

## Assignment-2.4

### Task 1: Zero-shot Prompt – Fibonacci Series Generator

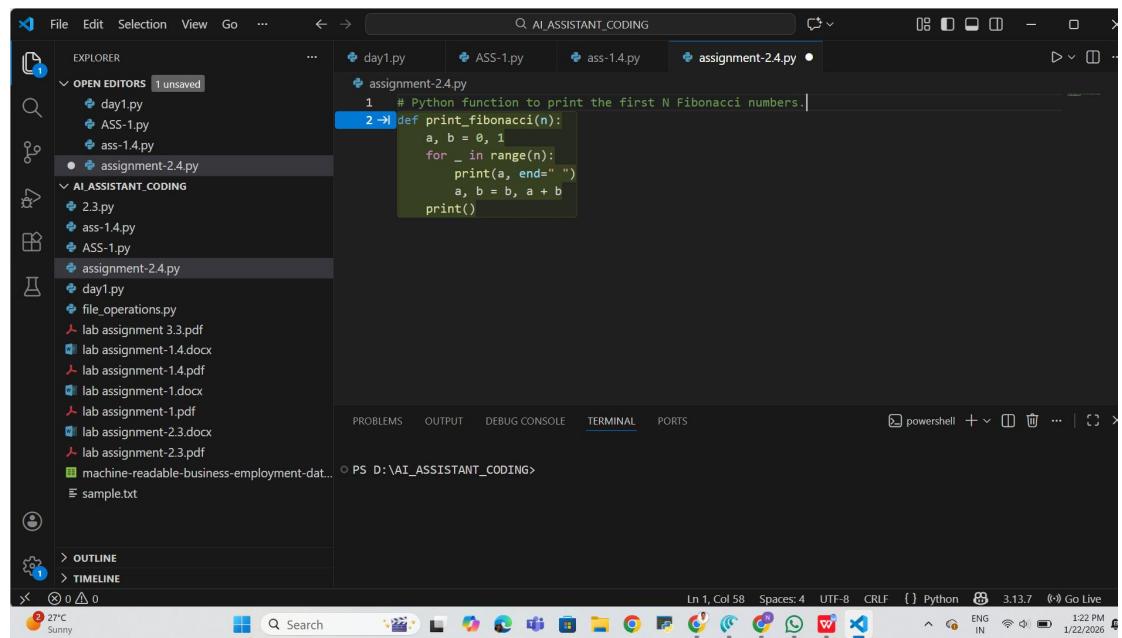
#### Task Description #1

- 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

#### PROMPT:



```
1  # Python function to print the first N Fibonacci numbers.
2  def print_fibonacci(n):
3      a, b = 0, 1
4      for _ in range(n):
5          print(a, end=" ")
6          a, b = b, a + b
7      print()
```

#### CODE:

```
def fibonacci_sequence(n):
    """
    Generates the first N Fibonacci numbers.

    Args:
        n: Integer, number of Fibonacci numbers to generate

    Returns:
        List of first N Fibonacci numbers
    """
    fib_sequence = []
    a, b = 0, 1
    for _ in range(n):
        fib_sequence.append(a)
        a, b = b, a + b
    return fib_sequence
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
```

```

        return True
        return True
# Test the function
num = int(input("Enter the number of Fibonacci numbers to generate: "))
fib_numbers = fibonacci_sequence(num)
print(f"The first {num} Fibonacci numbers are: {fib_numbers}")

```

OUTPUT :

The screenshot shows a terminal window with the following content:

```

lab assignment-1.4.pdf
lab assignment-1.docx
lab assignment-1.pdf
lab assignment-2.3.docx
lab assignment-2.3.pdf
machine-readable-business-employment-dat...
sample.txt

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achandi sai teja/AppData/Local/Programs/Python/Python313/python.exe" ./AI_ASSISTANT_CODING/assignment-2.4.py
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achandi sai teja/AppData/Local/Programs/Python/Python313/python.exe" ./AI_ASSISTANT_CODING/assignment-2.4.py
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achandi sai teja/AppData/Local/Programs/Python/Python313/python.exe" ./AI_ASSISTANT_CODING/assignment-2.4.py
Enter the number of Fibonacci numbers to generate: 7
The first 7 Fibonacci numbers are: [0, 1, 1, 2, 3, 5, 8]
PS D:\AI_ASSISTANT_CODING>

```

The terminal window also shows icons for powershell and Python.

### 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:

The screenshot shows a code editor with the following content:

```

2.3.py
ass-1.4.py
ASS-1.py
assignment-2.4.py
day1.py
file_operations.py
lab assignment 3.3.pdf

26     fib_numbers = fibonacci_sequence(num)
27     print(f"The first {num} Fibonacci numbers are: {fib_numbers}")
28
29     # Reverse a list by using an example of your choice.

```

CODE:

```

# Reverse a list by using an example of your choice
def reverse_list(input_list):
    """
    Reverses the given list.

    Args:
        input_list: List of elements to be reversed

    Returns:
        List: Reversed list
    """
    return input_list[::-1]

# Test the reverse_list function
sample_list = [1, 2, 3]
reversed_list = reverse_list(sample_list)
print(f"Original list: {sample_list}")
print(f"Reversed list: {reversed_list}")

```

## OUTPUT:



The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
Original list: [1, 2, 3, 4, 5]
Reversed list: [5, 4, 3, 2, 1]
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
Original list: [1, 2, 3]
Reversed list: [3, 2, 1]
PS D:\AI_ASSISTANT_CODING>
```

Below the terminal, the status bar shows: Line 45, Col 41 (462 selected), Spaces: 4, UTF-8, CR/LF, Python, 3.13.7, Go Live.

Adding **just one clear example** helps Copilot quickly infer:

what the function should do

the expected input and output format

This improves **accuracy and reduces ambiguity**, even with a minimal prompt.

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

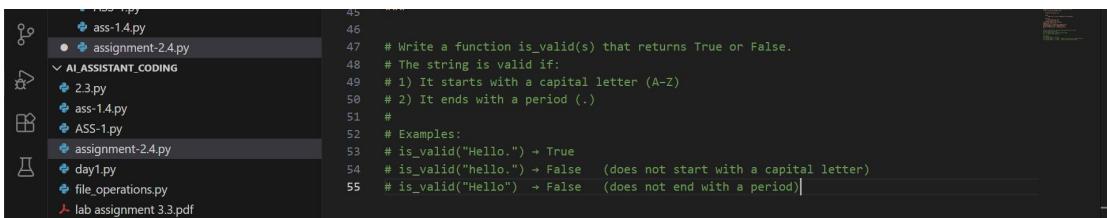
### Task Description #3

- 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:



The screenshot shows a code editor with the following file structure on the left:

- Assignment-2.4
- ass-1.4.py
- assignment-2.4.py
- AI\_ASSISTANT\_CODING
- 2.3.py
- ass-1.4.py
- ASS-1.py
- assignment-2.4.py
- day1.py
- file\_operations.py
- lab assignment 3.3.pdf

The `assignment-2.4.py` file contains the following code:

```
45
46
47 # Write a function is_valid(s) that returns True or False.
48 # The string is valid if:
49 # 1) It starts with a capital letter (A-Z)
50 # 2) It ends with a period (.)
51 #
52 # Examples:
53 # is_valid("Hello.") → True
54 # is_valid("Hello.") → False (does not start with a capital letter)
55 # is_valid("Hello") → False (does not end with a period)
```

## CODE:

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

    Args:
        s: Input string to validate
    Returns:
        Boolean: True if valid, False otherwise
    """
    if len(s) == 0:
        return False
    return s[0].isupper() and s[-1] == '.'

# Test the is_valid function
test_strings = ["Hello.", "hello.", "Hello", ""]
for string in test_strings:
    result = is_valid(string)
    print(f'is_valid("{string}") → {result}')
```

## OUTPUT:

The screenshot shows a terminal window with the following content:

```
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
is_valid("Hello.") → True
is_valid("Hello.") → False
is_valid("") → False
is_valid("") → False
```

Providing **multiple examples in comments** helps Copilot clearly understand:

what counts as valid

what should be rejected

This leads to **more accurate and relevant code generation**.

## 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:

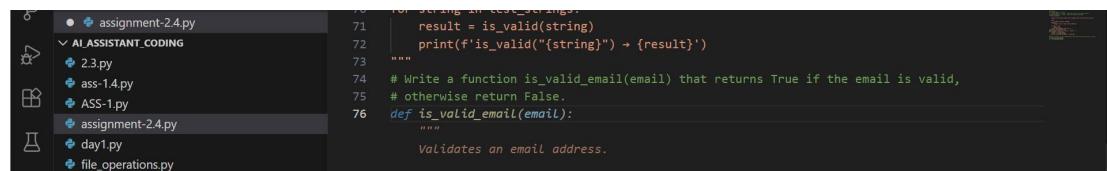
Zero-shot may result in basic or generic validation.

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

checking).

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

#### PROMPT:



The screenshot shows a code editor interface with a sidebar containing a file tree. The tree includes files like assignment-2.4.py, 2.3.py, ass-1.4.py, ASS-1.py, and day1.py. The main pane displays a snippet of Python code:

```
for string in test_strings:
    result = is_valid(string)
    print(f'is_valid("{string}") -> {result}')

"""
# Write a function is_valid_email(email) that returns True if the email is valid,
# otherwise return False.
def is_valid_email(email):
    """
    Validates an email address.
```

#### CODE:

```
def is_valid_email(email):
    """
    Validates an email address based on basic criteria.

    Args:
        email: String, email address to validate

    Returns:
        Boolean: True if valid, False otherwise
    """
    # Basic checks for presence of '@' and '.'
    if "@" not in email or "." not in email:
        return False

    # Split the email into local and domain parts
    local_part, domain_part = email.split("@", 1)

    # Local part should not be empty
    if len(local_part) == 0:
        return False

    # Domain part should contain at least one dot
    if "." not in domain_part:
        return False

    # Domain part should not start or end with a dot
    if domain_part.startswith(".") or domain_part.endswith("."):
        return False

    return True

# Test the is_valid_email function
test_emails = ["test@example.com", "invalid.email", "another@domain.org",
"no_at_symbol.com"]
for email in test_emails:
    result = is_valid_email(email)
    print(f'is_valid_email("{email}") -> {result}')
```

#### OUTPUT:

```

lab assignment-2.3.docx
lab assignment-2.3.pdf
machine-readable-business-employment-dat...
sample.txt

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achandi sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
ms/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
is_valid_email("test@example.com") -> True
is_valid_email("invalid_email") -> False
is_valid_email("another@domain.org") -> True
is_valid_email("no_at_symbol.com") -> False
PS D:\AI_ASSISTANT_CODING>

```

In 77, Col 1 (1091 selected) Spaces: 4 UTF-8 CRLF {} Python 3.13.7 FNG

## PROMPT:

```

Ass-1.py
ass-1.4.py
assignment-2.4.py
AI ASSISTANT CODING
2.3.py
ass-1.4.py
ASS-1.py
assignment-2.4.py
day1.py
file_operations.py
lab assignment-3.3.pdf
lab assignment-1.4.docx

112
113     # Write a function is_valid_email(email) that checks whether an email is valid.
114     # Rules:
115     # - Must contain exactly one '@'
116     # - Must have characters before and after '@'
117     # - Must end with a valid domain containing '.'
118     #
119     # Examples:
120     # is_valid_email("user@example.com") -> True
121     # is_valid_email("user@example.com") -> False
122     # is_valid_email("user@com") -> False
123

```

## CODE:

```

def is_valid_email(email):
    if email.count("@") != 1:
        return False

    local, domain = email.split("@")
    if not local or "." not in domain:
        return False

    return True
# Test the is_valid_email function
test_emails = ["user@example.com", "userexample.com", "user@com"]
for email in test_emails:
    result = is_valid_email(email)
    print(f'is_valid_email("{email}") -> {result}')

```

## OUTPUT:

```

lab assignment-2.3.docx
lab assignment-2.3.pdf
machine-readable-business-employment-dat...
sample.txt

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achandi sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
ms/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
is_valid_email("no_at_symbol.com") -> False
is_valid_email("user@example.com") -> True
is_valid_email("userexample.com") -> False
is_valid_email("user@com") -> False
PS D:\AI_ASSISTANT_CODING>

```

Few-shot prompting **guides Copilot's reasoning** by showing:

valid vs invalid patterns

edge cases

This results in **more reliable, accurate, and meaningful code** compared to zero-shot prompts.

## PROMPT:

```
134     result = is_valid_email(email)
135     print(f'is_valid_email("{email}") -> {result}'))"""
136
137 # Write a function sum_of_digits(n) that returns the sum of digits of a number in user input
138 def sum_of_digits(n):
139     """
```

CODE:

```
def sum_of_digits(n):
    """
    Calculates the sum of digits of a given number.

    Args:
        n: Integer whose digits are to be summed

    Returns:
        Integer: Sum of the digits
    """
    return sum(int(digit) for digit in str(abs(n)))

# Test the sum_of_digits function
number = int(input("Enter a number to calculate the sum of its digits: "))
result = sum_of_digits(number)
print(f"The sum of the digits of {number} is: {result}")
```

OUTPUT:

```
is_valid_email("userexample.com") -> False
is_valid_email("user@com") -> False
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
Enter a number to calculate the sum of its digits: 123
The sum of the digits of 123 is: 6
PS D:\AI_ASSISTANT_CODING>
```

PROMPT:

```
154     """
155     # Write a function sum_of_digits(n) that returns the sum of all digits in a number.
156     # Example:
157     # sum_of_digits(123) -> 6
158
159     def sum_of_digits(n):
160         return sum(int(digit) for digit in str(abs(n)))
```

CODE:

```
155     # Write a function sum_of_digits(n) that returns the sum of all digits in a number.
156     # Example:
157     # sum_of_digits(123) -> 6
158     def sum_of_digits(n):
159         return sum(int(digit) for digit in str(abs(n)))
160
161     # Test the sum_of_digits function
162     test_numbers = [123]
163     for number in test_numbers:
164         result = sum_of_digits(number)
165         print(f"sum_of_digits({number}) -> {result}")
```

OUTPUT:

```
is_valid_email("userexample.com") -> False
is_valid_email("user@com") -> False
PS D:\AI_ASSISTANT_CODING> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
Enter a number to calculate the sum of its digits: 123
The sum of the digits of 123 is: 6
PS D:\AI_ASSISTANT_CODING>
```

