

ASSIGNMENT – 13

NAME: K.Bharath

HT.NO: 2403A52065

BATCH: AIB03

Task-1:

Prompt: correct the code.

Code:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a folder structure with 'task_1.py' selected. Other files include '13_lab', 'task_1.py', 'ASSIGNMENT-12.docx', and 'ASSIGNMENT-12.pdf'.
- Code Editor:** Displays the content of 'task_1.py'. The code defines a function `calculate_area` that takes a shape and dimensions. It handles shapes like rectangle, square, and circle, and ignores other shapes. It also handles missing parameters by raising a `ValueError`.
- Terminal:** Shows the command `circle with params (x=2, y=0) -> area = 12.566370614359172` and the path `PS C:\Users\varsh\Desktop\AI>`.
- Status Bar:** Shows the current file as 'task_1.py', the extension as 'Python', and the status as 'Python'.
- GitHub Copilot AI Panel:** On the right, there's a panel titled 'CHAT' with a message from 'Varshitha-bit' that says 'Enable GPT-5 mini for all clients'. Below it, a message from 'GitHub Copilot' says 'I'll create a short todo plan, then fix task_1.py, add a tiny main test, configure the Python environment, run the script to validate, and update the todo list as I go. Expected outcome: corrected file and a successful execution showing the area outputs. I'll start by writing the todo list and marking the first item in-progress.'
- Bottom Status:** Includes icons for 'Add Context...', 'Add context (@)', 'commands', 'Agent', 'GPT-4o', and 'Python'.

```

13_lab > task_1.py > ...
2 def calculate_area(shape, x, y=0):
3     """Calculate area for rectangle, square, or circle.
4
5     Args:
6         shape (str): One of 'rectangle', 'square', 'circle'.
7         x (float): For rectangle: width, for square: side, for circle: radius.
8         y (float, optional): For rectangle: height. Ignored for other shapes.
9
10    Returns:
11        float: The computed area.
12
13    Raises:
14        ValueError: If an unsupported shape is provided or required parameters are missing/invalid.
15
16    if not isinstance(shape, str):
17        raise ValueError("shape must be a string")
18
19    shape = shape.lower()
20
21    if shape == "rectangle":
22        if y is None or y == 0:
23            raise ValueError("rectangle requires both width (x) and height (y)")
24        return float(x) * float(y)
25    elif shape == "square":
26        return float(x) * float(x)
27    elif shape == "circle":
28        return math.pi * float(x) * float(x)
29    else:
30        raise ValueError(f"unsupported shape: {shape}")
31
32 if __name__ == "__main__":
33     # quick manual tests
34     examples = [
35         ("rectangle", 3, 4),
36         ("square", 5),
37         ("circle", 2),
38     ]
39
40     for ex in examples:
41         shape = ex[0]
42         x = ex[1]
43         y = ex[2] if len(ex) > 2 else 0
44         area = calculate_area(shape, x, y)
45         print(f"{shape} with params (x={x}, y={y}) -> area = {area}")

```

circle with params (x=2, y=0) -> area = 12.566370614359172
PS C:\Users\Varsh\Desktop\AI>

Output:

```

13_lab > task_1.py > calculate_area
1 import math
2
3 def calculate_area(shape, x, y=0):
4     """Calculate area for rectangle, square, or circle.
5
6     Args:
7         shape (str): One of 'rectangle', 'square', 'circle'.
8         x (float): For rectangle: width, for square: side, for circle: radius.
9         y (float, optional): For rectangle: height. Ignored for other shapes.
10
11    Returns:
12        float: The computed area.
13
14    Raises:
15        ValueError: If an unsupported shape is provided or required parameters are missing/invalid.
16
17    if not isinstance(shape, str):
18        raise ValueError("shape must be a string")
19
20    shape = shape.lower()
21
22    if shape == "rectangle":
23        if y is None or y == 0:
24            raise ValueError("rectangle requires both width (x) and height (y)")
25        return float(x) * float(y)
26    elif shape == "square":
27        return float(x) * float(x)
28    elif shape == "circle":
29        return math.pi * float(x) * float(x)
30    else:
31        raise ValueError(f"unsupported shape: {shape}")
32
33 if __name__ == "__main__":
34     # quick manual tests
35     examples = [
36         ("rectangle", 3, 4),
37         ("square", 5),
38         ("circle", 2),
39     ]
40
41     for ex in examples:
42         shape = ex[0]
43         x = ex[1]
44         y = ex[2] if len(ex) > 2 else 0
45         area = calculate_area(shape, x, y)
46         print(f"{shape} with params (x={x}, y={y}) -> area = {area}")

```

rectangle with params (x=3, y=4) -> area = 12.0
square with params (x=5, y=0) -> area = 25.0
circle with params (x=2, y=0) -> area = 12.566370614359172
PS C:\Users\Varsh\Desktop\AI>

Observation:

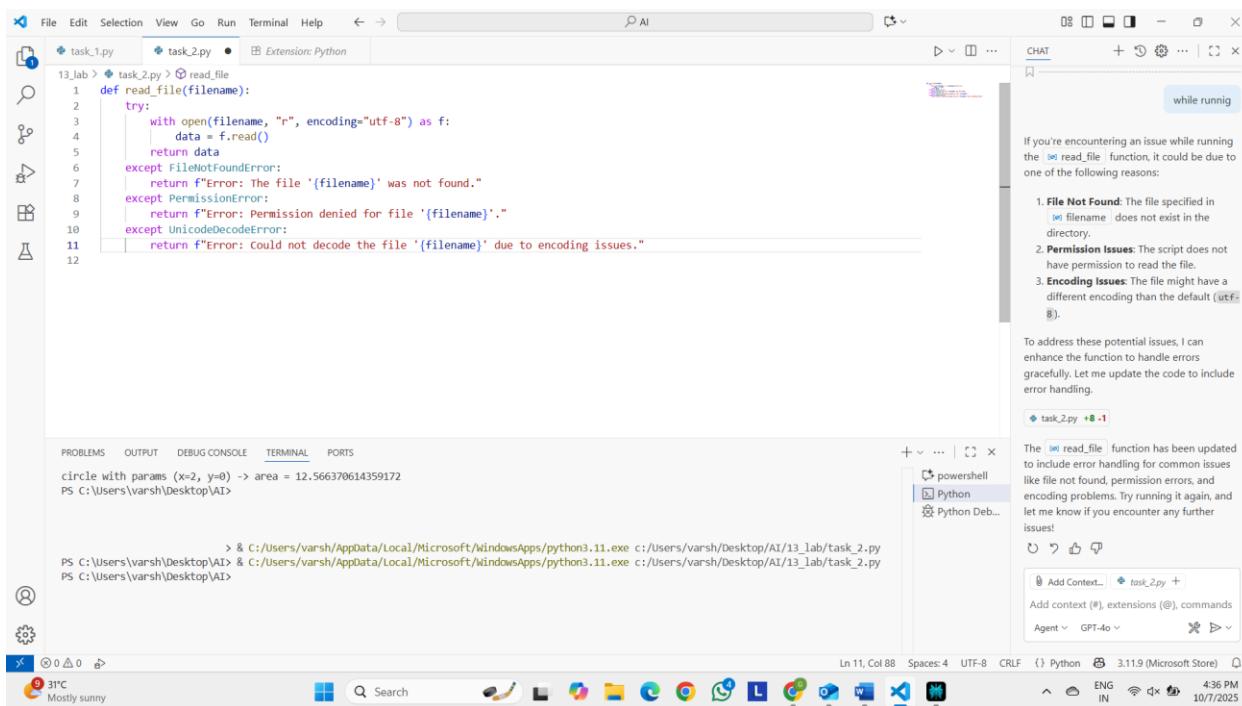
First when we write the code in the work space, then when we give the prompt to correct the code it immediately changed the whole code. The original code was hardly 6 lines but the generated code is more lines. It even included comments so that we can understand the code easily.

Now the code is in a correct structure.

Task-2:

Prompt: Correct my mistake

Code:



```
task_2.py
1 def read_file(filename):
2     try:
3         with open(filename, "r", encoding="utf-8") as f:
4             data = f.read()
5             return data
6     except FileNotFoundError:
7         return f"Error: The file '{filename}' was not found."
8     except PermissionError:
9         return f"Error: Permission denied for file '{filename}'."
10    except UnicodeDecodeError:
11        return f"Error: Could not decode the file '{filename}' due to encoding issues."
12
```

If you're encountering an issue while running the `read_file` function, it could be due to one of the following reasons:

- File Not Found:** The file specified in `filename` does not exist in the directory.
- Permission Issues:** The script does not have permission to read the file.
- Encoding Issues:** The file might have a different encoding than the default (`utf-8`).

To address these potential issues, I can enhance the function to handle errors gracefully. Let me update the code to include error handling.

The `read_file` function has been updated to include error handling for common issues like file not found, permission errors, and encoding problems. Try running it again, and let me know if you encounter any further issues!

task_2.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

circle with params (x=2, y=0) -> area = 12.566370614359172
PS C:\Users\varsh\Desktop\AI>

> & C:/Users/varsh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/varsh/Desktop/AI/13_lab/task_2.py
PS C:\Users\varsh\Desktop\AI> & C:/Users/varsh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/varsh/Desktop/AI/13_lab/task_2.py
PS C:\Users\varsh\Desktop\AI>

31°C Mostly sunny

Ln 11, Col 88 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 (Microsoft Store) ENG IN 4:36 PM 10/7/2025

Output:

The screenshot shows the Microsoft Visual Studio Code interface. In the top left, there are two tabs: 'task_1.py' and 'task_2.py'. The 'task_2.py' tab is active, displaying the following Python code:

```

13_lab> task_2.py > read_file
1  def read_file(filename):
2      try:
3          with open(filename, "r", encoding="utf-8") as f:
4              data = f.read()
5              return data
6      except FileNotFoundError:
7          return f"Error: The file '{filename}' was not found."
8      except PermissionError:
9          return f"Error: Permission denied for file '{filename}'."
10     except UnicodeDecodeError:
11         return f"Error: Could not decode the file '{filename}' due to encoding issues."
12

```

In the bottom right corner of the code editor, there is a 'while running' status bar with a message: 'If you're encountering an issue while running the `read_file` function, it could be due to one of the following reasons:'. It lists three items:

- File Not Found:** The file specified in `filename` does not exist in the directory.
- Permission Issues:** The script does not have permission to read the file.
- Encoding Issues:** The file might have a different encoding than the default (`utf-8`).

To the right of the code editor, there is a 'CHAT' sidebar with a message from AI: 'To address these potential issues, I can enhance the function to handle errors gracefully. Let me update the code to include error handling.' Below this, there is a preview of the updated code: 'task_2.py +8 -1'.

At the bottom of the interface, there is a terminal window showing command-line output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
circle with params (x=2, y=0) -> area = 12.566370614359172
PS C:\Users\varsh\Desktop\AI>

> & C:/Users/varsh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/varsh/Desktop/AI/13_lab/task_2.py
PS C:\Users\varsh\Desktop\AI> & C:/Users/varsh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/varsh/Desktop/AI/13_lab/task_2.py
PS C:\Users\varsh\Desktop\AI>

```

The bottom right corner shows system status: '31°C Mostly sunny', 'Add Context...', 'task_2.py +', 'Add context (#), extensions (@)', 'commands', 'Agent GPT-4o', '3.11.9 (Microsoft Store)', '4:36 PM', '10/7/2022', and language settings 'ENG IN'.

Observation:

In the give code we did not give a code with proper indentation. And file handling was also not perfect. But as we gave the prompt to correct it. It handled all that issues.

Task-3:

Prompt: correct my code. In this the attributes are name, age and marks.

Code:

The screenshot shows the Microsoft Visual Studio Code interface. In the center, there is a code editor window displaying Python code for a class named `Student`. The code includes docstrings and methods like `__init__`, `details`, and `total_marks`. To the right of the code editor, a "CHAT" panel shows an AI-generated message suggesting improvements: "AI improves naming (name, age, marks). • Adds docstrings. • Improves print readability. • Possibly uses sum(self.marks) if marks stored in a list." Below the code editor, the terminal tab is active, showing command-line output related to the Python environment. The bottom status bar indicates the file is 3.11.9 (Microsoft Store) and the date is 10/7/2025.

```

1 class Student:
2     """
3         A class to represent a student with their details and marks.
4     Attributes:
5         name (str): The name of the student.
6         age (int): The age of the student.
7         marks (list): A list of marks obtained by the student.
8     """
9     def __init__(self, name, age, marks):
10        """
11            Initialize the student with name, age, and marks.
12        Args:
13            name (str): The name of the student.
14            age (int): The age of the student.
15            marks (list): A list of marks obtained by the student.
16        """
17        self.name = name
18        self.age = age
19        self.marks = marks
20    def details(self):
21        """
22            Print the details of the student.
23            print(f"Name: {self.name}, Age: {self.age}")
24    def total_marks(self):
25        """
26            Calculate and return the total marks of the student.
27            return sum(self.marks)

```

Output:

This screenshot is identical to the one above, showing the Microsoft Visual Studio Code interface with the AI-assisted code completion feature for the `Student` class. The code editor, terminal output, and status bar are all the same, indicating no changes have occurred between the two screenshots.

Observation:

First when we write the code in the work space, then when we give the prompt to correct the code it immediately changed the whole code. The original code was hardly 6 lines but the generated code is more lines. It even included comments so that we can understand the code easily.

Now the code is in a correct structure

Task-4:

Prompt: Correct my mistake

Code:

The screenshot shows the Microsoft Visual Studio Code (VS Code) interface. The left sidebar (Explorer) shows a project structure with a folder 'AI' containing files 'task_1.py', 'task_2.py', 'task_3.py', and 'task_4.py'. Below these are 'ASSIGNMENT-12.docx' and 'ASSIGNMENT-12.pdf'. The main editor area displays the following Python code:

```
13_lab > task_4.py > ...
1 nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 squares = [i * i for i in nums]
3 print(squares)
```

The right-hand panel contains several sections: 'CHAT' (empty), 'If no output is seen when running the script, it could be due to one of the following reasons:' (with three numbered items: 1. Script Not Executing Properly, 2. Output Suppression, 3. Syntax or Runtime Errors), 'Debugging Steps' (with three numbered steps: 1. Add a Debug Print, 2. Run the Script Manually, 3. Check for Errors), and a 'Terminal' section showing command-line history:

```
Users/vrsh/Desktop/AI/13_lab/task_3.py
PS C:\Users\vrsh\Desktop\AI> & C:/Users/vrsh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/vrsh/Desktop/AI/13_lab/task_4.py
PS C:\Users\vrsh\Desktop\AI> ^
PS C:\Users\vrsh\Desktop\AI> python "c:/Users/vrsh/Desktop/AI/13_lab/task_4.py"
PS C:\Users\vrsh\Desktop\AI>
```

The bottom status bar indicates the file is 'task_4.py', the language is 'Python', and the version is '3.11.9 (Microsoft Store)'. The date and time shown are '10/7/2025 4:48 PM'.

Output:

The screenshot shows the Microsoft Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'AI' containing files: task_1.py, task_2.py, task_3.py, task_4.py, ASSIGNMENT-12.docx, and ASSIGNMENT-12.pdf. The main editor window displays the following Python code:

```
13_lab > task_4.py > ...
1 nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 squares = [i * i for i in nums]
3 print(squares)
```

The terminal tab at the bottom shows the command line output:

```
Users/vansh/Desktop/AI/13_lab/task_3.py
PS C:\Users\varsh\Desktop\AI> & C:/Users/varsh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/vansh/Desktop/AI/13_lab/task_4.py
PS C:\Users\varsh\Desktop\AI> ^
PS C:\Users\varsh\Desktop\AI> python "c:/Users/varsh/Desktop/AI/13_lab/task_4.py"
PS C:\Users\varsh\Desktop\AI>
```

The status bar at the bottom right indicates: Ln 3, Col 15, Spaces: 4, UTF-8, CRLF, Python, 3.11.9 (Microsoft Store), ENG IN, 4:48 PM, 10/7/2025.

Observation:

First when we write the code in the work space, then when we give the prompt to correct the code it immediately changed the whole code. The original code was hardly 6 lines but the generated code is more lines. It even included comments so that we can understand the code easily.

Now the code is in a correct structure