

ASSIGNMENT – 3.4

2303A51850

BATCH – 13

Task 1: Zero-shot Prompt – Fibonacci Series Generator

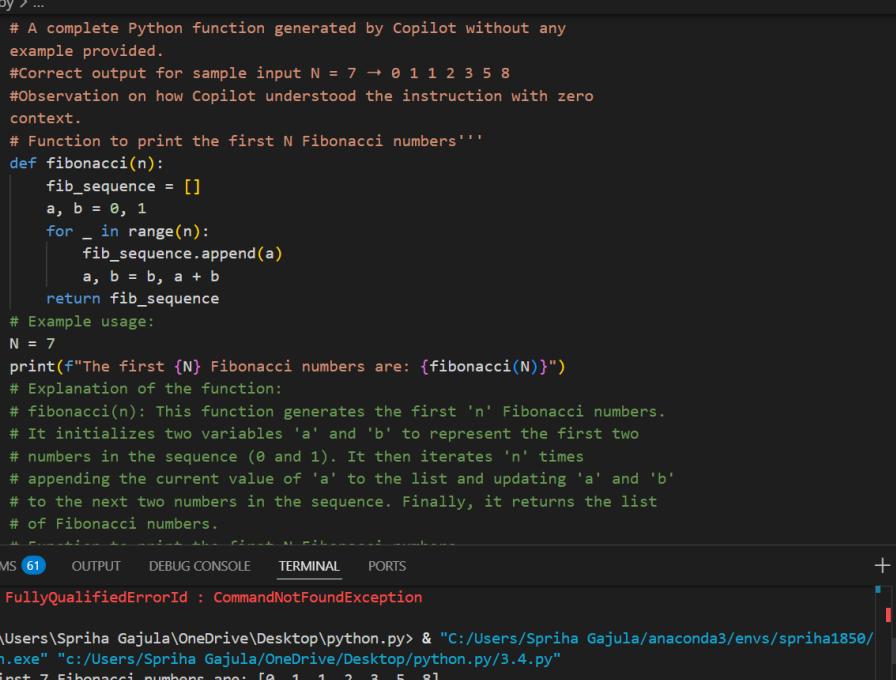
Task Description #1

- 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.

CODE:



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

- File Tabs:** The tabs at the top include: .. .py, dsa1.py, day9hmwk.py, day10.py, day11.py, aicoding3.1.py 9+, 3.4.py, and 3.4.py > ...
- Code Content:** The code is a Python script named `python.py` that defines a function `fibonacci(n)` to generate the first `n` Fibonacci numbers. It includes a docstring explaining the function's purpose, logic, and variable names.
- Terminal Output:** Below the code editor, the terminal output shows three command-line executions of the script, all resulting in the same output: `[0, 1, 1, 2, 3, 5, 8]`.

JUSTIFICATION:

`fibonacci(n)`: This function generates the first 'n' Fibonacci numbers.

It initializes two variables 'a' and 'b' to represent the first two numbers in the sequence (0 and 1). It then iterates 'n' times appending the current value of 'a' to the list and updating 'a' and 'b' to the next two numbers in the sequence. Finally, it returns the list of Fibonacci numbers.

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.

CODE:

```
34
35     '''Task 2: One-shot Prompt List Reversal Function
36     Task Description #2
37     • Write a comment prompt to reverse a list and provide one
38     example below the comment to guide Copilot.
39     Expected Output #2
40     • Copilot-generated function to reverse a list using slicing or loop.
41     • Output: [3, 2, 1] for input [1, 2, 3]
42     • Observation on how adding a single example improved Copilot's
43     accuracy.
44     # Function to reverse a list...
45     def reverse_list(lst):
46         return lst[::-1]
47     # Example usage:
48     input_list = [1, 2, 3]
49     reversed_list = reverse_list(input_list)
50     print(f"The reversed list of {input_list} is: {reversed_list}")
51     # Explanation of the function:
52     # reverse_list(lst): This function takes a list 'lst' as input and
53     # returns a new list that is the reverse of the input list using slicing.
54     # Function to reverse a list
55
```

PROBLEMS 61 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"

- PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
The reversed list of [1, 2, 3] is: [3, 2, 1]
- PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
The reversed list of [1, 2, 3] is: [3, 2, 1]
- PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py

Ln 50, Col 55 Spaces: 4 UTF-8 CRLF {} Python 3.12.9 (spriha1850) (↻) Go Live 🔍

JUSTIFICATION:

`reverse_list(lst)`: This function takes a list 'lst' as input and returns a new list that is the reverse of the input list using slicing.

Function to reverse a list Function to compute the factorial of a given number

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.

```

python.py → python.py
py      ↗ dsa1.py    ↗ day9hmwk.py    ↗ day10.py    ↗ day11.py    ↗ aicoding3.1.py 9+    ↗ 3.4.py    X    ▶    ⌂    ...
3.4.py > ...
53     # Output: True or False based on input.
54
55     # Students reflect on how multiple examples guide Copilot to generate more accurate code
56
57     # Function to check if a string starts with a capital letter and ends with a period ''
58     def is_valid(s):
59         return s[0].isupper() and s.endswith('.')
60
61     # Example usage:
62     test_strings = [
63         "Hello world.",
64         "hello world.",
65         "Hello world",
66         "Python is great."
67     ]
68
69     for string in test_strings:
70         print(f"Is the string '{string}' valid? {is_valid(string)}")
71
72     # Explanation of the function:
73     # is_valid(s): This function checks if the input string 's' starts with a capital letter
74     # by using the isupper() method on the first character and checks if it ends with a
75     # period using the endswith() method. It returns True if both conditions are met,
76     # otherwise it returns False.
77
78     # Function to check if a string starts with a capital letter and ends with a period
79     # Function to compute the factorial of a given number
80
81
82
83
84
85
86

```

PROBLEMS 61 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
● PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
Is the string 'Hello world.' valid? True
Is the string 'hello world.' valid? False
Is the string 'Hello world' valid? False
Is the string 'Python is great.' valid? True
○ PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py>

```

Ln 74, Col 2 Spaces: 4 UTF-8 CRLF { } Python 3.12.9 (spriha1850) (→) Go Live 🔍

JUSTIFICATION :

`is_valid(s)`: This function checks if the input string 's' starts with a capital letter by using the `isupper()` method on the first character and checks if it ends with a period using the `endswith()` method. It returns True if both conditions are met, otherwise it returns False.

Function to check if a string starts with a capital letter and ends with a period

Function to compute the factorial of a given number

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.

CODE:

Zero-shot Prompting (Email Validator)

The screenshot shows a code editor with a Python file named `3.4.py`. The code implements a function to validate email addresses using regular expressions. It prompts the user for an email address, prints a validation message, and handles invalid inputs. The terminal below shows the script being run and a valid email being entered.

```
3.4.py > ...
92     Few-shot may result in basic or generic validation.
93     Few-shot gives detailed and specific logic (e.g., @ and domain
94     checking).
95     • Submit both code versions and note how few-shot improves
96     reliability.''
97 #Zero-shot Prompting (Email Validator)
98 import re
99
100 def is_valid_email(email):
101     pattern = r'^[a-zA-Z0-9_.%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
102     return re.match(pattern, email) is not None
103
104 # User input
105 email = input("Enter an email address: ")
106
107 # Validation result
108 if is_valid_email(email):
109     print("Valid email address")
110 else:
111     print("Invalid email address")
112
113
114
115
```

PROBLEMS 61 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Spriha Gajula\OneDrive\Desktop> python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
● PS C:\Users\Spriha Gajula\OneDrive\Desktop> python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
Enter an email address: Sprihagajula@gmail.com
Valid email address
● PS C:\Users\Spriha Gajula\OneDrive\Desktop> python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
Enter an email address: 506@gmail.com
```

JUSTIFICATION :

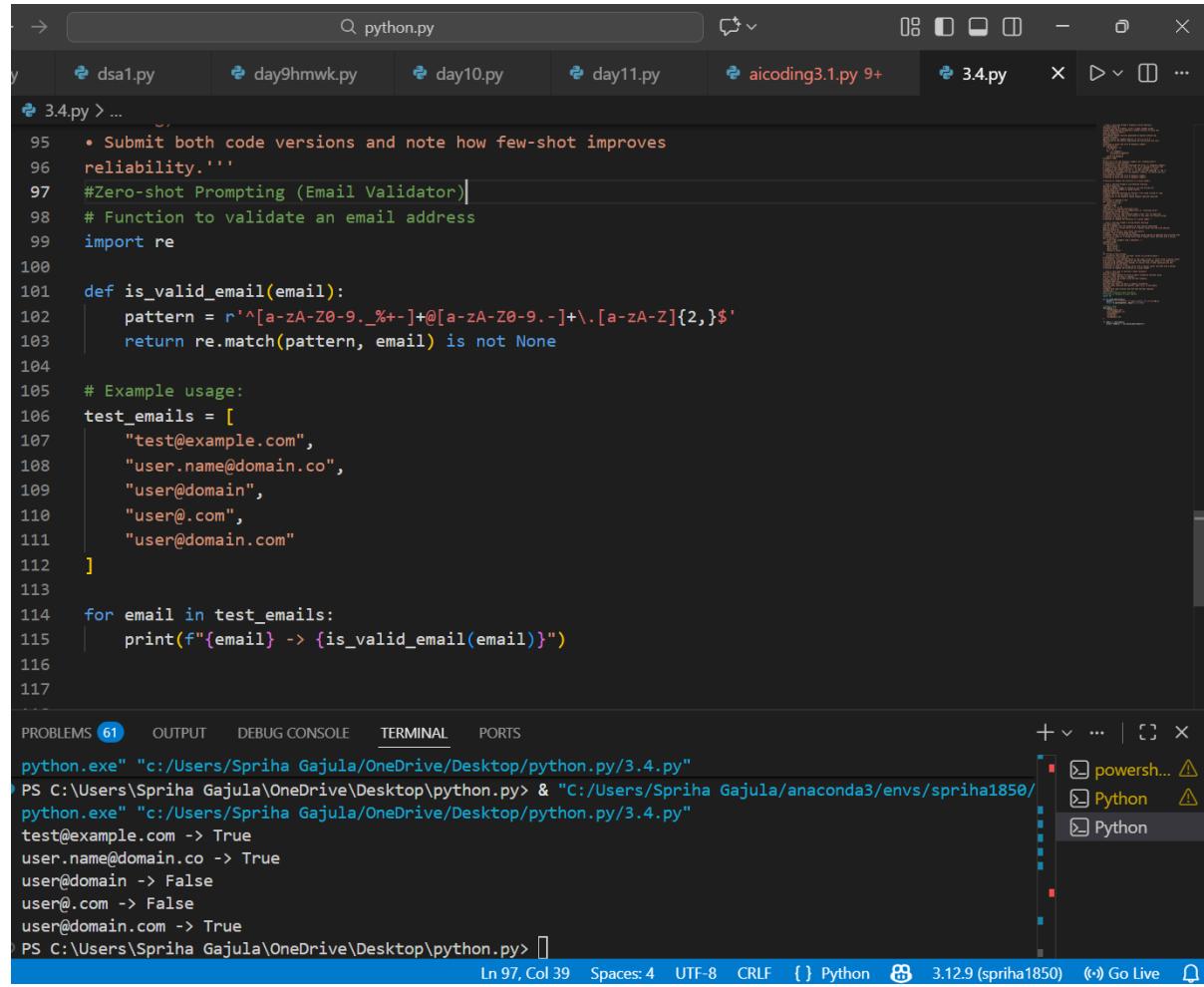
This program uses regular expressions to validate the structure of an email address.

It checks for the presence of valid characters, an @ symbol, and a proper domain format.

User input makes the program flexible and reusable for different email addresses.

The function returns a Boolean value, ensuring clear and efficient validation.

Few-shot Prompting (Email Validator)



```
python.py
DSA DAY 9 HOMEWORK - DAY 10 HOMEWORK - DAY 11 HOMEWORK - AI CODING 3.1 - 3.4.py
3.4.py > ...
95     • Submit both code versions and note how few-shot improves
96     reliability.'''
```

```
#Zero-shot Prompting (Email Validator)
# Function to validate an email address
import re

def is_valid_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None

# Example usage:
test_emails = [
    "test@example.com",
    "user.name@domain.co",
    "user@domain",
    "user@.com",
    "user@domain.com"
]

for email in test_emails:
    print(f"{email} -> {is_valid_email(email)}")
```

```
PROBLEMS 61 OUTPUT DEBUG CONSOLE TERMINAL PORTS
python.exe "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/3.4.py"
test@example.com -> True
user.name@domain.co -> True
user@domain -> False
user@.com -> False
user@domain.com -> True
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py> []
Ln 97, Col 39  Spaces: 4  UTF-8  CRLF  { } Python  3.12.9 (sprisha1850)  (→) Go Live  ⚡
```

JUSTIFICATION :

This program checks whether an email address is valid using a regular expression.

The `is_valid_email()` function defines a pattern that allows letters, numbers, and special characters before @, followed by a valid domain name and extension.

A list of sample email addresses is provided to test different cases, including valid and invalid formats.

Each email is passed to the function, and the result (True or False) is printed to show whether the email is valid.

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

CODE:

Style 1: Generic task prompt

```
105     email = input("Enter an email address: ")  
106  
107     # Validation result  
108     if is_valid_email(email):  
109         print("Valid email address")  
110     else:  
111         print("Invalid email address")'''  
112  
113 Task 5: Prompt Tuning □ Summing Digits of a Number  
114 → Task Description #5 import Task  
115 # Experiment with 2 different prompt styles to generate a function  
116 # that returns the sum of digits of a number.  
117 Style 1: Generic task prompt  
118 def sum_of_digits(n):  
119     total = 0  
120     n = abs(n) # handles negative numbers  
121     while n > 0:  
122         total += n % 10  
123         n //= 10  
124     return total  
125 Style 2: Task + Input/Output example  
126 # Function to compute the sum of digits of a number  
127 def sum_of_digits(n):  
128     total = 0  
PROBLEMS 107 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"  
● PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/Untitled-1.py"  
The sum of digits of 1234 is: 10  
The sum of digits of -5678 is: 26  
The sum of digits of 0 is: 0  
The sum of digits of 9 is: 9  
○ PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py <|  
Ln 113, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.9 (spriha1850) Go Live
```

Style 2: Task + Input/Output example

```
24     return total  
25 Style 2: Task + Input/Output example  
26 # Function to compute the sum of digits of a number  
27 def sum_of_digits(n):  
28     total = 0  
29     n = abs(n) # handles negative numbers  
30     while n > 0:  
31         total += n % 10  
32         n //= 10  
33     return total  
34  
35 # Example usage  
36 test_numbers = [1234, -5678, 0, 9]  
37  
38 for number in test_numbers:  
39     print(f"The sum of digits of {number} is: {sum_of_digits(number)}")  
40  
41 # Explanation of the function:  
42 # sum_of_digits(n): This function calculates the sum of the digits of an integer 'n'  
43 # It first converts 'n' to its absolute value to handle negative numbers.  
44 # It then iteratively extracts each digit using modulus and integer division,  
45 # adding each digit to the total sum. Finally, it returns the computed sum.  
46 # Function to compute the sum of digits of a number  
47  
PROBLEMS 107 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/aicoding3.1.py"  
PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py & "C:/Users/Spriha Gajula/anaconda3/envs/spriha1850/python.exe" "c:/Users/Spriha Gajula/OneDrive/Desktop/python.py/Untitled-1.py"  
The sum of digits of 1234 is: 10  
The sum of digits of -5678 is: 26  
The sum of digits of 0 is: 0  
The sum of digits of 9 is: 9  
○ PS C:\Users\Spriha Gajula\OneDrive\Desktop\python.py <|  
Ln 113, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.9 (spriha1850) Go Live
```

JUSTIFICATION:

The sum_of_digits function calculates the total of all digits in a number. First, it takes the absolute value so that negative numbers are handled correctly. It then looks at each digit one by one, starting from the last digit, and adds it to a running total. After all digits have been added, the function gives back the total sum. Finally, the code tests this function with different numbers and prints the results.