

Introduction to

GitHub Copilot

T2065

04/07/2025

Agenda:

Topics Covered Today

Key Learnings / Concepts Understood:

Key Concepts with Definitions/ Code Snippet – Hands-on Practice

Challenges / Debugging Experience

Tasks/Assignments Completed

Additional Learning Resources / Notes

Q&A

Plan for Tomorrow

List of the topics to be Covered:

1. Introduction to GitHub Copilot

2. Core Features & Benefits
3. Use Cases & Applications
4. Getting Started: Installation & Setup
5. Effective Prompting Techniques
6. Advanced Features & Extensions
7. Practical Application & Projects
8. Performance & Limitations

Introduction to GitHub Copilot



GitHub Copilot

What is GitHub Copilot?

GitHub Copilot is one of the most reliable and advanced AI code completion tools. Using OpenAI's GPT-4 natural language prediction model, Copilot analyzes code context and suggests lines for you.

GitHub Copilot

is an advanced AI coding assistant



How GitHub Copilot works

GitHub Copilot uses Machine Learning capabilities to understand your code context and suggest code completions.

It's best used from the start of a project, generating functions, variable names, and algorithms that match your coding style.

It learns from the code you write and gets better over time.

GitHub Copilot supports various programming languages, enabling developers to utilize its functionality across various projects. Some of the supported languages include:

- Python
- JavaScript
- TypeScript
- Ruby
- Go
- PHP
- Swift
- Kotlin
- Rust
- C#
- C++
- Java
- HTML/CSS
- SQL

Key Features of GitHub Copilot

Key Features Of GitHub Copilot

Vast code repository and access to real-world examples

Faster coding

AI-based code suggestions

Comprehensive code snippets

Direct integration

Excellent understanding of programming idioms & patterns



Uses Real-World Code: It learned from all the code on GitHub, so it gives you relevant examples that real developers use.

Knows Best Practices: It understands common ways of writing code, so it suggests code that follows good patterns.

Gives You Whole Functions: It doesn't just suggest one line; it can write entire functions or code blocks for you.

Smart Suggestions: It reads your code to understand the context and gives you smart suggestions that fit your project.

Code Faster: It types code for you, which speeds up your work and reduces typos.

Works in Your Editor: It integrates directly into popular code editors like VS Code, so you get help right where you code.

GitHub Copilot use cases

GitHub Copilot offers a multitude of use cases, empowering developers to write code faster and more efficiently. Some specific use cases of Copilot include:

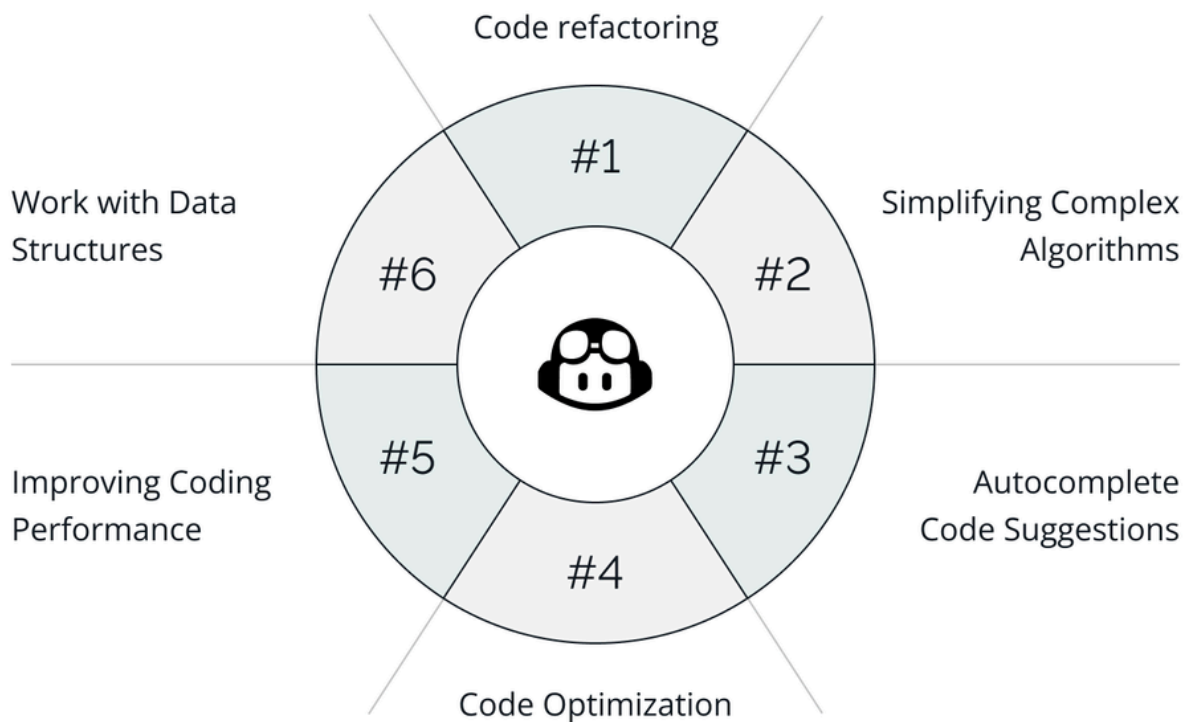
Autocomplete code suggestions to expedite coding and reduce manual typing.

Intelligent function and variable suggestions allow developers to focus on high-level logic rather than boilerplate code.

Efficient code refactoring and optimization suggestions, enhancing code quality and readability.

Simplifying complex algorithms and data structures by providing well-documented code examples from GitHub's extensive repository.

GitHub Copilot Use Cases



GitHub Copilot Limitations

- **Can Lack Creativity:** Copilot suggests code based on what it has seen before. This is helpful but might not offer a new or creative solution to a problem. You still need to think for yourself.
- **Not Great with All Languages:** It works best with popular languages like Python and JavaScript. For less common languages (like Elixir), the suggestions might not be as good or as frequent.

Applicability


Use Cases & Applicability




Aspect	ChatGPT	GitHub Copilot
Use Cases	Boilerplate code generation	Access to real-world examples
	Algorithm implementation	Programming patterns and idioms
	Data transformations	Specialized domains and libraries
Applicability	Text-based natural language processing	Specific programming scenarios and established patterns
	General-purpose coding tasks	


Performance Evaluation

Performance Evaluation

**ChatGPT**

- ✓ Code efficiency
- ✓ Code readability



**GitHub Copilot**

- ✓ Code accuracy
- ✓ Relevance to context
- ✓ Completeness of solutions

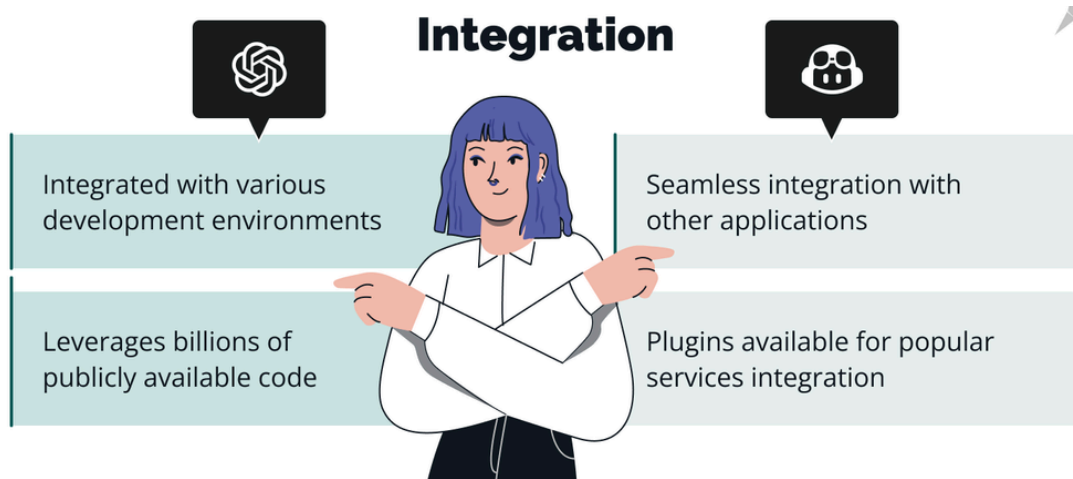
Examining the code offered by Copilot is crucial to understanding its performance in terms of generating code and comprehension. Evaluation methods may include:

Code Accuracy: Verifying the correctness and reliability of the code offered by Copilot through testing, debugging, and code review processes.

Relevance to Context: Assessing how well Copilot understands the context and requirements developers provide, ensuring that the generated code aligns with the intended functionality.

Completeness of Solutions: Evaluating whether Copilot offers comprehensive and robust code snippets that cover all necessary aspects of the given coding task.

Integration



GitHub Copilot

- ☐ GitHub Copilot is tightly integrated with several development environments, including VS Code, Visual Studio, Neovim, and JetBrains IDEs.
- ☐ This enables developers to use Copilot's code generation and completion features seamlessly within these IDEs.
- ☐ GitHub Copilot users can analyze and make software code and draw from billions of publicly available code across multiple programming languages.

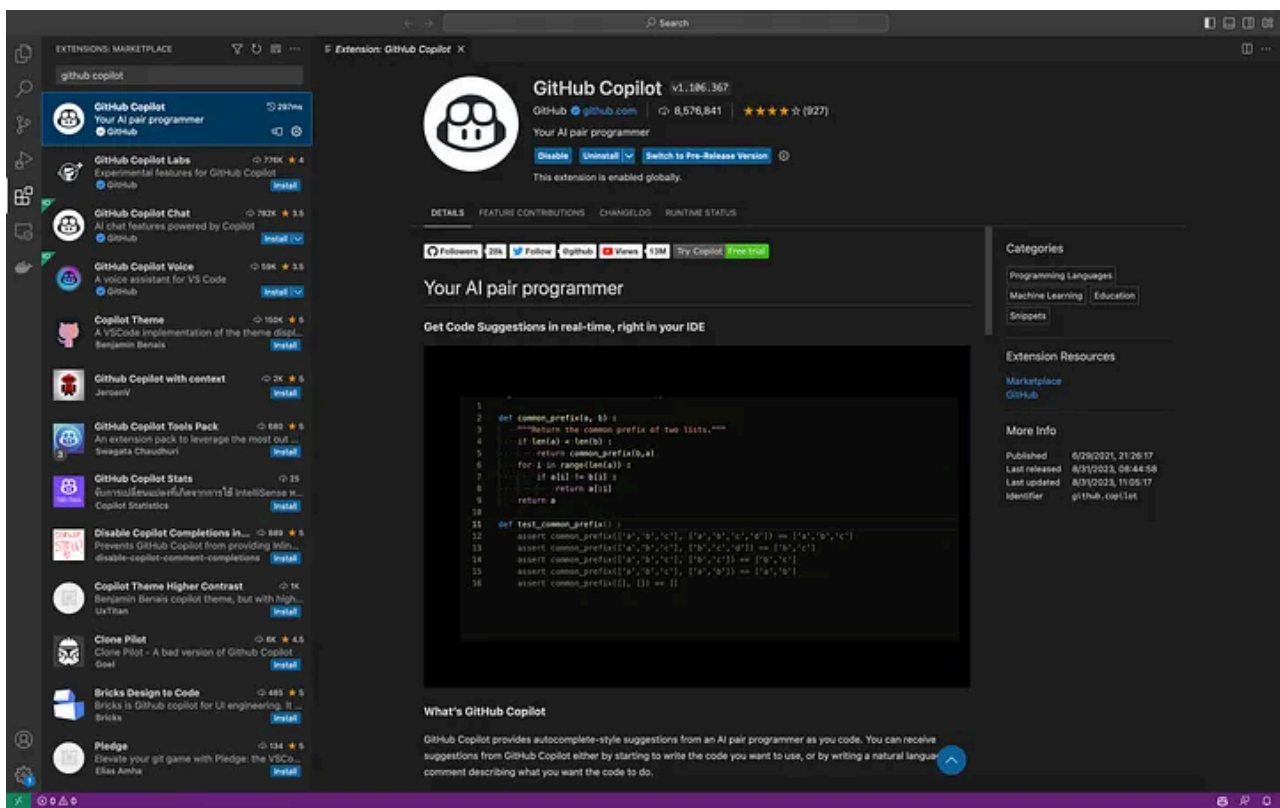
Integrating GitHub Copilot with VSCode

Before diving in, ensure your account has access to the GitHub Copilot service.

While this tutorial is about integrating with both editors, I suggest trying it first in VSCode. This will help you confirm access and functionality of GitHub Copilot.

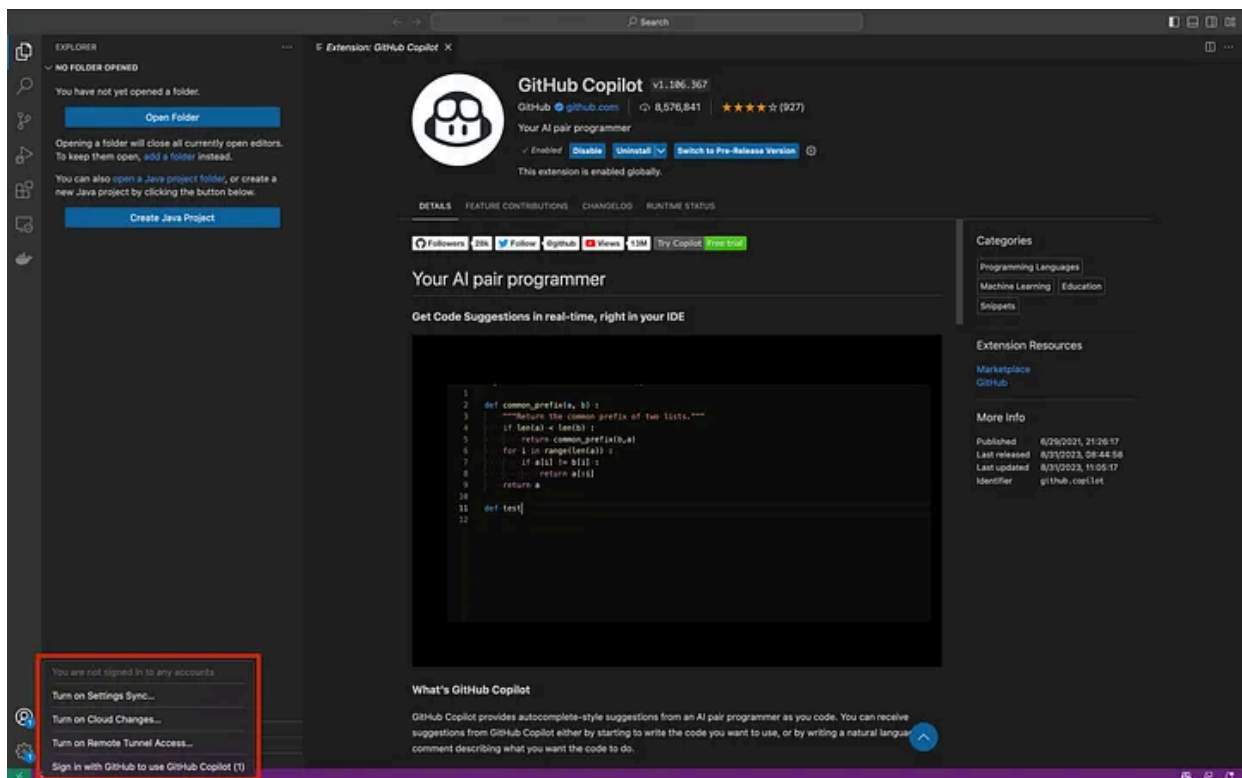
Here are the steps:

1. Install VSCode on your computer.
2. Click on ***Extensions*** in VSCode -> Search for GitHub Copilot in the ***Extensions Marketplace*** -> Install GitHub Copilot.



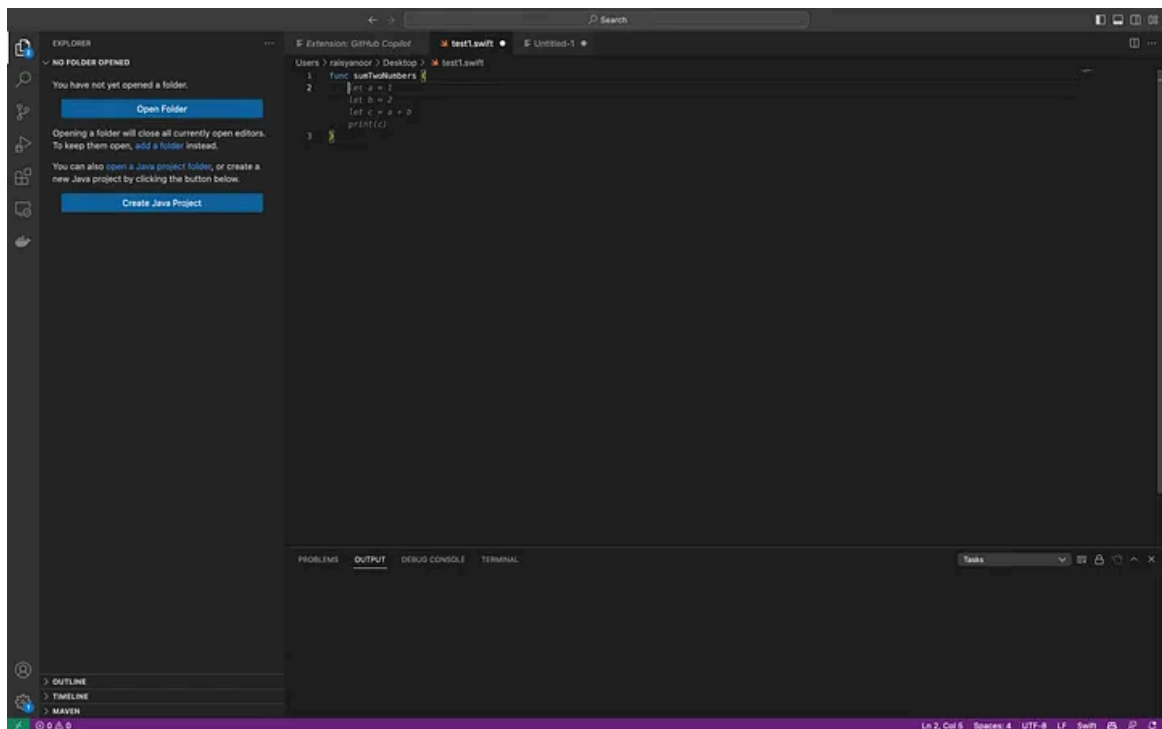
3. Ensure you enable GitHub Copilot after installing it.

4. In the bottom-left corner of VSCode, there's an icon to link your GitHub Account.

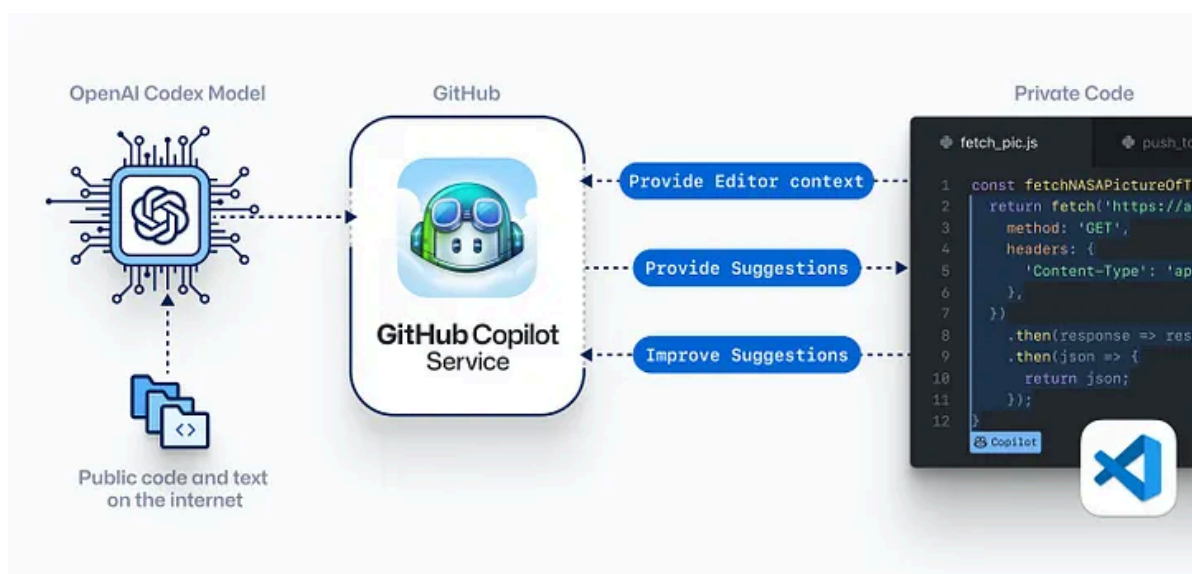


5. Click on “*Sign in with GitHub*” to activate GitHub Copilot.

6. If your GitHub account is already signed in on your default browser, the process should proceed automatically.
7. Create a new file in VSCode. For demonstration, I will use a .swift file since I develop iOS applications with Swift.
8. With GitHub Copilot activated, it'll suggest code. For instance, when I typed *“func sumTwoNumbers {}”*, GitHub Copilot provided a snippet to sum two numbers.



How does GitHub Copilot work?



It can suggest complete lines of code or entire functions by analyzing how you code.



1. GitHub Copilot can assemble code from user comments and predicts your code by just reading the function name you have declared.
2. It allows you to cycle through alternative suggestions and manually edit the suggested code.
3. It autofill repetitive code, or create unit tests for your methods.
4. The GitHub Copilot editor extension sends your comments and code to the GitHub Copilot service, which then uses OpenAI Codex to synthesize and suggest code.
5. it actually works by reading through all the open-source code on the GitHub repos worldwide and then collect the data and tries to find the best possible code related to it!
6. It is said to work great with repetitive code patterns so users can let it generate the rest of the code.
7. The AI assistant can also help you learn a new programming language.

Prompting Techniques

1. Explain Your Goal in a Comment

Before you write a function, write a comment that says exactly what it should do.

- **Bad:** // make a function
- **Good:** // function that adds two numbers together

2. Use Clear Names

Name your functions and variables descriptively. Copilot reads these names to understand your goal.

- **Bad:** function calc(d)
- **Good:** function calculateTotalPrice(items)

3. Give Step-by-Step Instructions

For a complex task, write comments inside your function like a to-do list. Copilot will help you with each step.

```
function submitOrder() {  
  // 1. Get items from the cart  
  
  // 2. Check if the user is logged in  
  
  // 3. Send to payment  
}
```

4. Keep Your Code Tidy

An organized code file helps Copilot understand the context of your project. This leads to much smarter suggestions.

In short: **Clear instructions = Better code from Copilot.**

Copilot Labs & Extensions

The old Copilot Labs tools are now part of Copilot Chat.

1. Copilot Chat (Inside your code editor)

This is a chat window where you can ask for help with your code.

- To Explain Code: Highlight any code and ask, "What does this do?"
- To Create Tests: Highlight a function and ask, "Write tests for this."

2. Copilot in the CLI (For your terminal)

This helps you find terminal commands without having to Google them.

- Example: Type ?? find all files larger than 5MB and it will give you the correct command to run.

Real-World Projects Using Github Copilot



Web App Development

- Build a to-do list or weather app with HTML/CSS/JS
- Copilot assists with form logic, local storage, APIs.

Automation Scripts

- Automate repetitive file handling, data scraping.

Legacy Code Refactoring

- Analyze and simplify old code
- Replace deprecated APIs with modern alternatives.

Copilot for Teams & Enterprises

Copilot for Business

- Centralized billing, usage analytics, and policy enforcement.
- Team-wide access to GitHub Copilot.

Pair Programming

- Developer A writes comments, Copilot suggests code.
- Developer B reviews suggestions and validates logic.

Repository Collaboration

- Used in shared repositories.
- Works well with GitHub PRs, Issues, and Actions.

Ethics, Licensing, and Legal Aspects

AI-Generated Code Licensing

- Some code might resemble open-source snippets.
- Be cautious about license compatibility.

Attribution & Open Source

- GitHub recommends reviewing Copilot code for attribution requirements.

Responsible AI Usage

- Don't blindly trust suggestions.
- Validate logic, test thoroughly.
- Avoid using Copilot for sensitive or proprietary logic.

Additional Learning Resources:

- ChatGPT
- scaler

- gfg
- tutorial points

Plan for Tomorrow: