

Assignment-3.4

Task 1: Zero-shot Prompt – Fibonacci Series Generator

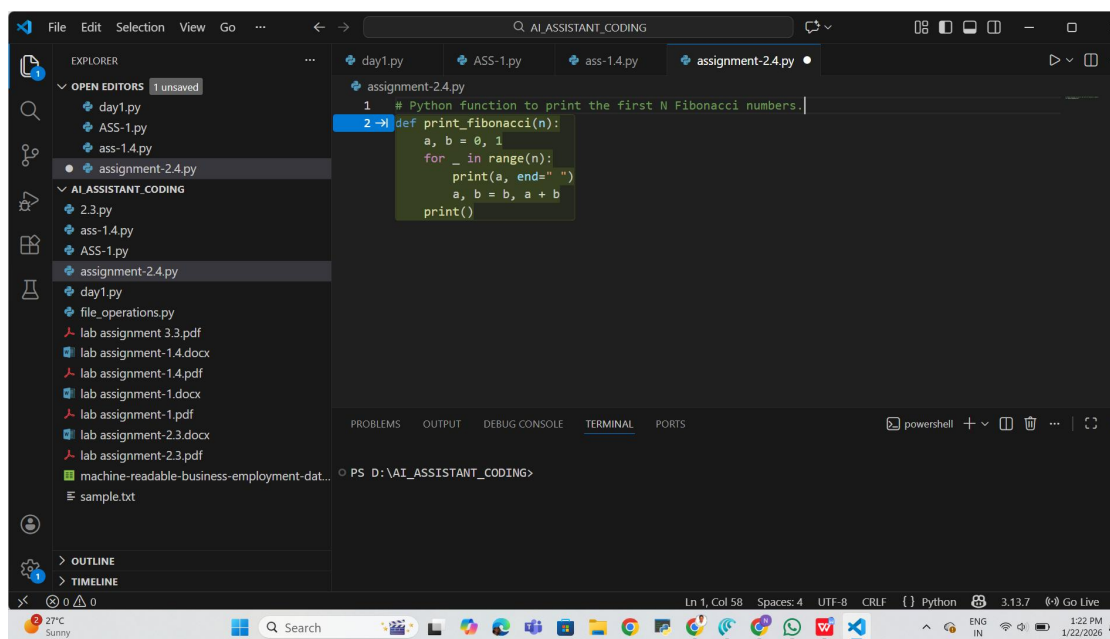
Task Description #1

- 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 → 0 1 1 2 3 5 8
- Observation on how Copilot understood the instruction with zero Context

PROMPT:



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
```

```

13     a, b = 0, 1
14     for _ in range(n):
15         fib_sequence.append(a)
16         a, b = b, a + b
17     return fib_sequence
18     for i in range(2, int(n ** 0.5) + 1):
19         if n % i == 0:
20             return False
21
22     return True
23     return True
24 # Test the function
25 num = int(input("Enter the number of Fibonacci numbers to generate: "))
26 fib_numbers = fibonacci_sequence(num)
27 print(f"The first {num} Fibonacci numbers are: {fib_numbers}")
28

```

OUTPUT :

```

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
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 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>

```

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:

```

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:

```

29 #Reverse a list by using an example of your choice
30 """def reverse_list(input_list):
31
32     Reverses the given list.
33
34     Args:
35         input_list: List of elements to be reversed
36
37     Returns:
38         List: Reversed list
39     return input_list[::-1]
40 # Test the reverse_list function
41 sample_list = [1, 2, 3]
42 reversed_list = reverse_list(sample_list)
43 print(f"Original list: {sample_list}")
44 print(f"Reversed list: {reversed_list}")
45 """

```

OUTPUT:

```

PS D:\VAI_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:\VAI_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:\VAI_ASSISTANT_CODING>

```

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

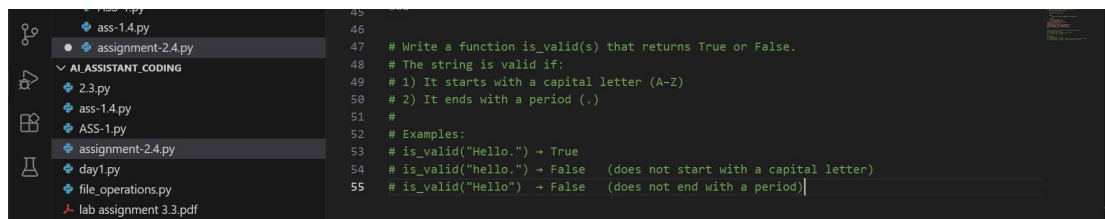
Task Description #3

- 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

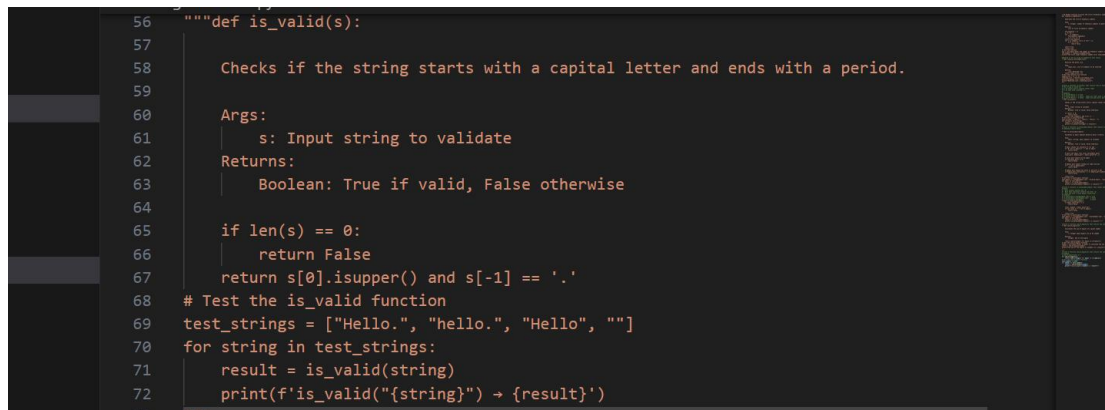
- A function `is_valid()` that checks the pattern.
- Output: True or False based on input.
- Students reflect on how multiple examples guide Copilot to generate more accurate code.

PROMPT:



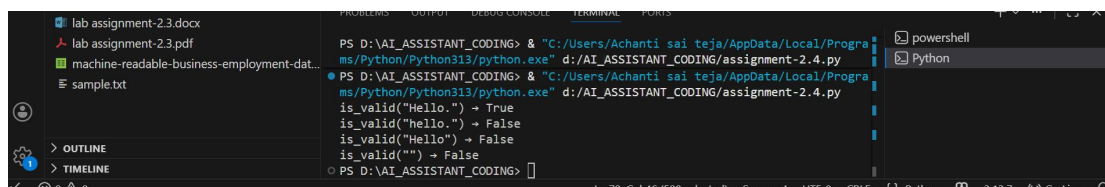
```
45 # Write a function is_valid(s) that returns True or False.
46 # The string is valid if:
47 # 1) It starts with a capital letter (A-Z)
48 # 2) It ends with a period (.)
49 #
50 # Examples:
51 # is_valid("Hello.") → True
52 # is_valid("hello.") → False (does not start with a capital letter)
53 #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

CODE:



```
56 """def is_valid(s):
57     Checks if the string starts with a capital letter and ends with a period.
58
59     Args:
60         s: Input string to validate
61     Returns:
62         Boolean: True if valid, False otherwise
63
64     if len(s) == 0:
65         return False
66     return s[0].isupper() and s[-1] == '.'
67
68 # Test the is_valid function
69 test_strings = ["Hello.", "hello.", "Hello", ""]
70 for string in test_strings:
71     result = is_valid(string)
72     print(f'is_valid("{string}") → {result}')
```

OUTPUT:



```
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
ms/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py
is_valid("Hello.") → True
is_valid("hello.") → False
is_valid("Hello") → 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

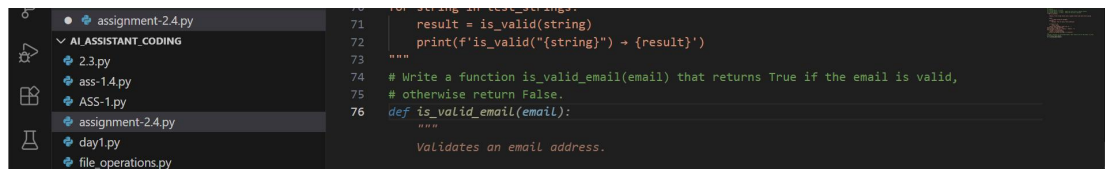
- Compare both outputs:

Zero-shot may result in basic or generic validation.

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

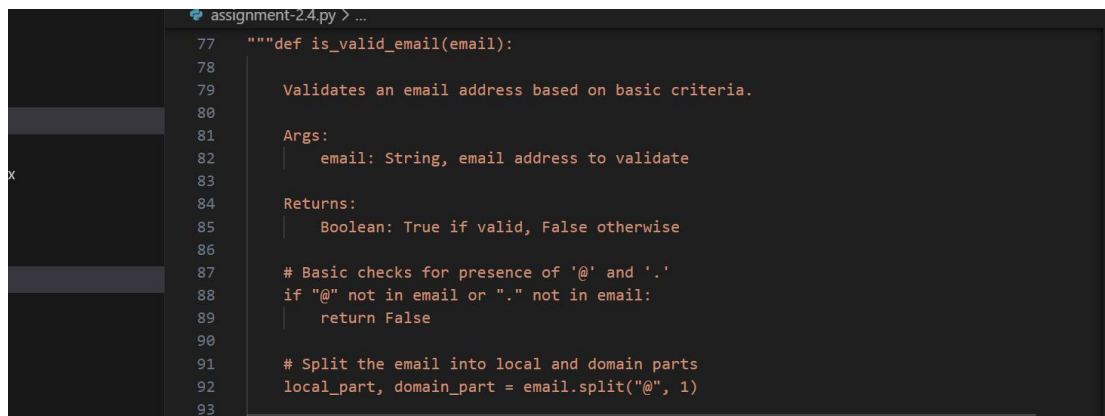
- Submit both code versions and note how few-shot improves reliability.

PROMPT:

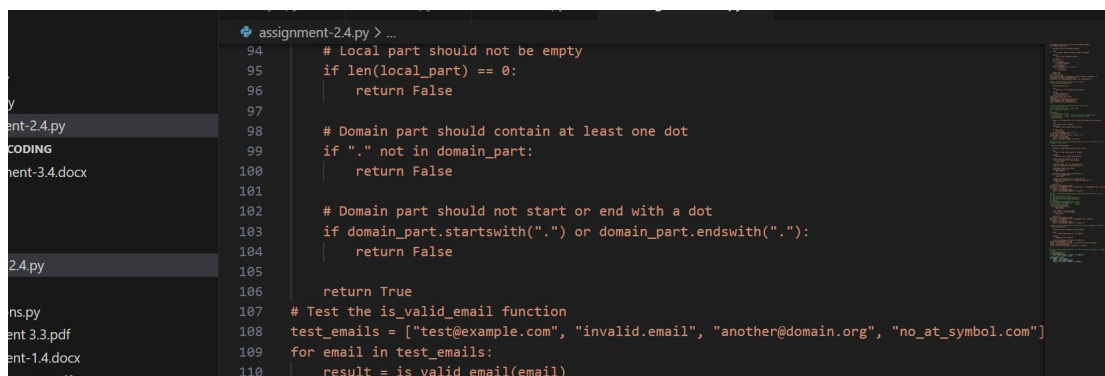


```
70 for string in test_strings:
71     result = is_valid(string)
72     print(f'is_valid("{string}") -> {result}')
73 """
74 # Write a function is_valid_email(email) that returns True if the email is valid,
75 # otherwise return False.
76 def is_valid_email(email):
77     """
78     Validates an email address.
```

CODE:

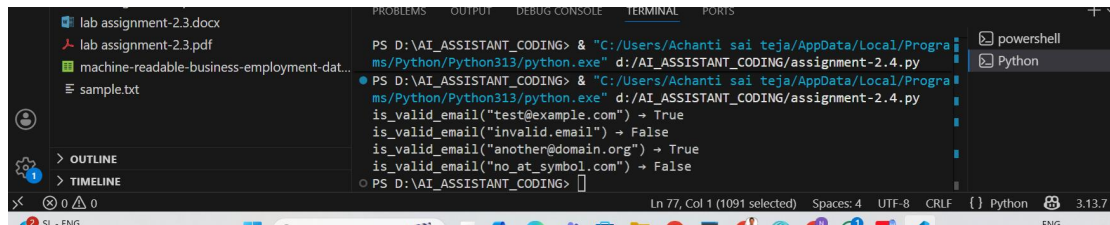


```
77 """def is_valid_email(email):
78
79     Validates an email address based on basic criteria.
80
81     Args:
82         email: String, email address to validate
83
84     Returns:
85         Boolean: True if valid, False otherwise
86
87     # Basic checks for presence of '@' and '.'
88     if "@" not in email or "." not in email:
89         return False
90
91     # Split the email into local and domain parts
92     local_part, domain_part = email.split("@", 1)
93
```



```
94     # Local part should not be empty
95     if len(local_part) == 0:
96         return False
97
98     # Domain part should contain at least one dot
99     if "." not in domain_part:
100         return False
101
102     # Domain part should not start or end with a dot
103     if domain_part.startswith(".") or domain_part.endswith("."):
104         return False
105
106     return True
107
108 # Test the is_valid_email function
109 test_emails = ["test@example.com", "invalid.email", "another@domain.org", "no_at_symbol.com"]
110 for email in test_emails:
111     result = is_valid_email(email)
```

OUTPUT:



PROMPT:



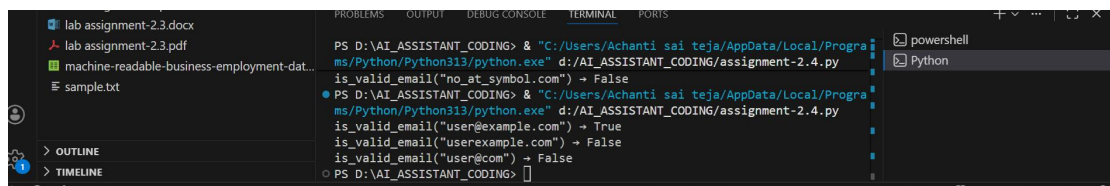
CODE:

```
"""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:



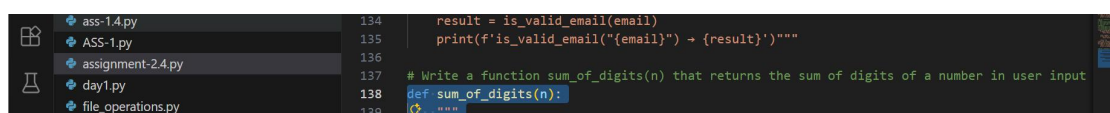
Few-shot prompting **guides Copilot's reasoning** by showing:

valid vs invalid patterns

edge cases

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

PROMPT:



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:

```
sample.txt  
is_valid_email("user@example.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:

```
day1.py  
ASS-1.py  
ass-1.4.py  
assignment-2.4.py  
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
```

CODE:

```
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  
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 # Test the sum_of_digits function  
161 test_numbers = [123]  
162 for number in test_numbers:  
163     result = sum_of_digits(number)  
164     print(f"sum_of_digits({number}) → {result}")  
165
```

OUTPUT:

```
sample.txt  
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> & "C:/Users/Achanti sai teja/AppData/Local/Programs/Python/Python313/python.exe" d:/AI_ASSISTANT_CODING/assignment-2.4.py  
sum_of_digits(123) → 6  
PS D:\AI_ASSISTANT_CODING>
```

