F-16 - GitHub Copilot Edits

SUMMARY

In this lesson, we explored the **GitHub Copilot Edits** feature. This powerful tool is designed to behave like an AI programmer, allowing us to edit multiple files across our entire code base based on a specific prompt. Unlike typical suggestions from Copilot, the edit mode directly applies changes to the code, offering us the choice to either accept or revert those modifications. We learned how to activate and utilize this mode effectively, emphasizing the importance of providing the full context of the code base to ensure accurate and comprehensive edits.

To enable Copilot Edits:

1. **Select Edit Mode**: Use the dropdown menu to choose the edit mode.
2. **Add Context**: Ensure that the entire folder or significant parts of the code base are included for context, allowing the tool to make necessary changes across different files.
3. **Provide Prompt**: Enter a specific command or request for change (e.g., refactoring, renaming functions).
4. **Review Edits**: View the diff to see proposed changes and decide whether to accept or discard them.

WHAT WE LEARNED

- GitHub Copilot Edits can edit multiple files based on a single prompt.
- It directly applies code changes instead of merely suggesting them.
- Context is crucial; without it, Copilot might not edit all relevant files.
- Diff view provides a clear representation of what changes are being made.

HOW WE CAN APPLY IT

- **Refactoring**: Apply changes across multiple files efficiently.
- **Feature Addition**: Add new features that span across various parts of the code base.
- **Codebase Modernization**: Update legacy code to adhere to modern coding practices.
- **Bug Fixing**: Resolve issues that affect multiple interconnected files.

TIPS AND TRICKS

- **Full Context**: Always include the entire code base or relevant sections to ensure comprehensive edits.
- **Preview Changes**: Use the diff view to understand the impact of edits before applying them.
- **Rollback Capability**: If changes are not satisfactory, utilize the rollback feature to revert edits.
- **Combine Modes**: Use ask mode for smaller, isolated changes and edit mode for broader updates.

EXAMPLES

Refactoring a Function

```
# Initial inefficient function
def calculate_product(a, b):
    result = 0
    for _ in range(b):
        result += a
    return result

# After Copilot Edit
def multiply(a, b):
    return a * b
```

Adding a Feature Across Files

1. **Prompt**: "Add a new field for tasks called `dueDate`."

Models.py File:

```
class Task:
    def __init__(self, task_id, title, description, completed, due_date):
        self.task_id = task_id
        self.title = title
        self.description = description
        self.completed = completed
        self.due_date = due_date
```

Update UI and Logic: Ensure user inputs cover the new field, and logic processes it accurately throughout relevant files.

Including External Data

```
def fetch_gold_price():
    # Edits included finding a suitable API that doesn't require a key
    response = requests.get('https://api.example.com/gold')
    return response.json()['goldPrice']
```

This lesson empowers us with the efficiency of AI-driven coding, significantly reducing the time and effort required to make extensive edits across complex code bases.