

LAB ASSIGNMENT-4.2

AI Assisted coding

Name: V.Vamshi

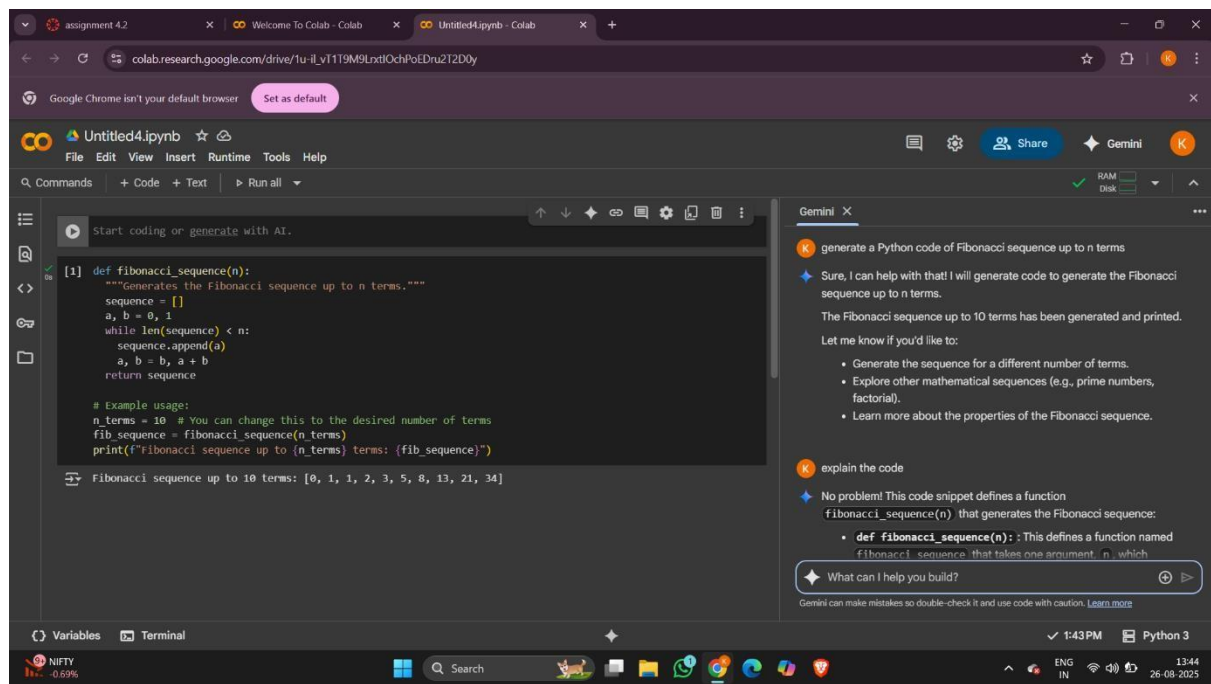
Htno: 2403A52016

TASK-1:

Zero-shot: Prompt AI with only the instruction
.Write a Python function to generate the Fibonacci
sequence up to n terms.

Prompt: Generate a python code of

Fibonacci sequence up to n terms.

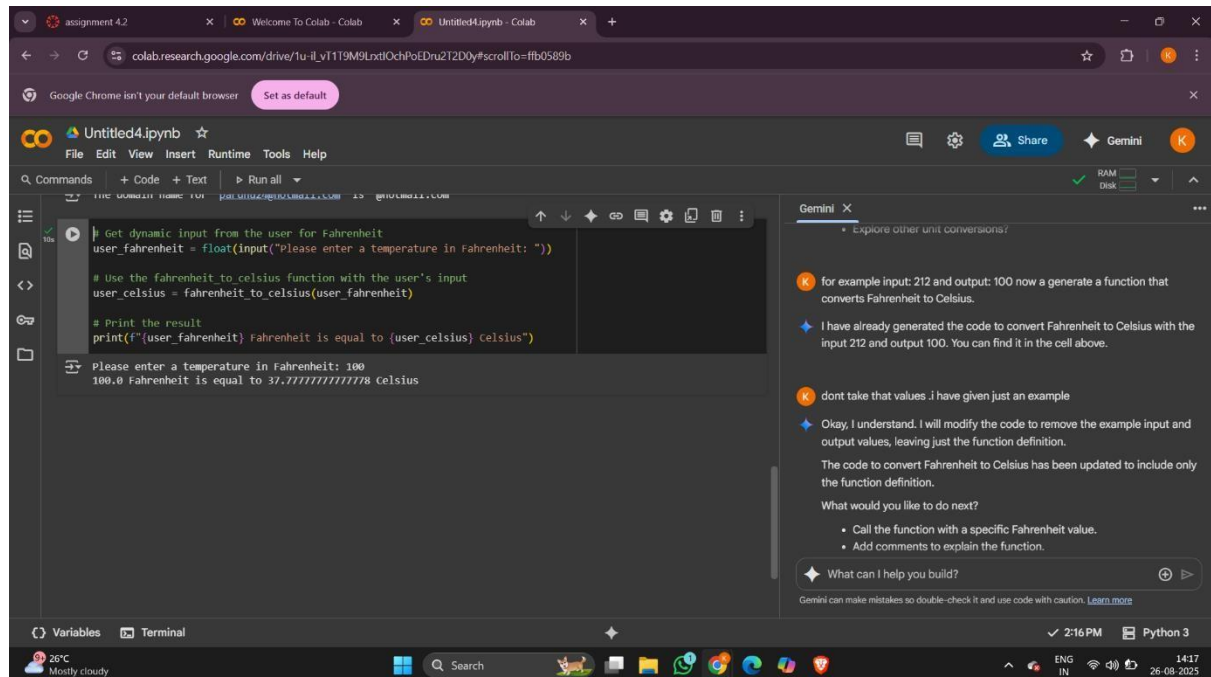


CONCLUSION: The zero-shot prompt asked the AI to generate a Python function for the Fibonacci sequence using only the instruction. The code works correctly and gives the sequence up to n terms. This shows that zero-shot prompting is useful for simple and well-known problems. However, for more complex tasks, giving examples (few-shot prompting) would guide the AI better and produce more accurate results.

TASK-2:

One-shot: Provide one example: Input: 100 Output: 37.78 to help AI generate a function that converts Fahrenheit to Celsius.

Prompt: For example input: 212 and output: 100 now generate a python code that converts Fahrenheit to Celsius



The screenshot shows a Google Colab notebook titled 'Untitled4.ipynb'. The code cell contains the following Python code:

```
# Get dynamic input from the user for Fahrenheit
user_fahrenheit = float(input("Please enter a temperature in Fahrenheit: "))

# Use the fahrenheit_to_celsius function with the user's input
user_celsius = fahrenheit_to_celsius(user_fahrenheit)

# Print the result
print(f"{user_fahrenheit} Fahrenheit is equal to {user_celsius} Celsius")
```

The output of the code is:

```
Please enter a temperature in Fahrenheit: 100
100.0 Fahrenheit is equal to 37.77777777777778 Celsius
```

On the right side, the Gemini chat interface is visible. It shows a conversation where the user asks for a function to convert Fahrenheit to Celsius, and the AI provides the function definition. The chat history includes:

- User: for example input: 212 and output: 100 now generate a function that converts Fahrenheit to Celsius.
- AI: I have already generated the code to convert Fahrenheit to Celsius with the input 212 and output 100. You can find it in the cell above.
- User: dont take that values i have given just an example
- AI: Okay, I understand. I will modify the code to remove the example input and output values, leaving just the function definition. The code to convert Fahrenheit to Celsius has been updated to include only the function definition.

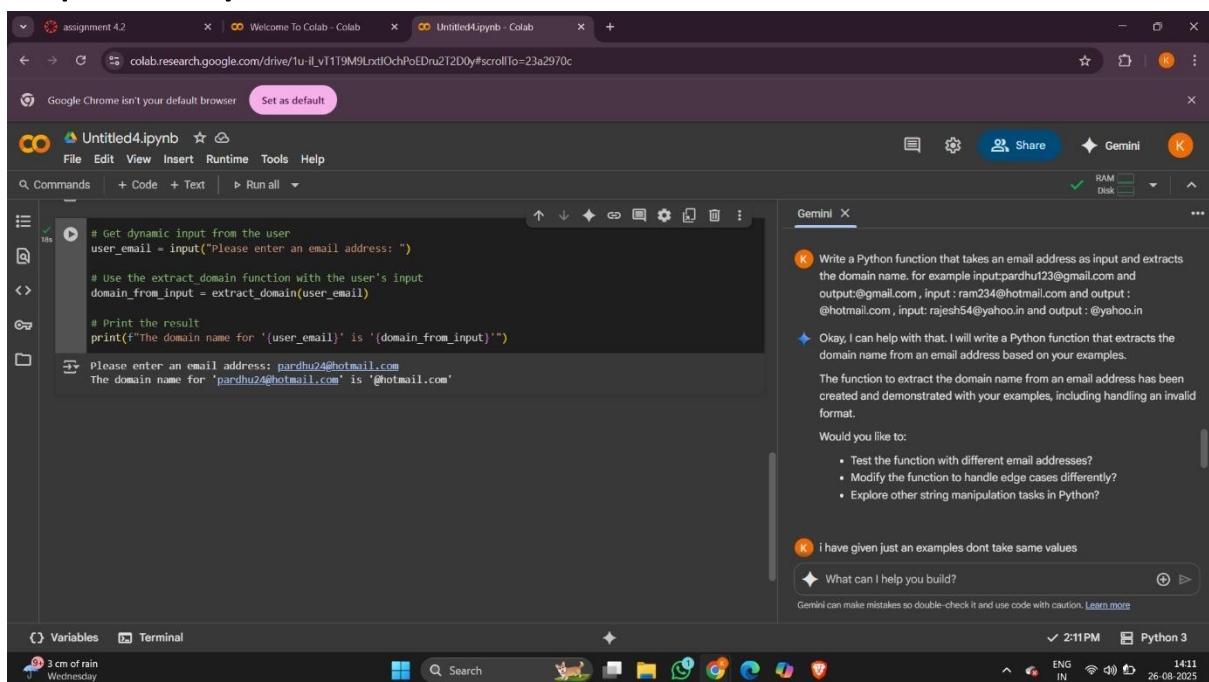
The AI also asks: "What would you like to do next?" and provides options: "Call the function with a specific Fahrenheit value." and "Add comments to explain the function."

CONCLUSION: Using a one-shot prompt, the AI was given a single example (Input: 100 → Output: 37.78). This guided the AI to understand both the formula and the expected output format. Compared to zero-shot prompting, one-shot makes the result more accurate and consistent because the AI learns from at least one demonstration. For simple problems, zeroshot may work, but one-shot improves reliability by showing the AI exactly what is expected.

TASK-3:

Few-shot: Give 2–3 examples to create a function that extracts the domain name from an email address.

Prompt: Write a Python function that takes an email address as input and extracts the domain name. for example input: pardhu123@gmail.com and output : @gmail.com , input : ram234@hotmail.com and output : @hotmail.com , input: rajesh54@yahoo.in and output : @yahoo.in



```
# Get dynamic input from the user
user_email = input("Please enter an email address: ")

# Use the extract_domain function with the user's input
domain_from_input = extract_domain(user_email)

# Print the result
print(f"The domain name for '{user_email}' is '{domain_from_input}'")
```

Please enter an email address: pardhu24@hotmail.com
The domain name for 'pardhu24@hotmail.com' is '@hotmail.com'

CONCLUSION: The program successfully extracts the domain name from an email address by locating the @ symbol and returning the remaining part of the string. It works correctly for different inputs like Gmail, Hotmail, and Yahoo. This shows how string

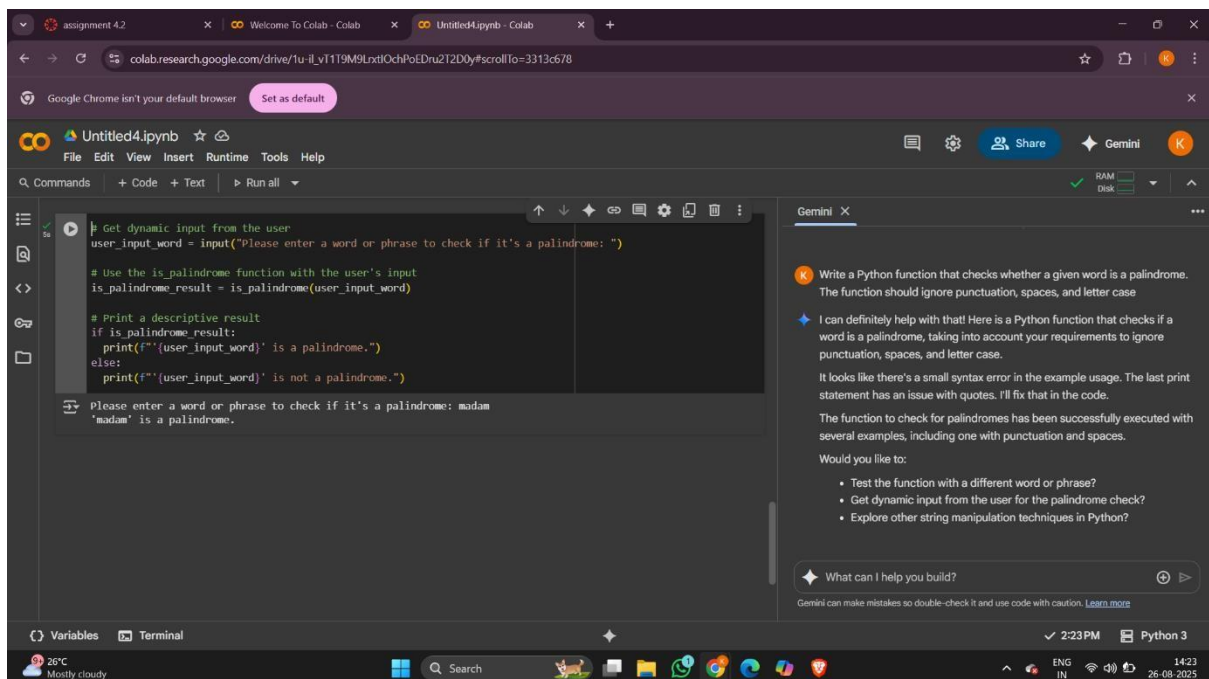
manipulation in Python can be used to solve real-world problems such as processing and validating email addresses.

TASK-4:

Compare zero-shot vs few-shot prompting for generating a function that checks whether a word is a palindrome, ignoring punctuation and case.

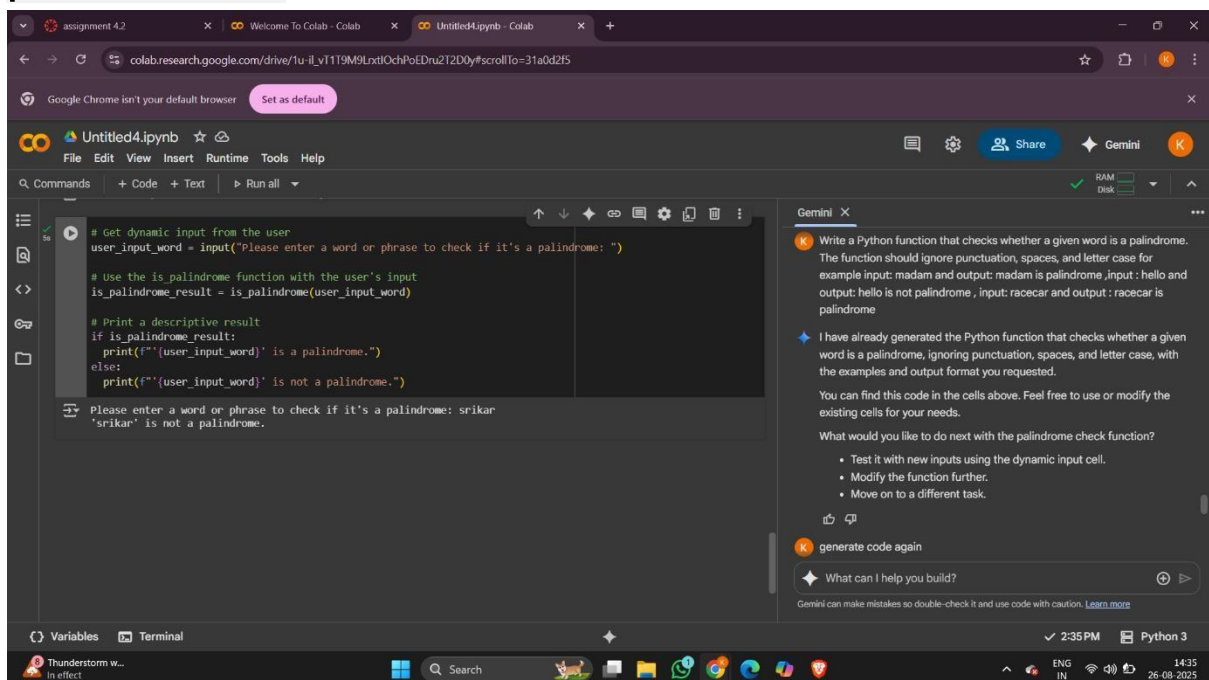
Prompt:

ZERO-SHOT: Write a Python function that checks whether a given word is a palindrome. The function should ignore punctuation, spaces, and letter case.



FEW-SHOT:

Write a Python function that checks whether a given word is a palindrome. The function should ignore punctuation, spaces, and letter case for example input: madam and output: madam is palindrome ,input : hello and output: hello is not palindrome , input: racecar and output : racecar is palindrome.



The screenshot displays a Google Colab notebook titled 'Untitled4.ipynb'. The code in the notebook defines a function `is_palindrome` that takes a string `user_input_word` and returns a boolean result. The function uses `input()` to get user input and `print()` to display the result. The output shows that 'srikar' is not a palindrome. To the right of the code editor is a Gemini chat interface. The prompt asks to write a Python function to check for palindromes, ignoring punctuation, spaces, and letter case, with examples: 'madam' is a palindrome, 'hello' is not, and 'racecar' is. The Gemini response provides the function code and suggests further actions like testing with new inputs or modifying the function.

```
# Get dynamic input from the user
user_input_word = input("Please enter a word or phrase to check if it's a palindrome: ")

# Use the is_palindrome function with the user's input
is_palindrome_result = is_palindrome(user_input_word)

# Print a descriptive result
if is_palindrome_result:
    print(f'{user_input_word} is a palindrome.')
else:
    print(f'{user_input_word} is not a palindrome.')

Please enter a word or phrase to check if it's a palindrome: srikar
'srikar' is not a palindrome.
```

CONCLUSION:

Zero-shot prompting is faster and works if the task is simple and unambiguous, but risks misinterpretation. **Few-shot prompting** is more reliable because examples clarify expectations and edge cases, leading to better, more accurate code. For a task like palindrome checking (where rules

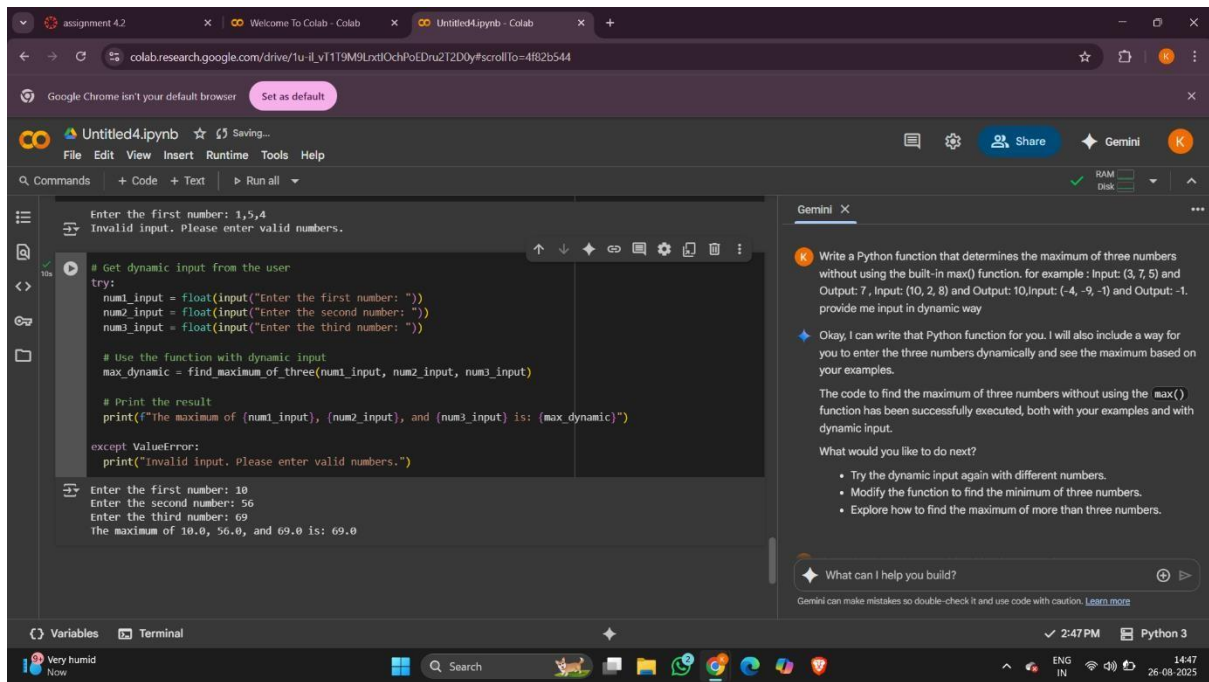
like ignoring punctuation and case matter), **few-shot prompting is superior.**

TASK-5:

Use few-shot prompting with 3 sample inputs to generate a function that determines the maximum of three numbers without using the built-in `max()` function.

Prompt:

Write a Python function that determines the maximum of three numbers without using the built-in `max()` function. for example : Input: (3,7, 5) and Output: 7 , Input: (10, 2, 8) and Output: 10,Input: (-4, -9, -1) and Output: -1. provide me input in dynamic way.



CONCLUSION:

The program correctly determines the largest of three numbers without using the built-in `max()` function. By comparing the numbers step by step with conditional statements, it can handle positive, negative, and mixed inputs. This shows how logical thinking and control structures in Python can replace built-in functions to solve problems efficiently.

