AI Coding Assistance Tools Hands-on Project Using GitHub Copilot Student Handout

Overview

In this handout, we will explore how to use GitHub Copilot, an AI coding assistant tool, for writing, debugging, and optimizing code. This handout assumes no prior knowledge, so we will break everything down into simple, digestible steps.

Key Learning Points

- What GitHub Copilot is and how it works.
- Setting up GitHub Copilot in your coding environment (VS Code).
- Using Copilot to write, debug, and refactor code.
- Understanding the benefits of AI in coding.
- Hands-on project timeline for effective practice.

What is GitHub Copilot?

GitHub Copilot is an AI-powered tool that helps developers by:

- **Suggesting Code**: Predicts what code you will write next.
- **Debugging**: Helps to detect and correct errors in your code.
- **Optimizing**: Refactors your code to make it more efficient.

GitHub Copilot can be integrated into many code editors, including **Visual Studio Code (VS Code)**.

Why Use GitHub Copilot?

Using GitHub Copilot is like having an expert guiding you while you code. It helps save time, reduce errors, and ensures that you're following best practices in coding.

Indian Analogy: Think of Copilot as an assistant in your kitchen while making an Indian dish. Before you even ask, your assistant starts preparing the next ingredients for you. It helps you stay ahead, just like Copilot does with coding.

Step-by-Step Project Using GitHub Copilot

Step 1: Setting Up GitHub Copilot (10 minutes)

- **Install VS Code** if you haven't done so already.
- **Install GitHub Copilot**:
- Open VS Code.
- Go to Extensions (`Ctrl+Shift+X`) and search for "GitHub Copilot."
- Click **Install** and sign in with your GitHub account.
- **Enable Copilot** in your VS Code settings.

You will now see suggestions as you start typing code in the editor.

Step 2: Writing Code with GitHub Copilot (20 minutes)

Once Copilot is installed and active:

- **Create a new file** in VS Code, e.g., `example.py`.
- **Start writing code**. As you type, Copilot will suggest lines of code to complete your function or logic.

For example:

```
def calculate_sum(a, b):

# Your code here
```

Copilot might suggest:

```
return a + b
```

Accept the suggestion by pressing `Tab`.

Indian Analogy: It's like making chai (tea) with an assistant who anticipates your next steps and starts adding the right spices before you even ask.

Step 3: Debugging with GitHub Copilot (20 minutes)

To see Copilot in action during debugging:

- **Introduce an error** in your code, such as forgetting a variable:

```
def calculate_sum(a):

return a + b
```

- **Run the code**. The error will be flagged because `b` is not defined.
- **Fix the error** by accepting Copilot's suggestion for the correct code.

Step 4: Optimizing and Refactoring Code (20 minutes)

You can also use Copilot to improve your code's efficiency:

- **Write inefficient code** such as:

```
def multiply_numbers(x, y):

result = 0

for i in range(y):

result += x

return result
```

- **Accept Copilot's refactor** for a cleaner solution:

```
def multiply_numbers(x, y):
  return x * y
```

Timeline for Project Activities

Here’s a suggested timeline to guide your practice:

- **Setting up GitHub Copilot (10 mins)**: Install and configure Copilot in VS Code.
- **Writing code with Copilot (20 mins)**: Use Copilot to help you write code.
- **Debugging with Copilot (20 mins)**: Deliberately introduce and fix errors using Copilot.
- **Optimizing with Copilot (20 mins)**: Refactor inefficient code with Copilot’s suggestions.
- **Reflection & Review (20 mins)**: Review how Copilot helped improve your workflow.

Key Takeaways

- **GitHub Copilot** is an intelligent tool that helps you write, debug, and optimize code more efficiently.
- It saves time and reduces errors but does not replace the need to understand programming basics.
- Practicing with Copilot will improve your coding skills, making you more productive.
