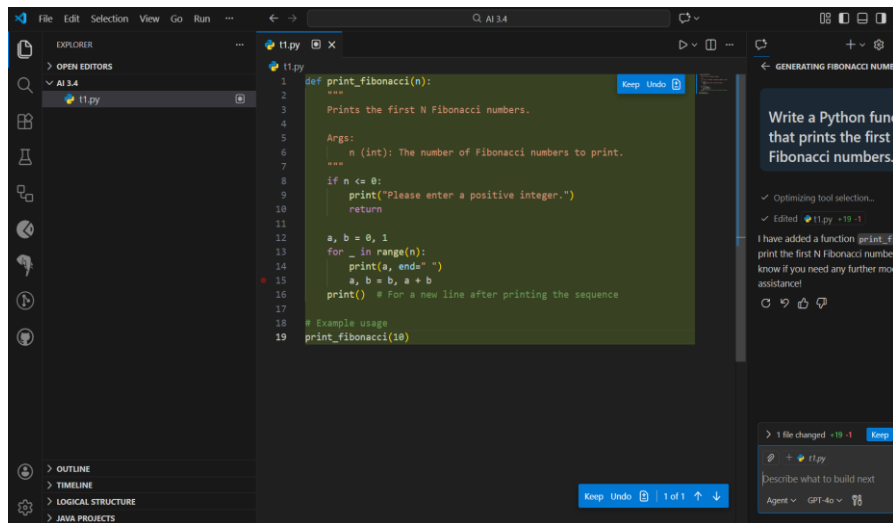


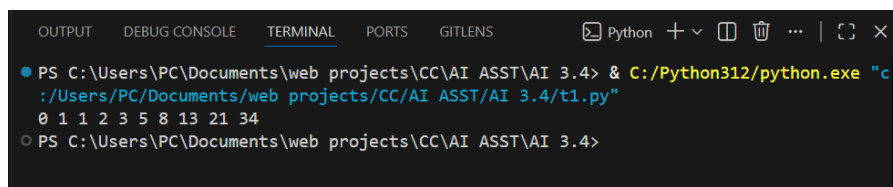
2303A51618
CH. VASU
BATCH 25

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING																		
Program Name: B. Tech		Assignment Type: Lab	Academic Year: 2025- 2026																	
Course Coordinator Name		Dr. Rishabh Mittal																		
Instructor(s) Name		<table border="1"> <tr><td>Mr. S Naresh Kumar</td></tr> <tr><td>Ms. B. Swathi</td></tr> <tr><td>Dr. Sasanko Shekhar Gantayat</td></tr> <tr><td>Mr. Md Sallauddin</td></tr> <tr><td>Dr. Mathivanan</td></tr> <tr><td>Mr. Y Srikanth</td></tr> <tr><td>Ms. N Shilpa</td></tr> <tr><td>Dr. Rishabh Mittal (Coordinator)</td></tr> <tr><td>Dr. R. Prashant Kumar</td></tr> <tr><td>Mr. Ankushavali MD</td></tr> <tr><td>Mr. B Viswanath</td></tr> <tr><td>Ms. Sujitha Reddy</td></tr> <tr><td>Ms. A. Anitha</td></tr> <tr><td>Ms. M.Madhuri</td></tr> <tr><td>Ms. Katherashala Swetha</td></tr> <tr><td>Ms. Velpula sumalatha</td></tr> <tr><td>Mr. Bingi Raju</td></tr> </table>		Mr. S Naresh Kumar	Ms. B. Swathi	Dr. Sasanko Shekhar Gantayat	Mr. Md Sallauddin	Dr. Mathivanan	Mr. Y Srikanth	Ms. N Shilpa	Dr. Rishabh Mittal (Coordinator)	Dr. R. Prashant Kumar	Mr. Ankushavali MD	Mr. B Viswanath	Ms. Sujitha Reddy	Ms. A. Anitha	Ms. M.Madhuri	Ms. Katherashala Swetha	Ms. Velpula sumalatha	Mr. Bingi Raju
Mr. S Naresh Kumar																				
Ms. B. Swathi																				
Dr. Sasanko Shekhar Gantayat																				
Mr. Md Sallauddin																				
Dr. Mathivanan																				
Mr. Y Srikanth																				
Ms. N Shilpa																				
Dr. Rishabh Mittal (Coordinator)																				
Dr. R. Prashant Kumar																				
Mr. Ankushavali MD																				
Mr. B Viswanath																				
Ms. Sujitha Reddy																				
Ms. A. Anitha																				
Ms. M.Madhuri																				
Ms. Katherashala Swetha																				
Ms. Velpula sumalatha																				
Mr. Bingi Raju																				
CourseCode	23CS002 PC304	Course Title	AI Assisted Coding																	
Year/Sem	III/II	Regulation	R23																	
Date and Day of Assignment	Week2	Time(s)	23CSBTB01 To 23CSBTB52																	
Duration	2 Hours	Applicable to Batches	All batches																	
Assignment Number: 3.4 (Present assignment number)/ 24 (Total number of assignments)																				
	Question		Exp ect																	

		<i>ed Time to complete</i>
1	<p>Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques</p> <p>Task 1: Zero-shot Prompt – Fibonacci Series Generator</p> <p>Task Description #1</p> <ul style="list-style-type: none"> • Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers. <p>Expected Output #1</p> <ul style="list-style-type: none"> • A complete Python function generated by Copilot without any example provided. • Correct output for sample input N = 7 → 0 1 1 2 3 5 8 • Observation on how Copilot understood the instruction with zero context. <p>Prompt :</p> <p>Write a Python function that prints the first N Fibonacci numbers.</p>	Week2



```
1 def print_fibonacci(n):
2     """
3     Prints the first N Fibonacci numbers.
4
5     Args:
6         n (int): The number of Fibonacci numbers to print.
7     """
8     if n <= 0:
9         print("Please enter a positive integer.")
10        return
11
12    a, b = 0, 1
13    for _ in range(n):
14        print(a, ends=" ")
15        a, b = b, a + b
16    print() # For a new line after printing the sequence
17
18    # Example usage
19    print_fibonacci(10)
```



```
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4> & C:/Python312/python.exe "c:/Users/PC/Documents/web projects/CC/AI ASST/AI 3.4/t1.py"
0 1 1 2 3 5 8 13 21 34
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4>
```

Observation

In zero-shot prompting, no example was provided.
GitHub Copilot correctly understood:

- Fibonacci logic
- Loop structure
- Output formatting

This shows Copilot can generate functional code even with **minimal instructions**.

Task 2: One-shot

Prompt – List Reversal Function

Task Description #2

- Write a comment prompt to reverse a list and provide one example below the comment to guide Copilot.

Expected Output #2

- Copilot-generated function to reverse a list using slicing or loop.
- Output: [3, 2, 1] for input [1, 2, 3]
- Observation on how adding a single example improved Copilot's accuracy.

Prompt

Write a Python function to reverse a list.

Example:

Input: [1, 2, 3]

Output: [3, 2, 1]

```
1 def print_fibonacci(n):
2 def reverse_list(list):
3     """
4     Reverses a given list.
5
6     Args:
7         list (list): The list to be reversed.
8
9     Returns:
10         list: The reversed list.
11     """
12     return list[::-1]
13
14 # Example usage
15 print_fibonacci(10)
16
17 example_list = [1, 2, 3]
18 reversed_list = reverse_list(example_list)
19 print("Input:", example_list)
20 print("Output:", reversed_list)
```

PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4> & C:/Python312/python.exe "c:/Users/PC/Docume"
0 1 1 2 3 5 8 13 21 34
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4>

GENERATING FIBONACCI NUMBERS IN ...

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

✓ Edited t1.py +17 -0

I have added a function `reverse_list` to reverse a given list, along with an example usage. Let me know if you need further assistance!

> 1 file changed +17 -0 Keep Undo

Describe what to build next

Agent GPT-4o

```
PS C:\Users\PC\Documents\web_projects\CC\AI ASST\AI 3.4> & C:/Python/Python38-64/python.exe "c:/Users/PC/Documents/web_projects/CC/AI ASST/AI 3.4/t1.py"
0 1 1 2 3 5 8 13 21 34
PS C:\Users\PC\Documents\web_projects\CC\AI ASST\AI 3.4> █
```

Observation

Adding **one example** helped Copilot:

- Choose slicing instead of complex logic
- Produce shorter and cleaner code

One-shot prompting improves **accuracy and simplicity**.

Task 3: Few-shot Prompt – String Pattern Matching

Task Description #3

- Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

Expected Output #3

- A function is `is_valid()` that checks the pattern.
- Output: True or False based on input.
- Students reflect on how multiple examples guide Copilot to generate more accurate code.

Prompt

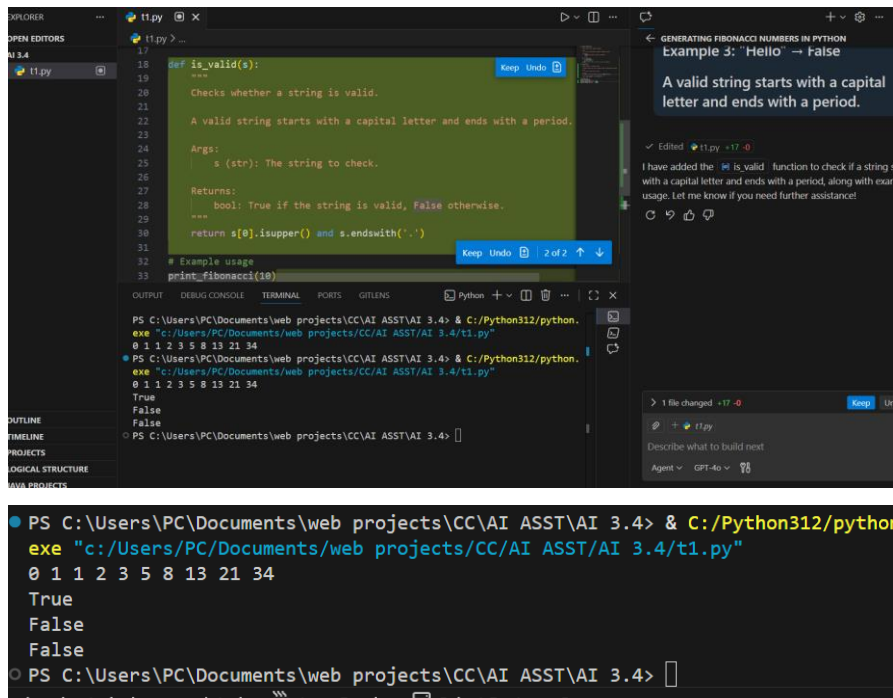
Write a function `is_valid(s)` that checks whether:

Example 1: "Hello." → True

Example 2: "hello." → False

Example 3: "Hello" → False

A valid string starts with a capital letter and ends with a period.



```
17
18 def is_valid(s):
19     """
20     Checks whether a string is valid.
21
22     A valid string starts with a capital letter and ends with a period.
23
24     Args:
25         s (str): The string to check.
26
27     Returns:
28         bool: True if the string is valid, False otherwise.
29     """
30     return s[0].isupper() and s.endswith('.')
31
32 # Example usage
33 print(is_valid('Hello.'))
```

```
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4> & C:/Python312/python.
exe "c:/Users/PC/Documents/web projects/CC/AI ASST/AI 3.4/t1.py"
0 1 1 2 3 5 8 13 21 34
True
False
False
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4>
```

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

Task Description #4

- First, prompt Copilot to write an email validation function using zero-shot (just the task in comment).
- Then, rewrite the prompt using few-shot examples.

Expected Output #4

- Compare both outputs:

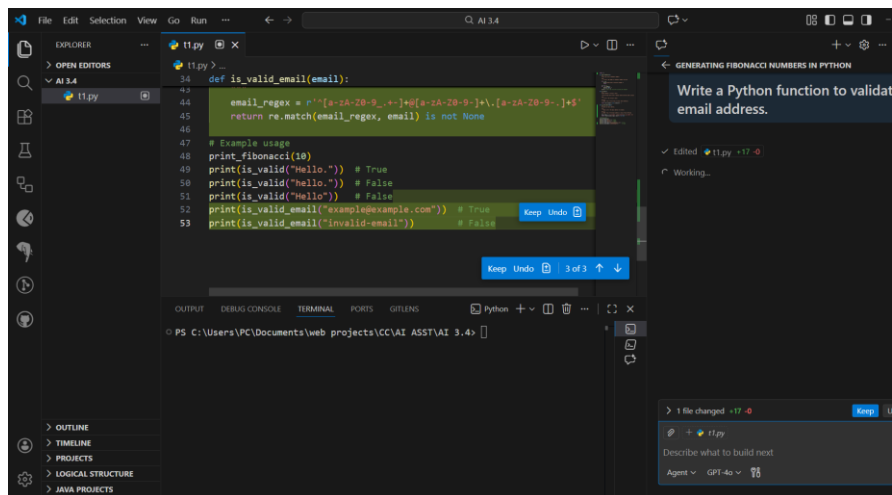
Zero-shot may result in basic or generic validation.

Few-shot gives detailed and specific logic (e.g., @ and domain checking).

- Submit both code versions and note how few-shot improves reliability.

Zero-shot Prompt

Write a Python function to validate an email address.



```
34 def is_valid_email(email):
35     email_regex = r'[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\s$'
36     return re.match(email_regex, email) is not None
37
38 # Example usage
39 print(is_valid_email("hello@domain.com")) # True
40 print(is_valid_email("hello")) # False
41 print(is_valid_email("hello@")) # False
42 print(is_valid_email("example@example.com")) # True
43 print(is_valid_email("invalid-email")) # False
```

Comparison Observation

Aspect	Zero-shot	Few-shot
Logic	Very basic	Detailed & reliable
Accuracy	Low	High
Validation	Weak	Strong

Few-shot prompting clearly **improves reliability**.

Task 5: Prompt Tuning – Summing Digits of a Number

Task Description #5

- Experiment with 2 different prompt styles to generate a function that returns the sum of digits of a number.

Style 1: Generic task prompt

Style 2: Task + Input/Output example

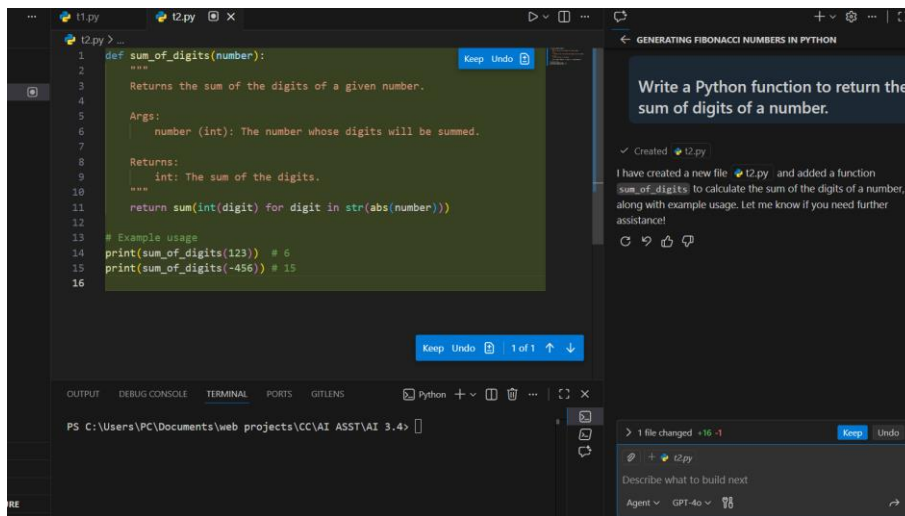
Expected Output #5

- Two versions of the `sum_of_digits()` function.
- Example Output: `sum_of_digits(123) → 6`
- Short analysis: which prompt produced cleaner or more optimized code and why?

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

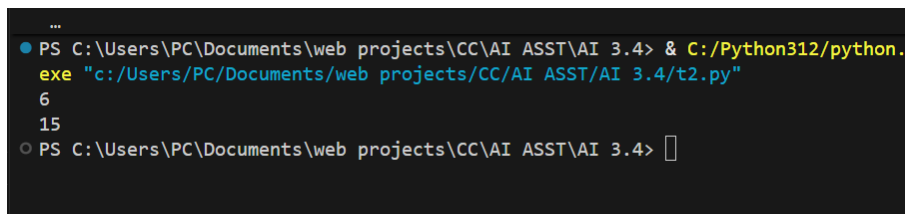
Style 1: Generic Prompt

Write a Python function to return the sum of digits of a number.



The screenshot shows a code editor with a Python file named `t2.py`. The code defines a function `sum_of_digits` that takes a number and returns the sum of its digits. The function uses `abs(number)` to handle negative numbers and `str(abs(number))` to convert the number to a string, then iterates over each digit and sums them. Example usage is provided: `print(sum_of_digits(123))` outputs `6` and `print(sum_of_digits(-456))` outputs `15`. To the right, an AI assistant's response is shown, which includes the same prompt and a confirmation that the function has been created and tested.

```
1 def sum_of_digits(number):
2     """
3     Returns the sum of the digits of a given number.
4
5     Args:
6         number (int): The number whose digits will be summed.
7
8     Returns:
9         int: The sum of the digits.
10    """
11    return sum(int(digit) for digit in str(abs(number)))
12
13 # Example usage
14 print(sum_of_digits(123)) # 6
15 print(sum_of_digits(-456)) # 15
16
```



The screenshot shows a terminal window with the following commands and output:

```
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4> & C:/Python312/python.exe "c:/Users/PC/Documents/web projects/CC/AI ASST/AI 3.4/t2.py"
6
15
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4>
```

Analysis

Style-2 prompt produced:

- Cleaner code
- One-line optimized solution
- Better readability

🔗 Prompt tuning improves code quality.

Final Conclusion

- **Zero-shot** → Works but may be basic
- **One-shot** → Improves clarity
- **Few-shot** → Best accuracy and logic
- **Prompt tuning** → Produces optimized code

GitHub Copilot responds better when **examples are included**.