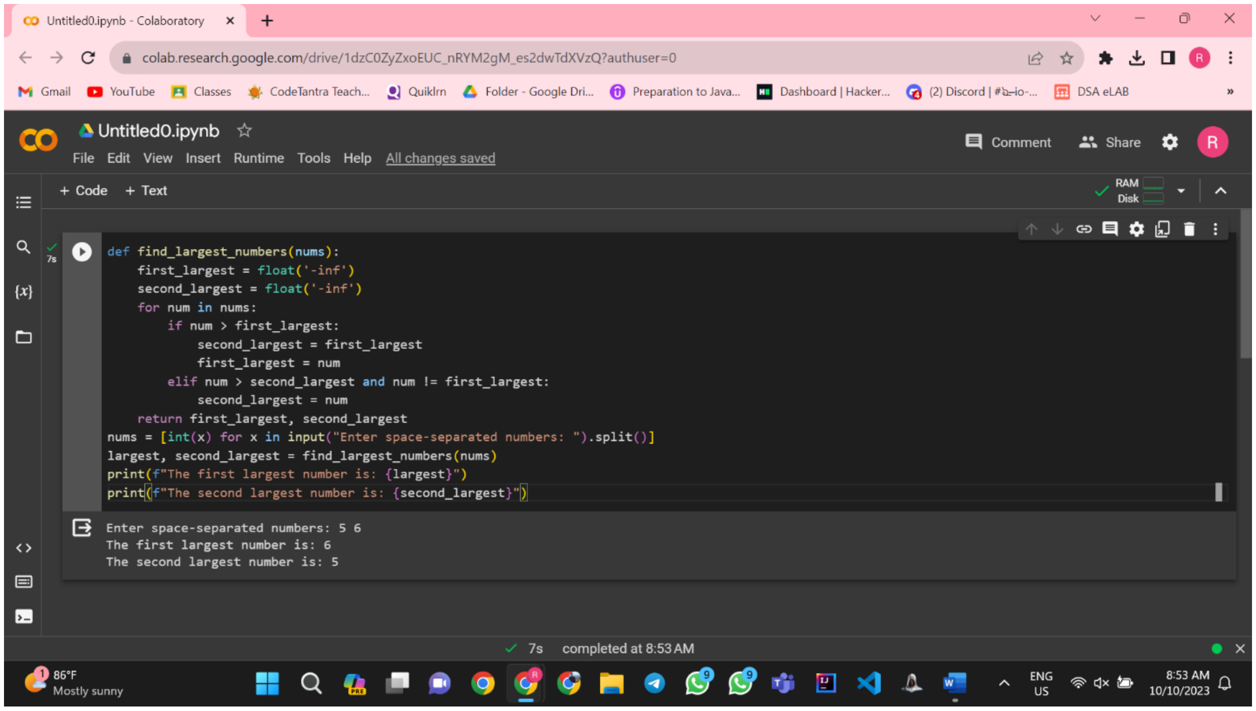
Rohan Soni

RA2211003012027

Week 11

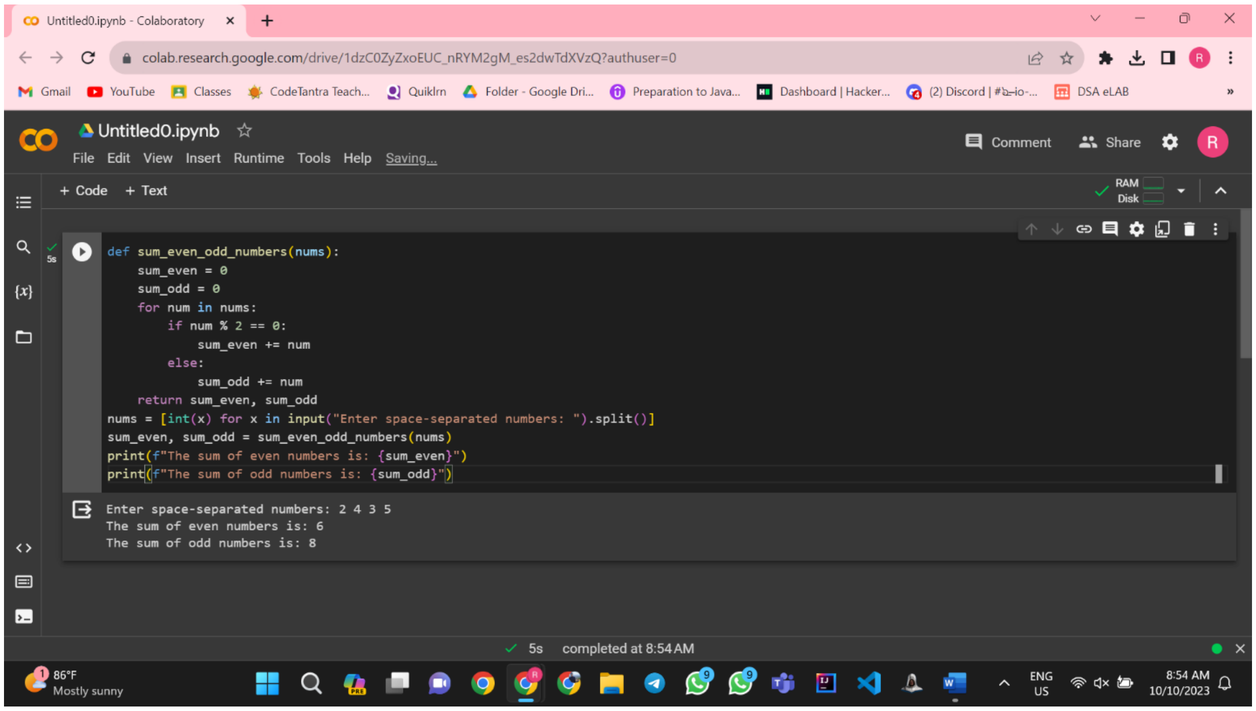
1. Implement a python program to find the first largest and second largest numbers in an Array. Note: should not use any built-in sorting functions or libraries.



```python
def find_largest_numbers(nums):
    first_largest = float('-inf')
    second_largest = float('-inf')
    for num in nums:
        if num > first_largest:
            second_largest = first_largest
            first_largest = num
        elif num > second_largest and num != first_largest:
            second_largest = num
    return first_largest, second_largest
nums = [int(x) for x in input("Enter space-separated numbers: ").split()]
largest, second_largest = find_largest_numbers(nums)
print(f"The first largest number is: {largest}")
print(f"The second largest number is: {second_largest}")
```

```
Enter space-separated numbers: 5 6
The first largest number is: 6
The second largest number is: 5
```

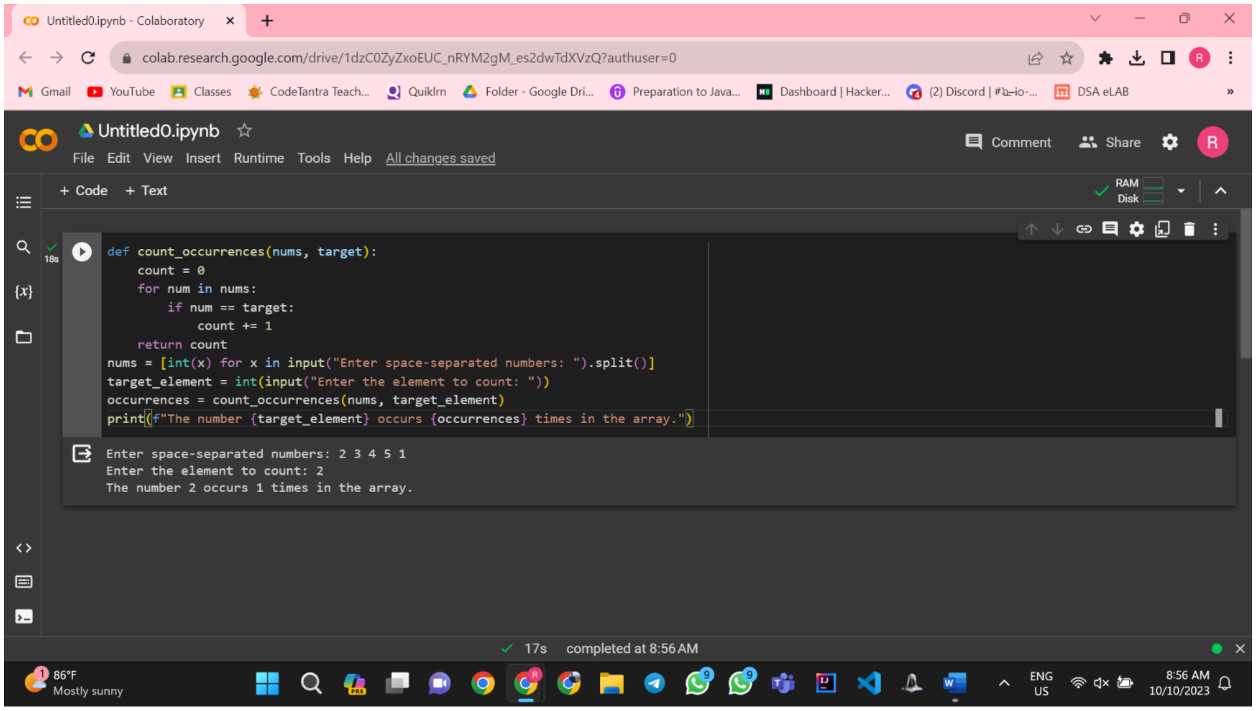
2. Write a Python program to calculate the sum of even numbers and the sum of odd numbers in an array.



```python
def sum_even_odd_numbers(nums):
    sum_even = 0
    sum_odd = 0
    for num in nums:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num
    return sum_even, sum_odd
nums = [int(x) for x in input("Enter space-separated numbers: ").split()]
sum_even, sum_odd = sum_even_odd_numbers(nums)
print(f"The sum of even numbers is: {sum_even}")
print(f"The sum of odd numbers is: {sum_odd}")
```

```
Enter space-separated numbers: 2 4 3 5
The sum of even numbers is: 6
The sum of odd numbers is: 8
```

3. Write a python program to count the Occurrences of a Specific Element in an Array.

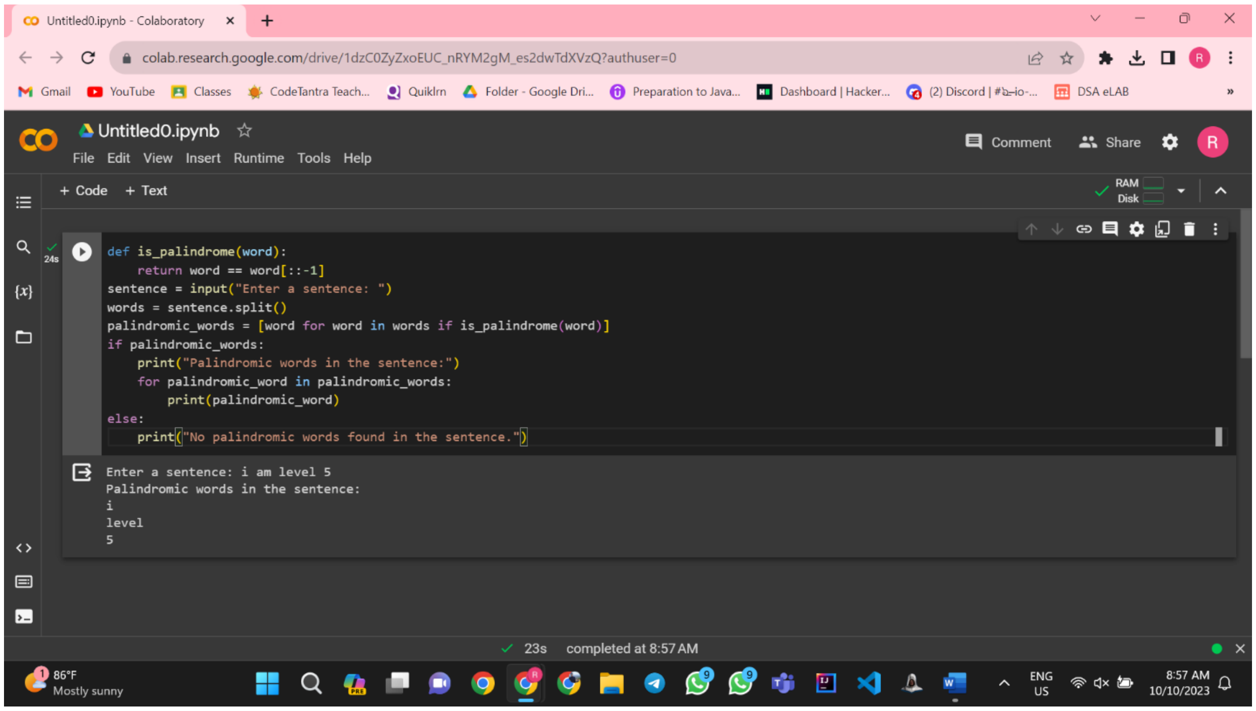


```
def count_occurrences(nums, target):
    count = 0
    for num in nums:
        if num == target:
            count += 1
    return count
nums = [int(x) for x in input("Enter space-separated numbers: ").split()]
target_element = int(input("Enter the element to count: "))
occurrences = count_occurrences(nums, target_element)
print(f"The number {target_element} occurs {occurrences} times in the array.")
```

```
Enter space-separated numbers: 2 3 4 5 1
Enter the element to count: 2
The number 2 occurs 1 times in the array.
```

Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1dzC0ZyZxoEUC_nRYM2gM_es2dwTdXVzQ?authuser=0
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
17s completed at 8:56 AM
86°F Mostly sunny
8:56 AM 10/10/2023

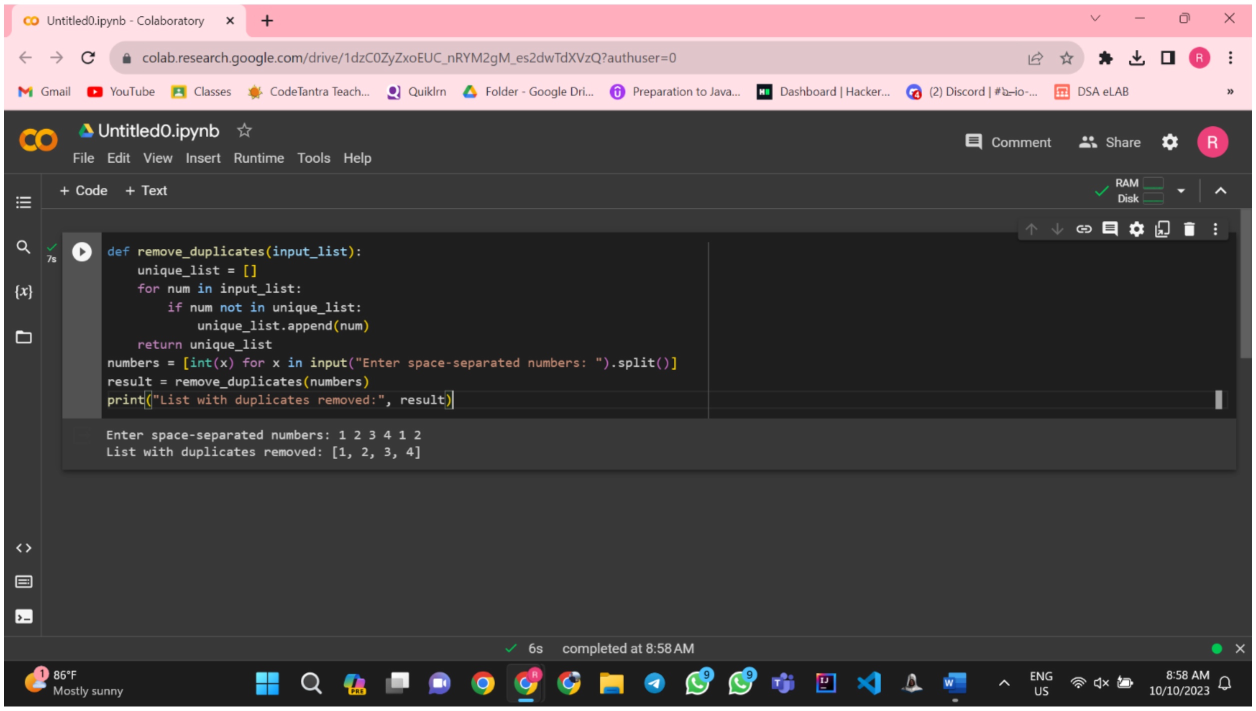
4. Write a Python program that takes a sentence as input and identifies and prints all the palindromic words in the sentence. Use an array to store the palindromic words.



```python
def is_palindrome(word):
    return word == word[::-1]
sentence = input("Enter a sentence: ")
words = sentence.split()
palindromic_words = [word for word in words if is_palindrome(word)]
if palindromic_words:
    print("Palindromic words in the sentence:")
    for palindromic_word in palindromic_words:
        print(palindromic_word)
else:
    print("No palindromic words found in the sentence.")
```

```
Enter a sentence: i am level 5
Palindromic words in the sentence:
i
level
5
```

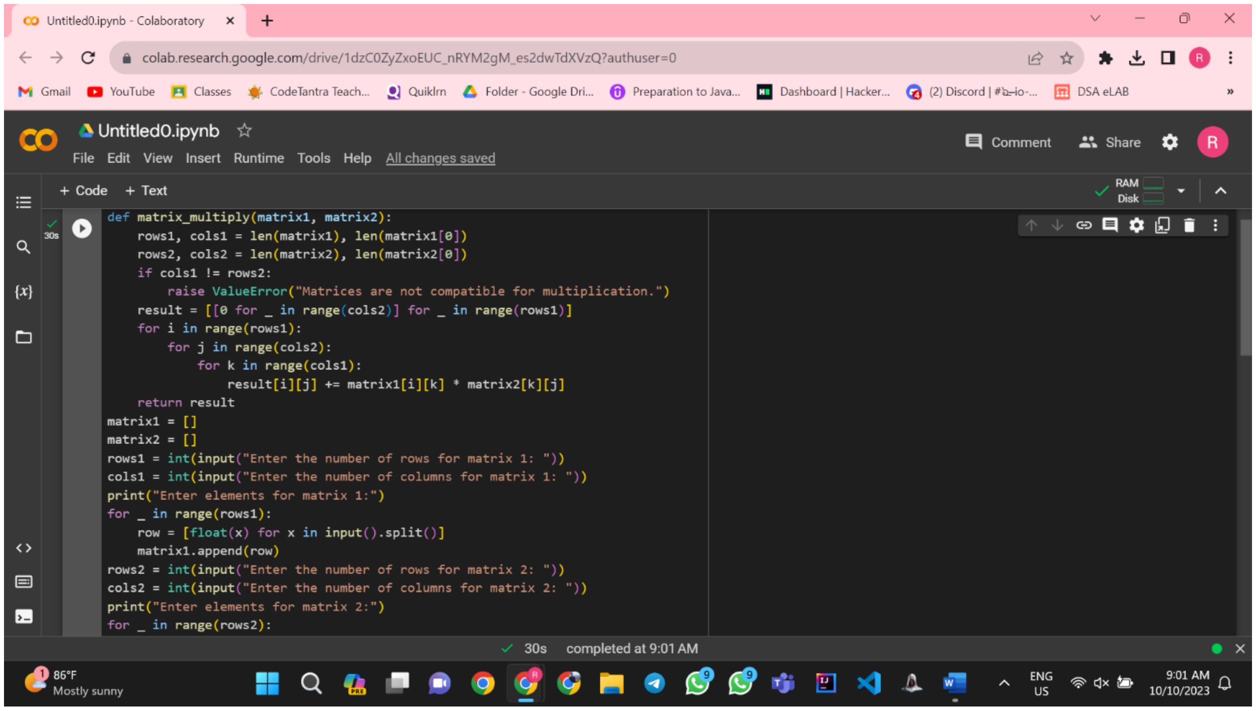
5. Write a Python program that takes a list of numbers and removes all duplicates from the list, preserving the original order of elements.



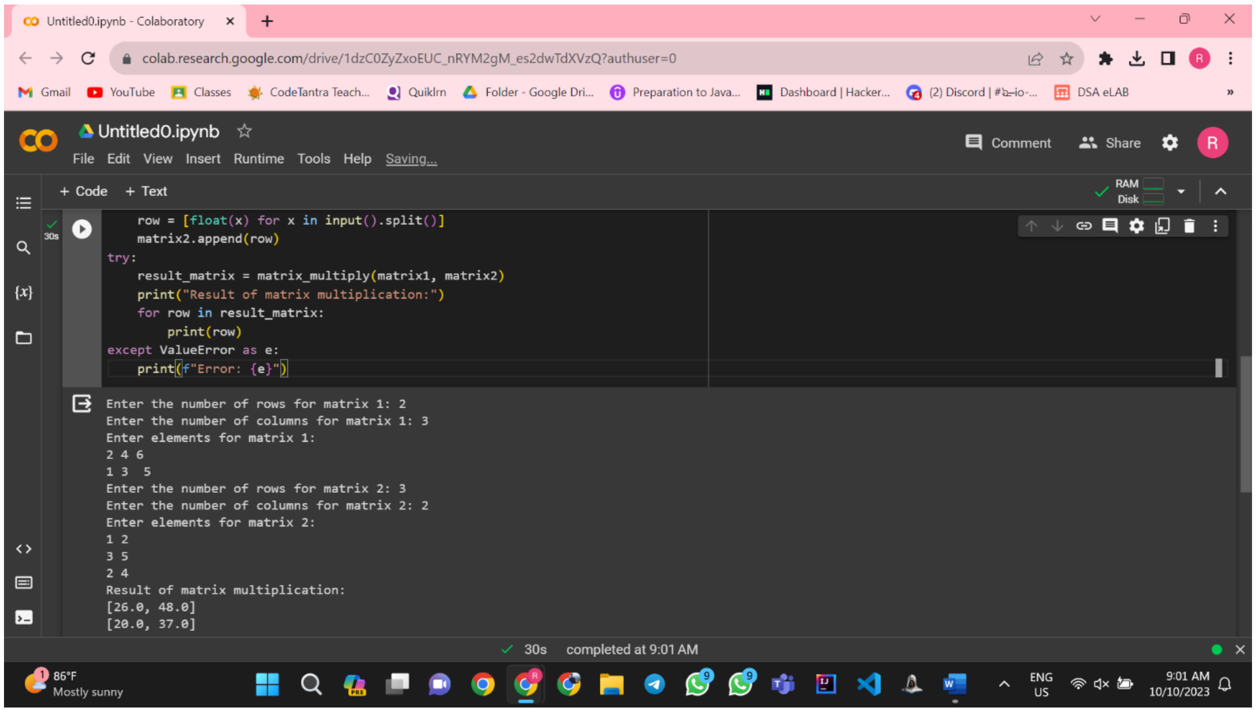
```python
def remove_duplicates(input_list):
    unique_list = []
    for num in input_list:
        if num not in unique_list:
            unique_list.append(num)
    return unique_list
numbers = [int(x) for x in input("Enter space-separated numbers: ").split()]
result = remove_duplicates(numbers)
print("List with duplicates removed:", result)
```

```
Enter space-separated numbers: 1 2 3 4 1 2
List with duplicates removed: [1, 2, 3, 4]
```

6. Write a Python program that performs matrix multiplication. Ask the user to input two matrices as lists of lists (2D arrays) and then multiply them if possible. Make sure to check if the matrices are compatible for multiplication and handle errors gracefully.



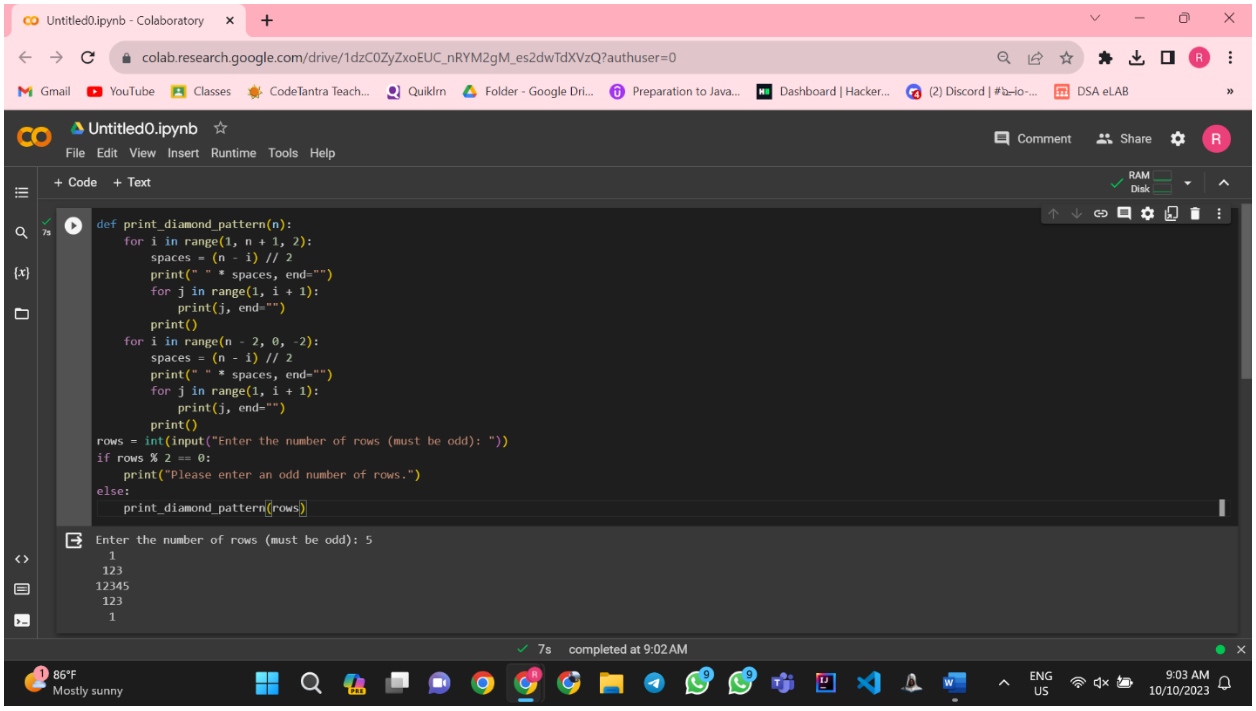
```python
def matrix_multiply(matrix1, matrix2):
    rows1, cols1 = len(matrix1), len(matrix1[0])
    rows2, cols2 = len(matrix2), len(matrix2[0])
    if cols1 != rows2:
        raise ValueError("Matrices are not compatible for multiplication.")
    result = [[0 for _ in range(cols2)] for _ in range(rows1)]
    for i in range(rows1):
        for j in range(cols2):
            for k in range(cols1):
                result[i][j] += matrix1[i][k] * matrix2[k][j]
    return result
matrix1 = []
matrix2 = []
rows1 = int(input("Enter the number of rows for matrix 1: "))
cols1 = int(input("Enter the number of columns for matrix 1: "))
print("Enter elements for matrix 1:")
for _ in range(rows1):
    row = [float(x) for x in input().split()]
    matrix1.append(row)
rows2 = int(input("Enter the number of rows for matrix 2: "))
cols2 = int(input("Enter the number of columns for matrix 2: "))
print("Enter elements for matrix 2:")
for _ in range(rows2):
```



```python
    row = [float(x) for x in input().split()]
    matrix2.append(row)
try:
    result_matrix = matrix_multiply(matrix1, matrix2)
    print("Result of matrix multiplication:")
    for row in result_matrix:
        print(row)
except ValueError as e:
    print(f"Error: {e}")
```

```
Enter the number of rows for matrix 1: 2
Enter the number of columns for matrix 1: 3
Enter elements for matrix 1:
2 4 6
1 3  5
Enter the number of rows for matrix 2: 3
Enter the number of columns for matrix 2: 2
Enter elements for matrix 2:
1 2
3 5
2 4
Result of matrix multiplication:
[26.0, 48.0]
[20.0, 37.0]
```

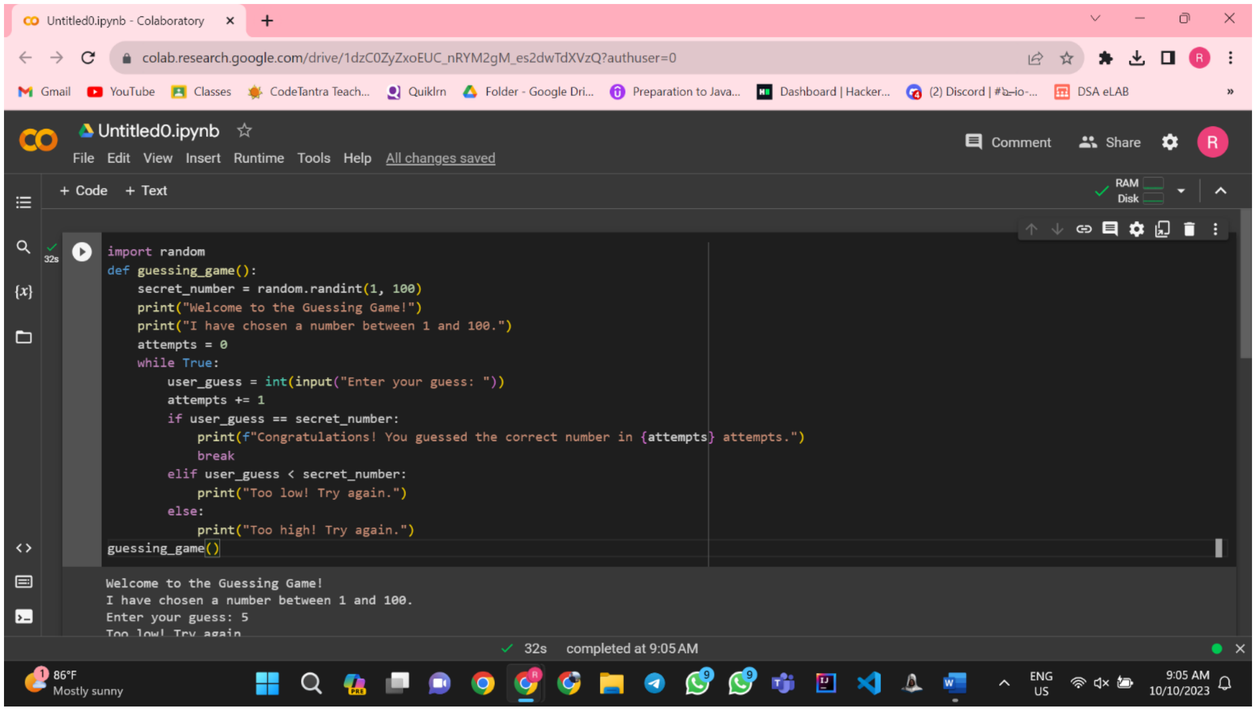
7. Write a python program to print diamond number pattern using Nested Loops.



```python
def print_diamond_pattern(n):
    for i in range(1, n + 1, 2):
        spaces = (n - i) // 2
        print(" " * spaces, end="")
        for j in range(1, i + 1):
            print(j, end="")
        print()
    for i in range(n - 2, 0, -2):
        spaces = (n - i) // 2
        print(" " * spaces, end="")
        for j in range(1, i + 1):
            print(j, end="")
        print()
rows = int(input("Enter the number of rows (must be odd): "))
if rows % 2 == 0:
    print("Please enter an odd number of rows.")
else:
    print_diamond_pattern(rows)
```

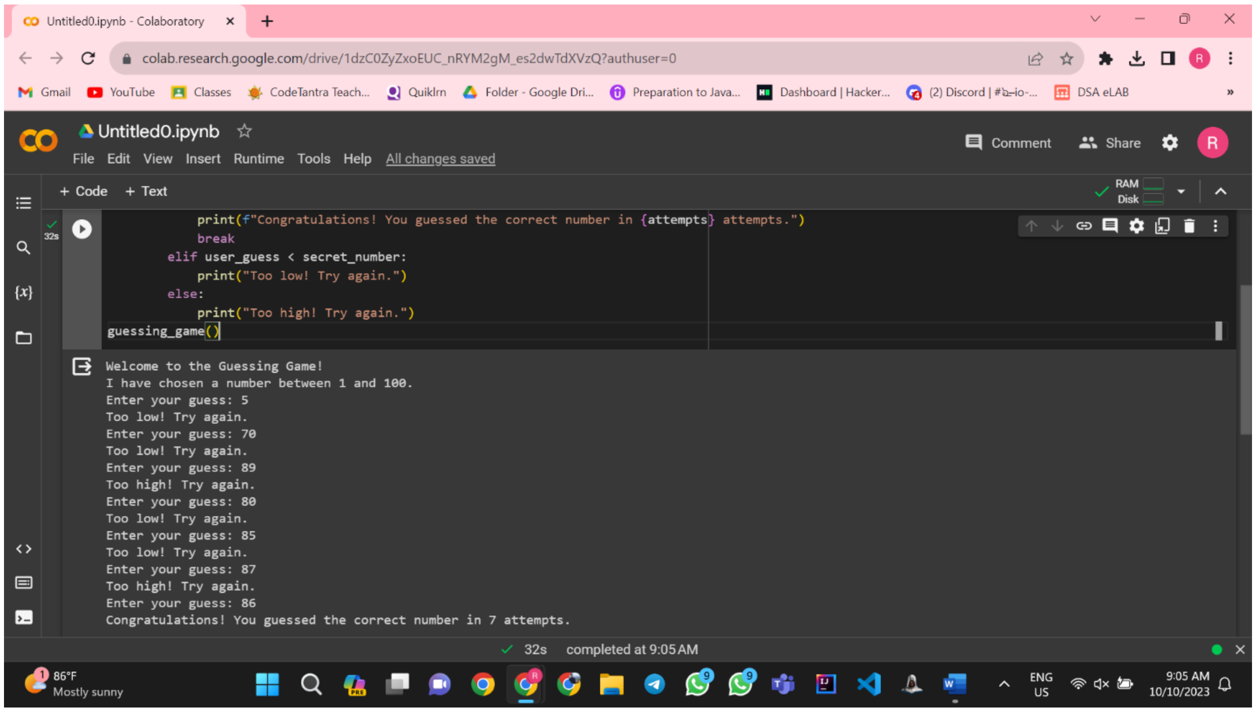
```
Enter the number of rows (must be odd): 5
  1
 123
12345
 123
  1
```

8. Write a Python program that simulates a simple guessing game. Generate a random number and have the user guess it. Provide hints like “too high” or “too low” until they guess correctly.



```python
import random
def guessing_game():
    secret_number = random.randint(1, 100)
    print("Welcome to the Guessing Game!")
    print("I have chosen a number between 1 and 100.")
    attempts = 0
    while True:
        user_guess = int(input("Enter your guess: "))
        attempts += 1
        if user_guess == secret_number:
            print(f"Congratulations! You guessed the correct number in {attempts} attempts.")
            break
        elif user_guess < secret_number:
            print("Too low! Try again.")
        else:
            print("Too high! Try again.")
guessing_game()
```

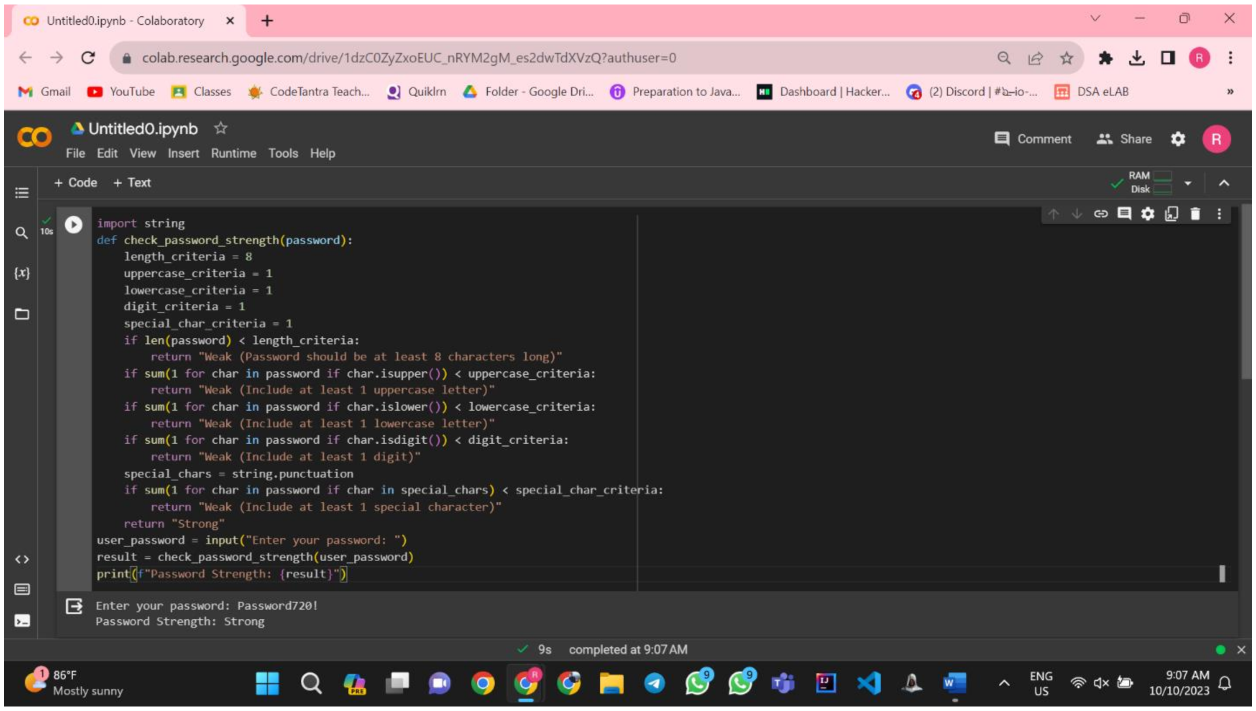
```
Welcome to the Guessing Game!
I have chosen a number between 1 and 100.
Enter your guess: 5
Too low! Try again
```



```python
            print(f"Congratulations! You guessed the correct number in {attempts} attempts.")
            break
        elif user_guess < secret_number:
            print("Too low! Try again.")
        else:
            print("Too high! Try again.")
guessing_game()
```

```
Welcome to the Guessing Game!
I have chosen a number between 1 and 100.
Enter your guess: 5
Too low! Try again.
Enter your guess: 70
Too low! Try again.
Enter your guess: 89
Too high! Try again.
Enter your guess: 80
Too low! Try again.
Enter your guess: 85
Too low! Try again.
Enter your guess: 87
Too high! Try again.
Enter your guess: 86
Congratulations! You guessed the correct number in 7 attempts.
```

9. Write a Python program that checks the strength of a password entered by a user. The program should assess the password based on criteria like length, use of uppercase and lowercase letters, digits, and special characters. Use control structures and arrays to provide a detailed evaluation.

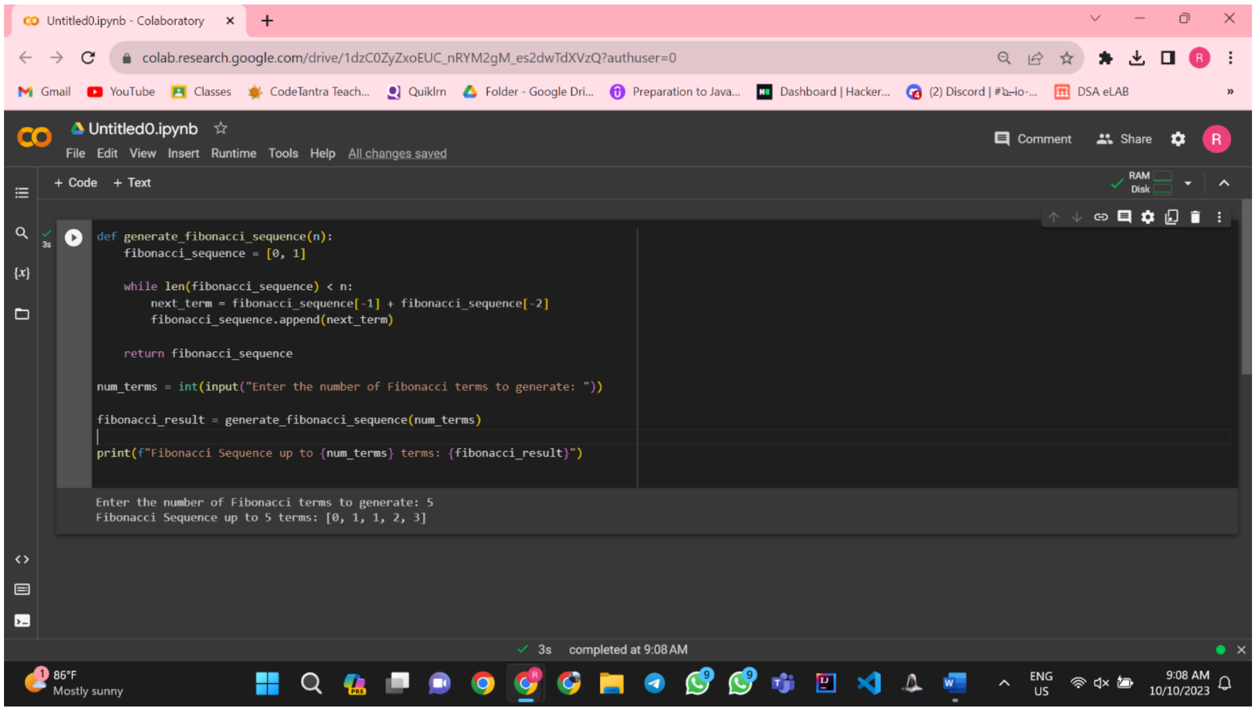


```python
import string
def check_password_strength(password):
    length_criteria = 8
    uppercase_criteria = 1
    lowercase_criteria = 1
    digit_criteria = 1
    special_char_criteria = 1
    if len(password) < length_criteria:
        return "Weak (Password should be at least 8 characters long)"
    if sum(1 for char in password if char.isupper()) < uppercase_criteria:
        return "Weak (Include at least 1 uppercase letter)"
    if sum(1 for char in password if char.islower()) < lowercase_criteria:
        return "Weak (Include at least 1 lowercase letter)"
    if sum(1 for char in password if char.isdigit()) < digit_criteria:
        return "Weak (Include at least 1 digit)"
    special_chars = string.punctuation
    if sum(1 for char in password if char in special_chars) < special_char_criteria:
        return "Weak (Include at least 1 special character)"
    return "Strong"
user_password = input("Enter your password: ")
result = check_password_strength(user_password)
print(f"Password Strength: {result}")
```

```
Enter your password: Password720!
Password Strength: Strong
```

Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1dzC0ZyZxoEUC_nRYM2gM_es2dwTdXVzQ?authuser=0
Untitled0.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
9s completed at 9:07 AM
9:07 AM 10/10/2023

10. Write a Python program that generates the Fibonacci sequence up to a specified number of terms using a loop and stores it in an array.



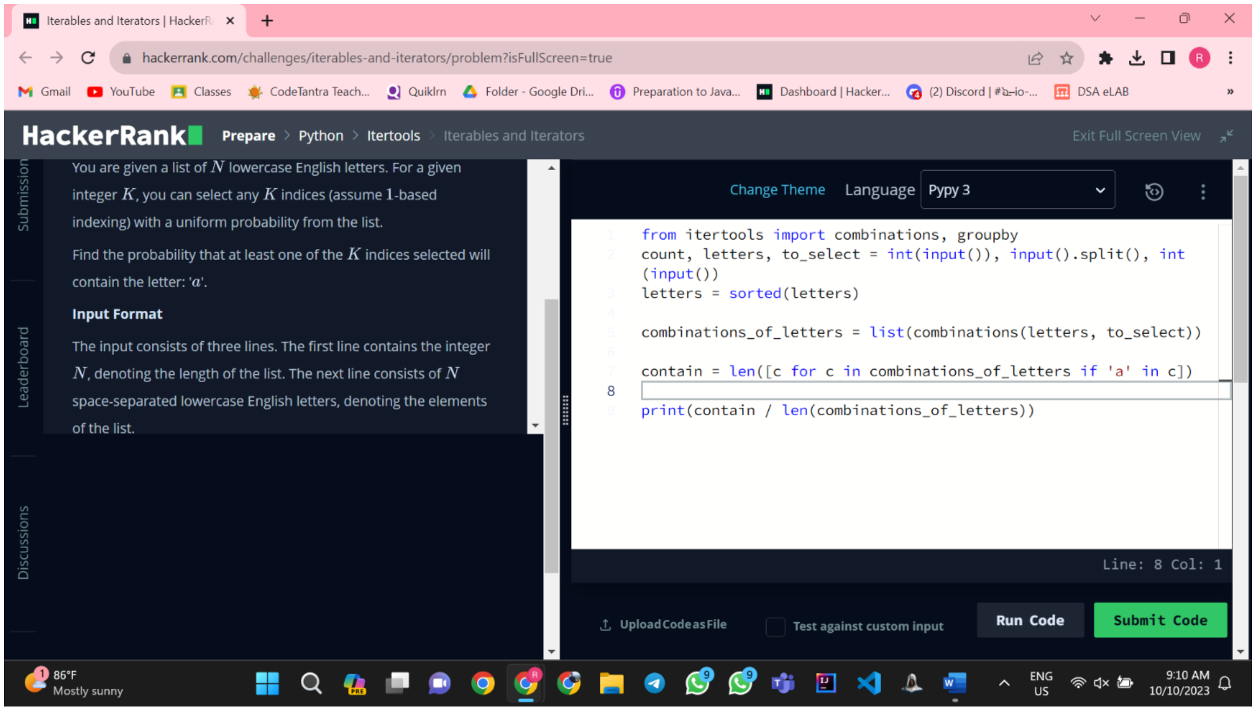
```
def generate_fibonacci_sequence(n):
    fibonacci_sequence = [0, 1]

    while len(fibonacci_sequence) < n:
        next_term = fibonacci_sequence[-1] + fibonacci_sequence[-2]
        fibonacci_sequence.append(next_term)

    return fibonacci_sequence

num_terms = int(input("Enter the number of Fibonacci terms to generate: "))

fibonacci_result = generate_fibonacci_sequence(num_terms)

print(f"Fibonacci Sequence up to {num_terms} terms: {fibonacci_result}")
```

```
Enter the number of Fibonacci terms to generate: 5
Fibonacci Sequence up to 5 terms: [0, 1, 1, 2, 3]
```

Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1dzC0ZyZxoEUC_nRYM2gM_es2dwTdXVzQ?authuser=0
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
3s completed at 9:08 AM
86°F Mostly sunny
9:08 AM 10/10/2023

HACKERRANK

https://www.hackerrank.com/challenges/iterables-and-iterators/problem?isFullScreen=true



Iterables and Iterators | HackerRank

hackerrank.com/challenges/iterables-and-iterators/problem?isFullScreen=true

HackerRank Prepare > Python > Itertools > Iterables and Iterators

You are given a list of N lowercase English letters. For a given integer K, you can select any K indices (assume 1-based indexing) with a uniform probability from the list.

Find the probability that at least one of the K indices selected will contain the letter: 'a'.

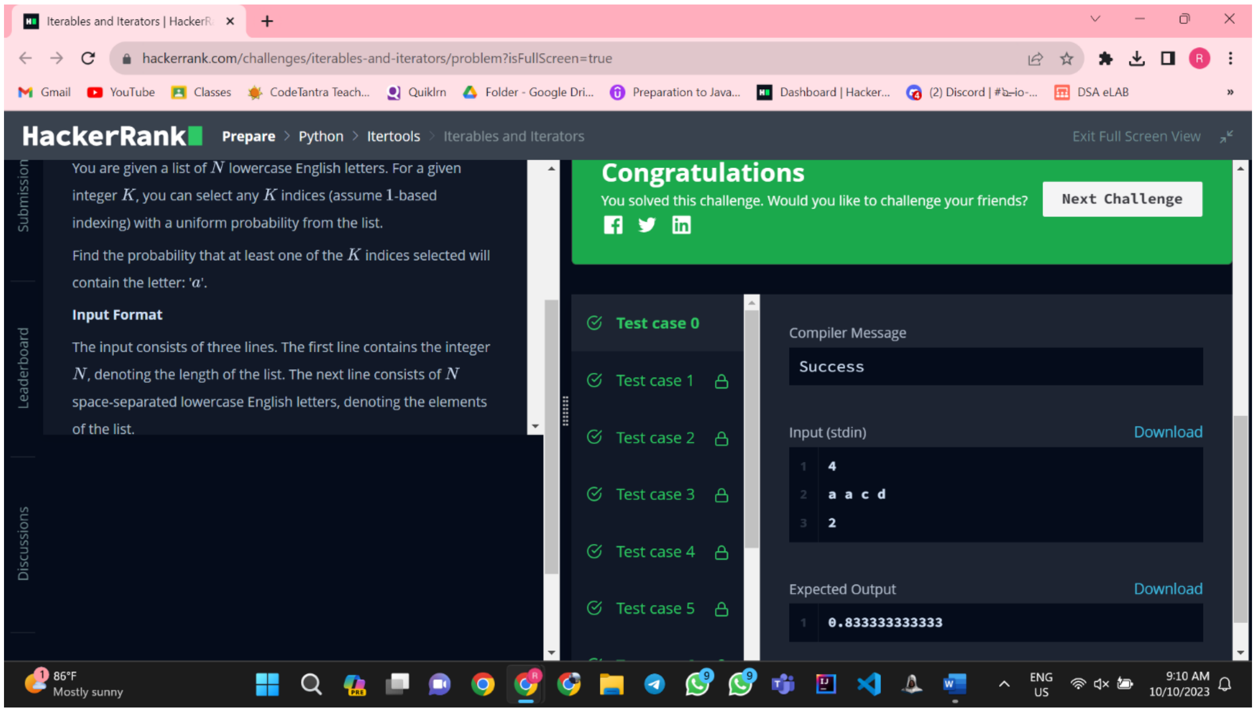
Input Format

The input consists of three lines. The first line contains the integer N, denoting the length of the list. The next line consists of N space-separated lowercase English letters, denoting the elements of the list.

Language: Pypy 3

```python
from itertools import combinations, groupby
count, letters, to_select = int(input()), input().split(), int(input())
letters = sorted(letters)

combinations_of_letters = list(combinations(letters, to_select))

contain = len([c for c in combinations_of_letters if 'a' in c])

print(contain / len(combinations_of_letters))
```

Line: 8 Col: 1



Iterables and Iterators | HackerRank

HackerRank Prepare > Python > Itertools > Iterables and Iterators

Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

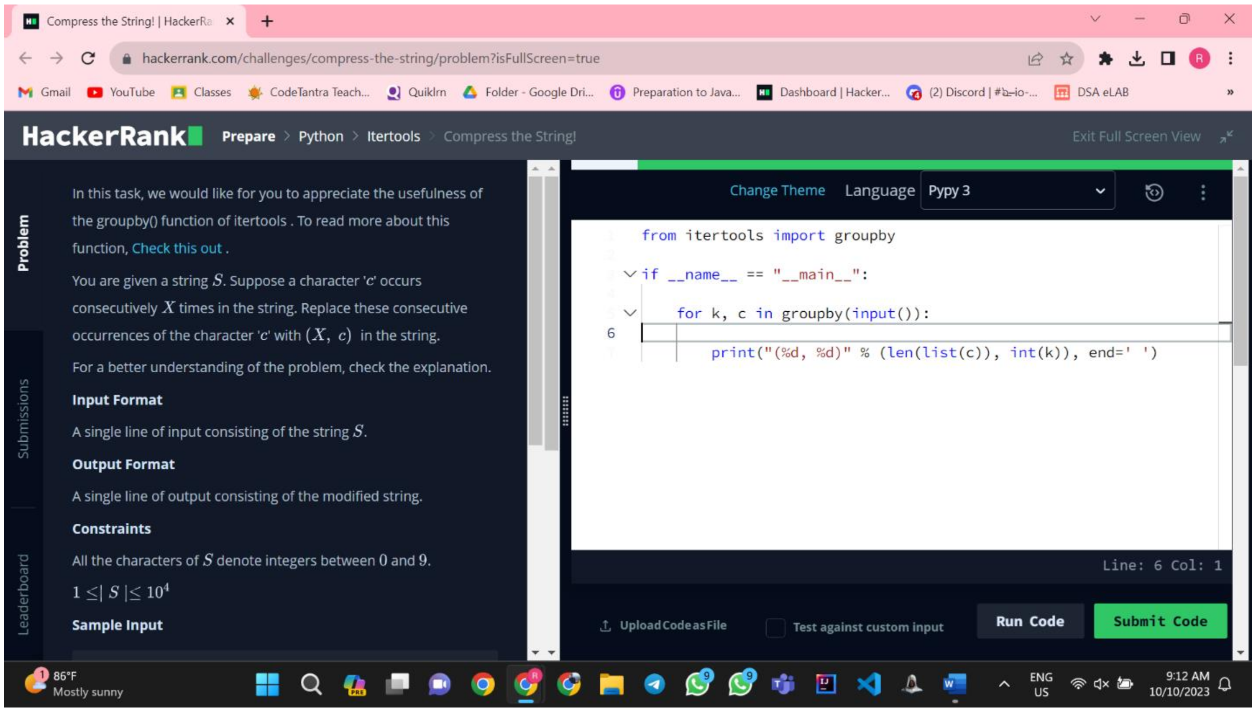
Input (stdin)

```
4
a a c d
2
```

Expected Output

```
0.833333333333
```

https://www.hackerrank.com/challenges/compress-the-string/problem?isFullScreen=true



In this task, we would like for you to appreciate the usefulness of the groupby() function of itertools . To read more about this function, Check this out .

You are given a string S. Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with (X, c) in the string.

For a better understanding of the problem, check the explanation.

Input Format

A single line of input consisting of the string S.

Output Format

A single line of output consisting of the modified string.

Constraints

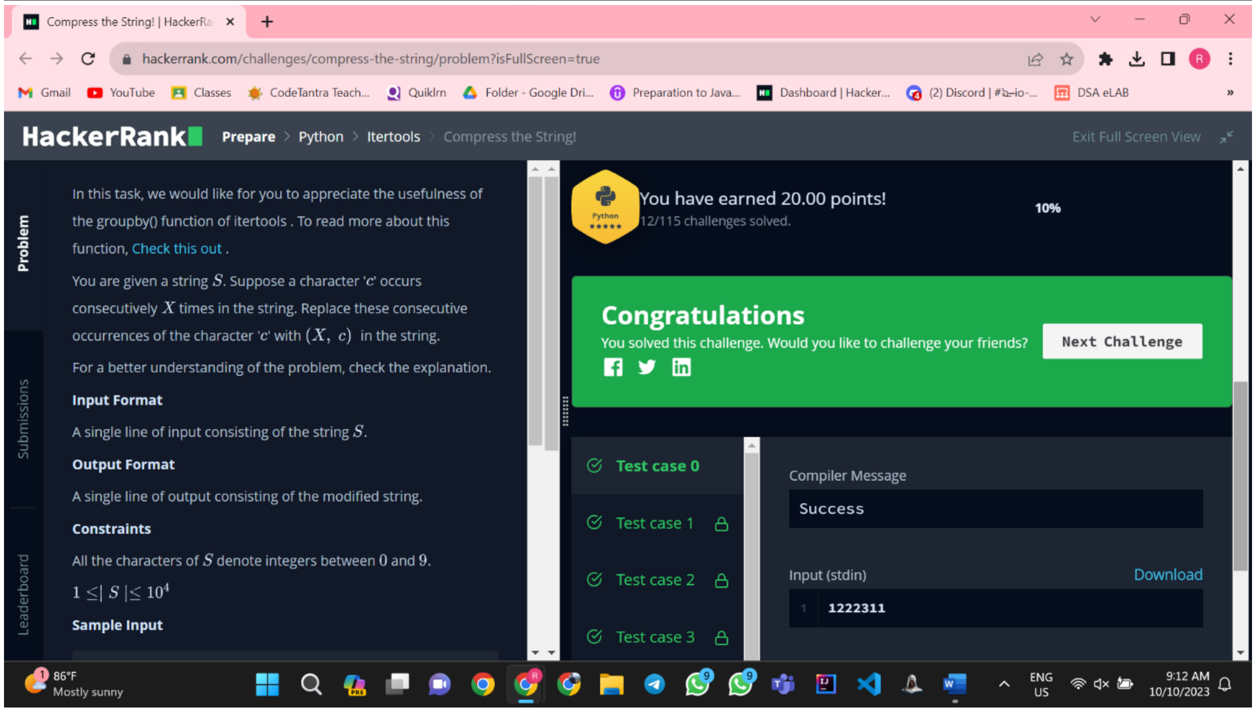
All the characters of S denote integers between 0 and 9.

$1 \leq |S| \leq 10^4$

Sample Input

```
from itertools import groupby

if __name__ == "__main__":

    for k, c in groupby(input()):

        print("(%d, %d)" % (len(list(c)), int(k)), end=' ')
```

Line: 6 Col: 1


Compress the String! | HackerRa
hackerrank.com/challenges/compress-the-string/problem?isFullScreen=true
HackerRank
Prepare > Python > Itertools > Compress the String!
Exit Full Screen View
Problem
Submissions
Leaderboard
Python
Next Challenge
86°F
Mostly sunny
ENG US
9:12 AM 10/10/2023

In this task, we would like for you to appreciate the usefulness of the groupby() function of itertools . To read more about this function, Check this out .

You are given a string S. Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with (X, c) in the string.

For a better understanding of the problem, check the explanation.

Input Format

A single line of input consisting of the string S.

Output Format

A single line of output consisting of the modified string.

Constraints

All the characters of S denote integers between 0 and 9.

$1 \leq |S| \leq 10^4$

Sample Input

You have earned 20.00 points!
12/115 challenges solved.
10%

Congratulations
You solved this challenge. Would you like to challenge your friends?

Test case 0
Test case 1
Test case 2
Test case 3

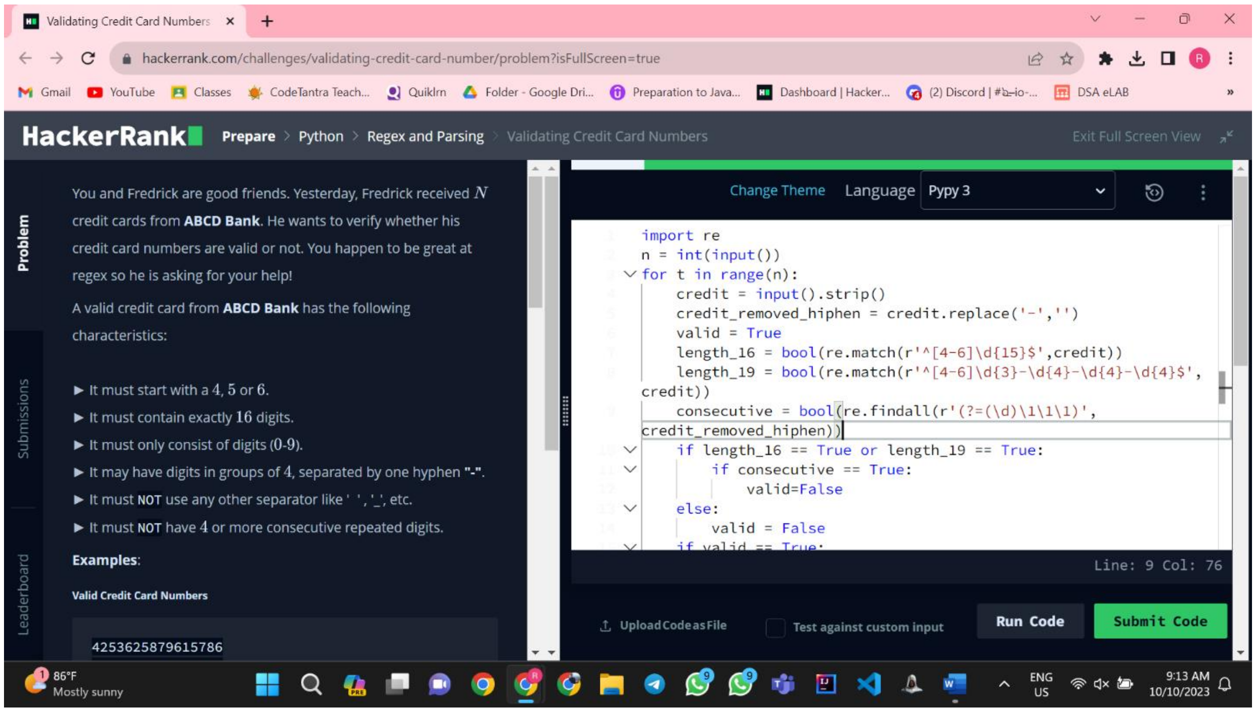
Compiler Message

```
Success
```

Input (stdin) Download

```
1222311
```

https://www.hackerrank.com/challenges/validating-credit-card-number/problem?isFullScreen=true



You and Fredrick are good friends. Yesterday, Fredrick received N credit cards from **ABCD Bank**. He wants to verify whether his credit card numbers are valid or not. You happen to be great at regex so he is asking for your help!

A valid credit card from **ABCD Bank** has the following characteristics:

- ► It must start with a 4, 5 or 6.
- ► It must contain exactly 16 digits.
- ► It must only consist of digits (0-9).
- ► It may have digits in groups of 4, separated by one hyphen "-".
- ► It must NOT use any other separator like ' ', '_', etc.
- ► It must NOT have 4 or more consecutive repeated digits.

Examples:

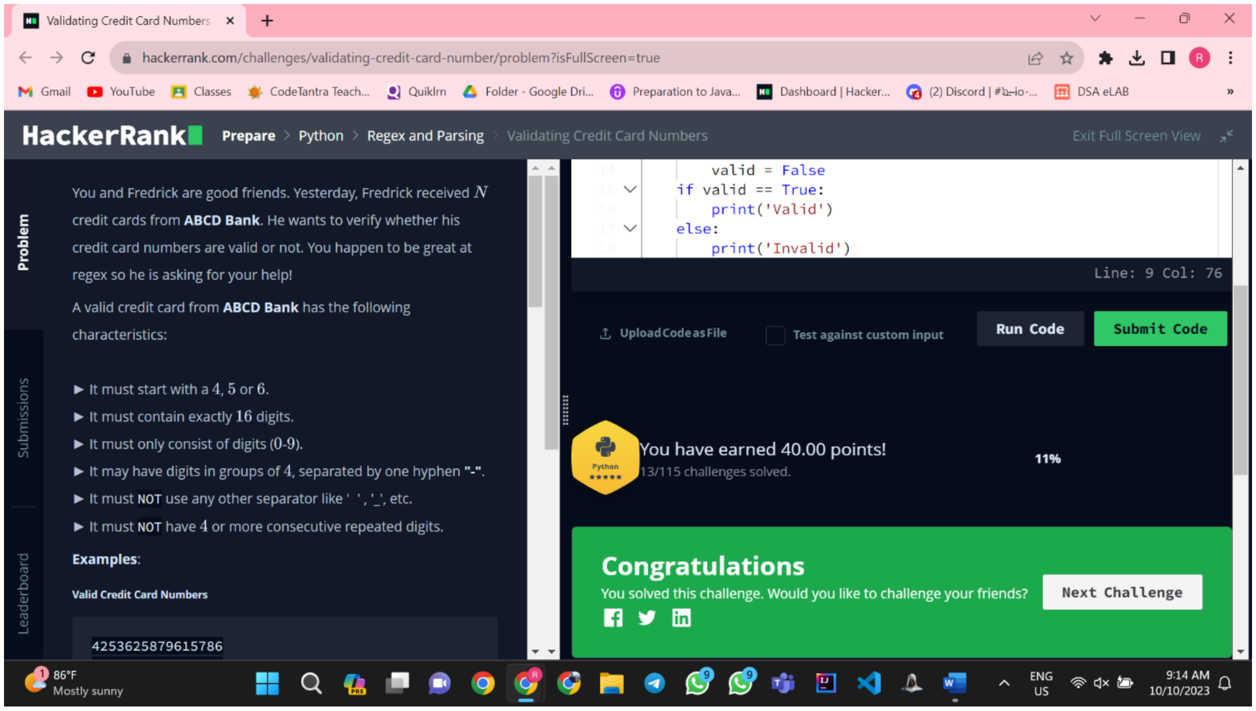
Valid Credit Card Numbers

```
4253625879615786
```

```
import re
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
```

Line: 9 Col: 76

Validating Credit Card Numbers
hackerrank.com/challenges/validating-credit-card-number/problem?isFullScreen=true
HackerRank
Prepare > Python > Regex and Parsing > Validating Credit Card Numbers
Exit Full Screen View
Problem
Submissions
Leaderboard
Change Theme
Language Pypy 3
Upload Code as File
Test against custom input
Run Code
Submit Code
86°F Mostly sunny
9:13 AM 10/10/2023



```
        valid = False
    if valid == True:
        print('Valid')
    else:
        print('Invalid')
```

Line: 9 Col: 76

You have earned 40.00 points!
13/115 challenges solved.
11%

Congratulations
You solved this challenge. Would you like to challenge your friends?

Validating Credit Card Numbers
hackerrank.com/challenges/validating-credit-card-number/problem?isFullScreen=true
HackerRank
Prepare > Python > Regex and Parsing > Validating Credit Card Numbers
Exit Full Screen View
Run Code
Submit Code
Next Challenge
86°F Mostly sunny
9:14 AM 10/10/2023