F-01 - AI Code Completion in GitHub Copilot

SUMMARY

In this lesson, we explored GitHub Copilot's **AI code completion**, also known as **ghost text**. This feature predicts the continuation of code as we type, offering suggestions word-by-word, line-by-line, or even for multiple lines. Here's how it works and how we can make the most of it:

1. **Activating Copilot**: Ensure GitHub Copilot is set up in your editor, then open any code file. As you start typing, Copilot will begin suggesting completions. For instance, in Python, typing `def calculate_area_of_rectangle` leads Copilot to suggest a function with parameters `length` and `width` and a return statement that calculates the area.

2. **Accepting Suggestions**:

    - Press `Tab` to accept the full ghost text.
    - If you want partial acceptance, use `Ctrl + Right Arrow` to navigate specific parts you wish to accept.

3. **Viewing Alternative Suggestions**:

    - Use `Alt + [ or ]` or `Ctrl + Enter` to toggle through different suggestions for your current function or code block.

4. **Forcing Suggestions**:

    - If suggestions aren't appearing, press `Alt + Back Slash` to prompt Copilot to generate completions.

5. **Customizing Settings**: If Copilot's suggestions become overwhelming, you can tailor its settings:

    - Navigate to **Copilot** > **Configure Code Completions** > **Edit Settings**.
    - Adjust settings to enable/disable Copilot for specific languages.

WHAT WE LEARNED

- How GitHub Copilot's AI code completion functions.
- Key shortcuts for managing suggestions.
- How to view and select alternative code suggestions.
- Forcing code completions.

- Customizing Copilot settings for different programming languages.

HOW WE CAN APPLY IT

- **Function Definitions**: Quickly generate function skeletons including parameters and return statements.
- **Script Writing**: Use partial suggestions for designing scripts efficiently.
- **Documentation**: Create detailed and varied docstrings.
- **Experiment with Code Variations**: Explore different ways to implement similar functionalities.

TIPS AND TRICKS

- **Precise Prompts**: The better described your initial code, the more accurate and helpful the predictions.
- **Customizing Output**: Modify Copilot settings for specific file types to streamline your workflow.
- **Shortcuts**: Familiarize yourself with keyboard shortcuts to swiftly manage suggestions.

EXAMPLES

```
# Example 1: Simple Function
def calculate_area_of_rectangle(length, width):
    """
    Calculate the area of a rectangle.

    Parameters:
    length (float): The length of the rectangle.
    width (float): The width of the rectangle.

    Returns:
    float: The area of the rectangle.
    """
    return length * width


# Example 2: Date Difference Function
def calculate_date_difference(date1, date2, difference_type='days'):
    """
    Calculate the difference between two dates in different time units.

    Parameters:
    date1 (datetime): The first date.
    date2 (datetime): The second date.
```

```
    difference_type (str): The unit of difference ('days', 'weeks', 'months', 'years', 'minutes',

    Returns:
    int: The difference in the specified unit.
    """
    if difference_type == 'days':
        return (date2 - date1).days
    elif difference_type == 'weeks':
        return (date2 - date1).days // 7
    elif difference_type == 'months':
        # Implement month difference logic
        pass
    elif difference_type == 'years':
        return date2.year - date1.year
    elif difference_type == 'minutes':
        return (date2 - date1).total_seconds() // 60
    elif difference_type == 'hours':
        return (date2 - date1).total_seconds() // 3600
```

By utilizing GitHub Copilot's AI code completion, we can significantly accelerate coding productivity and discover creative solutions for programming challenges.