

AI ASSI CODING

ASSIGNMENT-3.4

H Ti.NO:2303A510B4

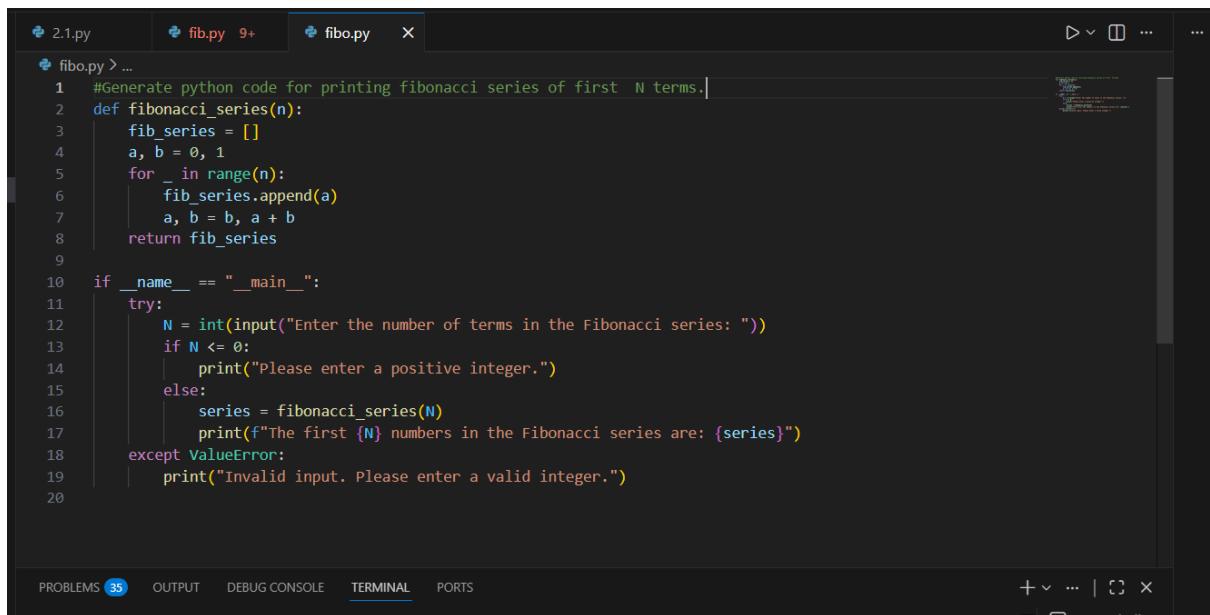
Batch:14

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

Prompt

#Generate python code for printing fibonacci series of first N terms.

Code:



```
 1 #Generate python code for printing fibonacci series of first N terms.
 2 def fibonacci_series(n):
 3     fib_series = []
 4     a, b = 0, 1
 5     for _ in range(n):
 6         fib_series.append(a)
 7         a, b = b, a + b
 8     return fib_series
 9
10 if __name__ == "__main__":
11     try:
12         N = int(input("Enter the number of terms in the Fibonacci series: "))
13         if N <= 0:
14             print("Please enter a positive integer.")
15         else:
16             series = fibonacci_series(N)
17             print(f"The first {N} numbers in the Fibonacci series are: {series}")
18     except ValueError:
19         print("Invalid input. Please enter a valid integer.")
```

Output:



```
PS C:\Users\nithe\OneDrive\Documents\python learning> & C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
":::/Users/nithe/OneDrive/Documents/python learning/fibo.py"
Enter the number of terms in the Fibonacci series: 10
The first 10 numbers in the Fibonacci series are: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
PS C:\Users\nithe\OneDrive\Documents\python learning>
```

Justification:

Copilot correctly interpreted the term “Fibonacci numbers” without any examples.

It generated an efficient iterative solution using variable swapping.

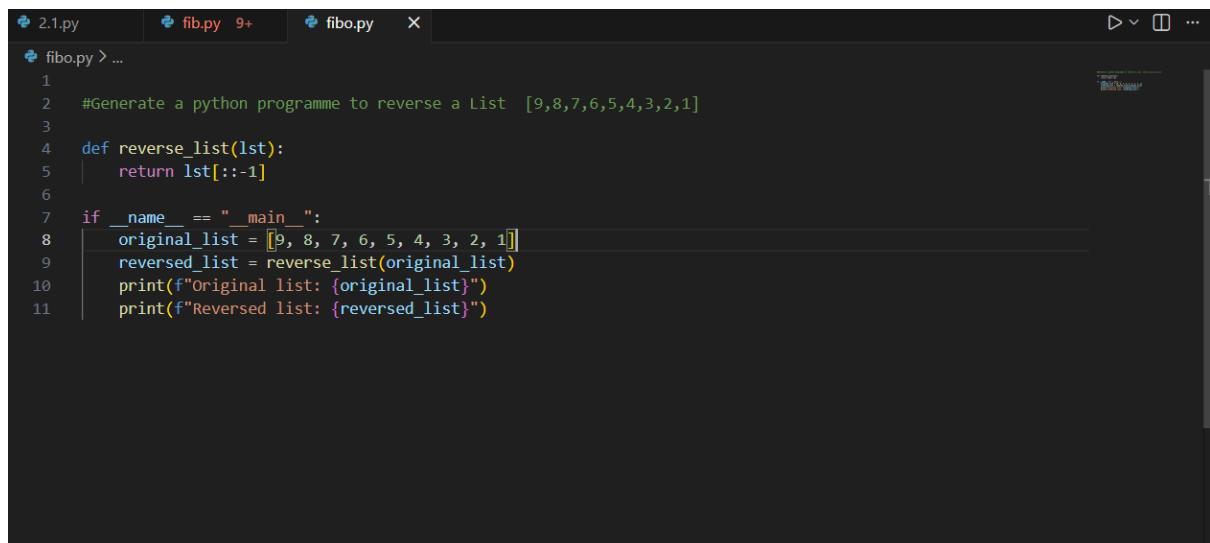
This demonstrates Copilot’s ability to rely on pretrained knowledge in a zero-shot prompting scenario.

Task-2: One-shot Prompt – List Reversal Function

Prompt:

```
#Generate a python programme to reverse a
List [9,8,7,6,5,4,3,2,1]
```

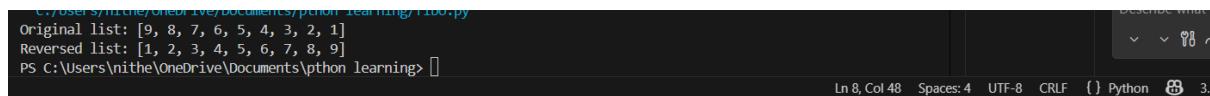
Code:



The screenshot shows a code editor window with three tabs at the top: '2.1.py', 'fib.py 9+', and 'fibo.py'. The 'fibo.py' tab is active, showing the following Python code:

```
1
2 #Generate a python programme to reverse a List [9,8,7,6,5,4,3,2,1]
3
4 def reverse_list(lst):
5     return lst[::-1]
6
7 if __name__ == "__main__":
8     original_list = [9, 8, 7, 6, 5, 4, 3, 2, 1]
9     reversed_list = reverse_list(original_list)
10    print(f"Original list: {original_list}")
11    print(f"Reversed list: {reversed_list}")
```

Output:



The screenshot shows a terminal window with the following text:

```
C:\Users\nithe\OneDrive\Documents\python learning> f100.py
Original list: [9, 8, 7, 6, 5, 4, 3, 2, 1]
Reversed list: [1, 2, 3, 4, 5, 6, 7, 8, 9]
PS C:\Users\nithe\OneDrive\Documents\python learning>
```

Justification:

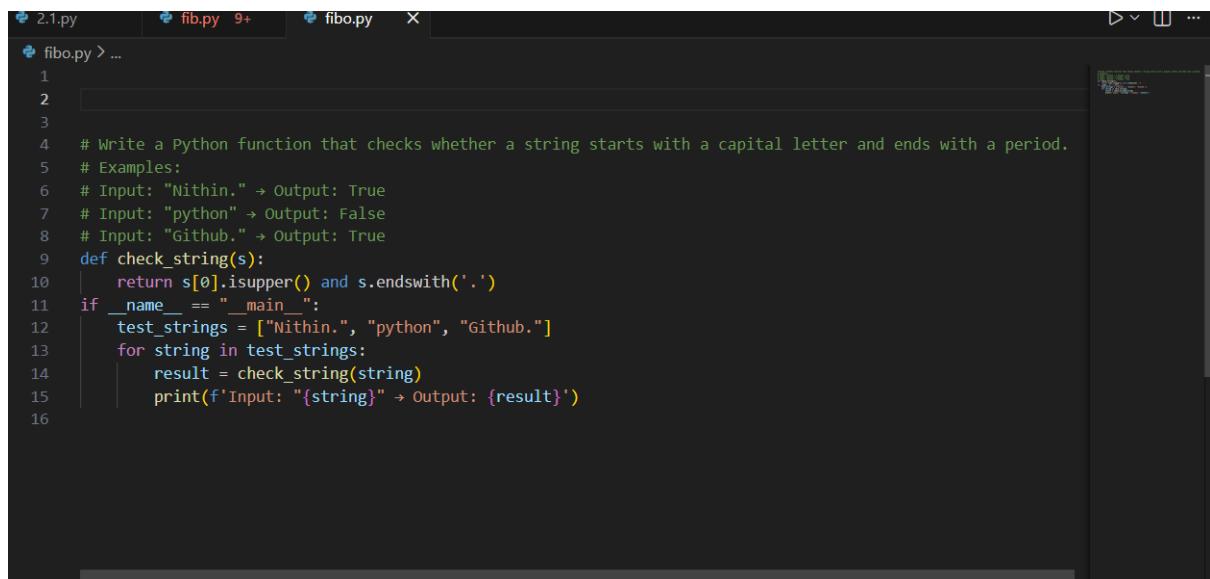
The single example clearly defined the expected input–output behavior.

Copilot accurately inferred the requirement to reverse the list order.

Providing one example improved precision compared to zero-shot prompting.

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

Prompt:

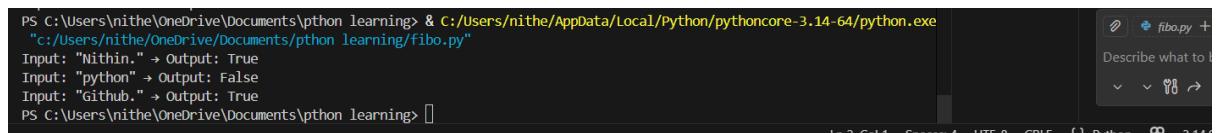


```
2.1.py    fib.py  9+  fibo.py  X
fibo.py > ...
1
2
3
4 # Write a Python function that checks whether a string starts with a capital letter and ends with a period.
5 # Examples:
6 # Input: "Nithin." → Output: True
7 # Input: "python" → Output: False
8 # Input: "Github." → Output: True
9 def check_string(s):
10     return s[0].isupper() and s.endswith('.')
11 if __name__ == "__main__":
12     test_strings = ["Nithin.", "python", "Github."]
13     for string in test_strings:
14         result = check_string(string)
15         print(f'Input: "{string}" → output: {result}')



```

Output:



```
PS C:\Users\nithe\OneDrive\Documents\python learning> & c:/users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
"c:/users/nithe/OneDrive/Documents/python learning/fibo.py"
Input: "Nithin." → Output: True
Input: "python" → Output: False
Input: "Github." → Output: True
PS C:\Users\nithe\OneDrive\Documents\python learning> []
```

Justification:

Multiple examples clearly defined both valid and invalid string patterns.

Copilot inferred the exact conditions using capitalization and ending character.

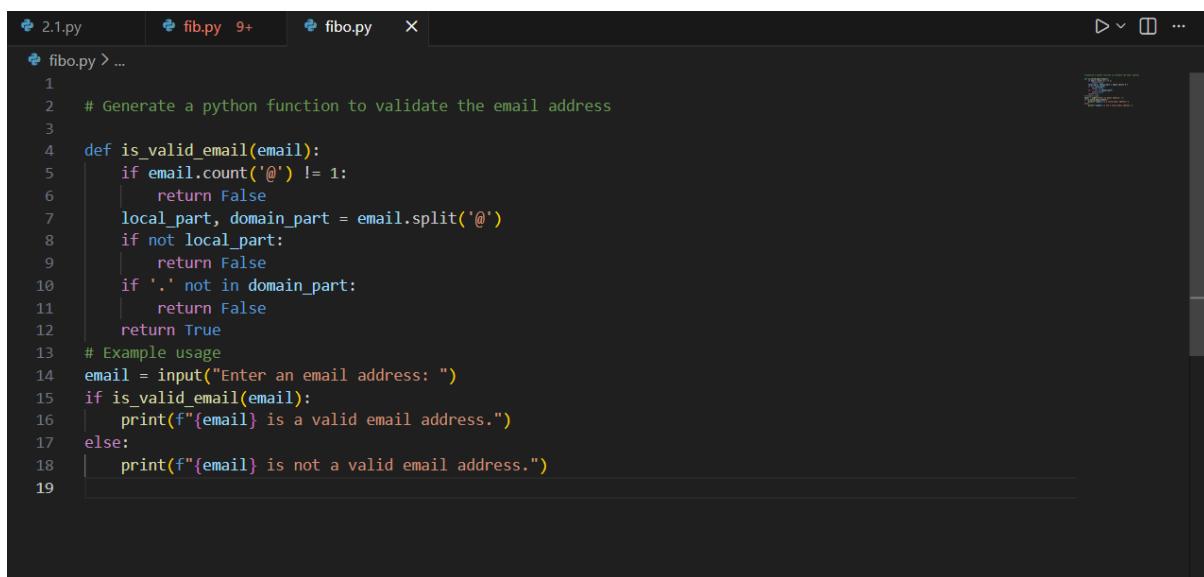
Few-shot prompting improved accuracy by reducing ambiguity compared to zero- or one-shot prompts

Task-4: Zero-shot vs Few-shot – Email Validator

Prompt:

Zero-shot:

Code:



A screenshot of a code editor window titled "fibo.py > ...". The code is as follows:

```
 1  # Generate a python function to validate the email address
 2
 3
 4 def is_valid_email(email):
 5     if email.count('@') != 1:
 6         return False
 7     local_part, domain_part = email.split('@')
 8     if not local_part:
 9         return False
10     if '.' not in domain_part:
11         return False
12     return True
13
14 # Example usage
15 email = input("Enter an email address: ")
16 if is_valid_email(email):
17     print(f"{email} is a valid email address.")
18 else:
19     print(f"{email} is not a valid email address.")
```

Output:

```
"c:/Users/nithe/OneDrive/Documents/python learning/fibo.py"
Enter an email address: mohan4@gmail.com
mohan4@gmail.com is not a valid email address.
PS C:\Users\nithe\OneDrive\Documents\python learning> & C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
"c:/Users/nithe/OneDrive/Documents/python learning/fibo.py"
Enter an email address: nithenk3@gmail.com
nithenk3@gmail.com is a valid email address.
PS C:\Users\nithe\OneDrive\Documents\python learning> 
```

Few-shot:

Code:

```
#Generate a python function to validate the email address.
def is_valid_email(email):
    if email.count('@') != 1:
        return False
    local_part, domain_part = email.split('@')
    if not local_part:
        return False
    if '.' not in domain_part:
        return False
    return True
# Example usage
print(is_valid_email("nithenk3@gmail.com"))      # True
print(is_valid_email("mohangmail.com"))           # False
print(is_valid_email("bala@com"))                 # False
print(is_valid_email("akhil3@gmail.com"))         # True
```

Output:

```
& C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/nithe/OneDrive/Documents/python learning/fi
bo.py" is not a valid email address.
True
False
False
True
PS C:\Users\nithe\OneDrive\Documents\python learning> 
```

Justification:

- The zero-shot prompt resulted in a basic validation that only checks for symbols.
- Few-shot examples clarified valid and invalid email structures explicitly.
- Providing examples improved Copilot's reliability and produced more accurate validation logic.

Task-5: Prompt Tuning – Summing Digits of a Number

Prompt:

Style-1: Generic task Prompt

#generate a python function that returns the sum of digits of a number.

Code:

```
#generate a python function that returns the sum of digits of a number.
def sum_of_digits(n):
    return sum(int(digit) for digit in str(n))
# Example usage:
number = int(input("Enter a number: "))
result = sum_of_digits(number)
print(f"The sum of digits of {number} is {result}.")
```

Output:

```
PS C:\Users\nithe\OneDrive\Documents\python learning> & C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
"c:/Users/nithe/OneDrive/Documents/python learning/fibo.py"
Enter a number: 2345
The sum of digits of 2345 is 14.
PS C:\Users\nithe\OneDrive\Documents\python learning> & C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
"c:/Users/nithe/OneDrive/Documents/python learning/fibo.py"
Enter a number: 8356
The sum of digits of 8356 is 22.
PS C:\Users\nithe\OneDrive\Documents\python learning> []
```

Style-2:

Prompt:

generate a python function that returns the sum of digits of a given number.

Code:

```
1
2
3 #generate a python function that returns the sum of digits of a given number.
4 def sum_of_digits(n):
5     return sum(int(digit) for digit in str(abs(n)))
6 # Example usage:
7 print(sum_of_digits(1234)) # Output: 10
8 print(sum_of_digits(-567)) # Output: 18
9 print(sum_of_digits(0)) # Output: 0
```

Output:

```
PS C:\Users\nithe\OneDrive\Documents\python learning> & C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
"c:/Users/nithe/OneDrive/Documents/python learning/fibo.py"
10
18
0
PS C:\Users\nithe\OneDrive\Documents\python learning>
```

Justification:

- The generic prompt produced a correct but more verbose loop-based solution.
- Adding an input/output example guided the model toward a cleaner, more concise implementation.
- Prompt tuning with examples improves readability and optimization by clarifying expected behavior.

