

## AI-ASSISTED CODING ASSIGNMENT 2

SHRUTHI ADDAGUDI | 2303A52133 | BATCH 37

The screenshot shows a code editor interface with an AI-assisted coding feature. On the left, there is a code editor window with the following Python code:

```
def remove_negative_values(integer_list):
    """
    Removes all negative values from a list of integers.

    Args:
        integer_list (list): A list of integers.

    Returns:
        list: A new list containing only the non-negative integers.
    """
    return [num for num in integer_list if num >= 0]

# Example usage:
my_list = [1, -2, 3, 0, -5, 6, -1]
print("Original list:", my_list)

cleaned_list = remove_negative_values(my_list)
print("List after removing negative values:", cleaned_list)
```

On the right, there is a Gemini AI panel. The user has asked Gemini to "write a python function that removes all negative values from a list". Gemini has generated the same code as shown in the code editor. It also provides some context and execution details.

### Task 1

**Prompt Used :** Write a Python function that removes all negative values from a list of integers.

**Output :** Before : [1, -2, 3, 0, -5, 6, -1]

After : [1, 3, 0, 6]

**Understanding:** Gemini generated a clean and efficient function using lists. The logic was easy to understand and it required no need for manual correction.

The screenshot shows a code editor interface with an AI-assisted coding feature. On the left, there is a code editor window with the following Python code:

```
def count_characters(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in input_string:
        if char.isalpha(): # Check if the character is an alphabet
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1
        elif char.isdigit(): # Check if the character is a digit
            digit_count += 1

    return vowel_count, consonant_count, digit_count

# Sample Input
sample_string = "Hello World 123!"

# Get the counts
v, c, d = count_characters(sample_string)

# Display the output
print(f"Sample Input: '{sample_string}'")
print(f"Number of Vowels: {v}")
print(f"Number of Consonants: {c}")
```

On the right, there is a Gemini AI panel. The user has asked Gemini to "create a python function that counts the number of vowels, consonants and didits in a string provide sample input and output". Gemini has generated the same code as shown in the code editor. It also provides some context and execution details.



**Gemini**

5 create a python function that counts the number of vowels,consonants and digits in a string provide sample input and output

◆ Certainly! Here's a Python function that counts the number of vowels, consonants, and digits in a given string, along with a sample input and its output.

The code executed successfully! For the sample input 'Hello World 123!', the function correctly identified:

- Number of Vowels: 3 (e, o, o)
- Number of Consonants: 7 (H, l, l, W, r, l, d)
- Number of Digits: 3 (1, 2, 3)

## Task 2

**Prompt Used :** Create a Python function that counts the number of vowels, consonants, and digits in a string.

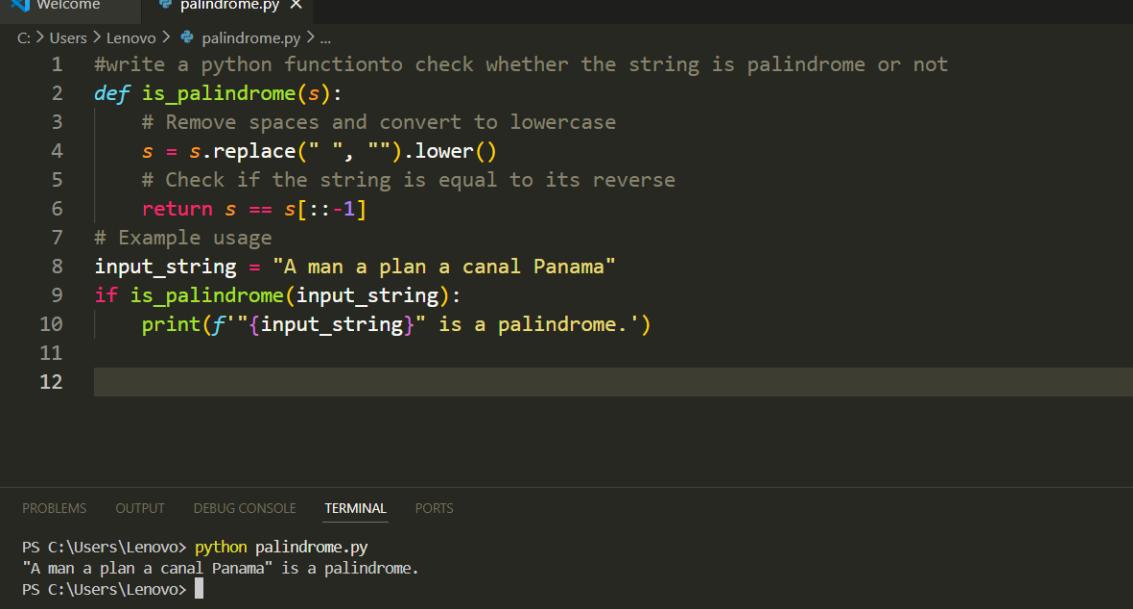
**Sample Input:** 'Hello World 123!'

Number of Vowels: 3

Number of Consonants: 7

Number of Digits: 3

**Student Comment:** Gemini clearly handled character classification using built-in Python methods like `isalpha()` and `isdigit()` metho



```
C:\> Users > Lenovo > palindrome.py > ...
1 #write a python functio to check whether the string is palindrome or not
2 def is_palindrome(s):
3     # Remove spaces and convert to lowercase
4     s = s.replace(" ", "").lower()
5     # Check if the string is equal to its reverse
6     return s == s[::-1]
7 # Example usage
8 input_string = "A man a plan a canal Panama"
9 if is_palindrome(input_string):
10     print(f'{input_string} is a palindrome.')
11
12
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Lenovo> python palindrome.py  
 "A man a plan a canal Panama" is a palindrome.  
 PS C:\Users\Lenovo>

The screenshot shows a Gemini AI interface. On the left, a code editor window displays Python code for checking if a string is a palindrome. The code includes importing re, defining a function is\_palindrome, cleaning the text by removing non-alphanumeric characters and converting it to lowercase, and comparing the cleaned text with its reverse. A sample usage section prints several examples. On the right, the Gemini interface shows a prompt about writing a Python function to check if a string is a palindrome. It provides a sample response with a working Python function and some test cases. Below the response are several interaction buttons like 'Test the palindrome function with a custom string' and 'What can I help you build?'. The overall interface is dark-themed.

```

Number of Digits: 3

[4] import re
def is_palindrome(text):
    # Remove non-alphanumeric characters and convert to lowercase
    cleaned_text = re.sub(r'[^\w\W]', '', text).lower()
    # Compare the cleaned string with its reverse
    return cleaned_text == cleaned_text[::-1]

# Sample usage:
print('"'Madam"' is a palindrome: {is_palindrome('Madam')}')
print('"'A man, a plan, a canal: Panama"' is a palindrome: {is_palindrome('A man, a plan, a canal: Panama')})
print('"'Hello"' is a palindrome: {is_palindrome('Hello')}')
print('"'Racecar"' is a palindrome: {is_palindrome('Racecar')}')
print('"'No lemon, no melon"' is a palindrome: {is_palindrome('No lemon, no melon')})"
...
  • 'Madam' is a palindrome: True
  • 'A man, a plan, a canal: Panama' is a palindrome: True
  • 'Hello' is a palindrome: False
  • 'Racecar' is a palindrome: True
  • 'No lemon, no melon' is a palindrome: True

```

**Gemini**

**S** write a python function to check whether the string is palindrom or not

◆ Certainly! Here's a Python function to check if a string is a palindrome, along with an example.

The code executed successfully! The `is_palindrome` function correctly identified:

- 'Madam' as a palindrome ( True )
- 'A man, a plan, a canal: Panama' as a palindrome ( True )
- 'Hello' as not a palindrome ( False )
- 'Racecar' as a palindrome ( True )
- 'No lemon, no melon' as a palindrome ( True )

[Test the palindrome function with a custom string](#)

[Explain how the is\\_palindrome function works](#)

[Write a function to reverse a string](#)

[What can I help you build?](#)

### TASK 3:

**Palindrome Check – Tool Comparison Prompt Used:** Write a Python function to check whether a string is a palindrome.

Tool	Code quality	readability
Gemini	Clear and beginner friendly	Explains logic
copilot	Short and optimised	Assumes prior python knowledge

The screenshot shows a Gemini AI interface. The user has asked to explain a Python palindrome checker function line by line. Gemini provides a breakdown of the code, explaining the purpose of each line. It defines the function, converts the input to lowercase, and compares it with its reverse. Below the main text are several interaction buttons like 'Explain the `re.sub` part of the `is\_palindrome` function', 'How can I make the `remove\_negative\_values` function more efficient', and 'What other string manipulation functions can I explore'. At the bottom, there's a text input field for the user to ask more questions and a footer note about Gemini's potential mistakes.

**Gemini**

**S** Explain the following Python palindrome checking function line by line: `def is_palindrome(s): s = s.lower() return s == s[::-1]`

◆ Certainly! Let's break down this Python palindrome checking function line by line:

1. `def is_palindrome(s):`
  - This line defines a function named `is_palindrome` that accepts one argument, `s`. The `s` here is expected to be a string that the function will check.
2. `s = s.lower()`
  - This line converts the input string `s` to lowercase. This is a crucial step because palindromes should ignore case differences (e.g., 'Madam' should be treated the same as 'madam'). By converting everything to lowercase, you ensure that 'Madam' and 'madam' both become 'madam' for the comparison.

[Explain the `re.sub` part of the `is\\_palindrome` function](#)

[How can I make the `remove\\_negative\\_values` function more efficient](#)

[What other string manipulation functions can I explore](#)

What can I help you build?

Gemini 2.5 Flash ▶

Gemini can make mistakes, so double-check it and use code with caution. [Learn more](#)

Gemini



### 3. `return s == s[::-1]`

- o This is the core of the palindrome check. It compares the modified string `s` with its reverse.
- o `s[::-1]` is a Python slice notation that creates a reversed copy of the string `s`. The `::` means 'step', and `-1` means to step backwards, effectively reversing the string.
- o The `==` operator then compares the original (lowercase) string with its reversed counterpart.
- o If they are identical, the expression `s == s[::-1]` evaluates to `True`, indicating that the string is a palindrome.
- o If they are not identical, it evaluates to `False`, meaning the string is not a palindrome.

In summary, this function efficiently checks for palindromes by normalizing the input string to

[Explain the 're.sub' part of the 'is\\_palindrome' function](#)

[How can I make the 'remove\\_negative\\_values' function more efficient](#)

[What other string manipulation functions can I explore](#)

## TASK 4

**Code Explanation Using Gemini Prompt Used:** Explain the following Python palindrome checking function line by line

`is_palindrome(s): s = s.lower() return s == s[::-1]` The explanation helped me understand slicing operations and how reversing of a string works. Gemini explanation was clear and it was easy to follow

## Conclusion

**In this lab, I explored AI-assisted coding using Google Gemini in Google Colab. Gemini proved to be effective in generating beginner-friendly code accompanied by clear and detailed explanations. GitHub Copilot was useful in producing more optimized and concise solutions, while Cursor AI assisted in refining and refactoring the code for better structure and readability. Overall, this lab enhanced my understanding of how different AI tools support programmers at various skill levels and stages of the development process.**

