ASSIGNMENT-3.4

Name: k.manjula

H NO:2303A51857

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

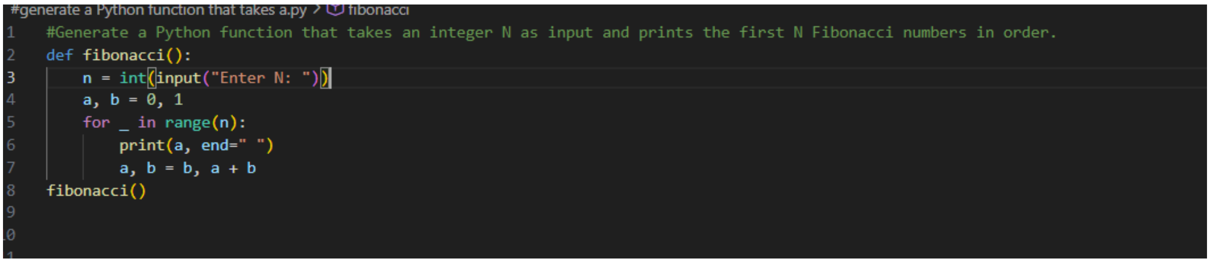
example provided.

- Correct output for sample input N = 7 → 0 1 1 2 3 5 8
- Observation on how Copilot understood the instruction with zero context.

PROMPT:

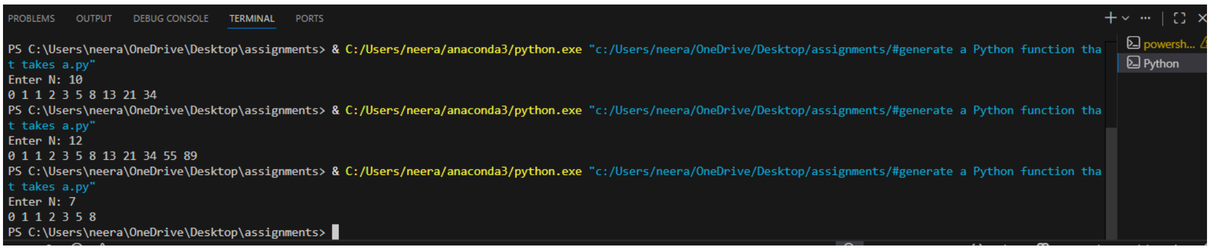
#Generate a Python function that takes an integer N as input and prints the first N Fibonacci numbers in order.

CODE:



```python
#Generate a Python function that takes an integer N as input and prints the first N Fibonacci numbers in order.
def fibonacci():
    n = int(input("Enter N: "))
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
fibonacci()
```

OUTPUT:



```
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#generate a Python function tha
t takes a.py"
Enter N: 10
0 1 1 2 3 5 8 13 21 34
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#generate a Python function tha
t takes a.py"
Enter N: 12
0 1 1 2 3 5 8 13 21 34 55 89
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#generate a Python function tha
t takes a.py"
Enter N: 7
0 1 1 2 3 5 8
PS C:\Users\neera\OneDrive\Desktop\assignments> 
```

JUSTIFICATION:

Observation :

Copilot correctly understood the instruction without any examples by:

- Recognizing the term *Fibonacci series*
- Identifying that the function should generate values iteratively
- Printing the sequence starting from 0
- Handling the input N as the number of terms

This shows that Copilot can infer logic and structure accurately even with minimal context, which is the essence of zero-shot prompting.

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]

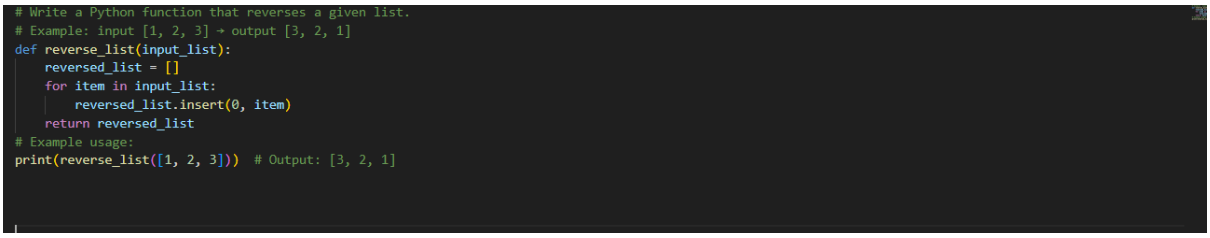
• Observation on how adding a single example improved Copilot's accuracy.

PROMPT:

Write a Python function that reverses a given list.

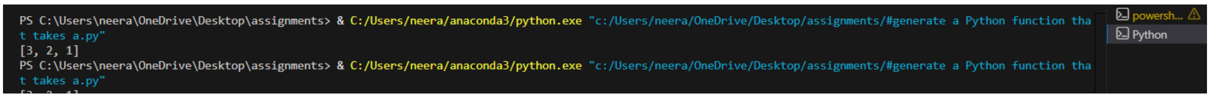
Example: input [1, 2, 3] → output [3, 2, 1]

CODE:



```
# Write a Python function that reverses a given list.
# Example: input [1, 2, 3] → output [3, 2, 1]
def reverse_list(input_list):
    reversed_list = []
    for item in input_list:
        reversed_list.insert(0, item)
    return reversed_list
# Example usage:
print(reverse_list([1, 2, 3]))  # Output: [3, 2, 1]
```

OUTPUT:



```
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#generate a Python function tha
t takes a.py"
[3, 2, 1]
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#generate a Python function tha
t takes a.py"
```

JUSTIFICATION:

Providing a single example helped Copilot clearly understand:

- The expected input and output format
- That the list order must be fully reversed
- The most efficient approach using slicing

The one-shot example reduced ambiguity and improved accuracy compared to zero-shot prompting.

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:

Write a Python function is_valid(text) that returns True if a string

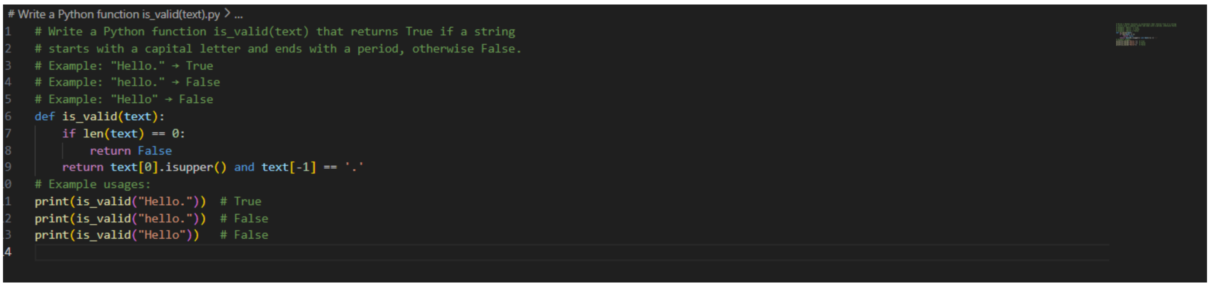
starts with a capital letter and ends with a period, otherwise False.

Example: "Hello." → True

Example: "hello." → False

Example: "Hello" → False

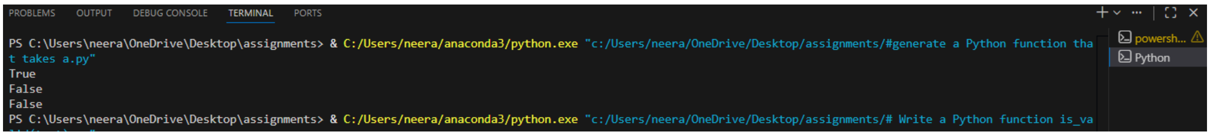
CODE:



```python
# Write a Python function is_valid(text) that returns True if a string
# starts with a capital letter and ends with a period, otherwise False.
# Example: "Hello." → True
# Example: "hello." → False
# Example: "Hello" → False
def is_valid(text):
    if len(text) == 0:
        return False
    return text[0].isupper() and text[-1] == '.'
# Example usages:
print(is_valid("Hello."))  # True
print(is_valid("hello."))  # False
print(is_valid("Hello"))   # False

```

OUTPUT:



```
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#generate a Python function tha
t takes a.py"
True
False
False
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/# Write a Python function is_va
```

JUSTIFICATION:

Providing multiple examples helped Copilot clearly identify:

- The importance of the first character being uppercase
- The requirement of a period at the end
- The exact conditions for True and False outputs

Few-shot prompting reduces misunderstanding and leads to more precise and reliable 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).

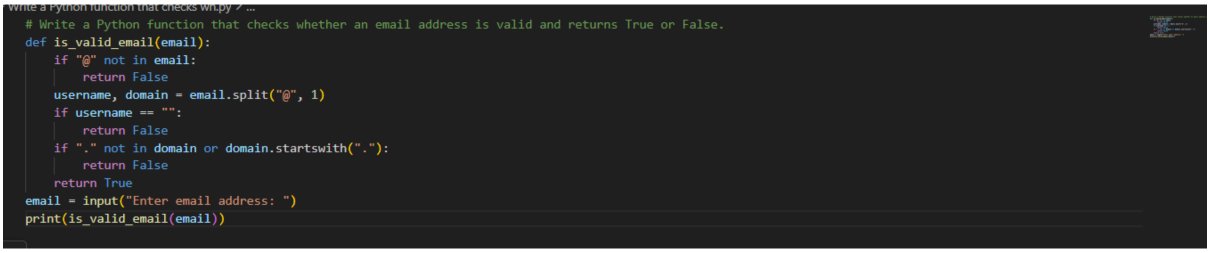
• Submit both code versions and note how few-shot improves reliability.

Zero-shot

PROMPT:

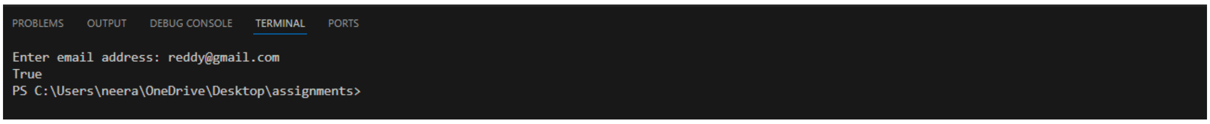
Write a Python function that checks whether an email address is valid and returns True or False.

CODE:



```python
# Write a Python function that checks whether an email address is valid and returns True or False.
def is_valid_email(email):
    if "@" not in email:
        return False
    username, domain = email.split("@", 1)
    if username == "":
        return False
    if "." not in domain or domain.startswith("."):
        return False
    return True
email = input("Enter email address: ")
print(is_valid_email(email))
```

OUTPUT:



```
Enter email address: reddy@gmail.com
True
PS C:\Users\neera\OneDrive\Desktop\assignments>
```

Few-shot

PROMPT

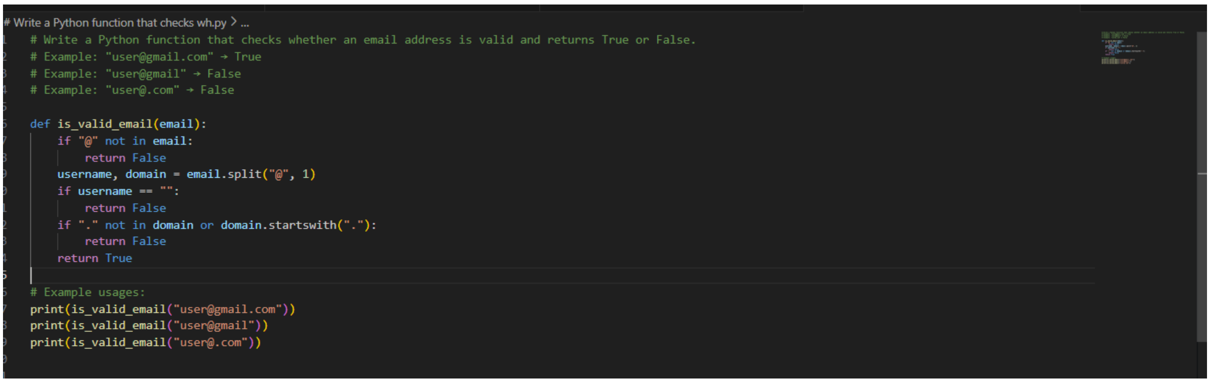
Write a Python function that checks whether an email address is valid and returns True or False.

Example: "user@gmail.com" → True

Example: "user@gmail" → False

Example: "user@.com" → False

CODE

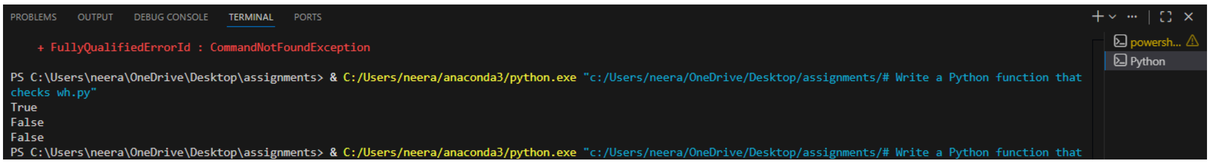


```python
# Write a Python function that checks whether an email address is valid and returns True or False.
# Example: "user@gmail.com" → True
# Example: "user@gmail" → False
# Example: "user@.com" → False

def is_valid_email(email):
    if "@" not in email:
        return False
    username, domain = email.split("@", 1)
    if username == "":
        return False
    if "." not in domain or domain.startswith("."):
        return False
    return True

# Example usages:
print(is_valid_email("user@gmail.com"))
print(is_valid_email("user@gmail"))
print(is_valid_email("user@.com"))
```

OUTPUT



```
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/# Write a Python function that
checks wh.py"
True
False
False
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/# Write a Python function that
```

JUSTIFICATION:

Zero-shot prompting provides only the task description, so Copilot generates a simple and generic email validation logic. This approach may miss several invalid cases. In few-shot prompting, example inputs and outputs are given along with the task. These examples help

Copilot understand the validation rules more clearly. As a result, few-shot prompting produces more accurate and reliable code than zero-shot prompting.

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

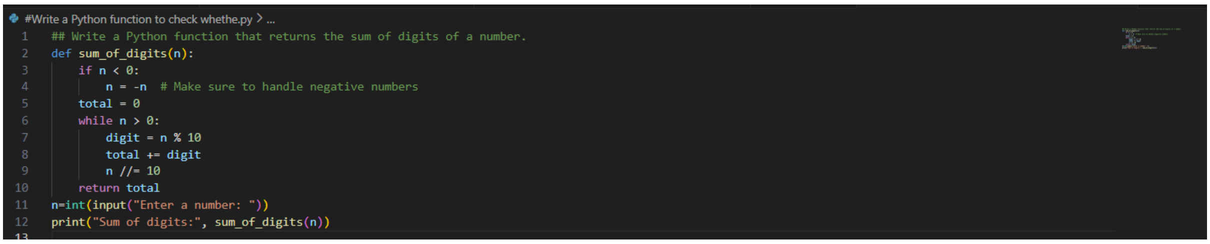
- Two versions of the sum_of_digits() function.
- Example Output: sum_of_digits(123) → 6
- Short analysis: which prompt produced cleaner or more optimized code and why?

#Style 1: Generic task prompt

PROMPT

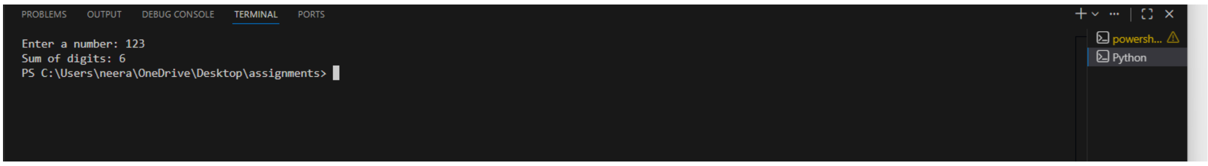
#Write a Python function that returns the sum of digits of a number.

CODE



```python
## Write a Python function that returns the sum of digits of a number.
def sum_of_digits(n):
    if n < 0:
        n = -n  # Make sure to handle negative numbers
    total = 0
    while n > 0:
        digit = n % 10
        total += digit
        n //= 10
    return total
n=int(input("Enter a number: "))
print("Sum of digits:", sum_of_digits(n))
```

OUTPUT



```
Enter a number: 123
Sum of digits: 6
PS C:\Users\neera\OneDrive\Desktop\assignments> 
```

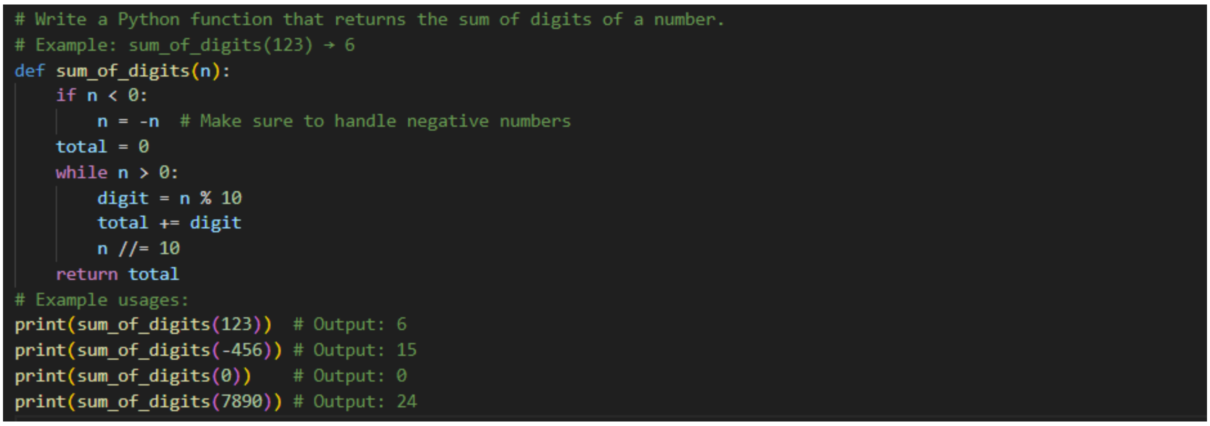
Style 2: Task + Input/Output example

PROMPT

# Write a Python function that returns the sum of digits of a number.

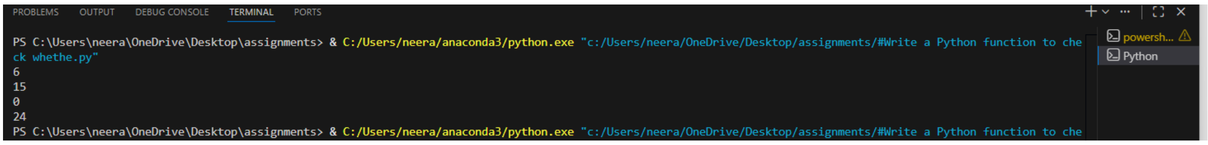
# Example: sum_of_digits(123) → 6

CODE



```python
# Write a Python function that returns the sum of digits of a number.
# Example: sum_of_digits(123) → 6
def sum_of_digits(n):
    if n < 0:
        n = -n  # Make sure to handle negative numbers
    total = 0
    while n > 0:
        digit = n % 10
        total += digit
        n //= 10
    return total
# Example usages:
print(sum_of_digits(123))  # Output: 6
print(sum_of_digits(-456)) # Output: 15
print(sum_of_digits(0))    # Output: 0
print(sum_of_digits(7890)) # Output: 24
```

OUTPUT



```
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#Write a Python function to che
ck whethe.py"
6
15
0
24
PS C:\Users\neera\OneDrive\Desktop\assignments> & C:/Users/neera/anaconda3/python.exe "c:/Users/neera/OneDrive/Desktop/assignments/#Write a Python function to che
```

JUSTIFICATION

The generic prompt produced a correct but longer solution using a loop.
The prompt with an example guided Copilot to generate a cleaner and more optimized one-line solution.
Providing input/output examples helps the model infer intent more clearly and produce concise, efficient code.