100 XP

Examine GitHub Copilot inline chat, smart actions, and quick chat

6 minutes

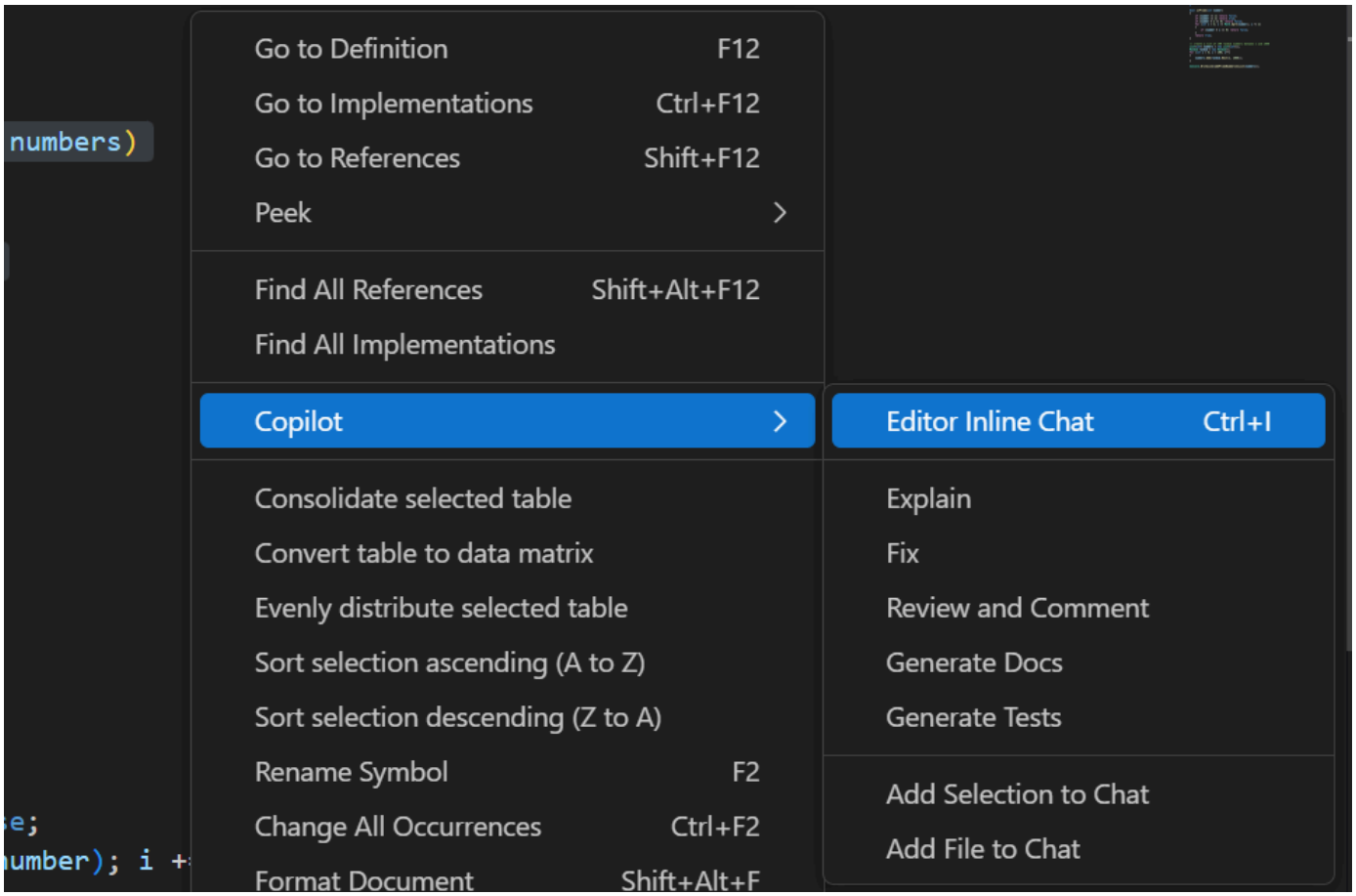
In addition to the Chat view, GitHub Copilot Chat provides several other ways to interact with the AI pair programmer. These include inline chat, smart actions, and quick chat.

Inline chat

One of GitHub Copilot Chat's key features is answering questions inline as you're coding. This allows you to harness the power of AI directly within your code editor, with minimal interruption to your development workflow.

The inline chat feature is only accessible when a file is open in the editor. There are several ways to open an inline chat session:

- Select **Editor Inline Chat** from the Copilot Chat menu.
- Use a keyboard shortcut: **Ctrl+I**.
- Right-click inside the code editor, select **Copilot** from the context menu, and then select **Editor Inline Chat**.



You can use an inline chat session to ask Copilot questions as you write or iterate your code.

When you open an inline chat session, the following interface is presented in the Editor.

Consider the following scenario:

- You are developing a C# application that processes a numeric list. The data is processed in batches of 100.
- You use the code snippet below to create a list of 100 random integers between 1 and 1000 that you can use for testing.

C#

```
// create a list of 100 random number between 1 and 1000
List<int> numbers = new List<int>();
Random random = new Random();
for (int i = 0; i < 100; i++)
{
```

```
    numbers.Add(random.Next(1, 1000));
}
```

The code is working fine. However, what if the requirements change? What if you need to ensure that the list doesn't include any duplicate numbers?

To update your code using GitHub Copilot auto completions, you would need to:

1. Delete the existing code.
2. Create a new comment that reflects your updated requirement.
3. Use GitHub Copilot to generate a new code snippet from the updated comment.
4. Review the suggestions and select the best option.

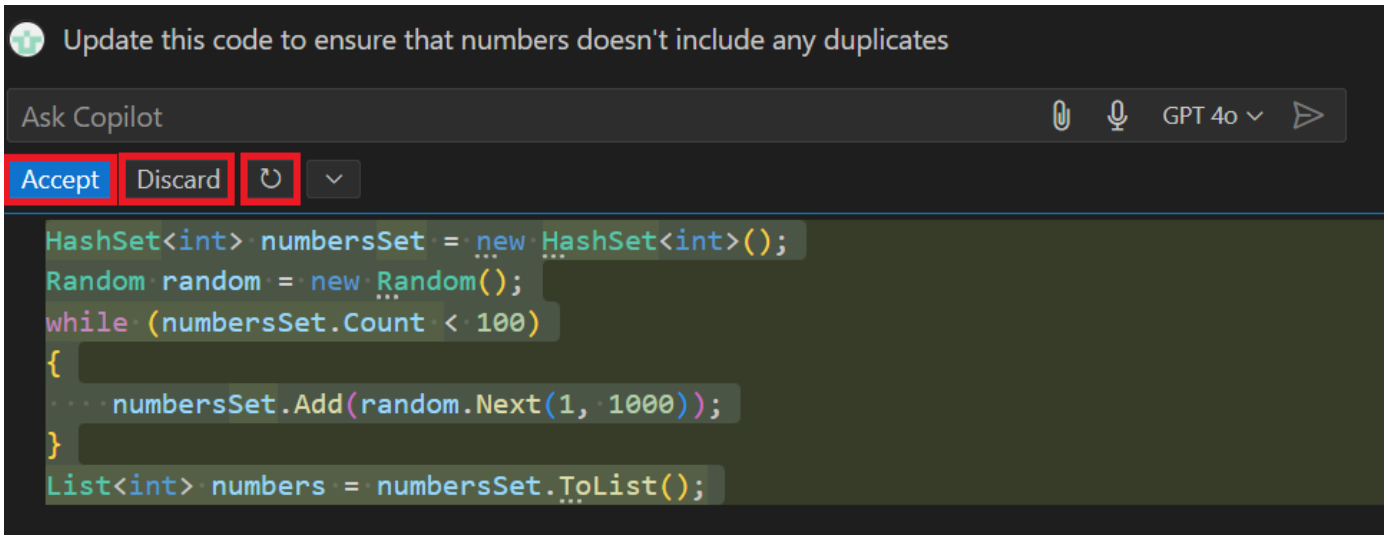
Luckily that isn't necessary. A better solution is to use GitHub Copilot Chat to suggest the required code updates.

You can use an inline chat session to suggest a code update as follows:

1. Highlight the code snippet that you want to update.
2. Press `Ctrl` + `I` to open the inline chat.
3. Ask GitHub Copilot Chat how to complete the update.
4. Review the suggestions and select the best option.

In this case, you could ask: `Update this code to ensure that numbers doesn't include any duplicates`

If you like the proposed code updates, you can select **Accept** and continue coding.



If you don't like the proposed updates, you can ask Copilot Chat to generate another suggestion by selecting the **Rerun...** icon. The Rerun icon is displayed as a circular arrow that appears below

the prompt textbox.

If you rerun the prompt and still don't get the results you need, try updating your prompt with additional context. Copilot Chat generates better suggestions when the prompt is clear, succinct, and accurate. You can choose **Discard** to close the inline chat without making any changes.

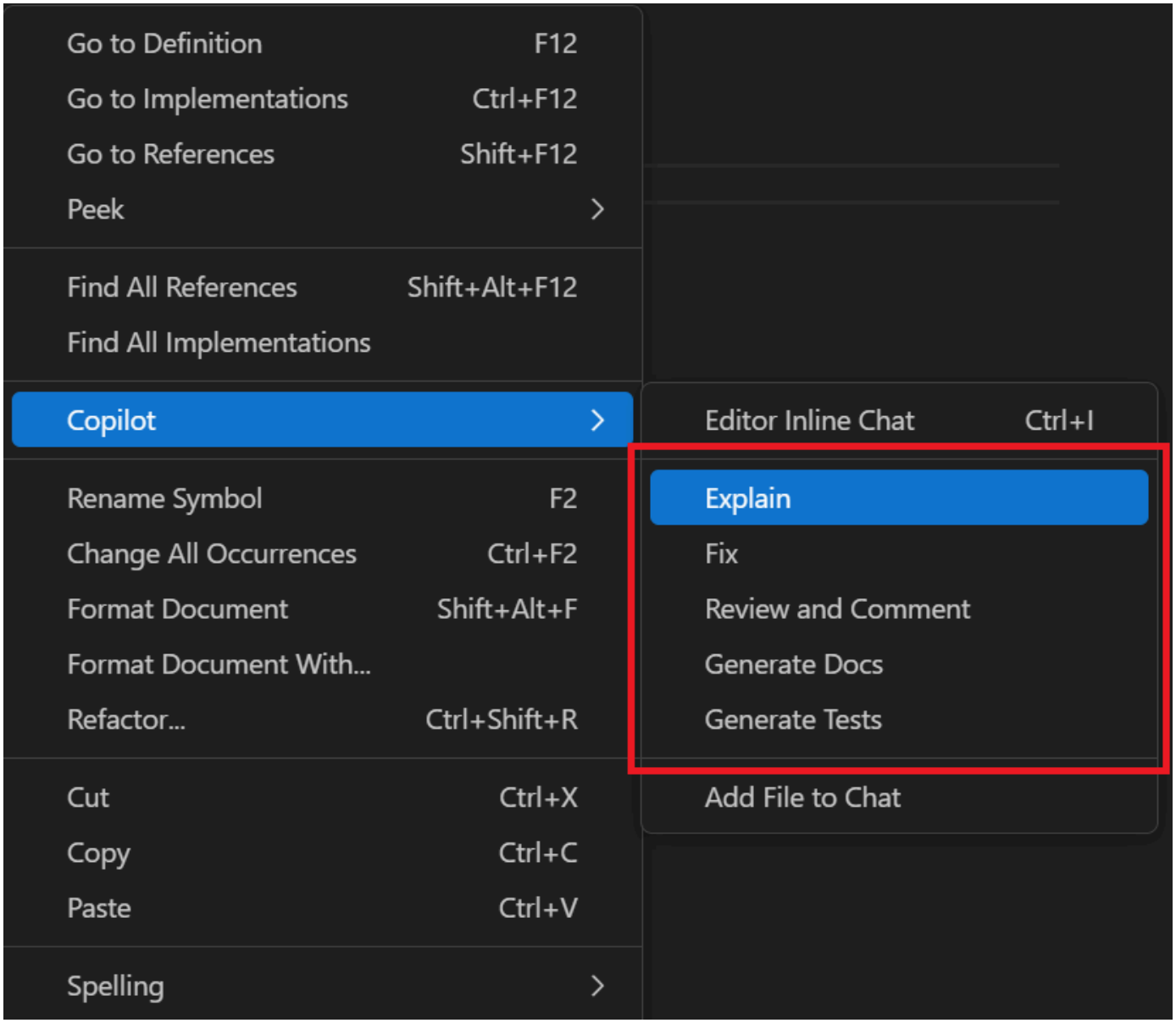
Smart actions

Some tasks are so common that they can be automated. GitHub Copilot Chat includes smart actions that allow you to complete certain tasks without having to write a prompt. Smart actions are predefined actions that you can run directly from the Visual Studio Code editor to perform common coding tasks quickly and easily.

The following smart actions are available:

- **Explain**: The `Explain` action generates a natural language description of the selected code. This can be useful for understanding the purpose and functionality of a code snippet. The `Explain` action is especially useful if you're working with unfamiliar code or need to explain the code to others.
- **Fix**: The `Fix` action suggests a fix for the selected code. This can be helpful if you encounter an error or issue in your code and need guidance on how to resolve it. Copilot Chat can suggest changes to variables, control structures, or function calls that might resolve the issue.
- **Review and Comment**: The `Review and Comment` action provides a code review of the selected code. This can be useful for identifying potential issues, improving code quality, and ensuring best practices are followed. The `Review and Comment` action can help you identify bugs, performance bottlenecks, and security vulnerabilities in your code.
- **Generate Docs**: The `Generate Docs` action creates documentation for the selected code. This can be useful for documenting your codebase, especially if you're working on a project with multiple contributors or need to share your code with others.
- **Generate Tests**: The `Generate Tests` action creates unit tests for the selected code. This can be helpful for ensuring the correctness and reliability of your code, especially if you're working on a project with complex logic or critical functionality.

To access the smart actions, right-click on a code snippet in the editor and select the desired action from the Copilot context menu.



Once again, let's consider the prime number app.

In the previous section, the inline chat suggested the following code to ensure that `numbers` doesn't include any duplicates:

C#

```
// create a list of 100 random numbers between 1 and 1000
List<int> numbers = new List<int>();
Random random = new Random();
while (numbers.Count < 100)
{
    int randomNumber = random.Next(1, 1000);
    if (!numbers.Contains(randomNumber))
    {
        numbers.Add(randomNumber);
```

```
    }
}
```

Although this code ensures a collection of unique elements, there could be opportunities for improvement.

You can use the **Review This** smart action to check code selections. To use the **Review This** smart action:

1. Select a code snippet that needs review. In this case, the code that generates the list of random numbers.
2. Select **Review This** from the Copilot context menu.
3. Review the suggested updates and select **Accept** to apply the changes.

For the code snippet that generates random numbers, the **Review This** smart action suggests the following:

- Initializing `List<int> numbers` with a capacity of `100` to avoid multiple resizes during population.
- Using a `HashSet<int>` rather than a `List<int>` for faster lookups when checking for duplicates.

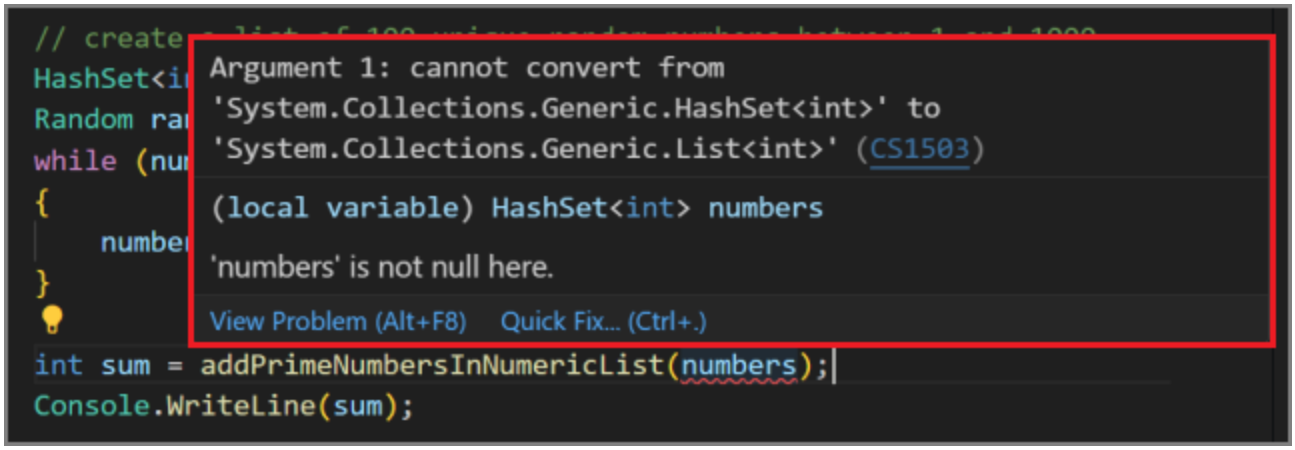
In this case, the **Review This** smart action leads you to using `HashSet<int>` rather than `List<int>` to improve performance.

C#

```
// create a list of 100 random numbers between 1 and 1000
HashSet<int> numbers = new HashSet<int>();
Random random = new Random();
while (numbers.Count < 100)
{
    int randomNumber = random.Next(1, 1000);
    if (!numbers.Contains(randomNumber))
    {
        numbers.Add(randomNumber);
    }
}

int sum = addPrimeNumbersInNumericList(numbers);
Console.WriteLine(sum);
```

Although using a `HashSet` works efficiently to ensure a collection of unique elements, the update creates a new problem. The `addPrimeNumbersInNumericList` method expects a `List<int>` as input, but the updated code creates a `HashSet<int>`. This results in the following compilation error:



Fortunately, you can use the **Fix** smart action to correct the error. To use the **Fix** smart action:

1. Right-click on the code snippet that needs to be fixed. In this case, the code underlined in red.
2. Select **Fix** from the Copilot context menu.
3. Review the suggested fix and select **Accept** to apply the changes.

The `Fix` smart action generates one or more suggestions to fix an error. In this case, one of the suggestions uses the `ToList()` method to convert the `HashSet` to a `List` inside the call to `addPrimeNumbersInNumericList`. The resulting code will look similar to the following code snippet:

C#

```
// create a list of 100 random numbers between 1 and 1000
HashSet<int> numbers = new HashSet<int>();
Random random = new Random();
while (numbers.Count < 100)
{
    int randomNumber = random.Next(1, 1000);
    if (!numbers.Contains(randomNumber))
    {
        numbers.Add(randomNumber);
    }
}

int sum = addPrimeNumbersInNumericList(numbers.ToList());
Console.WriteLine(sum);
```

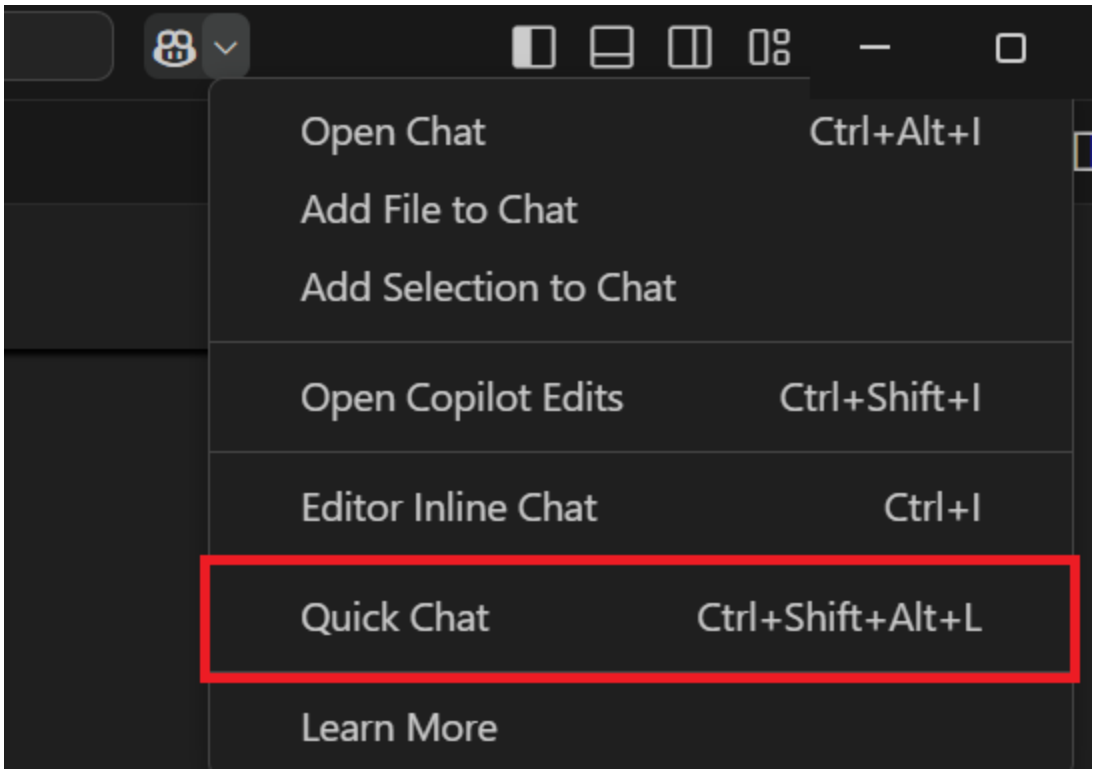
As always, review the suggestion provided by Copilot Chat. There may better solutions. You can select **Accept** to update your code with the proposed fix.

Quick Chat

The third option that GitHub Copilot Chat provides for chat interactions is Quick Chat.

You can use the Quick Chat feature of GitHub Copilot Chat when you want to have a quick, interactive conversation with the AI. This can be useful when you're trying to solve a problem, need help with understanding a piece of code, or want to brainstorm ideas. It's designed to feel like you're chatting with a fellow developer, making it a great tool for collaborative problem-solving.

To open a Quick Chat window, you can press **Ctrl+Shift+Alt+L** in the Visual Studio Code editor, or you can open Quick Chat from the Copilot Chat menu.



Summary

GitHub Copilot Chat provides several ways to interact with the AI pair programmer, including inline chat, smart actions, and Quick Chat. These features allow you to ask questions, get code suggestions, and perform common coding tasks quickly and easily.

Next unit: Exercise - Examine GitHub Copilot settings and user interface features

Previous Next