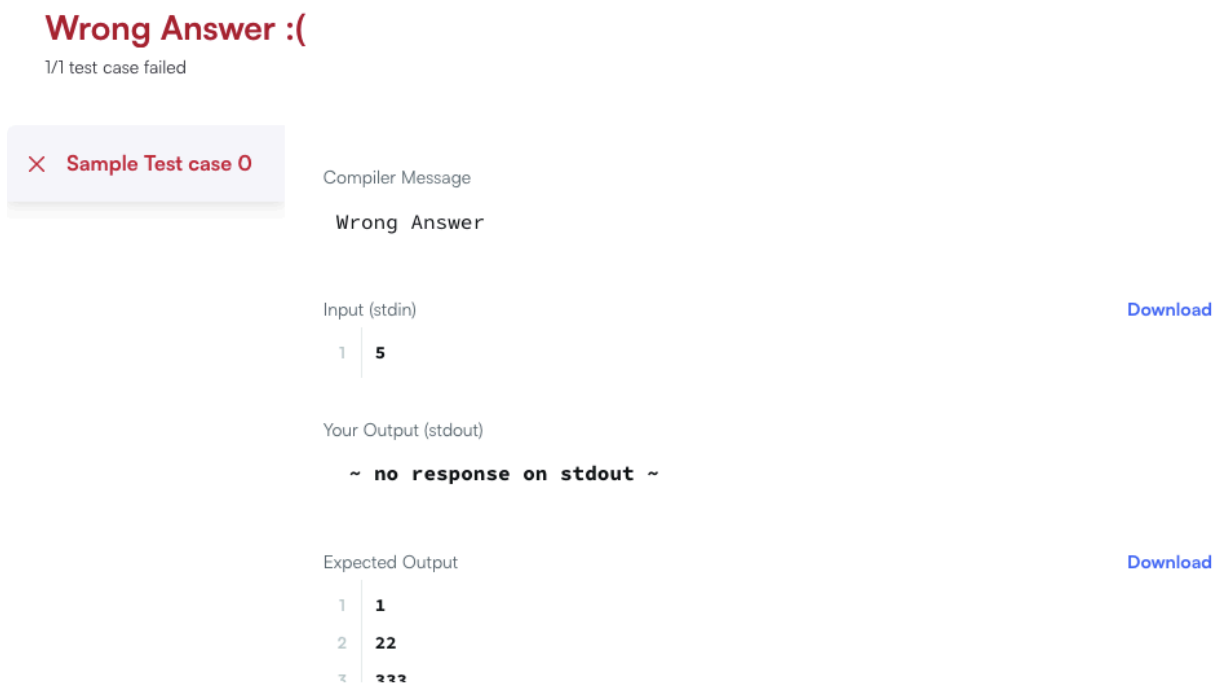
```python
# Complete the solve function below.
def solve(s):
    """
    Capitalizes the first letter of each word in a string.

    Args:
        s: The input string containing the full name.

    Returns:
        The string with each word capitalized.
    """
    # Use a list comprehension for a more concise and efficient approach.
    # The split() method correctly handles multiple spaces by treating them as
    # delimiters and can be more robust than manual character iteration.
    words = s.split(' ')
    capitalized_words = [word.capitalize() if word else '' for word in words]

    return ' '.join(capitalized_words)

if __name__ == '__main__':…
```

Medium

Simplistic prompt: Write a python function given a positive integer . Print a numerical triangle of height N-1

```python
def print_triangle(N):
    """
    Prints a numerical triangle of a specified height, using only
    arithmetic operations, a single for loop, and a print statement.

    Args:
        N: A positive integer representing the height of the triangle.
    """
```

```
    # Check for a positive integer, as required by the problem.
    if N <= 0:
        print("Input must be a positive integer.")
        return

    # Use a single for loop to iterate through each row of the triangle.
    for i in range(1, N):
        # The key to this solution is a mathematical trick.
        # (10**i - 1) // 9 creates a number like 1, 11, 111, etc.
        # Multiplying by 'i' generates the sequence for each row (1, 22,
333, etc.).
        print((10**i // 9) * i)
```

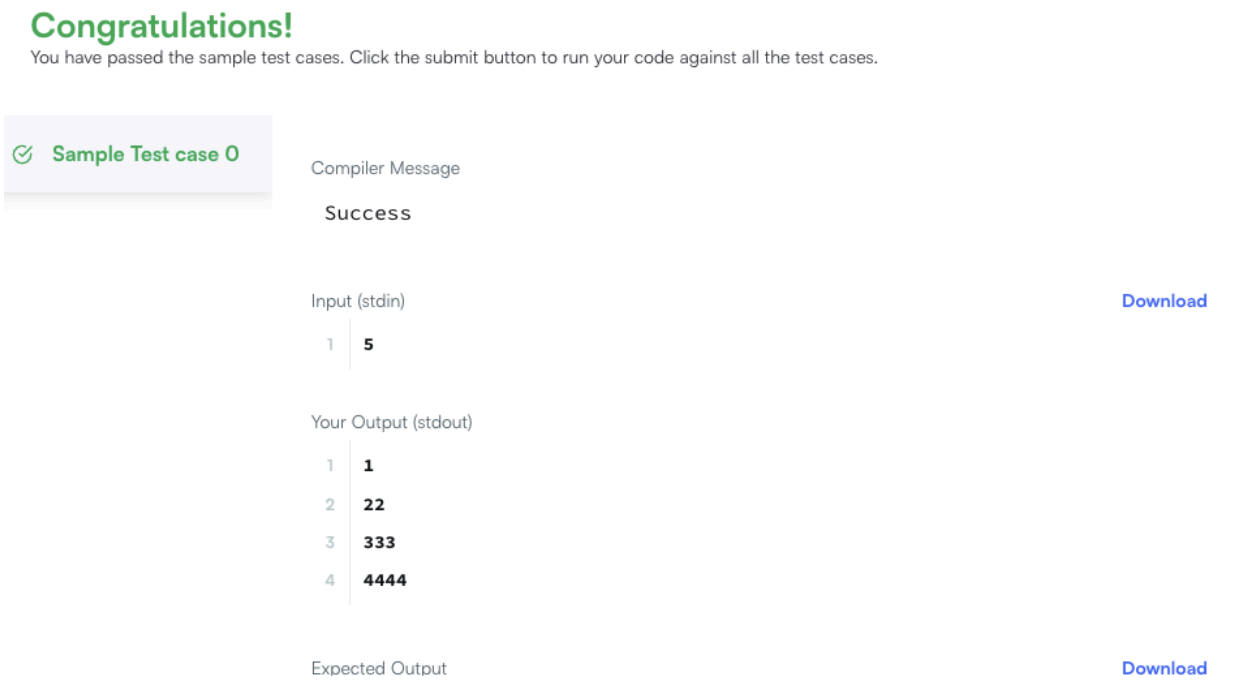


Prompt Engineering techniques:

1 - Can you do it using only arithmetic operations, a single for loop and print statement?

2 - for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank line also Print - You have to complete the print statement.

```
for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank
also
    print ((10**i // 9) * i)
```



Hard

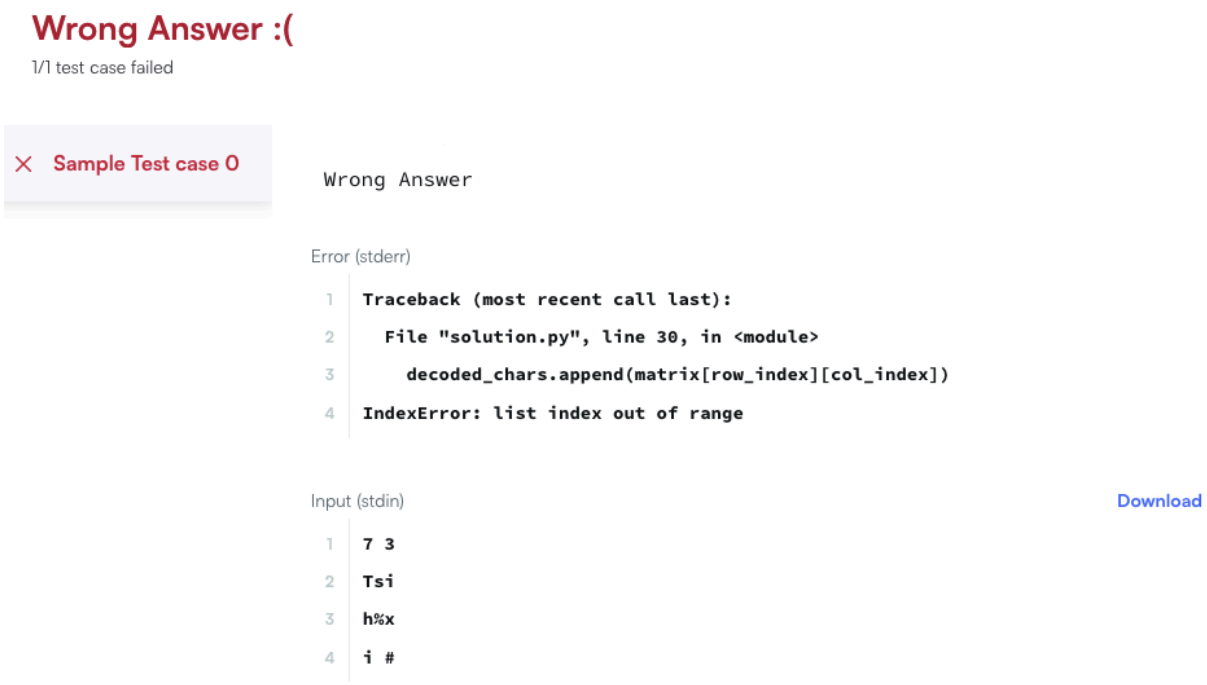
Simplistic prompt: Write a python function for Neo has a complex matrix script. The matrix script is a X grid of strings. It consists of alphanumeric characters, spaces and symbols (!,@,#,$,%,&).To decode the script, Neo needs to read each column and select only the alphanumeric characters and connect them. Neo reads the column from top to bottom and starts reading from the leftmost column.

If there are symbols or spaces between two alphanumeric characters of the decoded script, then Neo replaces them with a single space ' ' for better readability.

Neo feels that there is no need to use 'if' conditions for decoding.

Alphanumeric characters consist of: [A-Z, a-z, and 0-9].

```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)
    # Initialize a list to hold the characters of the decoded string.
    decoded_chars = []

    # Iterate through the columns of the matrix.
    for col_index in range(m):
        # Iterate through the rows to read one column at a time.
        for row_index in range(n):
            decoded_chars.append(matrix[row_index][col_index])

    # Join the characters to form a single string.
    decoded_string = "".join(decoded_chars)

    # Use a regular expression to replace sequences of non-alphanumeric
    # characters with a single space. The pattern (?<=\w)\W+(?=\w) uses
    # lookbehind (?<=\w) and lookahead (?=\w) assertions to ensure that
    # the replacement happens only between two alphanumeric characters.
    # This fulfills the "no 'if' conditions' requirement.
    final_string = re.sub(r'(?<=\w)\W+(?=\w)', ' ', decoded_string)

    # Print the final decoded string to standard output.
    print(final_string)

```



Prompt Engineering techniques:

1 - Input Format

The first line contains space-separated integers (rows) and (columns) respectively.

The next lines contain the row elements of the 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)

decoded_chars = []

for col_index in range(m):
    for row_index in range(n):
        decoded_chars.append(matrix[row_index][col_index])

decoded_string = "".join(decoded_chars)

final_string = re.sub(r'(?<=\w)\W+(?=\w)', ' ', decoded_string)

print(final_string)

```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
7 3
Tsi
h%x
i #
sM
$a
#t%
ir!
```

Your Output (stdout)

```
This is Matrix#  %!
```