

# AI ASSISTED CODING

Name : G. Sanjay

Hall Ticket No: 2303A510B9

Batch:13

Assignment - 3.4 Task-1. Zero-shot  
Prompt – Fibonacci Series Generator

Task Description :

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

Prompt :

```
# Write a Python code that prints the first N  
Fibonacci numbers
```

Code

```

Print first N Fibonacci numbers in - https://github.com/gandri/fib.py

fib.py
41 lines - 872 bytes

1  """
2  """
3  Print the first N Fibonacci numbers (starting from 0, 1, 1, 2, ...).
4
5  Usage:
6  - Run and enter N when prompted:
7      $ python fib.py
8      Enter N: 7
9      0 1 1 2 3 5 8
10     ...
11
12     from typing import List
13
14     def fibonacci(n: int) -> List[int]:
15         """Return a list containing the first n Fibonacci numbers (starting with 0)."""
16         if n <= 0:
17             return []
18         seq: List[int] = [0]
19         a, b = 0, 1
20         for _ in range(n):
21             seq.append(a)
22             a, b = b, a + b
23         return seq
24
25     def main() -> None:
26         try:
27             n: str = input("Enter N: ").strip()
28             n = int(n)
29         except ValueError:
30             print("Please enter a valid integer for N.")
31             return
32
33         nums = fibonacci(n)
34         if not nums:
35             print("No numbers to show (N <= 0).")
36         else:
37             # Print numbers separated by spaces
38             print(*nums)
39
40     if __name__ == "__main__":
41         main()

```

## Out Put:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\gandri\Documents\Year-2Sem\AI assistants Coding> & C:/Users/gandri/AppData/Local/Programs/Python/Python314/py
then, exec "C:/Users/gandri/Documents/Year-2Sem/AI assistants Coding/Ass-3.4.py"
Enter N: 7
0 1 1 2 3 5 8
○ PS C:\Users\gandri\Documents\Year-2Sem\AI assistants Coding>

```

## Justification:

In this task, Copilot was given only a single comment without any example. Despite the lack of context, it successfully generated a correct Fibonacci function, showing that zero-shot prompting works well for common and well-known problems. However, the logic was basic and depended entirely on Copilot's prior training. This demonstrates that zero-shot prompts are effective for simple, standard tasks but may lack customization or optimization.

## Task-2. One-shot Prompt – List Reversal Function

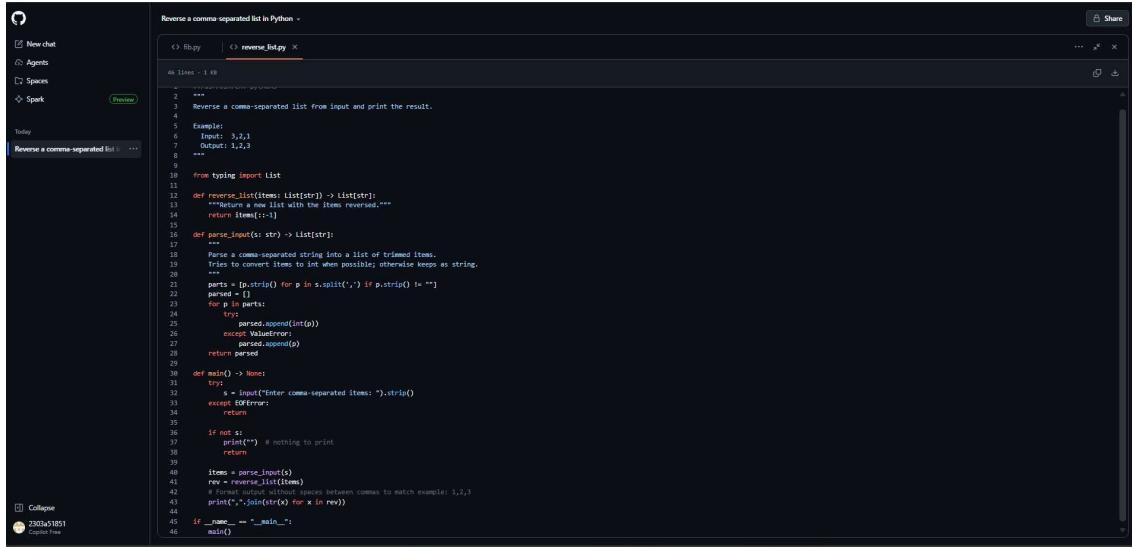
### Task Description #2

- Write a comment prompt to reverse a list and provide one

example below the comment to guide Copilot.

#Now Write a Python code for reverse a list Here Is the Example  
Input:3,2,1 Out Put is 1,2,3

## Code:



```
Reverse a comma-separated list in Python

reverse_list.py

46 Lines - 1 KB
Preview Share

1 #!/usr/bin/python
2 """
3     Reverse a comma-separated list from input and print the result.
4
5 Example:
6 Input: 3,2,1
7 Output: 1,2,3
8 """
9
10 from typing import List
11
12 def reverse_list(items: List[str]) -> List[str]:
13     """Return a new list with the items reversed."""
14     return items[::-1]
15
16 def parse_input(s: str) -> List[str]:
17
18     Parse a comma-separated string into a list of trimmed items.
19     Tries to convert items int when possible; otherwise keeps as string.
20
21     parts = [p.strip() for p in s.split(',') if p.strip() != ""]
22     parsed = []
23     for p in parts:
24         try:
25             parsed.append(int(p))
26         except ValueError:
27             parsed.append(p)
28     return parsed
29
30 def main() -> None:
31     try:
32         s = input("Enter comma-separated items: ").strip()
33     except EOFError:
34         return
35
36     if not s: # nothing to print
37         print("") # nothing to print
38         return
39
40     items = parse_input(s)
41     rev = reverse_list(items)
42     # Format output with one space between commas to match example: 1,2,3
43     print(','.join(str(x) for x in rev))
44
45 if __name__ == "__main__":
46     main()
```

## Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\gandr\Documents\3Year-25em\AI assistants coding & C:/Users/gandr/AppData/Local/Programs/Python/Python314/py
thon.exe "c:/Users/gandr/Documents/3Year-25em/AI assistants Coding/Ass-3.4.py"
● ps C:\Users\gandr\Documents\3Year-25em\AI assistants Coding & C:/Users/gandr/AppData/Local/Programs/Python/Python314/py
thon.exe "c:/Users/gandr/Documents/3Year-25em/AI assistants Coding/Ass-3.4.py"
Enter comma-separated items: 3,2,1
3,2,1
● ps C:\Users\gandr\Documents\3Year-25em\AI assistants Coding & C:/Users/gandr/AppData/Local/Programs/Python/Python314/py
thon.exe "c:/Users/gandr/Documents/3Year-25em/AI assistants Coding/Ass-3.4.py"
Enter comma-separated items: 3,4,5,6,7
7,6,5,4,3
● ps C:\Users\gandr\Documents\3Year-25em\AI assistants Coding & C:/Users/gandr/AppData/Local/Programs/Python/Python314/py
thon.exe "c:/Users/gandr/Documents/3Year-25em/AI assistants Coding/Ass-3.4.py"
7,6,5,4,3
Agent Auto ▾ 3.14.0 ↗ Go Live ↘ Go Live
```

## Justification:

By providing one example along with the comment, Copilot clearly understood the expected behavior of the function. The generated code accurately reversed the list using slicing or a loop. The example reduced ambiguity and improved correctness compared to a zero-shot approach.

This shows that even a single example can significantly enhance output quality.

## Task-3. Few-shot Prompt – String Pattern Matching

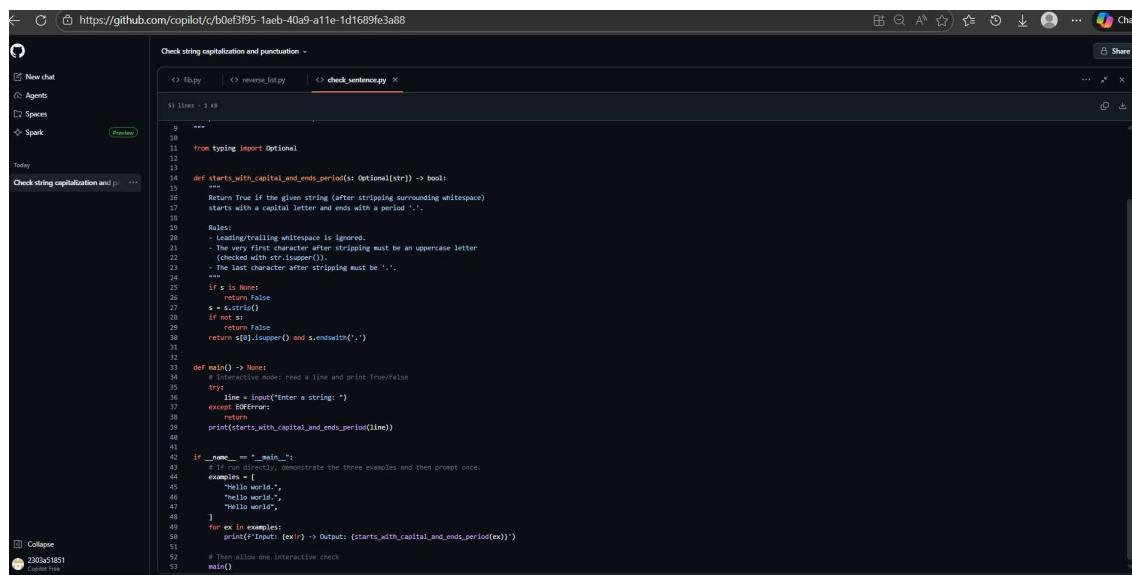
### Task Description #3

- 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.

### Prompt:

```
## Write a Python code that checks if a string starts with a capital letter and ends with a period
Example 1: # Input: "Hello world." # Output: True
# Example 2: # Input: "hello world." # Output: False
# Example 3: # Input: "Hello world" # Output: False
```

### Code:



```
Check string capitalization and punctuation

def starts_with_capital_and_ends_period(s: Optional[str]) -> bool:
    """
    Returns True if the given string (after stripping surrounding whitespace)
    starts with a capital letter and ends with a period '.'.

    Rules:
    - Leading/trailing whitespace is ignored.
    - The very first character after stripping must be an uppercase letter
      (checked with str.isupper()).
    - The last character after stripping must be '.'.

    If s is None, return False.
    If s is empty, return False.
    Otherwise, return s[0].isupper() and s.endswith('.')
    """
    if s is None:
        return False
    s = s.strip()
    if not s:
        return False
    return s[0].isupper() and s.endswith('.')

def main() -> None:
    # Interactive mode: read a line and print True/False
    try:
        line = input("Enter a string: ")
    except EOFError:
        return
    print(starts_with_capital_and_ends_period(line))

if __name__ == "__main__":
    # Optionally demonstrate the three examples and then prompt once.
    examples = [
        "Hello world.",
        "hello world.",
        "Hello world"
    ]
    for ex in examples:
        print(f"Input: {ex} > Output: {starts_with_capital_and_ends_period(ex)}")

    # Then allow one interactive check
    main()
```

### Output:



```
PS C:\Users\gandr\Documents\3year-2Sem\AI assistants Coding & C:\Users\gandr\AppData\Local\Programs\Python\Python314\python.exe "C:\Users\gandr\Documents\3year-2Sem\AI assistants Coding\Ass-3.4.py"
7,6,5,4,3
PS C:\Users\gandr\Documents\3year-2Sem\AI assistants Coding & C:\Users\gandr\AppData\Local\Programs\Python\Python314\python.exe "C:\Users\gandr\Documents\3year-2Sem\AI assistants Coding\Ass-3.4.py"
Input: 'Hello world.' > Output: True
Input: 'hello world.' > Output: False
Input: 'Hello world' > Output: False
Enter a string: The BIG bas
False
PS C:\Users\gandr\Documents\3year-2Sem\AI assistants coding>
```

## Justification:

With 2–3 examples, Copilot was able to identify the exact pattern requirements—starting with a capital letter and ending with a period. The generated `is_valid()` function was more precise and handled conditions correctly. Few-shot prompting guided Copilot to follow strict rules rather than guessing. This highlights how multiple examples help in pattern-based logic.

## 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.

### Prompt: Zero Short

```
# Write a Python function that validates if a string is a valid email address
```

### Code:

```

# This function validates an email address.
# prompt: Generate the function body
def is_valid_email_zero_shot(email_address):
    # Regular expression for a basic email validation
    # It checks for: characters before @, @ symbol, characters before ., . symbol, and characters after .
    # This pattern is a common and basic way to validate email structure.
    pattern = r"^[a-zA-Z0-9._%]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
    if re.match(pattern, email_address):
        return True
    else:
        return False

# Test email address
test_email = "test@example.com"

# Call the function and store the result
validation_result = is_valid_email_zero_shot(test_email)

# Print the test email and the validation result
print(f"Email: {test_email}, Is Valid: {validation_result}")

```

## Output:

```

Enter a string: The Big Bad Wolf
False

```

## Code: Few -Short:

```

# Check string capitalization and punctuation -
# https://www.geeksforgeeks.org/python-validate-email-address/
# Validate whether a string is a valid email address.
# This uses a conservative regular expression that accepts common valid emails like:
# - ten@example.com
# - ten@example.co
# and rejects obvious invalid ones:
# - Invalid-mail
# - Invalid-mail.
# - Invalid-mail@ (no top-level domain)
# Note: This is not a full RFC-5322 validator (these are very complex). It covers typical practical use.
# Return True if s is a valid email address by a practical regex, else False.
# Rules enforced:
# 1. Starts with letters, digits or + characters (not starting with checked).
# 2. Domain part allows letters, digits, hyphens and dots, and must contain a final dot +.
# 3. If no dot + then returns False.
# 4. If not s or not instanceof(s, str):
# 5. s is not None
# 6. s is not ''
# 7. Returns False
# return instanceof(EMAIL_REG_EXP, s)

# If name_ == "main":
#     examples = [
#         ("test@example.com", True),
#         ("test@example", False),
#         ("@example.com", False),
#         ("example.com", False),
#         ("", False)
#     ]
#     for email, expected in examples:
#         print(f"Input: {email} | Output: {is_valid_email(email)} | Expected: {expected}")
#     print("Input: (email) | Output: (result) | Expected: (expected))")

if __name__ == "__main__":
    examples = [
        ("test@example.com", True),
        ("test@example", False),
        ("@example.com", False),
        ("example.com", False),
        ("", False)
    ]
    for email, expected in examples:
        print(f"Input: {email} | Output: {is_valid_email(email)} | Expected: {expected}")
    print("Input: (email) | Output: (result) | Expected: (expected))")

```

## Out Put:

```

PS C:\Users\gandr\Documents\Year-2Sem\AI assistants Coding> & c:/users/gandr/appdata/local/programs/python/python314/python.exe "c:/users/gandr/documents/year-2sem/ai assistants coding/Ass-3.4.py"
NameError: name 'is_valid_email_zero_shot' is not defined. Did you mean: 'is_valid_email_few_shot'?
PS C:\Users\gandr\Documents\Year-2Sem\AI assistants Coding> & c:/users/gandr/appdata/local/programs/python/python314/python.exe "c:/users/gandr/documents/year-2sem/ai assistants coding/Ass-3.4.py"
Input: test@example.com
Output: True (expected: True)
Input: hello.world@domain.co
Output: True (expected: True)
Input: Invalid-email!
Output: False (expected: False)
Input: user@domain
Output: False (expected: False)

```

## Justification:

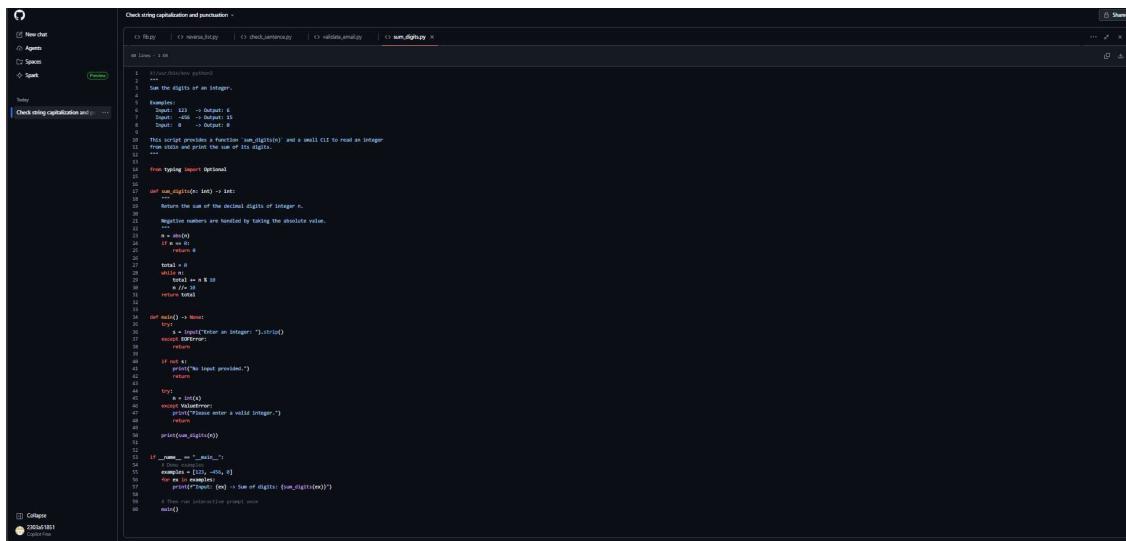
The zero-shot prompt resulted in a simple email check with minimal validation logic. In contrast, the few-shot prompt produced a more robust function that checked for proper structure such as @ symbol and domain format. The comparison shows that few-shot prompting improves reliability and reduces logical gaps. Examples help Copilot understand edge cases better.

## 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 Code:



```

Check string capitalization and punctuation ->
New chat Agents Spaces Spots Start (Previous) Today Check string capitalization and punctuation -> ...
1 #!/usr/bin/python3
2 ...
3 # Sum the digits of an integer.
4
5 def sum_digits(n):
6     """Sum the digits of integer n.
7     Input: 123 -> Output: 6
8     Input: -123 -> Output: 6
9     Input: 0 -> Output: 0
10
11     This script provides a function 'sum_digits()' and a small CLI to read an integer
12     from stdio and print the sum of its digits.
13
14     Note: Negative numbers are handled by taking the absolute value.
15
16     n = abs(n)
17     if n == 0:
18         return 0
19
20     total = 0
21     while n > 0:
22         total += n % 10
23         n /= 10
24     return total
25
26
27 def main() -> None:
28     try:
29         n = int(input("Enter an integer: "))
30     except ValueError:
31         print("Error: Please enter a valid integer")
32         return
33
34
35     if n < 0:
36         print("No input provided.")
37     else:
38         try:
39             n = int(n)
40         except ValueError:
41             print("Please enter a valid integer")
42             return
43
44         print(sum_digits(n))
45
46         if __name__ == "__main__":
47             examples = [(123, 6), (-123, 6)]
48             for n, expected in examples:
49                 print(f"Input: {n} -> Sum of digits: {sum_digits(n)}")
50
51     # Run user interactive prompt once
52     main()

```

Out Put:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding & C:/Users/gandr/AppData/Local/Programs/Python/python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/Ass-3.4.py"
5
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding & C:/Users/gandr/AppData/Local/Programs/Python/python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/Ass-3.4.py"
Input: 123 -> Sum of digits: 6
Input: -456 -> Sum of digits: 15
Input: 0 -> Sum of digits: 0
Enter an integer: 345
126
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding

```

# Prompt:

```
#Ok Now Write a Python code to Summing Digits of a  
NumberInput: 123 -> Sum of digits: 6 Input:  
-456 -> Sum of digits: 15 Input: 0 -> Sum of digits:  
0 Enter an integer: 345 12
```

Code:

## Out Put:

The screenshot shows a terminal window with several tabs open, all related to 'Year-2Sem\AI assistants Coding'. The tabs include:

- PS C:\Users\gandr\Documents\Year-2Sem\AI assistants Coding> `%`
- PS C:\Users\gandr\Documents\Year-2Sem\AI assistants Coding> `%`
- PS C:\Users\gandr\Documents\Year-2Sem\AI assistants Coding>
- PS C:\Users\gandr\Documents\Year-2Sem\AI assistants Coding> & C:/Users/gandr/AppData/Local/Programs/Python/python314/python.exe "c:/users/gandr/documents/year-2sem/ai assistants coding/Ass-1.4.py"

The command `& C:/Users/gandr/AppData/Local/Programs/Python/python314/python.exe "c:/users/gandr/documents/year-2sem/ai assistants coding/Ass-1.4.py"` has been run, and its output is displayed:

```
Input: 123 -> sum of digits: 6
Input: -456 -> sum of digits: 15
Input: 0 -> sum of digits: 0
Enter an integer: 234
9
```

The terminal also shows a sidebar with 'OUTLINE' and 'TIMELINE' sections, and a bottom status bar indicating 'Ln 55, Col 11'.

## Justification:

The generic prompt generated a working but straightforward solution. When input/output examples

were added, Copilot produced cleaner, more readable, and more accurate code. The example clarified expectations and improved function structure. This confirms that prompt tuning with examples leads to better-quality implementations.