

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	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasank 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		Expected

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

```

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
9             if n <= 0:
10                 print("Please enter a positive integer.")
11                 return
12
13             a, b = 0, 1
14             for _ in range(n):
15                 print(a, end=" ")
16                 a, b = b, a + b
17             print() # For a new line after printing the sequence
18
19             # Example usage
20             print_fibonacci(10)

```

OUTPUT DEBUG CONSOLE TERMINAL GITLENS

```

● 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]**

```
t1.py > ...
1 def print_fibonacci(n):
2     def reverse_list(lst):
3         """
4             Reverses a given list.
5
6             Args:
7                 lst (list): The list to be reversed.
8
9             Returns:
10                list: The reversed list.
11            """
12        return lst[::-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)

Keep Undo ⌛ | 2 of 2 ↑ ↓

OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + ⌛ ... | ⌂ X
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>

Indexing completed. Input Ready. Select Postgres Server.

Keep Undo ⌛ | 1 of 1 ⌂ X
> 1 file changed +17 -0 Keep Undo
@ + t1.py
Describe what to build next
Agent v GPT-4o v ⌂ X
GPT-4o
```

```
PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4> & C:/Pyd  
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_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.**

The screenshot shows a code editor interface with a Python file named `t1.py`. The code defines a function `is_valid(s)` that checks if a string starts with a capital letter and ends with a period. The function uses `s[0].isupper()` and `s.endswith('.')` to perform this check. The code also includes examples of how to use the function and prints the first 10 Fibonacci numbers.

```
def is_valid(s):
    """
    Checks whether a string is valid.
    A valid string starts with a capital letter and ends with a period.

    Args:
        s (str): The string to check.

    Returns:
        bool: True if the string is valid, False otherwise.
    """
    return s[0].isupper() and s.endswith('.')

# Example usage
print(fibonacci(10))
```

The terminal window below shows the output of running the script, which prints the first 10 Fibonacci numbers (0, 1, 1, 2, 3, 5, 8, 13, 21, 34) followed by three calls to the `is_valid` function with the values "True", "False", and "False".

```
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> & 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
```

**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.
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left, the Explorer sidebar shows an open folder named 'AI 3.4' containing a file 't1.py'. The main editor area displays the following Python code:

```
def is_valid_email(email):
    email_regex = r'^[a-zA-Z0-9.-]+@[a-zA-Z0-9.-]+\.[a-zA-Z0-9.-]+$'
    return re.match(email_regex, email) is not None

# Example usage
print(is_valid("Hello")) # True
print(is_valid("Hello.")) # False
print(is_valid("Hello")) # False
print(is_valid_email("example@example.com")) # True
print(is_valid_email("invalid-email")) # False
```

Below the code, the status bar indicates 'PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4>'. A floating 'GENERATING FIBONACCI NUMBERS IN PYTHON' window is visible in the center-right, with the text 'Write a Python function to validate email address.' and a 'Keep Undo' button. At the bottom right, a 'Copilot' window shows '1 file changed +17 -0' and a 'Keep' button.

## 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 require d, screenshots**

## Style 1: Generic Prompt

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

1.py t2.py

```
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
```

Keep Undo ⌛

GENERATING FIBONACCI NUMBERS IN PYTHON

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

✓ Created t2.py

I have created a new file t2.py and added a function `sum_of_digits` to calculate the sum of the digits of a number, along with example usage. Let me know if you need further assistance!

Keep Undo ⌛

OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\PC\Documents\web projects\CC\AI ASST\AI 3.4> [ ]

Keep Undo ⌛ 1 of 1 ↑ ↓

Python + ⌛

> 1 file changed +16 -1 Keep Undo

Describe what to build next

Agent GPT-4o

...  
● 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**.