

2303A1739

Gangula Navyasri

Task 1: Zero-shot Prompt – Fibonacci Series Generator

Task Description #1

- Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers.

Expected Output #1

- A complete Python function generated by Copilot without any example provided.
- Correct output for sample input $N = 7 \rightarrow 0 1 1 2 3 5 8$
- Observation on how Copilot understood the instruction with zero context.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The code editor displays a Python file named `lab ass 3.4.py` with the following content:

```
def print_fibonacci(n: int) -> None:
    if n <= 0:
        print("Please enter a positive integer.")
        return

    a, b = 0, 1

    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b

    print() # New line after the series
if __name__ == "__main__":
    print_fibonacci(10) # Output: 0 1 1 2 3 5 8 13 21 34
```

The terminal tab shows the command: `PS C:\Users\parva\OneDrive\Desktop\AI Assted>`. A floating comment box contains the task description: "Write a Python function that takes an integer N as input and prints the first N numbers of the Fibonacci series starting from 0, following correct Fibonacci logic and clean coding practices." Another floating box shows the generated code with some annotations. The status bar at the bottom right indicates "Spaces: 4" and "1/22/2026".

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

The screenshot shows the Visual Studio Code interface with the AI Assistant extension open. In the center, there's a code editor with Python code for generating a Fibonacci series and reversing lists. To the right, the AI Assistant panel displays a task card asking for a Python function to reverse a list. It includes examples and the generated code:

```
# task 2
def reverse_list(lst: list) -> list:
    return lst[::-1]

if __name__ == "__main__":
    print(fibonacci(10)) # Output: 0 1 1 2 3 5 8 13 21 34

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

The terminal at the bottom shows the execution of the script and its output:

```
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/lab ass 3.4.py"
0 1 1 2 3 5 8 13 21 34
0 1 1 2 3 5 8 13 21 34
Input: [1, 2, 3]
Output: [3, 2, 1]
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
```

1 and 2 tasks output

This screenshot shows the VS Code interface again, but this time the AI Assistant panel is more prominent. It shows the continuation of the task, including the definition of the `reverse_list` function and its usage. The terminal output is identical to the previous screenshot.

The AI Assistant panel also includes a note about adding a function to reverse a list to the existing file:

I'll add a function to reverse a list to your existing file:

...existing code...

```
# task 2
def reverse_list(lst: list) -> list:
    """
    Takes a list as input and re
    Args:
        lst (list): The list to be reversed.

    Returns:
        list: The reversed list.
    """
    return lst[::-1]
```

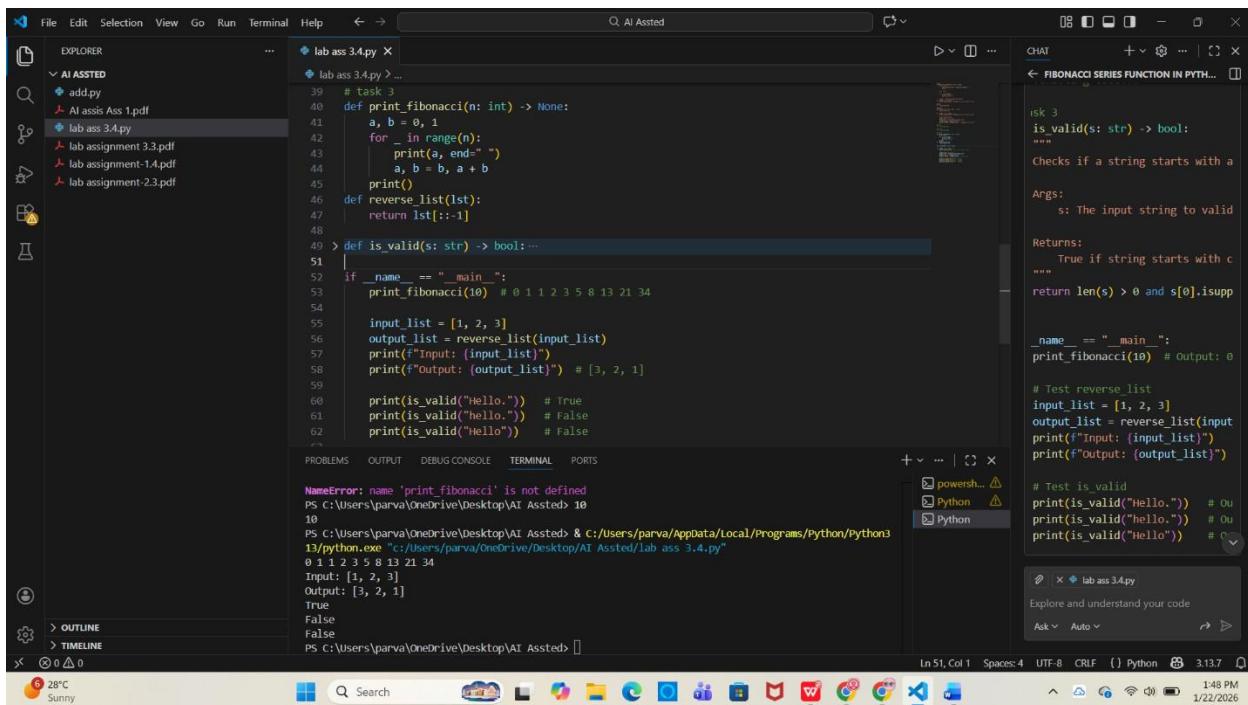
Task 3: Few-shot Prompt – String Pattern Matching

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.



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.

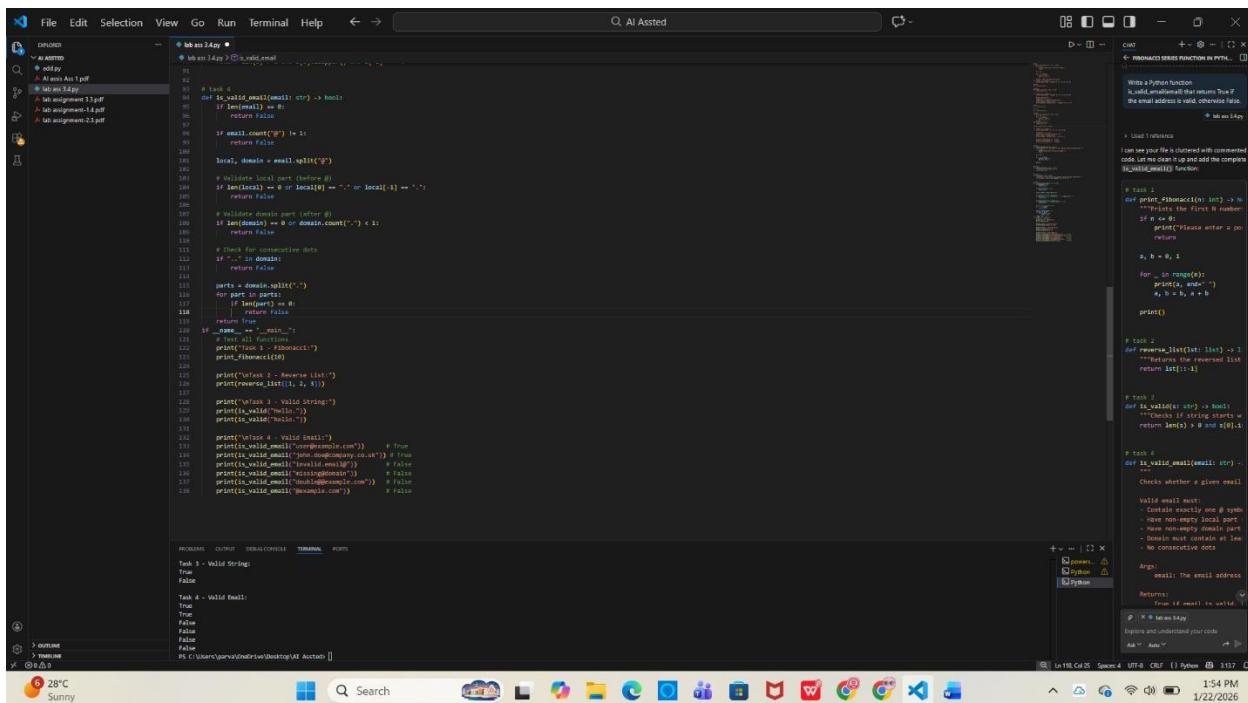
Expected Output #4

- 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.



```
140     #task4.2
141     # task 1
142     def print_fibonacci(n: int) -> None:
143         if n <= 0:
144             print("Please enter a positive integer.")
145             return
146
147         a, b = 0, 1
148
149         for _ in range(n):
150             print(a, end=" ")
151             a, b = b, a + b
152
153         print()
154
155
156     # task 2
157     def reverse_list(lst: list) -> list:
158         """Returns the reversed list."""
159         return lst[::-1]
160
161
162     # task 3
163     def is_valid(s: str) -> bool:
164         """Checks if string starts with capital letter and ends with period."""
165         return len(s) > 0 and s[0].isupper() and s[-1] == "."
166
167
168     # task 4
169     def is_valid_email(email: str) -> bool:
170         if len(email) == 0:
171             return False
172
173         if email.count "@" != 1:
174             return False
175
176         local, domain = email.split "@"
```

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** AI Assted.
- Explorer:** Shows files like AI ASSTED, add.py, AI assis Ass 1.pdf, lab assignment 3.3.pdf, lab assignment-1.4.pdf, and lab assignment-2.3.pdf.
- Code Editor:** The active file is lab ass 3.4.py, containing the following Python code:

```
def is_valid_email(email: str) -> bool:
    // ...
    if len(local) == 0 or local[0] == "." or local[-1] == ".":
        return False
    # Validate domain part (after @)
    if len(domain) == 0 or domain.count(".") < 1:
        return False
    # Check for consecutive dots
    if ".." in domain:
        return False
    parts = domain.split(".")
    for part in parts:
        if len(part) == 0:
            return False
    return True

if __name__ == "__main__":
    # Test all functions
    print("Task 1 - Fibonacci:")
    print_fibonacci(10)

    print("\nTask 2 - Reverse List:")
    print(reverse_list([1, 2, 3]))

    print("\nTask 3 - Valid String:")
    print(is_valid("Hello."))
    print(is_valid("hello."))

    print("\nTask 4 - Valid Email:")
    print(is_valid_email("user@example.com"))      # True
    print(is_valid_email("user@example.com"))        # False
    print(is_valid_email("user@.com"))               # False
    print(is_valid_email("user@domain"))             # False
    print(is_valid_email("john.doe@company.co.uk")) # True
```

Bottom Navigation: PROBLEMS 35, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS.

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?

Screenshot of the AI Assisted IDE interface. The left pane shows the file structure with files like `lab ass 3.4.py`, `add.py`, and several PDF documents. The main pane displays Python code for tasks 1 through 5. The terminal at the bottom shows the command `PS C:\Users\parva\OneDrive\Desktop\AI Assisted> & C:\users\parva\appdata\local\Programs\Python\Python313\python.exe "c:/users/parva/OneDrive/Desktop/AI Assisted/lab ass 3.4.py"` and output indicating Task 1 - Fibonaci completed successfully.

```
lab ass 3.4.py
...
# task 1
def print_fibonacci(n: int) -> None:
    if n < 0:
        print("Please enter a positive integer.")
        return
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
    print()

# task 2
def reverse_list(lst: list) -> list:
    """Returns the reversed list."""
    return lst[::-1]

# task 3
def is_valid(s: str) -> bool:
    """Checks if string starts with capital letter and ends with period."""
    return len(s) > 0 and s[0].isupper() and s[-1] == "."
    if len(s) == 0:
        return False
    if s[0].isupper():
        for _ in range(1, len(s)):
            if s[_].islower():
                return False
    if s[-1] != ".":
        return False
    if s.count(".") != 1:
        return False
    local, domain = s.split("@")
    if len(local) == 0 or len(domain) == 0:
        return False
    if len(domain) == 0 or domain.count(".") < 1:
        return False
    # Check for consecutive dots
    for i in range(len(domain) - 1):
        if domain[i] == "." and domain[i + 1] == ".":
            return False
    parts = domain.split(".")
    for part in parts:
        if len(part) == 0:
            return False
    return True

# task 4
def sum_of_digits(n: int) -> int:
    return sum(int(digit) for digit in str(abs(n)))
    if n < 0:
        n *= -1
    # Test all functions
    print("Task 1 - Fibonacci:")
    print_fibonacci(10)
    print("Task 2 - Reverse List:")
    print(reverse_list([1, 2, 3]))
    print("Task 3 - Valid String:")
    print(is_valid("Hello."))
    print(is_valid("Hello."))
    print("Task 4 - Valid Email:")
    print(is_valid("user@example.com"))
    print(is_valid("user@example.com"))
    print("Task 5 - Sum of Digits:")
    print(sum_of_digits(23))  # 5
    print(sum_of_digits(405))  # 9
```

Screenshot of the AI Assisted IDE interface, identical to the one above but with a different set of code in the editor. The terminal shows the command `PS C:\Users\Varva\OneDrive\Desktop\AI Assisted>` and the output indicates Task 1 - Fibonaci completed successfully.

```
lab ass 3.4.py
...
# task 1
def print_fibonacci(n: int) -> None:
    if n < 0:
        raise ValueError("Please enter a positive integer.")
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
    print()

# task 2
def reverse_list(lst: list) -> list:
    """Returns the reversed list."""
    return lst[::-1]

# task 3
def is_valid(s: str) -> bool:
    """Checks if string starts with capital letter and ends with period."""
    return len(s) > 0 and s[0].isupper() and s[-1] == "."
    if len(s) == 0:
        return False
    if s[0].isupper():
        for _ in range(1, len(s)):
            if s[_].islower():
                return False
    if s[-1] != ".":
        return False
    if s.count(".") != 1:
        return False
    local, domain = s.split("@")
    if len(local) == 0 or len(domain) == 0:
        return False
    if len(domain) == 0 or domain.count(".") < 1:
        return False
    # Check for consecutive dots
    for i in range(len(domain) - 1):
        if domain[i] == "." and domain[i + 1] == ".":
            return False
    parts = domain.split(".")
    for part in parts:
        if len(part) == 0:
            return False
    return True

# task 4
def sum_of_digits(n: int) -> int:
    return sum(int(digit) for digit in str(abs(n)))
    if n < 0:
        n *= -1
    # Test all functions
    print("Task 1 - Fibonacci:")
    print_fibonacci(10)
    print("Task 2 - Reverse List:")
    print(reverse_list([1, 2, 3]))
    print("Task 3 - Valid String:")
    print(is_valid("Hello."))
    print(is_valid("Hello."))
    print("Task 4 - Valid Email:")
    print(is_valid("user@example.com"))
    print(is_valid("user@example.com"))
    print("Task 5 - Sum of Digits:")
    print(sum_of_digits(23))  # 5
    print(sum_of_digits(405))  # 9
```