# Coding with Copilot

Coding with GitHub Copilot

## On this page

# Coding with GitHub Copilot

## GitHub Copilot vs GitHub Copilot Chat

GitHub Copilot and Copilot Chat serve different purposes, and the choice between them would be based on the specific use-case and the interaction you're looking for. Here are some scenarios in which you might choose one over the other.

GitHub Copilot:

- Direct Code Writing:
  - When you're actively writing code and want auto-suggestions inline within your IDE.
  - For quick code snippets, standard patterns, or commonly-used functions.
- Seamless IDE Integration:

  - Copilot is directly integrated into the Visual Studio Code (VS Code) environment, so if you're coding within VS Code, it provides a seamless experience.
- Solo Development:
  - If you're working on your own, Copilot can act like a pair-programming partner, suggesting code as you type.

Copilot Chat:

- In-Depth Assistance:
  - If you want a more interactive and explanatory assistance, akin to a chat with a mentor or experienced developer.
  - Great for understanding reasoning or getting more detailed explanations behind a piece of code.
- Learning and Teaching:
  - Beneficial for newcomers to a programming language or concept, as the chat format can provide a more guided, step-by-step walkthrough.
- Collaborative Scenarios:
  - If you have a complex problem and expect a bit of back-and-forth interaction to clarify and iterate on your requirements.
  - If you're working in a team setting and want to discuss a coding challenge or problem interactively, having a chat format might be beneficial to gather multiple perspectives.

While both tools are powered by advanced models and can aid in code generation, the choice boils down to the style of interaction you're seeking. For inline, direct coding suggestions, go with Copilot. For a more interactive, chat-based guidance, opt for Copilot Chat. Remember to always review and understand the code generated by either tool to ensure its correctness and appropriateness for your specific scenario.

## Create Workspace

Generate search parameters. `/search` is a keyword that can be used to generate code related to searching for resources in an API or database. When you type /search in your editor, GitHub Copilot may suggest code snippets that can be used to perform a search operation, such as querying an API endpoint with search parameters or filtering a database query based on search criteria.

Bring a workspace to life by using the `/createWorkspace` commands in copilot chat. Ask for a popular project type with `/createWorkspace` by listing out the type of project you need. For example, if you need a reactjs application with a mongodb database, it will send back the folder structure you need. After looking over the workspace copilot creates for you, you can select “create workspace” to create and open the project directory as a new workspace.

Just a friendly note, if you are not see the option listed when you type /, make sure your copilot chat is switched to the pre-release version and updated to the newest version in your marketplace. This was a new upgrade added to the copilot family and you may not see the option listed if you have not updated or using the release version.

## Create Notebook

When you use the `/createNotebook` command, Copilot generates a Jupyter notebook outline tailored to your requirements. If you find the outline suitable, you can simply click 'Create Notebook' to initiate the notebook creation process, and populate the code cells based on the suggested outline. Be sure to be specific with your requirements to make the most out of this slash command. For example, if you want to create a notebook for data analysis, you can specify the programming language, libraries, and data source you want to use. Copilot will then generate a notebook with the appropriate imports, and a code cell to load the data from the specified source. You can then add additional code cells to perform your analysis.

## Data Generation

To utilize Copilot for data generation, you can start by describing your data generation needs and write code that outlines the process. Copilot will then provide code suggestions based on the context you've provided.

Ways to generate data using GitHub Copilot:

1. Mock Data Generation: Chat with Copilot as if you were requesting data. For example: a. Can you help me generate a random name?
2. Generate Random Numbers: You can chat with Copilot to request random numbers, specify a range, or define the type of random data you need: a. I need a random integer between 1 and 100.
3. Data Validation: Request code to validate data or perform checks on data quality: a. Can you generate code to check if an email address is valid?
4. Data Transformation: Request code for transforming data, such as converting formats or applying mathematical operations: a. Convert this date to a different format.
5. Generate User Profiles: If you're working on a project that requires user profiles, you can have a conversation to generate user data: a. Can you create a user profile with a name, email, and age?
6. Database Query Generation: You can chat with Copilot to request code for generating database queries or mock database entries: a. Generate SQL code to insert a new record into the "users" table.

7. Complex Data Structures: For more complex data structures like JSON objects, you can have a conversation with Copilot to generate structured data: a. Create a JSON object with properties "name" and "address."
8. Data Formatting: Request code for formatting data according to specific requirements, such as dates, currency values, or other data types: a. Format this date to be displayed in a user-friendly format.

It's essential to review the generated code to ensure it aligns with your specific requirements and data generation rules. Copilot can save you time by providing boilerplate code and suggestions, but you should always validate and adapt the code as needed to fit your project's unique data generation needs.

## Testing with GitHub Copilot

### Unit Testing

This cutting-edge tool leverages the power of artificial intelligence to assist developers in creating robust unit tests for their code. Copilot Chat analyzes your codebase, understands the logic and flow, and then suggests test cases that cover various scenarios, from edge cases to common usage patterns.

One of the remarkable features of Copilot Chat is its ability to generate tests that are contextually relevant. It comprehends the intricacies of your code and proposes tests that exercise critical paths and corner cases, which can be time-consuming to identify manually. This capability significantly reduces the burden on developers, allowing them to focus on writing code, knowing that their tests are comprehensive and effective.

Moreover, Copilot Chat promotes best practices in testing. It encourages developers to think about potential edge cases and exceptional scenarios that might otherwise be overlooked. By doing so, it not only enhances the quality of your codebase but also instills a culture of thorough testing within your development team.

### Test-Driven Development

Test-driven development (TDD) with GitHub Copilot represents a powerful synergy of software development practices and cutting-edge AI assistance. With TDD, developers follow a process of writing tests before implementing their code. It actively assists in crafting test cases, generating test templates, and suggesting test scenarios based on the code being written.

## Copilot Refining and Refactoring

### Refactoring Code

Refactoring is a critical aspect of software development. It involves modifying existing code to improve its structure, readability, or performance, without altering its external behavior. GitHub Copilot Chat provides a new dimension to refactoring by offering AI-driven insights and suggestions. Copilot Chat offers an interactive environment where you can have real-time discussions with the AI model, ask questions, seek clarifications, and receive suggestions. When it comes to refactoring, this becomes an avenue for both seeking guidance on best practices and directly obtaining refactored code snippets.

In the example, we sought Copilot's assistance to handle an exception with a function that we wrote. Copilot was able to see the code and what we are trying to accomplish and provide us with an updated version that doesn't break the functionality.

Another really great example of refactoring your code with Copilot would be for example in our Calculator file earlier. If we Copilot suggested a calculator.html file that included the javascript embedded on that page, it would make testing a bit less readable and difficult. We can simply ask Copilot to refactor that code and extract the javascript needed into its own file, calculator.js

### Refining Code

CPU Usage auto insight in the profiler: A profiler can help you make informed decisions quickly by providing a visual depiction of execution times and CPU usage for your application. Copilot now gives you detailed information via CPU Usage auto insights. It now provides more detailed information and insights for specific methods and properties, including Enum.HasFlag, Enum.ToString, String.StartsWith, ConcurrentDictionary.Count and more. With Copilot, you can ask questions about functions on the identified hot paths in your code, which can help you produce more efficient or cost-effective code. Just click "Ask Copilot" to start exploring.

This feature is in Visual Studio

## Block Files

### .copilotignore

This feature allows users to block specific files and folders from being used as context for GitHub's AI-powered code suggestion tool, Copilot. When enabled, Copilot will not provide suggestions within the specified files and folders.

You can create a .copilotignore file in your project's repository and list the files, directories, or patterns that you want GitHub Copilot to exclude from its suggestions. This is particularly useful when you have test files, documentation, or other code-related files that you don't want to be considered when Copilot generates code completions or suggestions.

Some limitations to be aware of:

1. It does not work for Copilot Chat and Copilot Labs.
2. Every user accessing the repository containing the '.copilotignore' file MUST be under the feature flag. If not, there's a risk of Copilot accessing and providing suggestions in unwanted files.
3. The PoC has undergone only limited manual testing, so we cannot fully guarantee its efficacy at this stage.
4. We currently do not have a mechanism to provide definitive proof to users that Copilot has not accessed the files specified in the '.copilotignore' file.

### copilotignore repos

The Copilot Restricted Content feature allows repository owners to control the usage of Copilot across their codebases. This includes specific files, folders, or entire repositories, even if they're not hosted on GitHub. The goal is to provide CfB customers a flexible way of controlling what content Copilot can access for prompt crafting and where it can insert code.

To do this in your repository, you will go to your organization, click on “Copilot”, then click on “Restricted content” in the drop down. From there, you can specify the repositories and files within the repositories you wish to ignore. You can go based on a repository file, or specify a file across all repositories.

Use the YAML syntac with fnmatch notation to write rules and target specific paths.

The open beta is aimed for Github Universe 2023. The open beta will support VSCode code completion only, with other IDEs following shortly after.

## Debugging with GitHub Copilot

Copilot can provide you with meaningful help when you are debugging your code. For example, when an exception is thrown, it gives you the opportunity to start asking questions. Copilot has access to exceptions, call stack, local variables, and code. By forming good questions based on the right parts of the data Visual Studio has when you’re at an exception, Copilot Chat can provide useful insights and fixes for the issue.

Ways to GitHub Copilot can help you debug:

- **Error Messages and Fixes:** When you encounter errors in your code, GitHub Copilot can often provide suggestions to help you understand the error message and offer potential solutions. It might suggest code corrections to resolve common syntax or semantic issues.
- **Code Completion for Debugging Statements:** Copilot can assist you in writing debugging statements, such as console.log in JavaScript or print statements in Python. It can also generate code snippets to inspect variable values or print debugging information.
- **Refactoring:** Copilot can help you refactor your code to make it more readable and maintainable. By refactoring code, you may uncover and fix bugs in the process. Cleaner, well-organized code is often easier to debug.
- **Unit Testing:** Copilot can generate unit tests for your code. Writing comprehensive unit tests is a critical part of the debugging process, as it helps identify issues and verify that code behaves as expected.
- **Documentation and Comments:** Proper documentation can assist in understanding code, which can indirectly help with debugging. Copilot can help you write code comments, docstrings, and inline documentation to explain your code's functionality.
- **Code Review:** Copilot can assist with code review by suggesting improvements and identifying potential issues, including code smells and potential sources of bugs. Reviewing your code with Copilot's assistance can help you catch issues early.
- **Suggesting Debugging Techniques:** Although not a debugger itself, Copilot can suggest debugging techniques or approaches when you encounter issues. It may suggest you add breakpoints, examine specific variables, or trace the execution flow.

## GitHub REST API

GitHub Copilot REST APIs offer a wide range of capabilities including adding/removing teams to subscription, getting the seat assignment details, and more setting. They can easily retrieve crucial information about the number of Copilot seats available, ensuring that their subscription remains optimized for the team's needs.

These can be used for administrators and billing managers to gain information regarding the organizations subscription.

**Please note:** Only organization owners and members with admin permissions can configure and view details about the organization's Copilot for Business subscription.