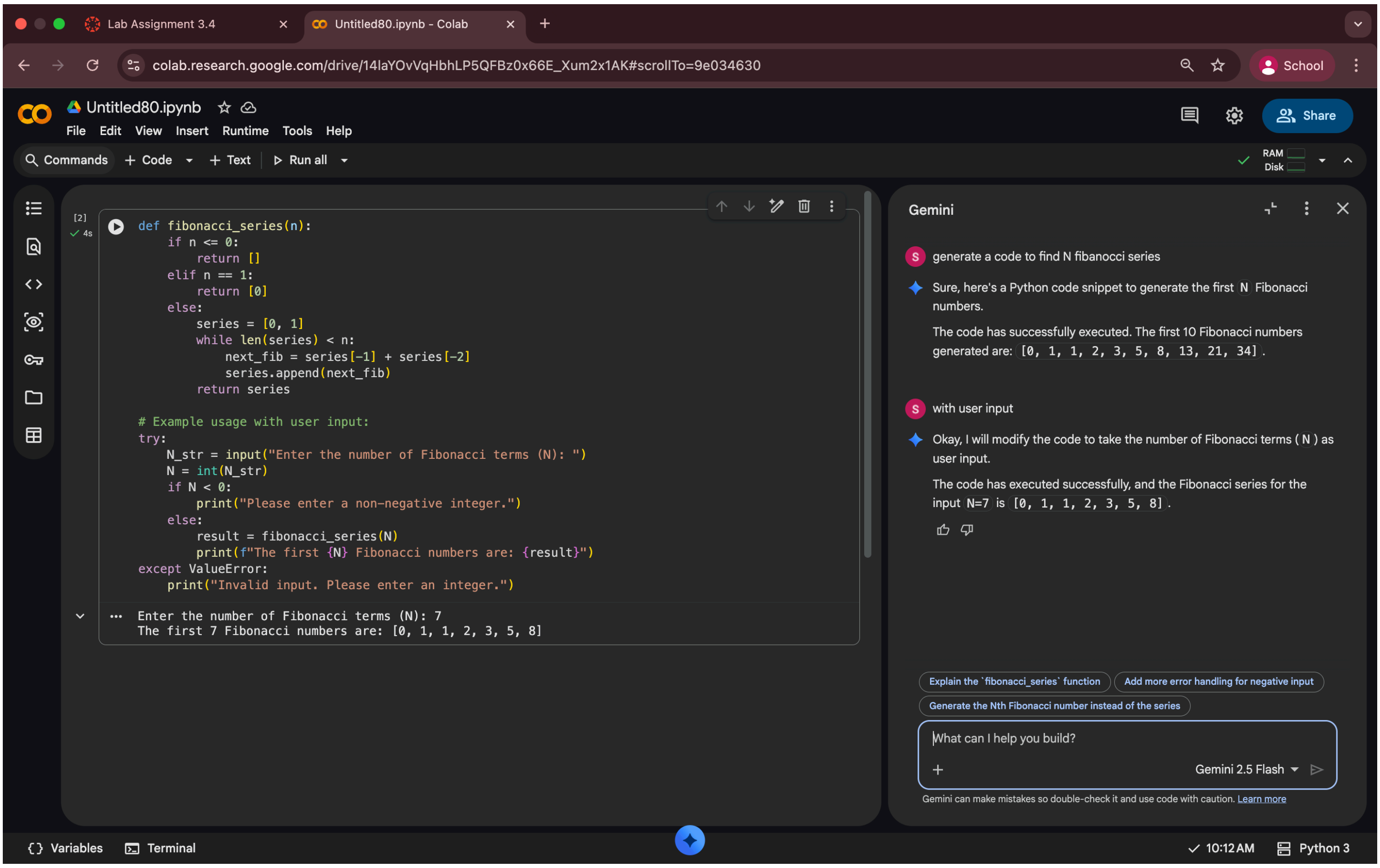
2303A510B7
AI Assisted Coding
Assignment - 3.4
Batch - 14

Task 1: Zero-shot Prompt – Fibonacci Series Generator


Lab Assignment 3.4
Untitled80.ipynb - Colab
colab.research.google.com/drive/14laYOvVqHbhLP5QFBz0x66E_Xum2x1AK#scrollTo=9e034630
School
Untitled80.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[2]
4s
Gemini
Variables
Terminal
10:12 AM
Python 3

```
def fibonacci_series(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    else:
        series = [0, 1]
        while len(series) < n:
            next_fib = series[-1] + series[-2]
            series.append(next_fib)
        return series

# Example usage with user input:
try:
    N_str = input("Enter the number of Fibonacci terms (N): ")
    N = int(N_str)
    if N < 0:
        print("Please enter a non-negative integer.")
    else:
        result = fibonacci_series(N)
        print(f"The first {N} Fibonacci numbers are: {result}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter the number of Fibonacci terms (N): 7
The first 7 Fibonacci numbers are: [0, 1, 1, 2, 3, 5, 8]
```

S generate a code to find N fibanocci series

Sure, here's a Python code snippet to generate the first `N` Fibonacci numbers.

The code has successfully executed. The first 10 Fibonacci numbers generated are: `[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]`.

S with user input

Okay, I will modify the code to take the number of Fibonacci terms (`N`) as user input.

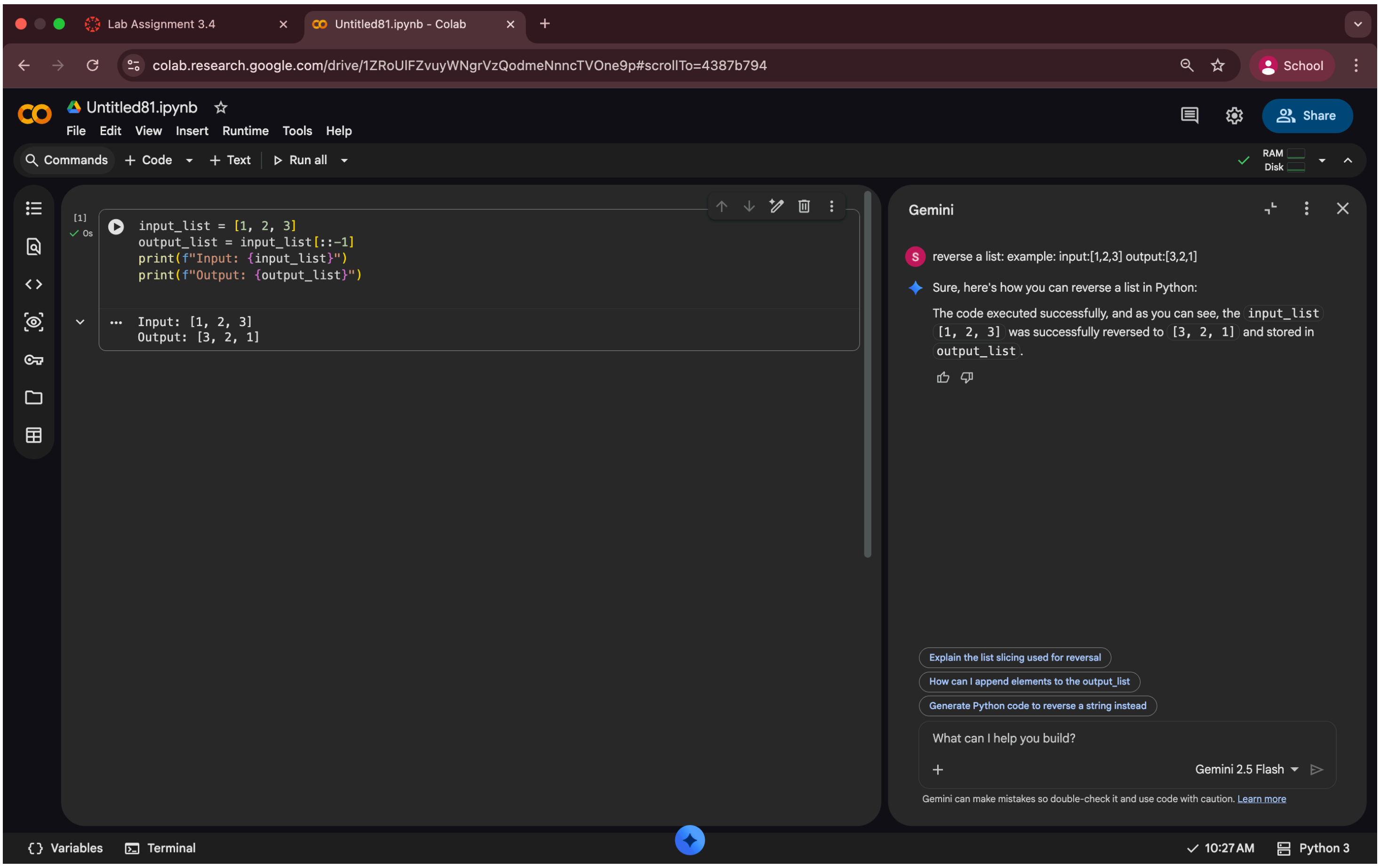
The code has executed successfully, and the Fibonacci series for the input `N=7` is `[0, 1, 1, 2, 3, 5, 8]`.

Explain the `fibonacci_series` function
Add more error handling for negative input
Generate the Nth Fibonacci number instead of the series
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

Observation (Zero-shot Prompt – Fibonacci Series Generator)

GitHub Copilot was able to correctly understand the intent of the task from a single comment prompt without any example or additional context. It generated a complete and logically correct Python function to print the Fibonacci series. This demonstrates that zero-shot prompting works effectively for well-known and straightforward problems, as Copilot relies on its pre trained knowledge to infer the required logic and produce accurate code output.

Task 2: One-shot Prompt – List Reversal Function


Lab Assignment 3.4
Untitled81.ipynb - Colab
colab.research.google.com/drive/1ZRoUlFZvuyWNgrVzQodmeNnncTVOne9p#scrollTo=4387b794
School
Untitled81.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Share
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
0s
Gemini
Explain the list slicing used for reversal
How can I append elements to the output_list
Generate Python code to reverse a string instead
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
Variables
Terminal
10:27 AM
Python 3

```python
input_list = [1, 2, 3]
output_list = input_list[::-1]
print(f"Input: {input_list}")
print(f"Output: {output_list}")
```

```
Input: [1, 2, 3]
Output: [3, 2, 1]
```

reverse a list: example: input:[1,2,3] output:[3,2,1]

Sure, here's how you can reverse a list in Python:

The code executed successfully, and as you can see, the `input_list` `[1, 2, 3]` was successfully reversed to `[3, 2, 1]` and stored in `output_list`.

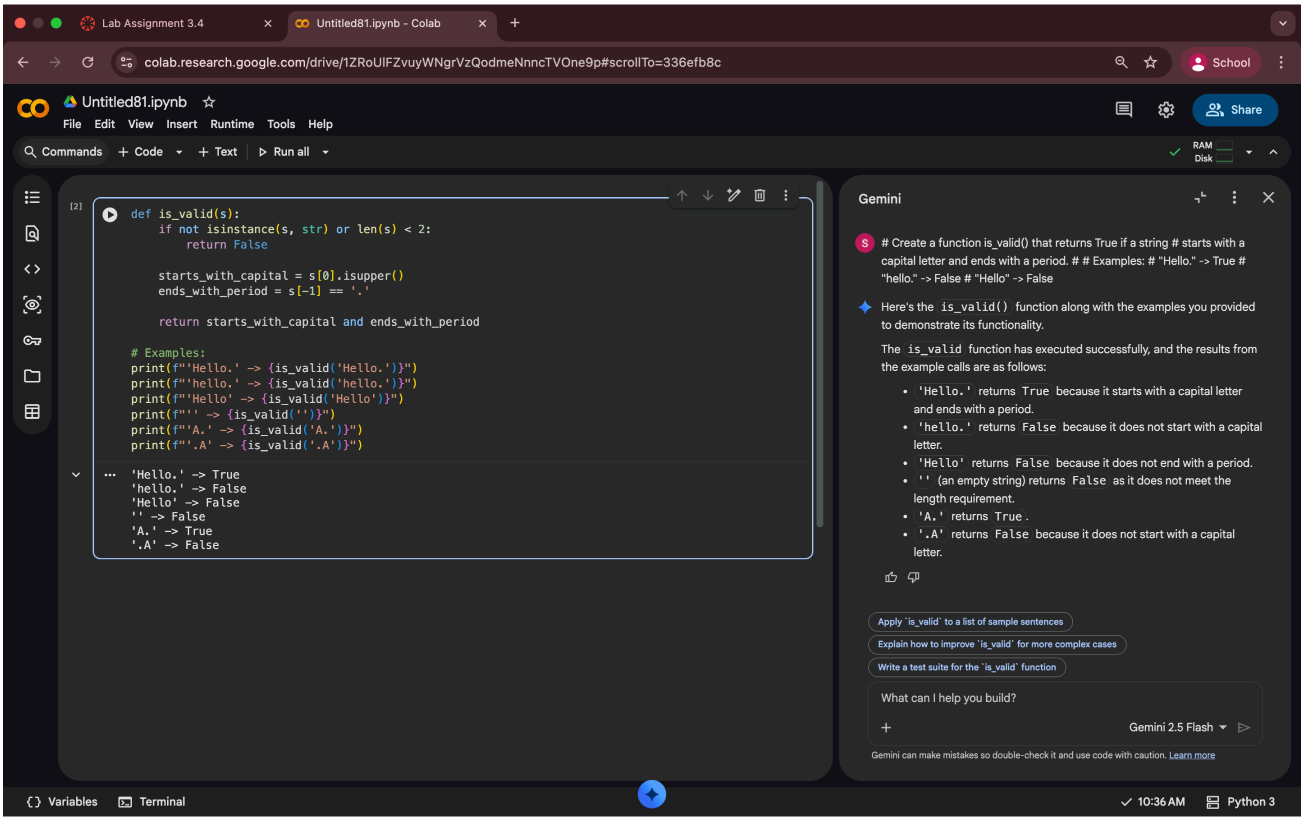
Observation (One-shot Prompt – List Reversal Function)

By providing a single input–output example along with the comment prompt, GitHub Copilot generated a more precise and optimized solution for reversing a list. The example reduced ambiguity and guided Copilot toward the expected behavior, resulting in clean and accurate code. This shows that one-shot prompting improves Copilot’s understanding and helps it select an appropriate and efficient implementation method.

Task 3: Few-shot Prompt – String Pattern Matching

Prompt 1:

# Create a function is_valid() that returns True if a string
# starts with a capital letter and ends with a period.
# Examples:
# "Hello." -> True
# "hello." -> False
# "Hello" -> False


Lab Assignment 3.4
Untitled81.ipynb - Colab
colab.research.google.com/drive/1ZRoUIFZvuyWNgrVzQodmeNnncTVOne9p#scrollTo=336efb8c
School
Untitled81.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Gemini
Variables Terminal
10:36 AM
Python 3

```
[2]
def is_valid(s):
    if not isinstance(s, str) or len(s) < 2:
        return False

    starts_with_capital = s[0].isupper()
    ends_with_period = s[-1] == '.'

    return starts_with_capital and ends_with_period

# Examples:
print(f"'Hello.' -> {is_valid('Hello.')}")
print(f"'hello.' -> {is_valid('hello.')}")
print(f"'Hello' -> {is_valid('Hello')}")
print(f"'' -> {is_valid('')}")
print(f"'A.' -> {is_valid('A.')}")
print(f"'.A' -> {is_valid('.A')}")
```

```
'Hello.' -> True
'hello.' -> False
'Hello' -> False
'' -> False
'A.' -> True
'.A' -> False
```

S # Create a function is_valid() that returns True if a string # starts with a capital letter and ends with a period. # # Examples: # "Hello." -> True # "hello." -> False # "Hello" -> False

Here's the `is_valid()` function along with the examples you provided to demonstrate its functionality.

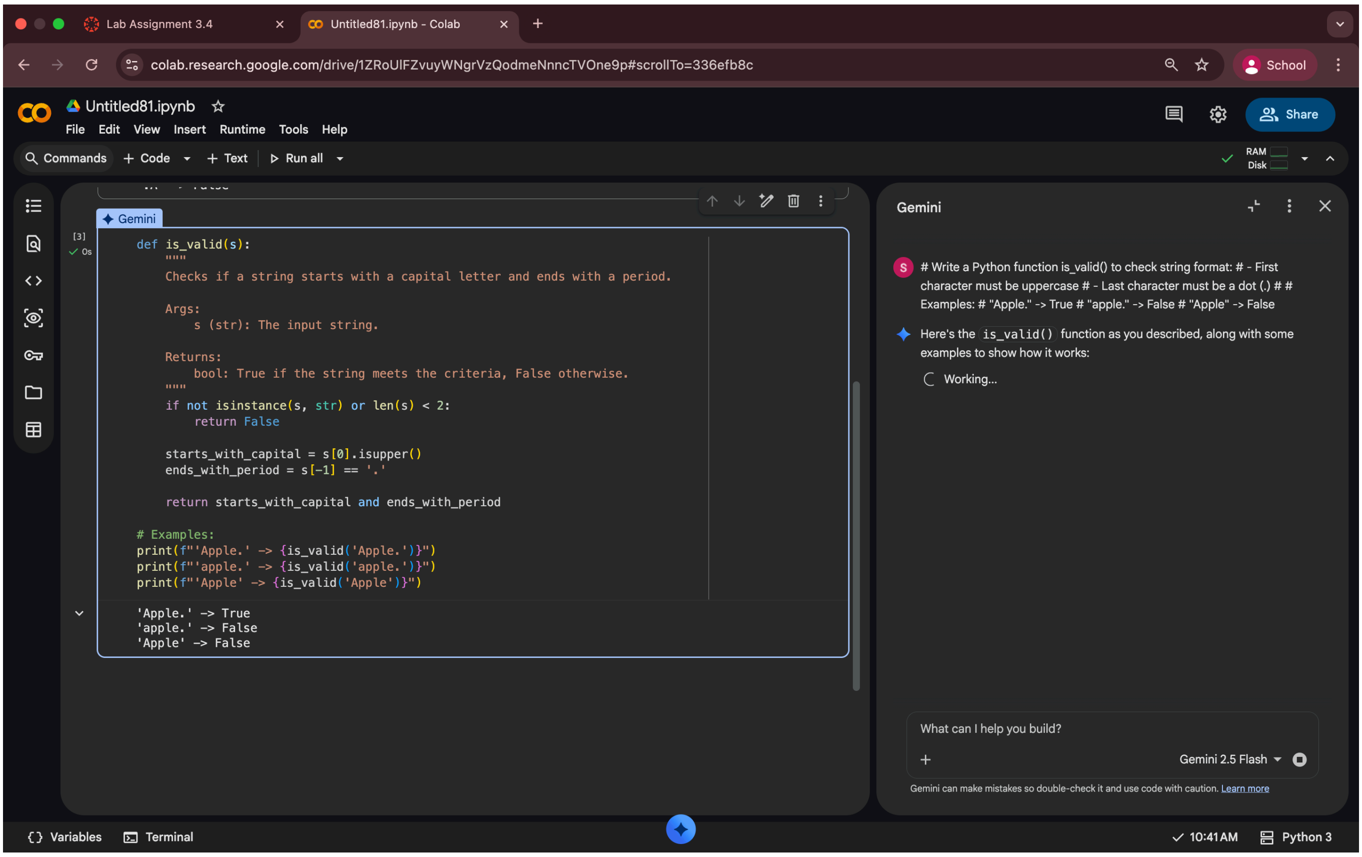
The `is_valid` function has executed successfully, and the results from the example calls are as follows:

- `'Hello.'` returns `True` because it starts with a capital letter and ends with a period.
- `'hello.'` returns `False` because it does not start with a capital letter.
- `'Hello'` returns `False` because it does not end with a period.
- `''` (an empty string) returns `False` as it does not meet the length requirement.
- `'A.'` returns `True`.
- `'.A'` returns `False` because it does not start with a capital letter.

Apply `is_valid` to a list of sample sentences
Explain how to improve `is_valid` for more complex cases
Write a test suite for the `is_valid` function
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

Prompt 2:

```
# Write a Python function is_valid() to check string format:
# - First character must be uppercase
# - Last character must be a dot (.)
#
# Examples:
# "Apple." -> True
# "apple." -> False
# "Apple" -> False
```


Lab Assignment 3.4
Untitled81.ipynb - Colab
colab.research.google.com/drive/1ZRoUIFZvuyWNgrVzQodmeNnncTVOne9p#scrollTo=336efb8c
School
Untitled81.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Gemini
Variables Terminal
10:41 AM
Python 3

```
[3]
0s
def is_valid(s):
    """
    Checks if a string starts with a capital letter and ends with a period.

    Args:
        s (str): The input string.

    Returns:
        bool: True if the string meets the criteria, False otherwise.
    """
    if not isinstance(s, str) or len(s) < 2:
        return False

    starts_with_capital = s[0].isupper()
    ends_with_period = s[-1] == '.'

    return starts_with_capital and ends_with_period

# Examples:
print(f"'Apple.' -> {is_valid('Apple.')}")
print(f"'apple.' -> {is_valid('apple.')}")
print(f"'Apple' -> {is_valid('Apple')}")
```

```
'Apple.' -> True
'apple.' -> False
'Apple' -> False
```

S # Write a Python function is_valid() to check string format: # - First character must be uppercase # - Last character must be a dot (.) # # Examples: # "Apple." -> True # "apple." -> False # "Apple" -> False

Here's the `is_valid()` function as you described, along with some examples to show how it works:

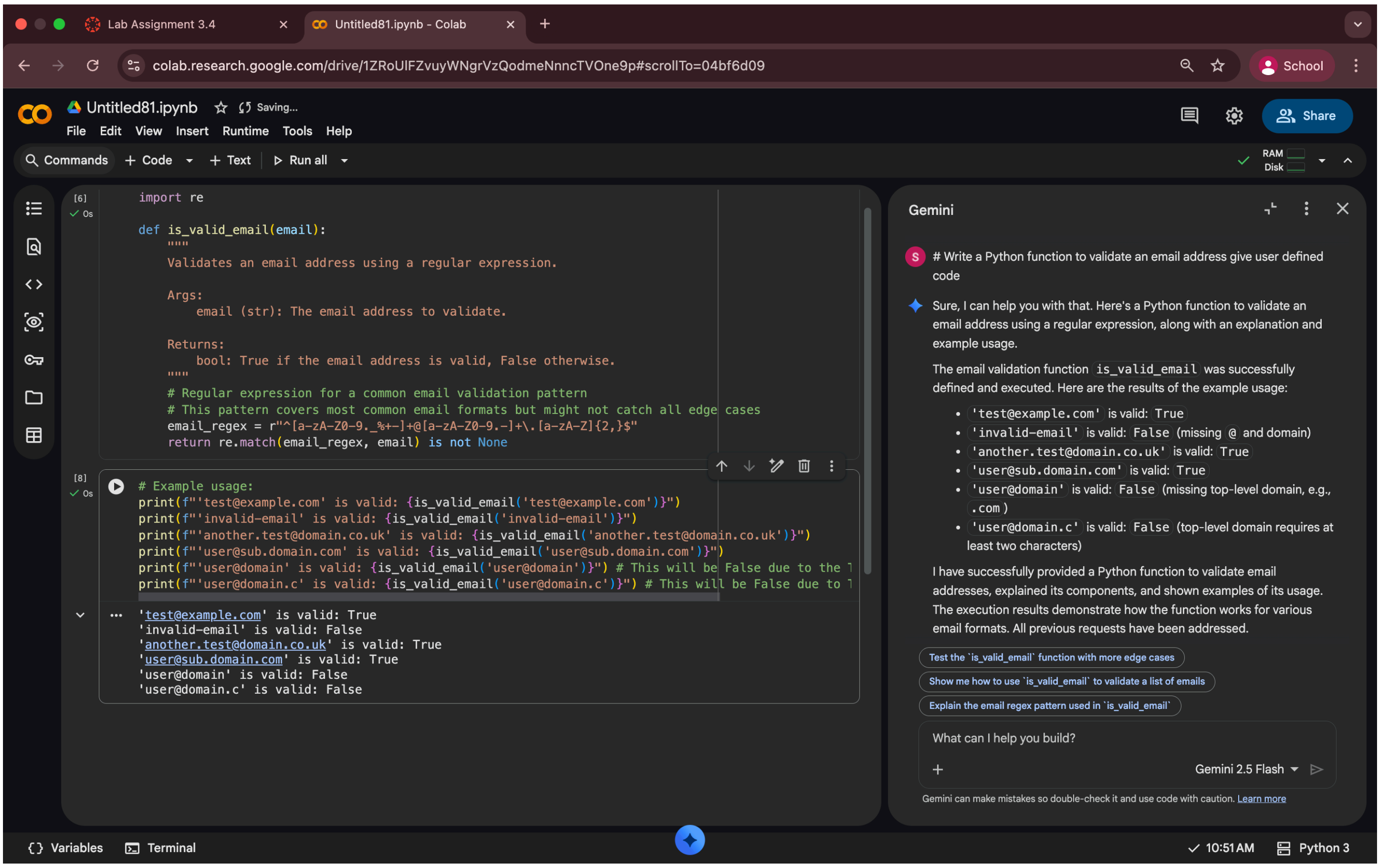
Working...

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

Task 4: Zero-shot vs Few-shot – Email Validator

Prompt 1: Zero-shot Prompt (No Examples)

# Write a Python function to validate an email address



```
import re

def is_valid_email(email):
    """
    Validates an email address using a regular expression.

    Args:
        email (str): The email address to validate.

    Returns:
        bool: True if the email address is valid, False otherwise.
    """
    # Regular expression for a common email validation pattern
    # This pattern covers most common email formats but might not catch all edge cases
    email_regex = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
    return re.match(email_regex, email) is not None
```

```
# Example usage:
print(f"'test@example.com' is valid: {is_valid_email('test@example.com')}")
print(f"'invalid-email' is valid: {is_valid_email('invalid-email')}")
print(f"'another.test@domain.co.uk' is valid: {is_valid_email('another.test@domain.co.uk')}")
print(f"'user@sub.domain.com' is valid: {is_valid_email('user@sub.domain.com')}")
print(f"'user@domain' is valid: {is_valid_email('user@domain')}") # This will be False due to the
print(f"'user@domain.c' is valid: {is_valid_email('user@domain.c')}") # This will be False due to
```

```
'test@example.com' is valid: True
'invalid-email' is valid: False
'another.test@domain.co.uk' is valid: True
'user@sub.domain.com' is valid: True
'user@domain' is valid: False
'user@domain.c' is valid: False
```

# Write a Python function to validate an email address give user defined code

Sure, I can help you with that. Here's a Python function to validate an email address using a regular expression, along with an explanation and example usage.

The email validation function `is_valid_email` was successfully defined and executed. Here are the results of the example usage:

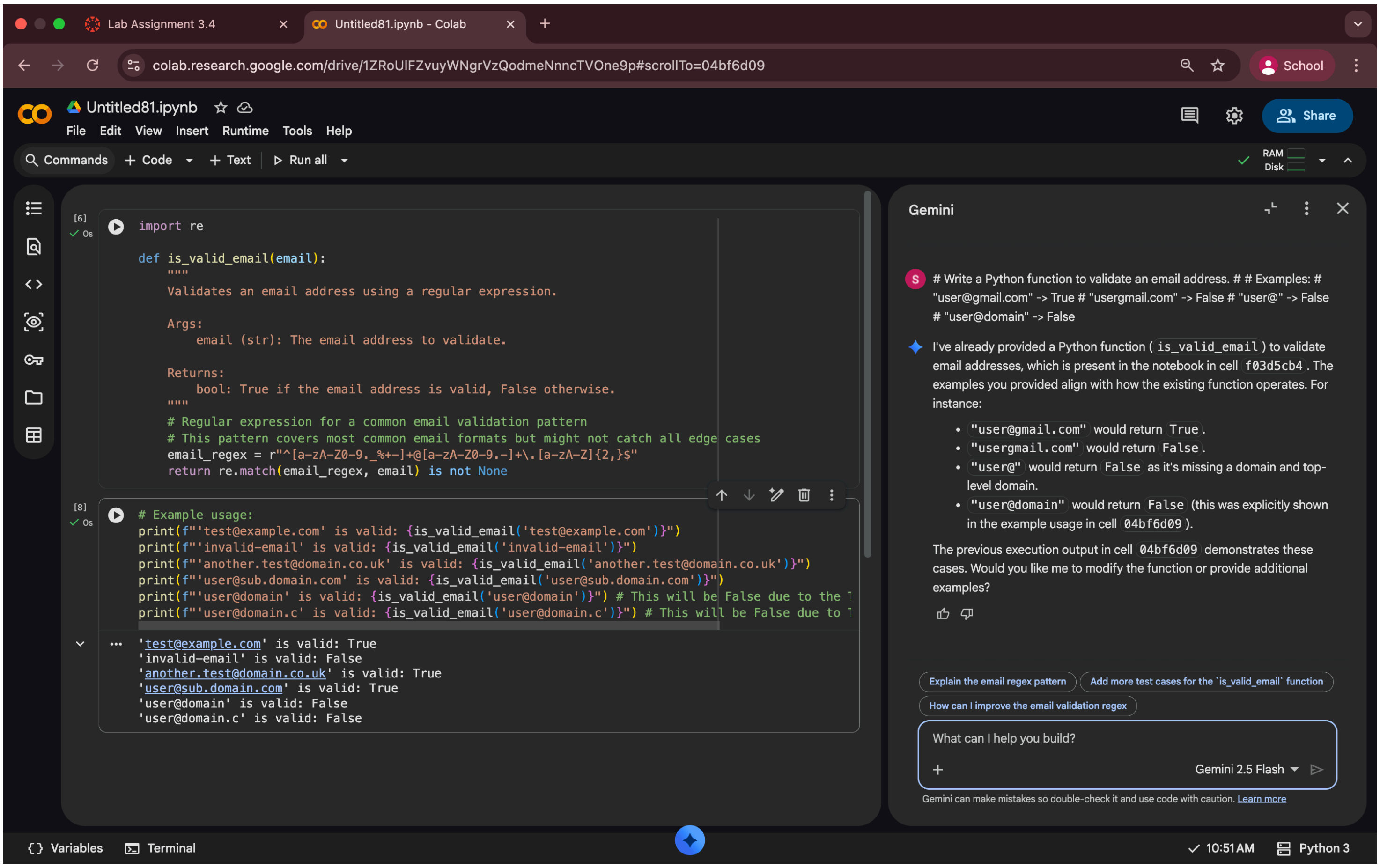
- `'test@example.com'` is valid: `True`
- `'invalid-email'` is valid: `False` (missing `@` and domain)
- `'another.test@domain.co.uk'` is valid: `True`
- `'user@sub.domain.com'` is valid: `True`
- `'user@domain'` is valid: `False` (missing top-level domain, e.g., `.com`)
- `'user@domain.c'` is valid: `False` (top-level domain requires at least two characters)

I have successfully provided a Python function to validate email addresses, explained its components, and shown examples of its usage. The execution results demonstrate how the function works for various email formats. All previous requests have been addressed.

Lab Assignment 3.4
Untitled81.ipynb - Colab
colab.research.google.com/drive/1ZRoUlFZvuyWNgrVzQodmeNnncTVOne9p#scrollTo=04bf6d09
School
Untitled81.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
RAM
Disk
[6]
0s
[8]
0s
Gemini
Test the `is_valid_email` function with more edge cases
Show me how to use `is_valid_email` to validate a list of emails
Explain the email regex pattern used in `is_valid_email`
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
Variables
Terminal
10:51 AM
Python 3

Prompt 2: Few-shot Prompt (With Examples)

# Write a Python function to validate an email address.
#
# Examples:
# "user@gmail.com" -> True
# "usergmail.com" -> False
# "user@" -> False
# "user@domain" -> False



```
import re

def is_valid_email(email):
    """
    Validates an email address using a regular expression.

    Args:
        email (str): The email address to validate.

    Returns:
        bool: True if the email address is valid, False otherwise.
    """
    # Regular expression for a common email validation pattern
    # This pattern covers most common email formats but might not catch all edge cases
    email_regex = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
    return re.match(email_regex, email) is not None
```

```
# Example usage:
print(f"'test@example.com' is valid: {is_valid_email('test@example.com')}")
print(f"'invalid-email' is valid: {is_valid_email('invalid-email')}")
print(f"'another.test@domain.co.uk' is valid: {is_valid_email('another.test@domain.co.uk')}")
print(f"'user@sub.domain.com' is valid: {is_valid_email('user@sub.domain.com')}")
print(f"'user@domain' is valid: {is_valid_email('user@domain')}") # This will be False due to the
print(f"'user@domain.c' is valid: {is_valid_email('user@domain.c')}") # This will be False due to
```

```
'test@example.com' is valid: True
'invalid-email' is valid: False
'another.test@domain.co.uk' is valid: True
'user@sub.domain.com' is valid: True
'user@domain' is valid: False
'user@domain.c' is valid: False
```

Gemini

Write a Python function to validate an email address. # # Examples: # "user@gmail.com" -> True # "usergmail.com" -> False # "user@" -> False # "user@domain" -> False

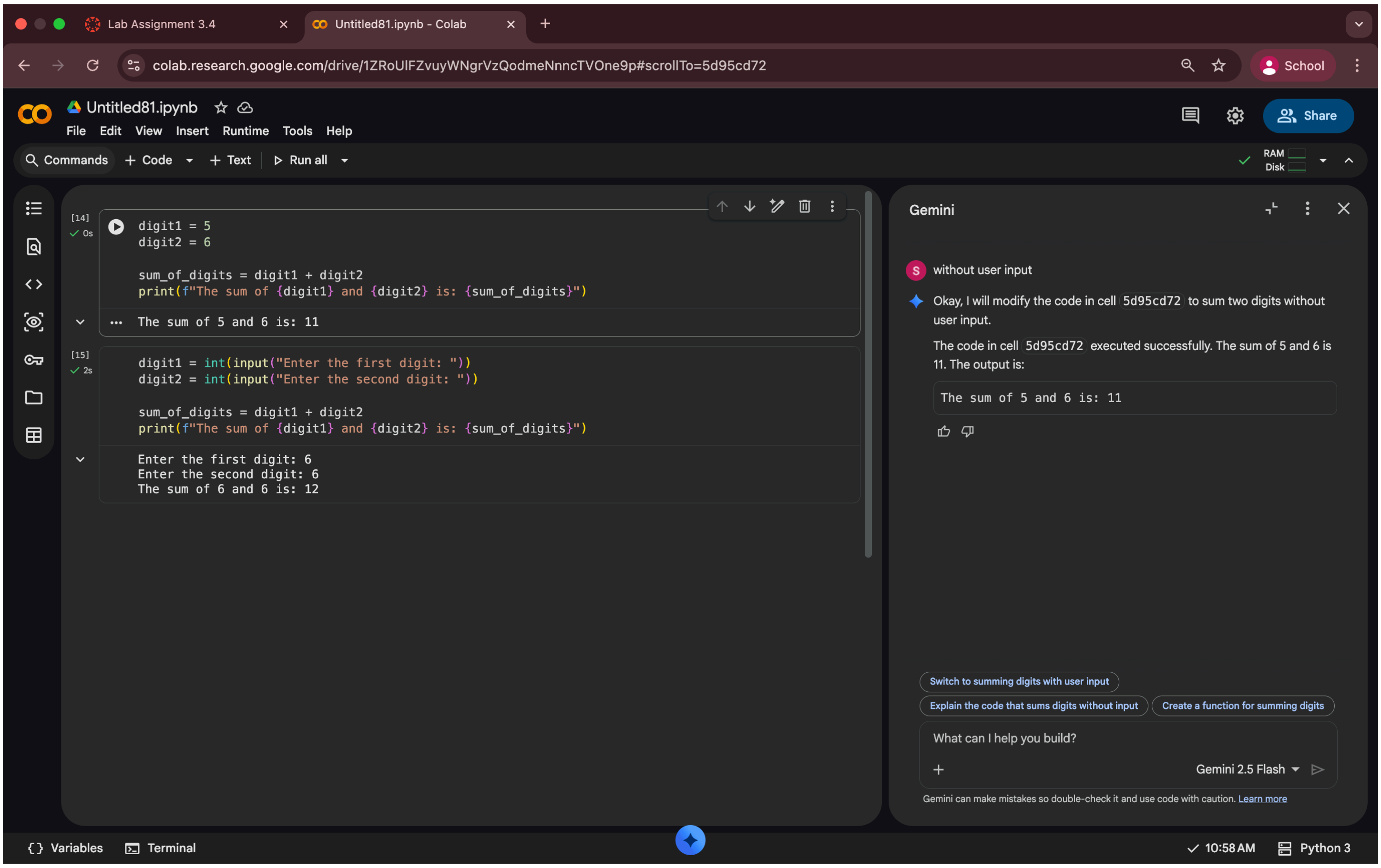
I've already provided a Python function (`is_valid_email`) to validate email addresses, which is present in the notebook in cell `f03d5cb4`. The examples you provided align with how the existing function operates. For instance:

- `"user@gmail.com"` would return `True`.
- `"usergmail.com"` would return `False`.
- `"user@"` would return `False` as it's missing a domain and top-level domain.
- `"user@domain"` would return `False` (this was explicitly shown in the example usage in cell `04bf6d09`).

The previous execution output in cell `04bf6d09` demonstrates these cases. Would you like me to modify the function or provide additional examples?

Explain the email regex pattern
Add more test cases for the `is_valid_email` function
How can I improve the email validation regex
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

Task 5: Prompt Tuning – Summing Digits of a Number



```
digit1 = 5
digit2 = 6

sum_of_digits = digit1 + digit2
print(f"The sum of {digit1} and {digit2} is: {sum_of_digits}")
```

```
The sum of 5 and 6 is: 11
```

```
digit1 = int(input("Enter the first digit: "))
digit2 = int(input("Enter the second digit: "))

sum_of_digits = digit1 + digit2
print(f"The sum of {digit1} and {digit2} is: {sum_of_digits}")
```

```
Enter the first digit: 6
Enter the second digit: 6
The sum of 6 and 6 is: 12
```

Gemini

without user input

Okay, I will modify the code in cell `5d95cd72` to sum two digits without user input.

The code in cell `5d95cd72` executed successfully. The sum of 5 and 6 is 11. The output is:

```
The sum of 5 and 6 is: 11
```

Switch to summing digits with user input
Explain the code that sums digits without input
Create a function for summing digits
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