

# AI Assisted Coding

## Assignment 3.4

Name: P. Vineeth Kumar

Hall ticket no: 2303A51256

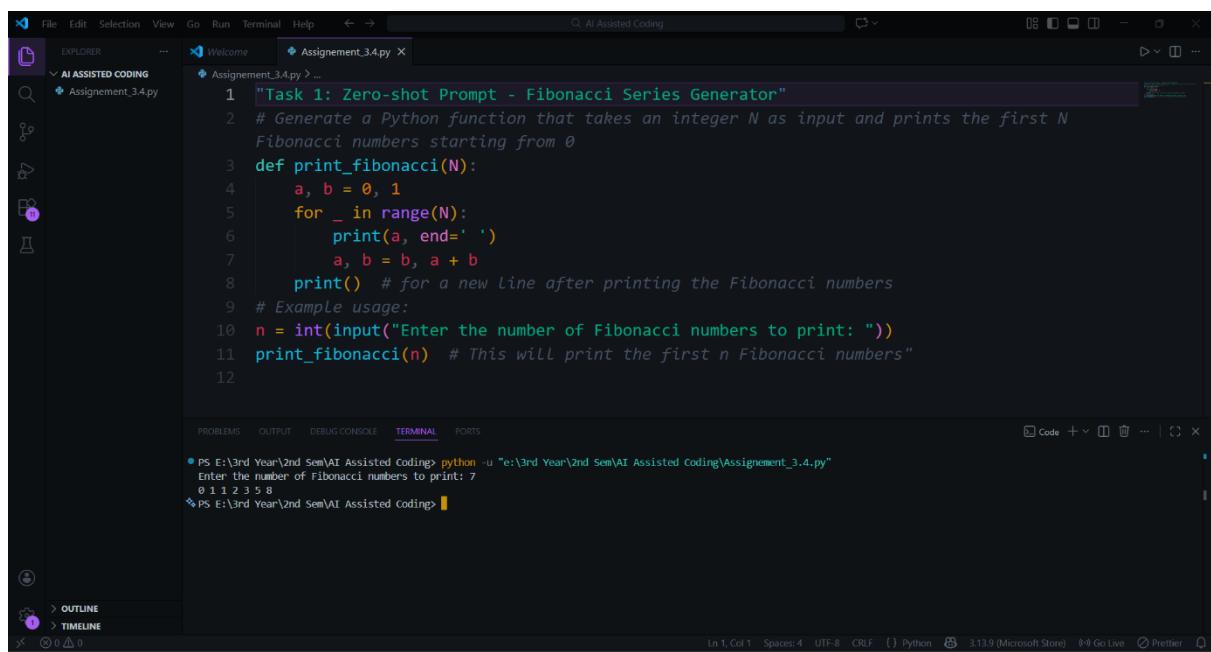
Batch no: 19

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

#### Prompt:

Generate a Python function that takes an integer N as input and prints the first N Fibonacci numbers starting from 0.

#### Code & Output:



The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several icons. The left sidebar has sections for Explorer, Welcome, AI ASSISTED CODING, and Assignment\_3.4.py. The main workspace shows the code for 'Assignment\_3.4.py' with the following content:

```
1  "Task 1: Zero-shot Prompt - Fibonacci Series Generator"
2  # Generate a Python function that takes an integer N as input and prints the first N
   Fibonacci numbers starting from 0
3  def print_fibonacci(N):
4      a, b = 0, 1
5      for _ in range(N):
6          print(a, end=' ')
7          a, b = b, a + b
8      print() # for a new Line after printing the Fibonacci numbers
9  # Example usage:
10 n = int(input("Enter the number of Fibonacci numbers to print: "))
11 print_fibonacci(n) # This will print the first n Fibonacci numbers
```

Below the code, the terminal tab shows the command 'python -u "E:\3rd Year\2nd Sem\AI Assisted Coding\Assignment\_3.4.py"' and its output: '0 1 1 2 3 5 8'. The bottom status bar indicates the code is saved in Python 3.13.9 (Microsoft Store) and shows file statistics like Ln 1, Col 1, Spaces: 4, and CRLF.

#### Explanation:

In this task, a zero-shot prompt was used where only the problem description was provided without any examples. Based on this instruction, the AI generated a function to print the Fibonacci series. The program starts with the initial values and iteratively calculates the next numbers in the sequence. This task demonstrates that the AI can correctly understand and solve a problem even when no examples are given.

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

### Prompt:

Write a Python function that takes a list as input and returns the reversed list.

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

### Code & Output:

The screenshot shows the Visual Studio Code interface with the 'AI ASSISTED CODING' extension active. The 'EXPLORER' sidebar shows a file named 'Assigment\_3.4.py'. The main code editor window contains the following Python code:

```
12
13 "Task 2: One-shot Prompt - List Reversal Function"
14 # Write a Python function that takes a list as input and returns the reversed list
15 # Example: input = [1, 2, 3], output = [3, 2, 1]
16 def reverse_list(input_list):
17     return input_list[::-1]
18 # Example usage:
19 example_list = list(map(int, input("Original List: ").split()))
20 print("Reversed list:", reverse_list(example_list))
```

The 'TERMINAL' tab at the bottom shows the command line output of running the script:

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assigment_3.4.py"
● Original list: 1 2 3
Reversed list: [3, 2, 1]
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

### Explanation:

In this task, a one-shot prompt was used by providing a single example along with the task description. The example helped the AI clearly understand the expected input and output format. As a result, the generated solution accurately reverses the list. This shows that adding one example improves the clarity and correctness of the AI-generated code.

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

### Prompt:

Write a Python function `is_valid(s)` that returns True if a string starts with a capital letter and ends with a period.

Examples:

"Hello world." → True

"hello world." → False

"Hello world" → False

### Code & Output:

The screenshot shows the Microsoft Visual Studio Code interface with the 'AI ASSISTED CODING' extension active. The 'TERMINAL' tab is selected, displaying a command-line session where the user runs a Python script named 'Assignment\_3.4.py'. The script contains a function 'is\_valid' that checks if a string starts with a capital letter and ends with a period. The terminal output shows three examples: 'Hello world.' (True), 'hello world.' (False), and 'Hello world' (False). The status bar at the bottom indicates the file is saved.

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_3.4.py"
● Enter a string: Hello world.
True
● PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_3.4.py"
Enter a string: hello world.
False
● PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_3.4.py"
Enter a string: Hello world
False
PS E:\3rd Year\2nd Sem\AI Assisted coding>
```

### Explanation:

In this task, few-shot prompting was used by providing multiple examples. These examples guided the AI to identify both conditions correctly: the string must start with a capital letter and end with a period. The presence of multiple examples helped the AI generate a more precise and reliable solution compared to zero-shot or one-shot prompting.

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

#### Zero-shot Prompt:

Write a Python function to validate whether an email address is valid or not.

#### Code & Output:

The screenshot shows a Microsoft Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward navigation icon. On the left is the Explorer sidebar with sections for Explorer, AI ASSISTED CODING, and the current file, Assignment\_3.4.py. The main editor area displays a Python script titled "Assignment\_3.4.py". The code implements a function to validate email addresses using regular expressions. Below the editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing command-line interactions with the Python interpreter. The bottom status bar indicates the file has 35 lines, 67 columns, 4 spaces, and is in UTF-8 format, using Python as the language, and shows the file size as 3.139 MB. There are also icons for Save, Undo, Redo, and other standard file operations.

```
File Edit Selection View Go Run Terminal Help < > AI Assisted Coding

EXPLORER
AI ASSISTED CODING
Assignment_3.4.py

Assignment_3.4.py > ...

34
35  "Task 4: Zero-shot vs Few-shot - Email Validator (Zero-shot Prompt)"
36  # Write a Python function to validate whether an email address is valid or not
37  def is_valid_email(email):
38      import re
39      pattern = r'^[a-zA-Z0-9.\%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
40      return re.match(pattern, email) is not None
41  # Example usage:
42  email_input = input("Enter an email address: ")
43  if is_valid_email(email_input):
44      print("Valid email address")
45  else:
46      print("Invalid email address")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_3.4.py"
● Enter an email address: srujanreddi.in
Valid email address
● PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_3.4.py"
Enter an email address: srujanmail.com
Invalid email address
PS E:\3rd Year\2nd Sem\AI Assisted Coding>

TERMINAL
```

## Few-shot Prompt:

Write a Python function `is_valid_email(email)` that returns True for valid emails and False otherwise.

## Examples:

"user@gmail.com" → True

"user123@yahoo.in" → True

"user@gmail.com" → False

"user@.com" → False

## Code & Output:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "AI ASSISTED CODING" containing "Assignment\_3.4.py".
- Editor:** The main editor window displays Python code for validating emails. The code defines a function `is\_valid\_email` that uses a regular expression to check if an email address is valid. It includes examples and a usage section.
- Terminal:** The terminal window at the bottom shows command-line interactions with the script. It prompts for an email address ("Enter an email address: ") and prints the result ("True" or "False").
- Status Bar:** Shows the current file path as "C:\Users\sruthi\OneDrive\Desktop\Assignment\_3.4.py", line 60, column 96, and other standard status bar information.

## Explanation:

In the zero-shot prompt, the AI produced a basic email validation logic because no examples were provided. In contrast, the few-shot prompt included valid and invalid examples, which helped the AI understand the structure of an email address more clearly. As a result, the few-shot approach generated a more accurate and reliable email validation solution.

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

### Style 1: Generic Task Prompt

#### Prompt:

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

#### Code & Output:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left has a 'AI ASSISTED CODING' section with a file named 'Assigment\_3.4.py'. The main editor area contains the following Python code:

```
61
62
63 "Task 5: Prompt Tuning - Summing Digits of a Number (Genereic Task Prompt)"
64 # Write a Python function that returns the sum of digits of a given number
65 def sum_of_digits(number):
66     total = 0
67     while number > 0:
68         total += number % 10
69         number //= 10
70     return total
71 # Example usage:
72 num = int(input("Enter a number: "))
73 print("Sum of digits:", sum_of_digits(num)) # This will print the sum of the digits of the number
```

The Terminal tab at the bottom shows the output of running the script:

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assigment_3.4.py"
● Enter a number: 123
Sum of digits: 6
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

### Style 2: Task + Input/Output Example Prompt

#### Prompt:

Write a Python function `sum_of_digits(n)` that returns the sum of all digits in a number.

Example: input = 123, output = 6

#### Code & Output:

The screenshot shows the Microsoft Visual Studio Code interface with the "AI ASSISTED CODING" extension active. The code editor displays a Python file named "Assignment\_3.4.py" containing the following code:

```
75
76
77 "Task 5: Prompt Tuning - Summing Digits of a Number (Task + Input/Output example)"
78 # Write a Python function sum_of_digits(n) that returns the sum of all digits in a number
79 # Example: input = 123, output = 6
80 def sum_of_digits_tuned():
81     return sum(int(digit) for digit in str(n))
82 # Example usage:
83 num_tuned = int(input("Enter a number: "))
84 print("Sum of digits (tuned):", sum_of_digits_tuned(num_tuned)) # This will print the sum of the
# digits of the number
```

The terminal below shows the execution of the script:

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignment_3.4.py"
● Enter a number: 123
Sum of digits (tuned): 6
PS E:\3rd Year\2nd Sem\AI Assisted coding>
```

### Explanation:

In this task, two different prompt styles were used. The generic prompt resulted in a straightforward solution, while the prompt with an input/output example produced a cleaner and more optimized implementation. This task highlights how prompt tuning can significantly improve code quality and readability.