



# GitHub Copilot Deep Dive



Presented by

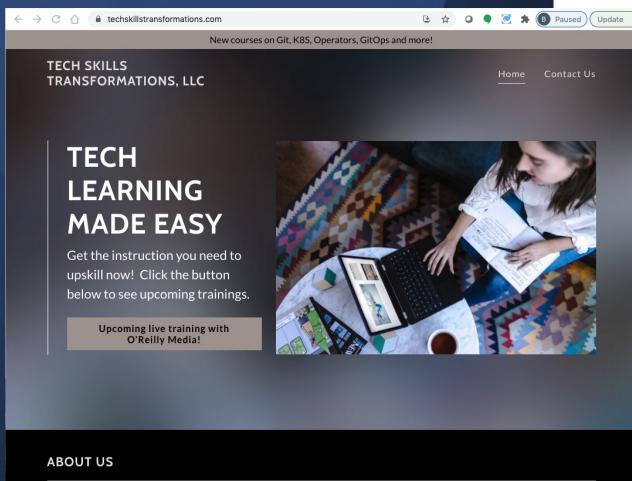
Tech Skills Transformations LLC

© 2023 Brent C. Laster & Tech Skills Transformations LLC



All rights reserved

# About me



- Founder, Tech Skills Transformations LLC
- R&D DevOps Director
- Global trainer – training (Git, Jenkins, Gradle, CI/CD, pipelines, Kubernetes, Helm, ArgoCD, operators)
- Author -
  - OpenSource.com
  - Professional Git book
  - Jenkins 2 – Up and Running book
  - Continuous Integration vs. Continuous Delivery vs. Continuous Deployment mini-book on O'Reilly
- <https://www.linkedin.com/in/brentlaster>
- @BrentCLaster
- GitHub: brentlaster



# Professional Git Book

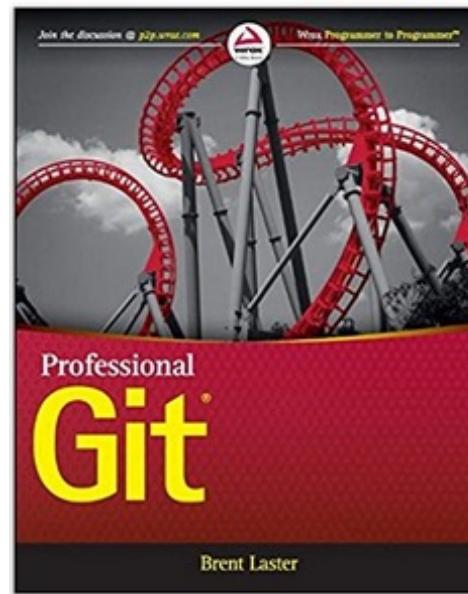
- Extensive Git reference, explanations,
- and examples
- First part for non-technical
- Beginner and advanced reference
- Hands-on labs

## Professional Git 1st Edition

by Brent Laster ▾ (Author)

★★★★★ ▾ 7 customer reviews

[Look inside](#) ↴



© 2023 Brent C. Laster &

Tech Skills Transformations LLC



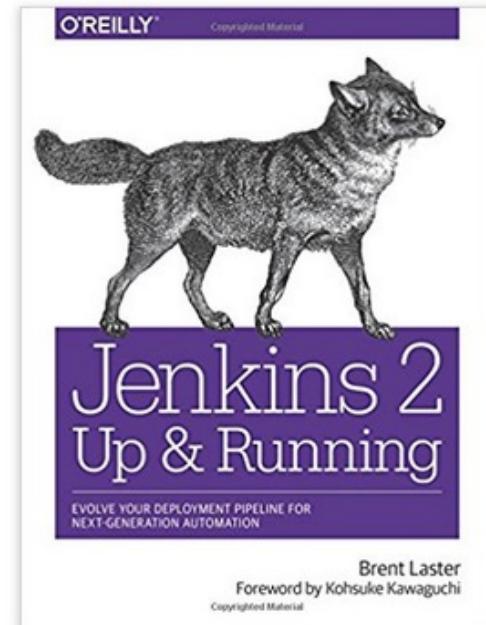
# Jenkins 2 Book

- Jenkins 2 – Up and Running
- “It’s an ideal book for those who are new to CI/CD, as well as those who have been using Jenkins for many years. This book will help you discover and rediscover Jenkins.” *By Kohsuke Kawaguchi, Creator of Jenkins*

★★★★★ 5 customer reviews

#1 New Release in Java Programming

[Look inside](#) ↴





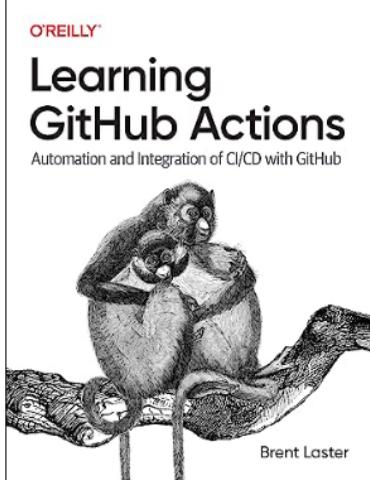
# GitHub Actions book

All Get the app Prime Day Back to School Add People Buy Again Gift Cards Recommendations IT Supplies Business Savings Amazon Basics EN Hello, Accou Group: Tech Skills Transfor

Books Advanced Search New Releases Best Sellers & More Children's Books Textbooks Textbook Rentals Best Books of the Month Your Company Bookshelf

 Watch now +

Books > Computers & Technology > Programming



O'REILLY

## Learning GitHub Actions

Automation and Integration of CI/CD with GitHub

Brent Laster

Roll over image to zoom in

Follow the Author


Brent Laster
Follow

### Learning GitHub Actions: Automation and Integration of CI/CD with GitHub 1st Edition

by Brent Laster (Author)

[See all formats and editions](#)

**Paperback**  
\$65.99 

1 New from \$65.99

Pre-order Price Guarantee. [Terms](#)

Automate your software development processes with GitHub Actions, the continuous integration and continuous delivery platform that integrates seamlessly with GitHub. With this practical book, open source author, trainer, and DevOps director Brent Laster explains everything you need to know about using and getting value from GitHub Actions. You'll learn what actions and workflows are and how they can be used, created, and incorporated into your processes to simplify, standardize, and automate your work in GitHub.

This book explains the platform, components, use cases, implementation, and integration points of actions, so you can leverage them to provide the functionality and features needed in today's complex pipelines and software development processes. You'll learn how to design and implement automated workflows that respond to common events like pushes, pull requests, and review updates. You'll understand how to use the components of the GitHub Actions platform to gain maximum automation and benefit.

With this book, you will:

- Learn what GitHub Actions are, the various use cases for them, and how to incorporate them into your processes
- Understand GitHub Actions' structure, syntax, and semantics
- Automate processes and implement functionality
- Create your own custom actions with Docker, JavaScript, or shell approaches
- Troubleshoot and debug workflows that use actions
- Combine actions with GitHub APIs and other integration options
- Identify ways to securely implement workflows with GitHub Actions
- Understand how GitHub Actions compares to other options

[^ Read less](#)



# What is GitHub Copilot?

- Cloud-based Generative AI tool
- Turns natural language prompts into coding suggestions
- Works across dozens of languages



- Trained on billions of lines of code
- Functions similar to an AI *pair* programmer\*
- Works on multiple IDEs
- Needs to be paired with Copilot Chat (in beta) for best experience/benefit



# How Copilot can help

- Code suggestions for nearly any current language
- Comment driven code creation
- Automatic test generation
- SQL generation (for queries, tables, indices, stored procs)
- Regular expression generation
- Pattern/mappings generation
- Document code
- Explain code
- Fixing code
- Reviewing code
- Translating code
- Generate K8s manifests
- What is the API to do...
- More!

- Generate boilerplate code
- Less time reading docs
- Less time recalling syntax
- Learning a new language
- Reduce cognitive load
- Let programmers focus on the problem more



Source: <https://github.com/orgs/community/discussions/49422>



# Challenges

- Should have human oversight/review
- Only as good as context/prompt
- Learning curve
- Sometimes needs "nudging"
- Hallucinations
- Date problem
- Length of response
- Always generative - not deterministic
- Some data going across



GitHub Copilot

Hi @techupskills, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.



# Signup individual #1

github.com/github-copilot/signup

Type ⌘ to search

gwstudent

Set status

Your profile

Your repositories

Your projects

Your codespaces

Your organizations

Your enterprises

Your stars

Your sponsors

Your gists

Upgrade

Try Enterprise

**Try Copilot**

Feature preview

Settings

Github Docs

Github Support

Sign out

## GitHub Copilot

Your AI pair programmer

### Start your 30-day free trial

GitHub Copilot uses the Codex AI model to offer coding suggestions.

- ✓ Get code suggestions in more than a dozen coding languages including Python, JavaScript, TypeScript, Go, and Ruby
- ✓ Plugs into IDEs including VS Code, Visual Studio, Neovim, and JetBrains
- ✓ Sign up for an annual subscription and get two months free!
- ✓ Cancel anytime before the trial ends to avoid getting billed

#### Pay frequency after trial

Monthly plan  
\$10/month

Yearly plan **Save \$20!**  
\$100/year

**Get access to GitHub Copilot**

Next: Confirm your payment details

GitHub Copilot is free for maintainers of popular open source software on GitHub and verified students. Check out the [free use FAQ](#) for more details.



# Signup Individual #2

github.com/github-copilot/signup/success

Paused

Type ⌘ to search

GitHub Copilot  
Your AI pair programmer

### Select your preferences

You can change these at any time from your account settings.

#### Suggestions matching public code

GitHub Copilot can allow or block suggestions matching public code. See [the GitHub Copilot documentation](#) to learn more.

Allow GitHub to use my code snippets for product improvements \*

Allow GitHub, its affiliates and third parties to use my code snippets to research and improve GitHub Copilot suggestions, related models and product features. More information in [Privacy FAQ](#).

Save and get started



# Signup Individual #3

The screenshot shows a web browser window with the URL [github.com/github-copilot/signup/settings](https://github.com/github-copilot/signup/settings). The page has a dark background with a futuristic, geometric pattern of colored shapes (purple, blue, green, yellow) in the foreground. At the top, there's a GitHub icon and a search bar with placeholder text "Type ⌘ to search". On the right side of the header, there are several icons and a user profile picture. The main content area features a large, stylized GitHub Copilot logo (a blue robot head with goggles). Below it, the text "GitHub Copilot is now ready to pair!" is displayed in bold white font, followed by the subtext "Add it to your editor and start building". A central call-to-action box contains the heading "Install the GitHub Copilot extension" with a gear icon. It provides instructions: "To use GitHub Copilot, you first need to install the GitHub Copilot extension. To install for your preferred code editor, check out these Getting Started guides:" followed by links to "Visual Studio Code", "Visual Studio", "JetBrains", and "Neovim". A pink callout bubble contains a tip: "If you have the extension already installed for your code editor and have it open, please restart your editor in order to access GitHub Copilot." Below this, a note says "Having problems installing? Visit the [Feedback forum](#)." A final tip at the bottom states: "Tip: Don't see your code editor? We currently only support the editors listed above. We're constantly looking to expand the editors we support, so stay tuned." At the very bottom of the box, there's a link to "Go to [Billing and plans](#) in your account settings to manage your GitHub Copilot subscription."



# GitHub Privacy Options

- Copilot collects
  - Prompts
  - Suggestions
  - User Engagement Data - usage info about events in IDE (such as suggestions accepted/dismissed, errors, general usage data, latency, features engagement)
- Data encrypted in transit and at rest, access-controlled
- Private code not shared
- Settings under

## GitHub Copilot

**Suggestions matching public code**

GitHub Copilot can allow or block suggestions matching public code. See [the GitHub Copilot documentation](#) to learn more.

Block ▼

**Allow GitHub to use my code snippets for product improvements \***

Allow GitHub, its affiliates and third parties to use my code snippets to research and improve GitHub Copilot suggestions, related models and product features. More information in [Privacy FAQ](#).

**Save**



# Post signup - billing info

The screenshot shows the GitHub Settings page for 'Billing' at [github.com/settings/billing](https://github.com/settings/billing). The user is Brent Laster (brentlaster). The left sidebar includes links for Public profile, Account, Appearance, Accessibility, Notifications, Access (Plans and usage, Spending limits, Payment information), Emails, Password and authentication, Sessions, and SSH and GPG keys. A yellow callout box on the right side informs about updates for LFS, stating: "Updates for LFS are coming soon: GitHub is changing the billing method for Git Large File Storage (Git LFS) from pre-purchased data packs to metered billing. More details [in this LFS billing FAQ](#)". The main content area is titled "Billing Summary". It shows the next payment of \$10.00 due by October 12, 2023. It notes that this amount does not include spend on usage of metered services. It also lists recurring charges for GitHub Copilot at \$10.00/month, due on Oct 12, 2023. Payment information links include Manage spending limit, View payment history, and Switch to yearly billing.

Click to go back, hold to see history

github.com/settings/billing

Settings

Brent Laster (brentlaster)

Your personal account [Switch to another account](#)

Go to your personal profile

Public profile

Account

Appearance

Accessibility

Notifications

Access

**Billing and plans**

Plans and usage

Spending limits

Payment information

Emails

Password and authentication

Sessions

SSH and GPG keys

**Updates for LFS are coming soon:**  
GitHub is changing the billing method for Git Large File Storage (Git LFS) from pre-purchased data packs to metered billing. More details [in this LFS billing FAQ](#).

**Billing Summary**

**Your next payment**

**\$10.00** Due by October 12, 2023

This amount does not include the spend on usage of metered service. View your [usage this month](#) below.

Payment information →

[Manage spending limit](#)

[View payment history](#)

[Switch to yearly billing](#)

**Your recurring charges & discounts**

GitHub Copilot \$10.00/month On Oct 12, 2023



# Setting allow/deny for public code match

15

The screenshot shows the GitHub organization settings page for 'skilldocs'. The left sidebar has a 'Policies and features' section selected under 'Access'. The main content area is titled 'GitHub Copilot policies and features'. It contains two sections: 'Policies' and 'Features'. In the 'Policies' section, there is a card for 'Suggestions matching public code' with a dropdown menu set to 'Allowed'. In the 'Features' section, there is a card for 'GitHub Copilot Chat' with a dropdown menu set to 'Enabled'.

skilldocs

Getting Started Actions - brentlaster...

https://github.com/organizations/skilldocs/settings/copilot/policies

Tech Skills Transformations LLC Public Trainings Docs  
Organization account Switch to another account

General

Access

- Billing and plans
- Repository roles
- Member privileges
- Import/Export
- Moderation

Code, planning, and automation

- Repository
- Codespaces
- Copilot

Access

- Policies and features
- Actions

GitHub Copilot policies and features

Policies

Suggestions matching public code

Copilot can allow or block suggestions matching public code.

Allowed

Features

GitHub Copilot Chat Beta

GitHub Copilot to explain a piece of code, propose bug fixes, or generate unit tests in a chat interface from your editor. If enabled, you agree to [pre-release terms](#).

Enabled



# Copilot for Individuals vs for Business

## Copilot for Individuals

\$ 10 per month /  
\$100 per year

- ✓ Plugs right into your editor
- ✓ Turns natural language prompts into code
- ✓ Offers multi-line function suggestions
- ✓ Speeds up test generation
- ✓ Blocks suggestions matching public code

[Start a free trial >](#)

## Copilot for Business new

\$ 19 per user  
per month

← Everything included in Copilot for Individuals, plus...

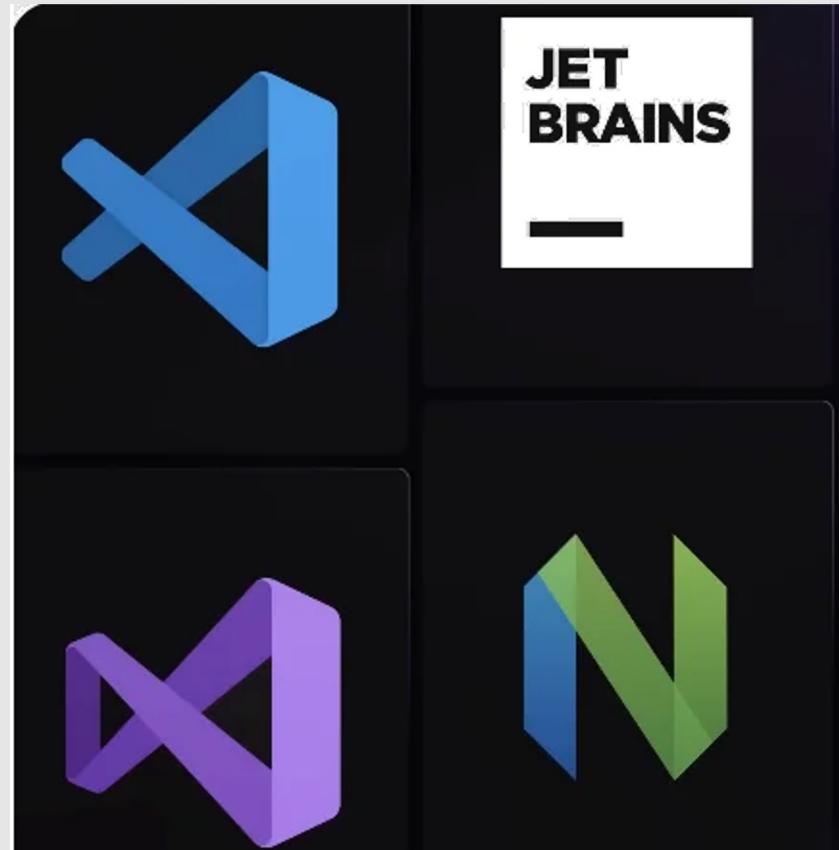
- ✓ Simple license management
- ✓ Organization-wide policy management
- ✓ Industry-leading privacy
- ✓ Corporate proxy support

[Buy Now >](#) [Contact Sales >](#)



# Supported IDEs

- Jet Brains
- VS Code
- Visual Studio Code
- Neovim
- Also Codespaces



# GitHub Copilot Extensions

**Extension: GitHub Copilot** v1.111.404



GitHub [github.com](#) | 8,953,788 | ★★★★★ (934)

Your AI pair programmer

Disable | Uninstall | Switch to Pre-Release Version | ⚙️

This extension is enabled globally.

DETAILS FEATURE CONTRIBUTIONS CHANLOG RUNTIME STATUS

Followers 29k Twitter Follow @github Views 13M Try Copilot Free trial

**Your AI pair programmer**

Get Code Suggestions in real-time, right in your IDE

```
1 def common_prefix(a, b):
2     """Return the common prefix of two lists."""
3     if len(a) < len(b):
4         return common_prefix(b, a)
5     for i, item in enumerate(a):
6         if item != b[i]:
7             return a[:i]
8     return a
```

**Extension: GitHub Copilot Chat** v0.7.1 Preview



GitHub [github.com](#) | 908,484 | ★★★★★ (57)

AI chat features powered by Copilot

Disable | Uninstall | Switch to Pre-Release Version | ⚙️

This extension is enabled globally.

DETAILS FEATURE CONTRIBUTIONS EXTENSION PACK RUNTIME STATUS

**GitHub Copilot Chat**

GitHub Copilot Chat is a companion extension to GitHub Copilot that houses preview Chat features. To learn more about GitHub Copilot Chat, [read our post](#).

Requirements

Review the [transparency note](#) to understand about usage, limitations and ways to improve Copilot Chat during the technical preview.

Extension Resources Marketplace Repository GitHub

**Extension: GitHub Copilot Labs** v0.15.1019 Preview



GitHub [github.com](#) | 816,556 | ★★★★★ (21)

Experimental features for GitHub Copilot

Install | ⚙️

DETAILS FEATURE CONTRIBUTIONS

Copilot Labs is a place for GitHub to experiment with new ideas before we take them into production. These experiments may never become part of the official product, or may change significantly before they become part of the supported product. Because they are experimental, features may break, not work as expected, or otherwise fail to meet our quality standards for production-ready code. All Copilot Labs features are provided "as-is" with no guarantees of support.

Finally, Copilot Labs is governed by separate terms. GitHub may collect more/different telemetry in Copilot Labs

**Extension: GitHub Copilot Voice** v0.21.1441 Preview



GitHub [github.com](#) | 66,824 | ★★★★★ (6)

A voice assistant for VS Code

Install | ⚙️

DETAILS FEATURE CONTRIBUTIONS CHANLOG

GitHub Copilot Voice Subscription Required

Categories

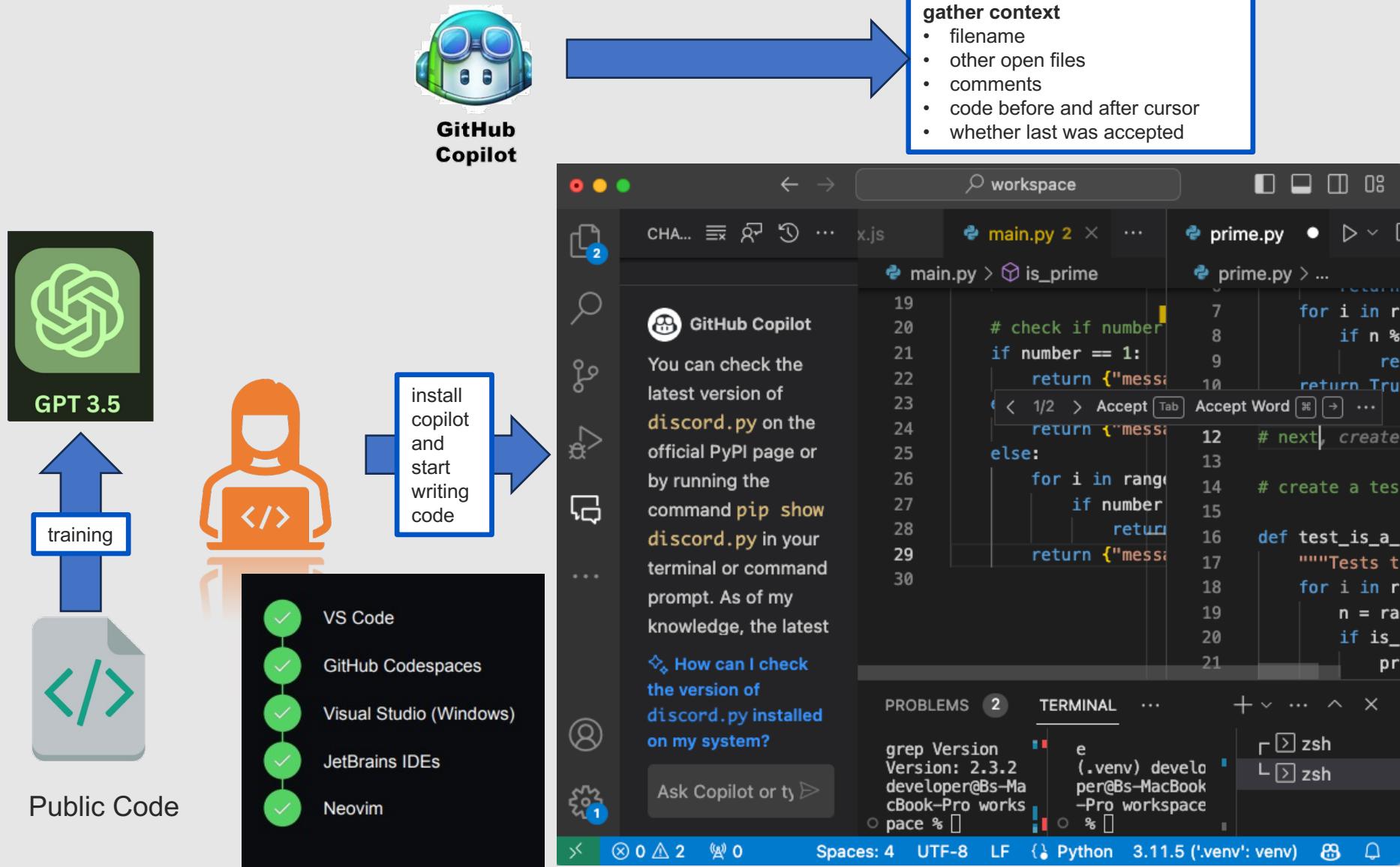
Other

Extension Resources Marketplace GitHub

**GitHub Copilot Voice**



# How Copilot works (high-level)

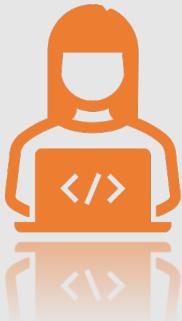




# How Copilot works (high-level)



**GitHub  
Copilot**



Public Code



gather context

- filename
- other open files
- comments
- code before and after cursor
- whether last was accepted

```

    main.py 2
    main.py > is_prime
    19
    20
    21
    22
    23
    24
    25
    26
    27
    28
    29
    30
    # check if number
    if number == 1:
        return {"messag
    < 1/2 > Accept Word
    return {"messag
else:
    for i in range(2, number):
        if number % i == 0:
            return {"messag
    return {"messag
    def test_is_a
        """Tests t
    for i in r
    n = ran
    if is_
    
```

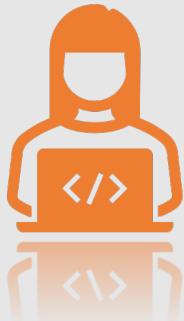
PROBLEMS 2 TERMINAL

grep Version  
Version: 2.3.2  
developer@Bs-Ma  
cBook-Pro works  
pace %

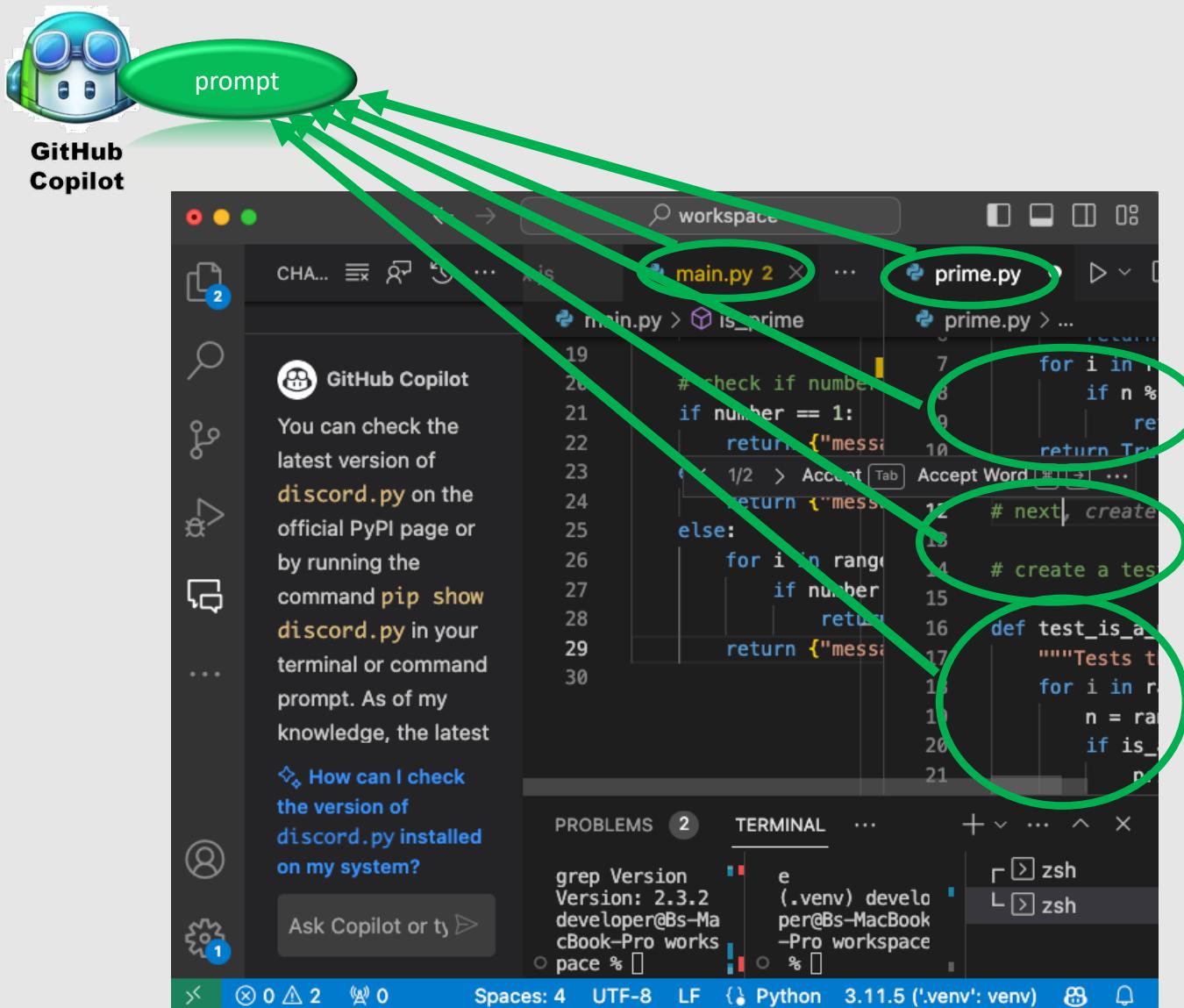
Spaces: 4 UTF-8 LF { Python 3.11.5 ('.venv': venv) 8 9



# How Copilot works (high-level)

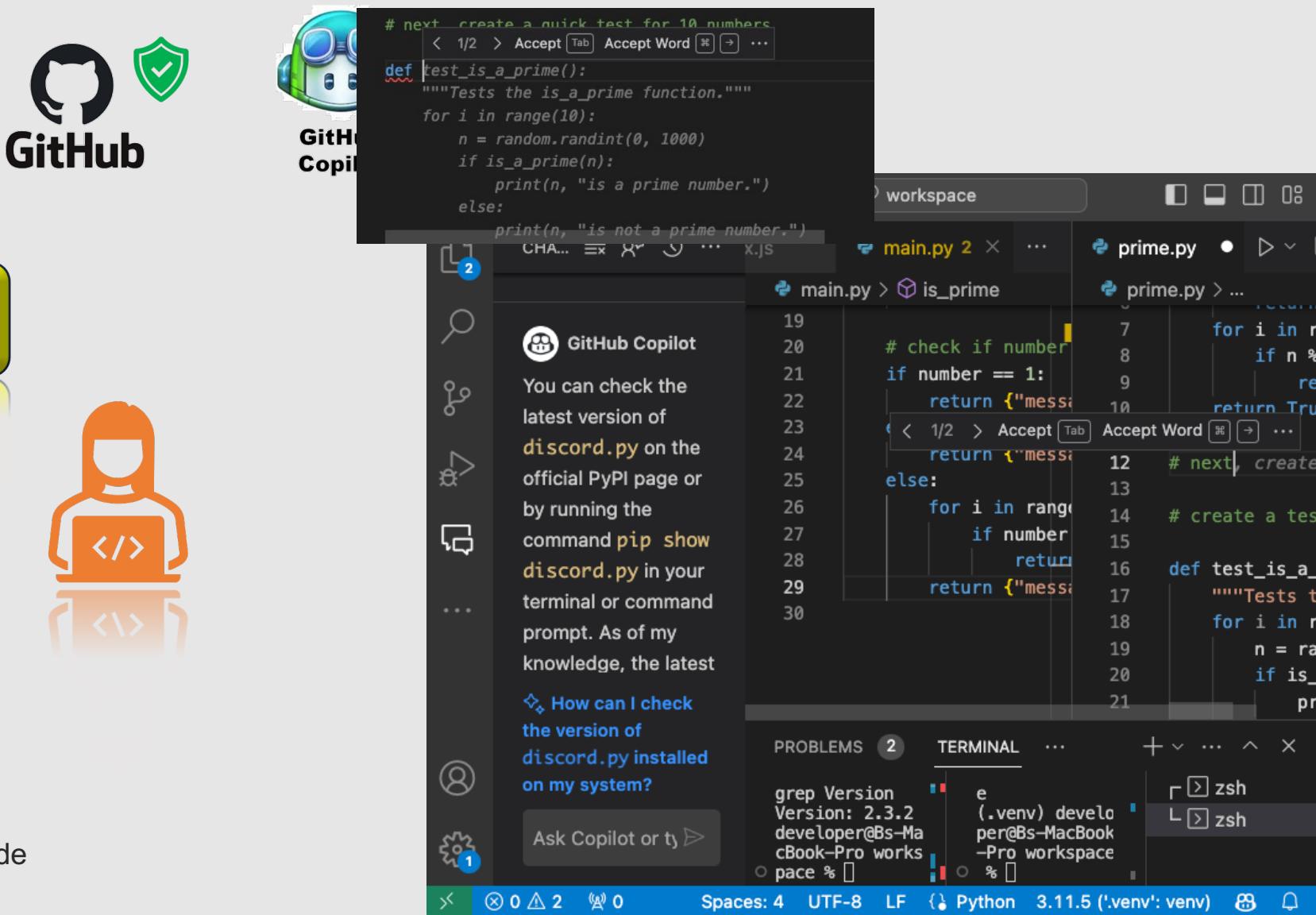


## Public Code



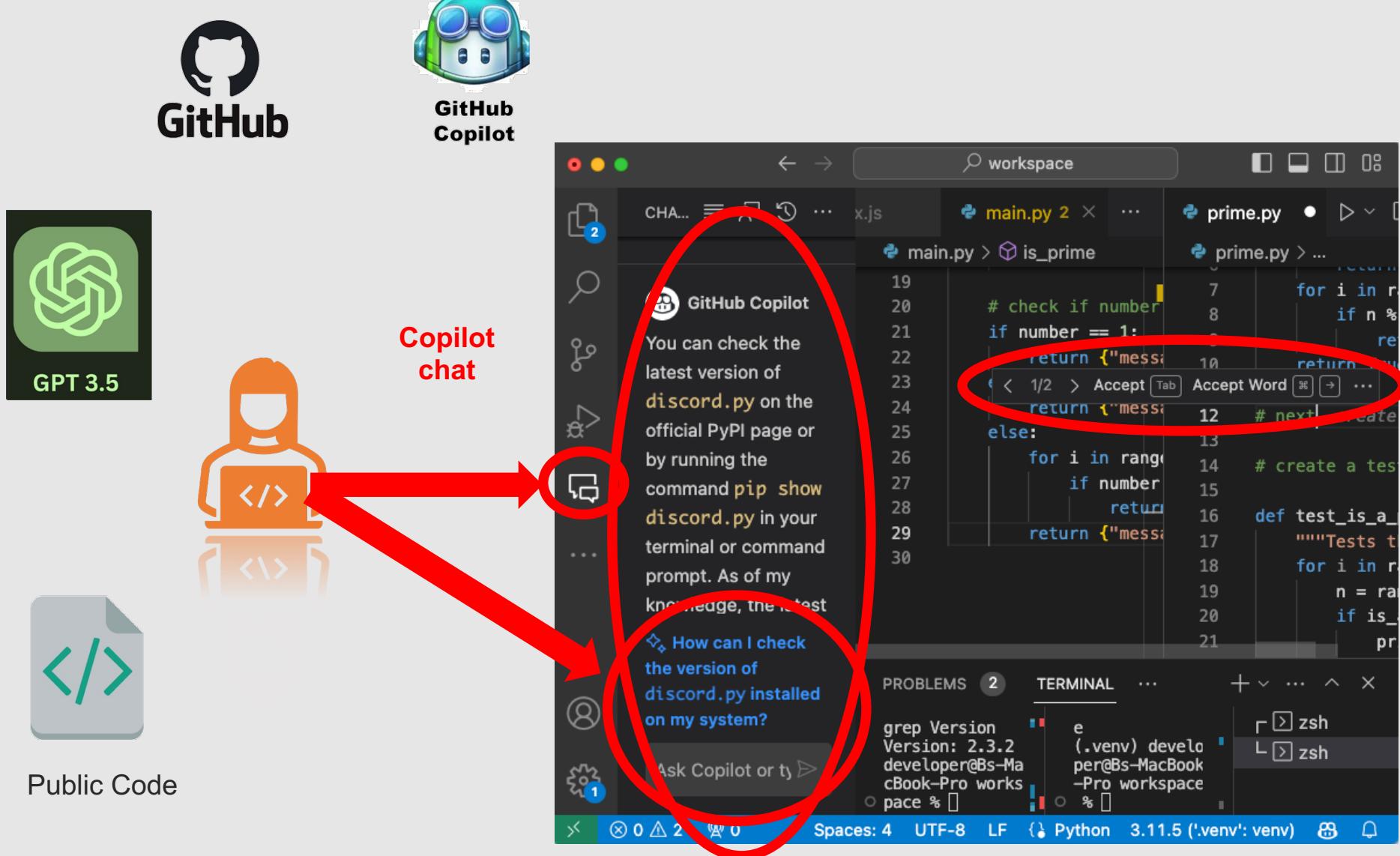


# How Copilot works (high-level)





# How Copilot works (high-level)





# Copilot shortcuts

A screenshot of the Visual Studio Code interface. The title bar shows tabs for 'Welcome', 'test-prime.py' (active), 'prime.py 2', and 'Untitled-1'. Below the tabs is a toolbar with icons for back, forward, accept, and word wrap, along with a dropdown menu and a '...' button. A red oval highlights the 'Accept' and 'Accept Word' buttons. The main editor area displays the code for 'test-prime.py':

```
1 # write a test to see if the function works
2
3
```

To the right of the editor, a floating toolbar shows 'Accept Line' and '✓ Always Show Toolbar'.

A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows the file name "prime.py". The status bar indicates "14, Col 4". A floating message box from GitHub Copilot is displayed, containing the text "Ask Copilot or type '/' for commands" with a blue arrow icon, and "Copilot generated code may be incorrect" with a warning icon. Below the message box, a keyboard shortcut legend is shown:

Show next inline suggestion	Alt + ]
Show previous inline suggestion	Alt + [
Trigger inline suggestion	Alt + \
Open GitHub Copilot (additional suggestions in separate pane)	Alt + Enter
Show next inline suggestion	Alt + ]
Show previous inline suggestion	Alt + [
Trigger inline suggestion	Alt + \
Open GitHub Copilot (additional suggestions in separate pane)	Alt + Enter



# Rotating through multiple options

```
prime.py 1 ● 3/3 > Accept Tab Accept Word ⌘ ⌘ ...
```

```
1 def is_prime(n):  
    if n == 1:  
        return False  
    for i in range(2, int(n**0.5)+1):  
        if not n % i:  
            return False  
    return True
```



# Accepting part of a suggestion

prime.py 2

```
1 def < 1/1 > Accept Tab Accept Word ⌘ → ...  
2     if n <= 1:  
         return False  
    for i in range(2, int(n**0.5)+1):  
        if n % i == 0:  
            return False  
    return True
```

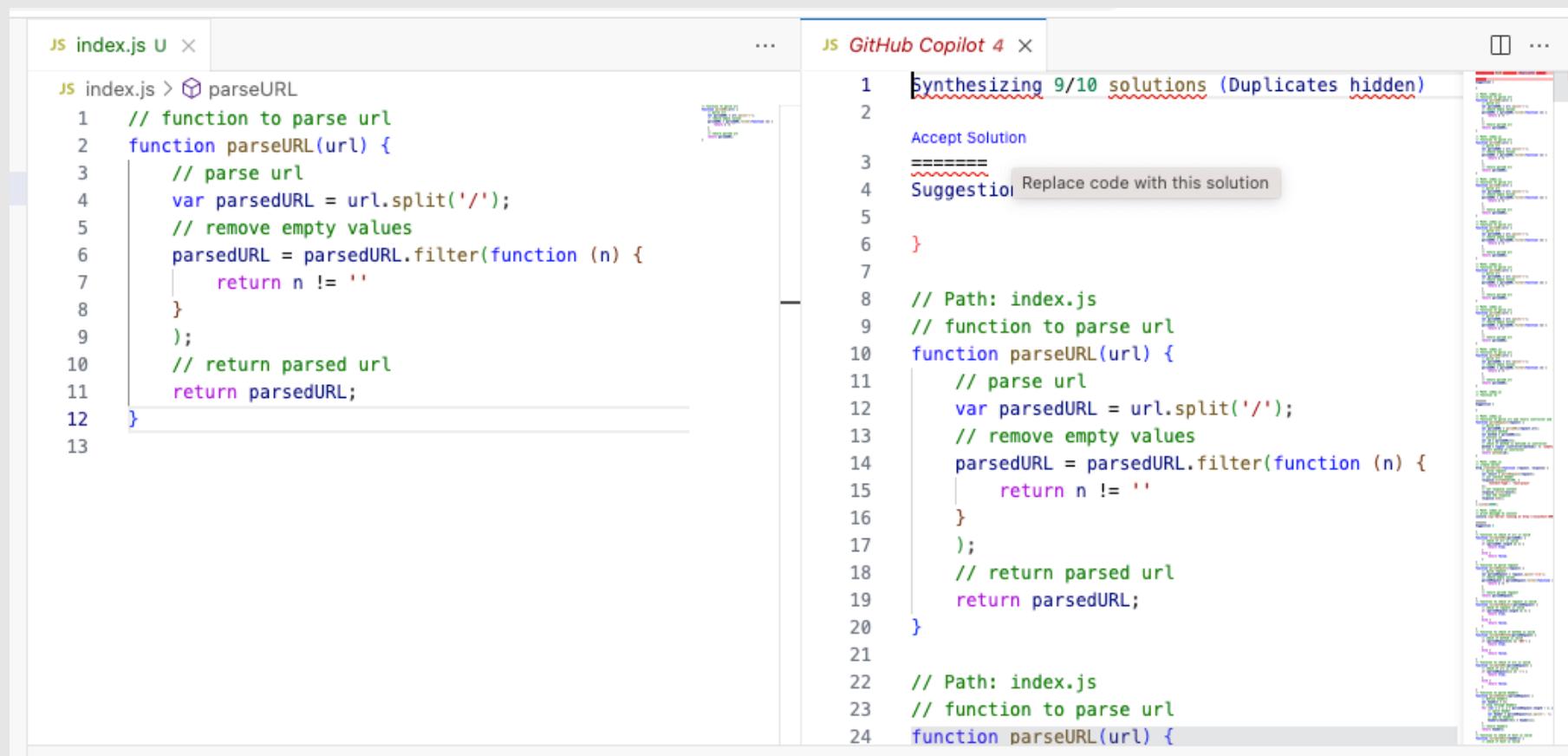
prime.py 1

```
1 def is_prime(n):  
2     if n <= 1:  
3         return False  
4     < 1/1 > Accept Tab Accept Word ⌘ → ...  
5     if n % i == 0:  
         return False  
    return True
```



# Seeing additional solutions

- CTRL + Enter
- Opens up new r/o window with other solutions
- Review and accept if fit (verify)



The screenshot shows two code editors side-by-side. The left editor is titled 'JS index.js U' and contains the following code:

```
1 // function to parse url
2 function parseURL(url) {
3     // parse url
4     var parsedURL = url.split('/');
5     // remove empty values
6     parsedURL = parsedURL.filter(function (n) {
7         return n != ''
8     })
9     );
10    // return parsed url
11    return parsedURL;
12}
13
```

The right editor is titled 'JS GitHub Copilot 4' and shows a suggestion for the same code. The status bar indicates 'Synthesizing 9/10 solutions (Duplicates hidden)'. A tooltip for 'Accept Solution' says 'Replace code with this solution'. The suggested code is identical to the original:

```
1 // Path: index.js
2 // function to parse url
3 function parseURL(url) {
4     // parse url
5     var parsedURL = url.split('/');
6     // remove empty values
7     parsedURL = parsedURL.filter(function (n) {
8         return n != ''
9     })
10    );
11    // return parsed url
12    return parsedURL;
13}
14
15 // Path: index.js
16 // function to parse url
17 function parseURL(url) {
18     // parse url
19     var parsedURL = url.split('/');
20     // remove empty values
21     parsedURL = parsedURL.filter(function (n) {
22         return n != ''
23     })
24    );
25    // return parsed url
26    return parsedURL;
27}
```



# About Labs

- Copilot is Generative AI
- Results may be different for you depending on context, etc.
- Queries in-line will give different results than queries through chat interfaces
- If not going through chat interfaces or using comment-directed prompts, may need to "nudge" in IDE (hit return multiple times and/or start typing first word/token of object (i.e. "import", "function" etc.)

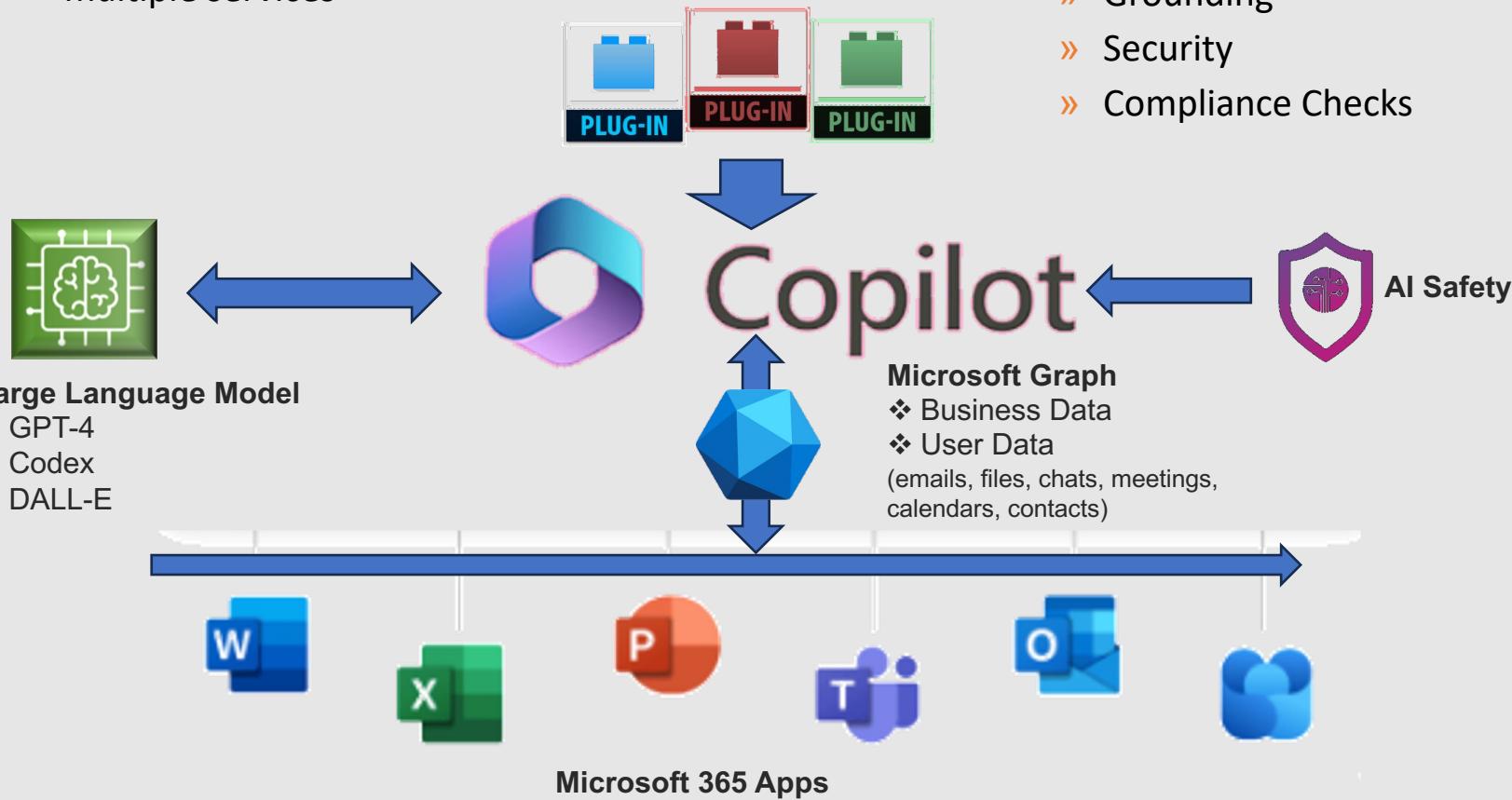




# Microsoft Copilot AI

- Data will not leave tenant
- Copilot will have real-time access to content and context in Microsoft Graph
  - Graph - API dev platform to connect multiple services

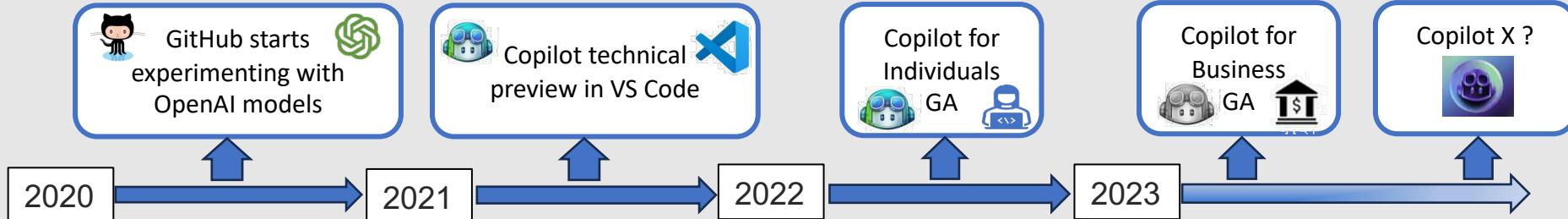
- Asking a question in app,
  - Copilot sends data to Graph to analyze / make sense of query
  - Data sent to GPT-4
  - Answer from ChatGPT sent back to Graph for
    - » Grounding
    - » Security
    - » Compliance Checks





# Brief History of Copilot

- Developed in collaboration with OpenAI
- Originally powered by OpenAI Codex
  - system created by OpenAI
  - more basis for source code context than GPT-3
  - trained on larger set of public source code
- Updated model
  - As of August, based on ChatGTP 3.5-turbo
    - » Originally trained in early 2022
    - » Model updated in March, 2023
  - Models trained and provided by OpenAI
    - » OpenAI protects models as a trade secret





# Copilot Chat Benefits

- Interactive Programming
  - Chat feature for more conversational interaction
  - Code creation
- Code Explanation
  - Useful especially for new coders or learning new language
- Code Review
  - Can review code and provide suggestions for improvements
- Learning
  - Ask Chat things like how to use certain functions, methods, etc. and see examples



# Chat commands

- "slash commands" for key functionality
- allow users to perform common tasks w/o having to retype prompt every time
- not all available in every IDE - VS Code has superset
- In some cases, invoke functionality that wouldn't be available via prompts alone

The screenshot shows a dark-themed chat interface with a blue header bar. The header bar contains the text '/help' on the left and 'Learn how to use Copilot Chat' on the right. Below the header, there is a list of slash commands, each preceded by a blue slash character. The commands listed are: /tests, /simplify, /fix, /explain, /extApi, /vscode, /createNotebook, /createWorkspace, /search, and /clear.

/help	Learn how to use Copilot Chat
/tests	
/simplify	
/fix	
/explain	
/extApi	
/vscode	
/createNotebook	
/createWorkspace	
/search	
/clear	



# Copilot Chat commands

Command	Purpose
/vscode	Questions about VS Code
/tests	Generate unit tests for the selected code
/simplify	Simplify the selected code
/fix	Propose a fix for the problems in the selected code
/explain	Explain step-by-step how the selected code works
/help	General help about Copilot
/clear	Clear chat output
/createWorkspace	Scaffold code for a new workspace
/extApi	Ask about VS Code extension development



# Using the chat interface

- Select code
- Select chat extension icon
- Type in the entry area
- Hit Enter
- View results

CHAT: GITHUB COPILOT

/simplify

GitHub Copilot

Here's a simplified version of the code that should work the same way:

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True
```

The changes I made are:

- Removed the check for `n == 2` since it's already covered by the loop
- Combined the check for `n == 1` and `n < 2` into a single check for `n < 2`
- Changed the condition in the loop to check for `i % i == 0` instead of `not n % i` for clarity
- Removed the extra blank lines at the end of the function

What is the difference between a while loop and a for loop?

/simplify Simplify the selected code

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

@brentlaster → /workspaces/copilot-dd (main) \$ code ir  
@brentlaster → /workspaces/copilot-dd (main) \$ code pr  
@brentlaster → /workspaces/copilot-dd (main) \$

Ln 1, Col 1 (191 selected) Spaces: 4 UTF-8



# From chat to editor

- Hover over results in Chat window
- Use popup menu to copy, insert, new file, etc.

The screenshot shows the GitHub Copilot interface. On the left is a sidebar with various icons. The main area has a title "CHAT: GITHUB COPILOT" and a search bar containing "/simplify". Below this is a message from "GitHub Copilot": "Here's a simplified version of the code that should work the same way:" followed by a code block:

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True
```

A red arrow points from the "Insert at Cursor" button in the Copilot message area to a red circle around the same button in the code editor's toolbar. The code editor shows the same code with some lines highlighted in blue. The tabs at the top of the editor are "JS index.js 1" and "prime.py".

Below the Copilot message, there is a section titled "The changes I made are:" with a bulleted list:

- Removed the check for `n == 2` since it's already covered by the loop
- Combined the check for `n == 1` and `n < 2` into a single check for `n < 2`
- Changed the condition in the loop to check for `n % i == 0` instead of `not n % i` for clarity



# Options from code box in chat

- Options available to "use" code provided by chat
- Hover over code in box
- Menu pops up with various options

The screenshot shows a GitHub Copilot interface. At the top, there's a GitHub Copilot icon and the text "GitHub Copilot". Below that, a message says "Here's a simplified version of the loop that would work the same way:". A red box highlights a portion of the Python code:

```
def is_prime(n):
    for i in range(2, n):
        if n % i == 0:
            return False
        return True
```

A context menu is open over the code, showing three options: "Insert Into New File", "Run in Terminal", and a keybinding "^Enter". The "Insert Into New File" option is highlighted. Three yellow callout bubbles provide additional context: "copy" points to the copy icon in the menu; "insert into editor pane at cursor" points to the "Insert Into New File" option; and "additional options" points to the "...".

The changes I made are:

- Removed the check for `n == 2` since it's already covered by the loop
- Combined the check for `n == 1` and `n < 2` into a single check for `n < 2`



# Quick actions with Copilot

- Click on lightbulb
- Select Option
- Answer appears inline for Accept or Discard

JS index.js 1 ● prime.py 1 ●

```

1  def is_prime(n):
2      if n < 2:
3          return False
4      for i in range(2, int(n**0.5)+1):
5          if x % i == 0:
6              return False
7      return True

```

JS index.js 1 ● prime.py 1 ●

```

1  def is_prime(n):
2      if n < 2:
3          return False
4      for i in range(2, int(n**0.5)+1):
5
6          if x % i == 0:
7              return False

```

Quick Fix

Fix using Copilot

Explain using Copilot

JS index.js 1 ● prime.py ●

```

1  def is_prime(n):
2      if n < 2:
3          return False
4      for i in range(2, int(n**0.5)+1):
5          if x % i == 0:
6              if n % i == 0:

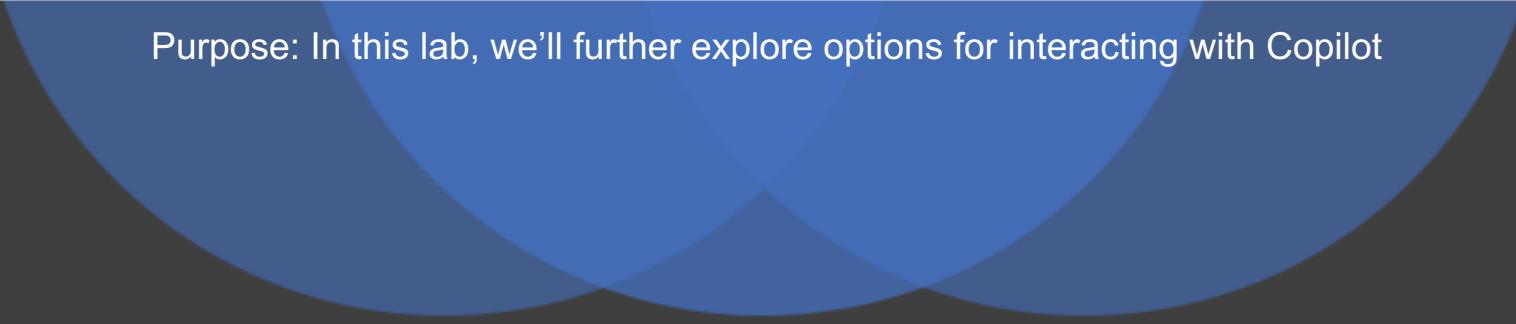
```

/fix "x" is not defined

Accept Discard ⌂ Changed 1 line



## Lab 2 – Using Copilot to simplify and explain code



Purpose: In this lab, we'll further explore options for interacting with Copilot



# Prompt

- Compilations of IDE code and relevant context fed back to model
- Series of algorithms select relevant code snippets or comments from current file
- Snippets are prioritized, filtered, and assembled into final prompt

Source:

<https://github.blog/2023-05-17-how-github-copilot-is-getting-better-at-understanding-your-code/>



# Helping Copilot with better "prompts"

- Open files
  - Copilot looks at and analyzes open files to get context and help improve suggestions
  - Helps to have related files open
- Top level comment
  - Brief, high-level comment in file you're working in helps Copilot understand overall context of what you're creating
- Appropriate includes and references
  - Best practice to set includes or references to modules manually
  - Gives Copilot context for frameworks, libraries, and versions to use
- Meaningful function names
  - Can help Copilot provide a function body that does what is needed
- Specific, well-scoped function comments
  - Use comments to help fill in details that help Copilot understand
- Prime Copilot with sample code
  - Copy and paste code that is close to what you're looking for
  - Once Copilot picks up on this, can delete the code
  - Helpful to get Copilot to understand a newer library version if it starts using old code



# Prompting Tips (from Copilot team)

- Single
  - Do one thing at a time
- Specific
  - Prompt should be specific
- Short
  - Answer that you expect back is short
- Using this approach, reliability for Copilot goes way up
- Copilot is really good at patterns
  
- May have to "nudge" it - start typing start of actual code vs comment
- Let Copilot handle the tedious stuff



# Giving Copilot direction via quick access chat dialog

44

- Cmd+I
- Use / commands or type question
- Output may appear in code editor

The screenshot shows a code editor interface with a Python file named prime.py open. The code defines a function `is_prime(n)` that checks if a number is prime. A tooltip from the GitHub Copilot quick access chat dialog is overlaid on the code editor, providing an explanation of the function's logic. The tooltip includes a summary, a detailed explanation of the loop condition, and a question for further improvement.

```
JS index.js 1 ● prime.py 2 ●
```

```
1 def is_prime(n):
2     if n < 2:
3         return False
4
5     for i in range(2, i):
6         if n % i == 0:
7             return False
8
9     return True
```

CHAT: GITHUB COPILOT  
brentlaster  
/explain

**GitHub Copilot**

The code above is a Python function that checks whether a given number `n` is prime or not. The function takes a single argument `n` and returns `True` if `n` is prime, and `False` otherwise.

The function first checks if `n` is less than 2, which is the smallest prime number. If `n` is less than 2, the function immediately returns `False`, since any number less than 2 cannot be prime.

If `n` is greater than or equal to 2, the function enters a loop that checks whether `n` is divisible by any number between 2 and the square root of `n`. The loop starts at 2, since any number less than 2 has already been ruled out, and ends at the square root of `n` plus one.

❖ Can you suggest any other improvements to the code for checking prime numbers?

Ask Copilot or type '/' for commands



# Giving Copilot directions via comments

- Necessary to tab and return and repeat

```
JS index.js 1 ● prime.py ● ▶
1 def is_prime(n):
2     if n < 2:
3         return False
4     for i in range(2, int(n**0.5)+1):
5         if n % i == 0:
6             return False
7     return True
8
9 # explain the code above line-by-line
10 # 1. def is_prime(n):
11 #     define a function named is_prime, which takes one argument n
12 # 2. if n < 2:
13 #     if n is less than 2, return False
14 # 3. for i in range(2, int(n**0.5)+1):
15 #     for i in range from 2 to the square root of n plus 1
16 # 4. if n % i == 0:
17 #     if n is divisible by i, return False
18 # 5. return True
19 #     if n is not divisible by any number between 2 and the square root of n plus 1, return
20
21
```



# Querying via comments

- Can ask anything you would ask via Chat with <comment character> q:
- Hit return/tab to see answer (<comment character> a):
- Comments will be different/more succinct than through chat

```
29      # test for a negative value
30  def test_is_not_prime_3():
31      response = client.get("/prime/-1")
32      assert response.status_code == 200
33      assert response.json() == {"number": -1, "is_prime": False}
34
35
36      # q: explain in detail what each line of this file does
37      # a: the first line imports pytest and the TestClient class from fastapi.testclient
38      # the second line imports the app from main.py
39      # the third line creates a TestClient instance with the app as an argument
40      # the fourth line defines a function that tests whether a number is prime
41      # the fifth line defines a function that tests whether a number is not prime
42      # the sixth line defines a function that tests whether a number is not prime
43      # the seventh line defines a function that tests whether a number is prime
44      # the eighth line defines a function that tests whether a number is not prime
45      # the ninth line defines a function that tests whether a number is not prime
46      # the tenth line defines a function that tests whether a number is prime
47      # the eleventh line defines a function that tests whether a number is not prime
48      # the twelfth line defines a function that tests whether a number is not prime
49      # the thirteenth line defines a function that tests whether a number is prime
50      # the fourteenth line defines a function that tests whether a number is not prime
51      # the fifteenth line defines a function that tests whether a number is not prime
52      # the sixteenth line defines a function that tests whether a number is prime
53      # the seventeenth line defines a function that tests whether a number is not prime
```



# Generating automatic doc

- /doc in chat interfaces
- regenerate button to see other options

The screenshot shows a code editor interface with two files open: `index.js` and `prime.py`. The `prime.py` file contains the following Python code:

```
1  def is_prime(n):
2+     """
3+         Returns True if the given number is prime, False otherwise.
4+
5+     Args:
6+         n (int): The number to check for primality.
7+
8+     Returns:
9+         bool: True if the given number is prime, False otherwise.
10+
11    if n < 2:
12        return False
13    for i in range(2, int(n**0.5)+1):
14        if n % i == 0:
15            return False
16    return True
```

A green callout box highlights the docstring and argument documentation for the `is_prime` function. At the bottom of the editor, there is a toolbar with buttons for `/doc`, `Accept`, `Discard`, and a `Regenerate` button. The `Regenerate` button is circled in red.

## Lab 3 – Using Copilot after the coding

Purpose: In this lab, we'll look at how Copilot can be used after code is already written including documenting and explaining code



# What does Copilot know about your project?

- Copilot looks at:
  - name of the current file
  - looks before and after cursor for current context
  - any open files in your editor during that session
    - » will grab snippets of code from other files to craft better suggestions
- Doesn't grab information from all files because of prompt sizes
  - working under constraints for system



# Does Copilot keep code?

- Code Snippets Data

"GitHub Copilot transmits snippets of your code from your IDE to GitHub to provide Suggestions to you. Code snippets data is only transmitted in real-time to return Suggestions, and is discarded once a Suggestion is returned. **Copilot for Business does not retain any Code Snippets Data.**"



# Features built-in to address data privacy

- GitHub won't retain code snippets, store, or share your code regardless if the data is from public repositories, private repositories, non-GitHub repositories, or local files.
- Filter to help detect and suppress Copilot suggestions that contain code that matches public code on GitHub
  - Checks code suggestions with surrounding code for matches or near matches (ignoring whitespace)
  - Checks against public code on GitHub of about 150 characters.
  - If match, the suggestion will not be shown to you.
- Feature (in private beta) to provide reference for suggestions that resemble public code on GitHub
  - Allows you to make a more informed decision about whether and how to use that code
  - Explore and learn how that code is used in other projects.
- AI-based vulnerability prevention system
  - Blocks insecure coding patterns in real-time to make GitHub Copilot suggestions more secure
  - Model targets the most common vulnerable coding patterns
    - Including hardcoded credentials, SQL injections, and path injections.

Source: <https://github.com/orgs/community/discussions/49422>



# Code referencing : private beta

52

- Includes updated filter that detects and shows context of code matching public code from GitHub
- If turned on, Copilot checks code suggestions with surrounding code of ~150 characters
- Compares against index of all public code on GitHub.com

- Matches w/ info re every repo they're in, are displayed in the editor
- Devs can choose to block or allow suggestions
- Match shows:
  - The matching code
  - The repositories where code appears
  - License governing each repo

The screenshot shows the VS Code interface with a dark theme. On the left is the sidebar with icons for file, search, repository, and other settings. The main area has a tab for 'binary-search.py'. The code editor contains the following Python code:

```
1 // implement a binary search algorithm
2
3 def binary_search(array, target):
4     # Write your code here.
5     return binary_search_helper(array, target, 0, len(array) - 1)
6
7 def binary_search_helper(array, target, left, right):
8     if left > right:
9         return -1
10    middle = (left + right) // 2
11    potential_match = array[middle]
12    if target == potential_match:
13        return middle
14    elif target < potential_match:
15        return binary_search_helper(array, target, left, middle)
16    else:
17        return binary_search_helper(array, target, middle + 1,
```

To the right of the editor is a panel titled 'Public code matches' for the file. It displays the following information:

**Public code matches for /Users/2percentsilk/binary-search.py**

*Matches will be tracked with their approximate line numbers below. As changes are made to the file, these references may become inaccurate. Matches will not persist when a file is closed.*

**Match**  
originally detected at line 8

Matched content:

```
:  
    return -1  
    middle = (left + right) // 2  
    potential_match = array[middle]  
    if target == potential_match:  
        return middle  
    elif target < potential_match:  
        return binary_search_helper(array, target, left, middle)  
    else:  
        return binary_search_helper(array, target,
```

**License Summary**

This snippet matches 4 references to public code. Below, you can find links to a sample of 50 of these references.

- MIT (2)

Source: <https://githubnext.com/projects/copilot-for-pull-requests>



# Using comments to generate tests

JS index.js 1 ● prime.py ●

```
12         return False
13     for i in range(2, int(n**0.5)+1):
14         if n % i == 0:
15             return False
16     return True
```

17 < 1/1 > Accept Tab Accept Word ⌘ → ... of the code above

```
19 def test_is_prime():
    assert is_prime(2) == True
    assert is_prime(3) == True
    assert is_prime(4) == False
    assert is_prime(5) == True
    assert is_prime(6) == False
```

20



# Building out functions using Copilot Fix

```
12         return False
13     for i in range(2, int(n**0.5)+1):
14         if n % i == 0:
15             return False
16     return True
17
```



JS index.js 1 ● prime.py ●

```
19
20     assert is_prime(84) == False
21     assert is_prime(85) == False
22     assert is_prime(86) == False
23     assert is_prime(87) == False
103+    assert is_prime(88) == False
104+    assert is_prime(90) == False
105+    assert is_prime(91) == False
106+    assert is_prime(92) == False
107+    assert is_prime(93) == False
108+    assert is_prime(94) == False
109+    assert is_prime(95) == False
110+    assert is_prime(96) == False
111+    assert is_prime(98) == False
112+    assert is_prime(99) == False
113+    assert is_prime(100) == False
114+
115+
116+
117+
```

19 118

/fix "(" was not closed

Accept

Discard



Changed 100 lines

# Generating tests from chat

- /tests in chat
- insert into new file
- good way to get more data generated

The screenshot shows the GitHub Copilot interface with two panes. The left pane is titled "CHAT: GITHUB COPILOT" and contains a message: "for the `is_prime` function:". The right pane shows a code editor with Python test code. A red circle highlights the function definition `def test_is_prime_with_prime_numbers()`. Another red circle highlights the "Insert Into New File" button in a tooltip. A third red circle highlights the input field at the bottom labeled "/tests Generate unit tests for the selected code".

```

CHAT: GITHUB COPILOT
for the is_prime function:

def test_is_prime_with_prime_numbers():
    assert is_prime(2) == True
    assert is_prime(3) == True
    assert is_prime(5) == True
    assert is_prime(7) == True
    assert is_prime(11) == True
    assert is_prime(13) == True
    assert is_prime(17) == True
    assert is_prime(19) == True
    assert is_prime(23) == True
    assert is_prime(29) == True

def test_is_prime_with_non_prime_numbers():
    assert is_prime(4) == False
    assert is_prime(6) == False
    assert is_prime(8) == False

# q: what does this function do?
# a: this function takes one argument

```

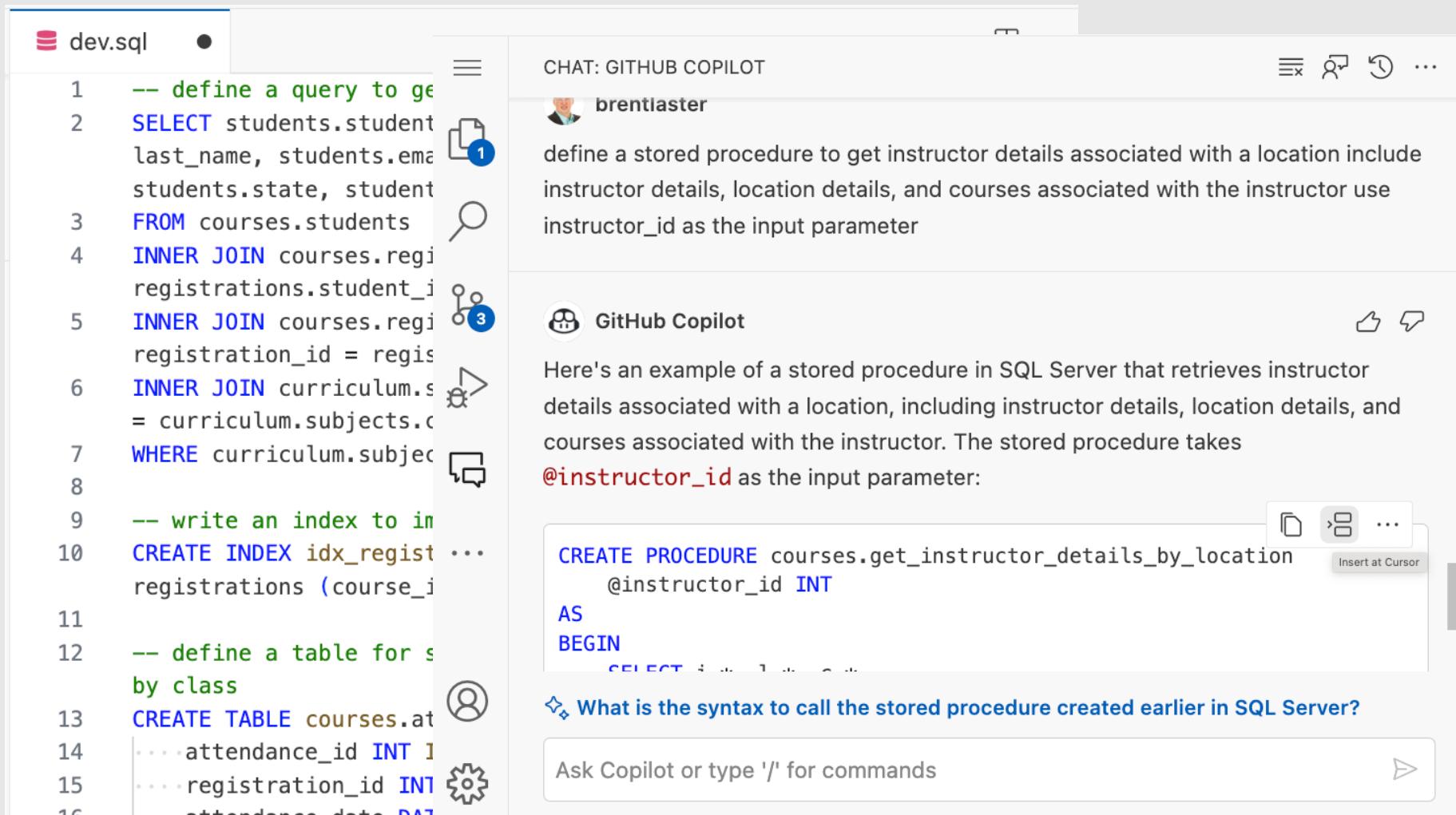
## Lab 4 – Using Copilot to generate tests

Purpose: In this lab, we'll see some examples of having Copilot generate tests



# Copilot and SQL - comments

- Can generate a lot of SQL items from comments



The screenshot shows a code editor window for a file named "dev.sql". The code contains several SQL statements and comments. A GitHub Copilot interface is overlaid on the right side of the editor.

**CHAT: GITHUB COPILOT**

brentlaster

define a stored procedure to get instructor details associated with a location include instructor details, location details, and courses associated with the instructor use instructor\_id as the input parameter

**GitHub Copilot**

Here's an example of a stored procedure in SQL Server that retrieves instructor details associated with a location, including instructor details, location details, and courses associated with the instructor. The stored procedure takes **@instructor\_id** as the input parameter:

```
CREATE PROCEDURE courses.get_instructor_details_by_location
    @instructor_id INT
AS
BEGIN
    SELECT ...
END
```

Insert at Cursor

❖ What is the syntax to call the stored procedure created earlier in SQL Server?

Ask Copilot or type '/' for commands



# Copilot and SQL - chat interface

- More flexibility and fuller responses

CHAT: GITHUB COPILOT

brentlaster

1 define a stored procedure to get instructor details associated with a location include instructor details, location details, and courses associated with the instructor use instructor\_id as the input parameter

GitHub Copilot 3

Here's an example of a stored procedure in SQL Server that retrieves instructor details associated with a location, including instructor details, location details, and courses associated with the instructor. The stored procedure takes **@instructor\_id** as the input parameter:

```
CREATE PROCEDURE courses.get_instructor_details_by_location
    @instructor_id INT
AS
BEGIN
    SELECT ...

```

Insert at Cursor

What is the syntax to call the stored procedure created earlier in SQL Server?

Ask Copilot or type '/' for commands



# Copilot and SQL

- Optimizing queries

```
58      — select * from courses.registration where year(registration_date) = 2023 and
      — month(registration_date) = 9
58+    CREATE INDEX idx_registration_date ON courses.registrations
      + (registration_date);
59+
60+  SELECT *
61+  FROM courses.registrations
62+  WHERE registration_date >= '2023-09-01'
63+  ... AND registration_date < '2023-10-01';
59  64
60  65
61  66  CREATE TABLE courses.students (
```

/optimize

Accept

Discard



Changed 7 lines



# Giving Copilot More Context

- Copilot gets context from a number of sources
  - Name of current file
  - Other open files in IDE
  - Contents of current file
  - Whether or not last suggestion was accepted
- Ways to provide more context
  - Open other files in editor
  - Copy and paste into current file

# Providing more context for better results

```

prime.py • dev.sql • create-tables.sql U
/wkspaces/copilot-dd/prime.py

1 -- define a select statement to get all
2   students enrolled in a course
3
4   select s.first_name, s.last_name, s.email, s.
5     phone, s.city, s.state, s.zip_code
6   from courses.students s
7   join courses.registrations r on s.student_id =
8     r.student_id
9
10  join courses.registration_items ri on r.
11    registration_id = ri.registration_id
12
13
14  -- create tables
15  CREATE TABLE training.categories (
16    training_id INT IDENTITY (1, 1) PRIMARY KEY,
17    training_name VARCHAR (255) NOT
18    NULL
19  );
20
21  CREATE TABLE curriculum.vendors (
22    brand_id INT IDENTITY (1, 1) PRIMARY KEY,
23    brand_name VARCHAR (255) NOT NULL
24  );

```

```

prime.py • dev.sql • create-tables.sql U
/wkspaces/copilot-dd/prime.py

1 -- create tables
2 <--> CREATE TABLE training.categories (
3   <--> training_id INT IDENTITY (1, 1) PRIMARY KEY,
4   <-->   training_name VARCHAR (255) NOT
5   <-->   NULL
6 );
7 <--> CREATE TABLE curriculum.vendors (
8   <--> brand_id INT IDENTITY (1, 1) PRIMARY KEY,
9   <-->   brand_name VARCHAR (255) NOT NULL
10 );
11 <--> CREATE TABLE curriculum.subjects (
12   <--> course_id INT IDENTITY (1, 1) PRIMARY KEY,
13   <-->   product_name VARCHAR (255) NOT NULL,
14   <-->   branch_id INT NOT NULL,
15   <-->   training_id INT NOT NULL,
16   <-->   model_year SMALLINT NOT NULL,
17   <-->   class_hours DECIMAL (10,2) NOT NULL,
18   <-->   FOREIGN KEY (training_id) REFERENCES
          curriculum.categories (training_id) ON
          DELETE CASCADE ON UPDATE CASCADE,
   <-->   FOREIGN KEY (brand_id) REFERENCES
          curriculum.vendors (brand_id) ON DELETE

```

## Lab 5 – Using Copilot to help with SQL

Purpose: In this lab, we'll see some examples of how to have Copilot help with writing SQL



# Copilot and updates

- query: write a function to see a random number generator

```
myfiles > -go explore.go
```

```
1 package main
2
3 import (
4     "math/rand"
5     "time"
6 )
7
8 func seedRand() {
9     rand.Seed(time.Now().UnixNano())
10 }
11
```



# Deprecated code

- Seed function is deprecated as of Go 1.20

rand.go

```
389 //  
390 // Deprecated: As of Go 1.20 there is no reason to call Seed with  
391 // a random value. Programs that call Seed with a known value to get  
392 // a specific sequence of results should use New(NewSource(seed)) to  
393 // obtain a local random generator.  
-----
```



# Giving Copilot a clue

- Can use doc to provide more context for Copilot

```
myfiles > -go explore.go
1      // Create and seed the generator.
2      // Typically a non-fixed seed should be
3      // used, such as time.Now().UnixNano().
4      // Using a fixed seed will produce the
5      // same output on every run.
6      r := rand.New(rand.NewSource(99))
7
8
```



# Redo query with extra context

```
myfiles > -go explore.go
1      // Create and seed the generator.
2      // Typically a non-fixed seed should be
3      // used, such as time.Now().UnixNano().
4      // Using a fixed seed will produce the
5      // same output on every run.
6
7      import (
8          "math/rand"
9          "time"
10         )
11
12     func seedRandom() *rand.Rand {
13         return rand.New(rand.NewSource(time.
14             Now().UnixNano())))
15     }
```

## Lab 6 – Teaching Copilot about updates

Purpose: In this lab, we'll see an example of what to do when Copilot does not have the most up-to-date information



# What version of Go does Copilot think is current? (The 2021 problem)

68

- Let's ask...



brentlaster

what's the latest version of Go?



GitHub Copilot

As an AI, I don't have real-time capabilities to fetch the latest version of Go. However, as of my last training data in September 2021, the latest stable version of Go is Go 1.17, released in August 2021. Please check the official Go website or its GitHub repository for the most up-to-date information.

❖ What are the new features in Go 1.17?

Ask Copilot or type '/' for commands



```
// q: what's the latest version  
of go?
```

```
// a: 1.16
```

11

12

// a: what's the latest version

< 2/2 > Accept Tab Accept Word ⌘ ⌛ ...

13

// a: 1.13.4

14



# Copilot and Kubernetes

- Copilot can also work with frameworks
- Can generate manifests
- Can explain how to run frameworks
- Can also provide API references



# Defining a Kubernetes manifest

- Provide basic information

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-service
5  spec:
6    type: NodePort
7    selector:
8      app: my-app
9    ports:
10      name: http
11      port: 80
12      targetPort: 8080
13      nodePort: 30100
14
```

write a k8s service manifest of type  
Nodeport with the nodePort = 30100

Accept

Discard ▾



Changed  
14 lines





# Explaining how to run frameworks

CHAT: GITHUB COPILOT

 GitHub Copilot

To start a Kubernetes service, you need to apply the service configuration using **kubectl**. The configuration is typically written in a YAML file.

Here's how you can start a service using **kubectl**:

1. Save your service configuration in a YAML file. In your case, it seems like you already have a service configuration in a file named **service.yaml**.
2. Open a terminal.
3. Run the following command to start the service:

```
kubectl apply -f service.yaml
```

❖ **What is the difference between a Kubernetes service and a deployment?**

How do I start a K8s service? 



# Getting the code to invoke K8s APIs

CHAT: GITHUB COPILOT



```
clientset, err := kubernetes.NewForConfig(config)
if err != nil {
    panic(err)
}

// Get the service
service, err := clientset.CoreV1().Services("default").Get()
if err != nil {
    panic(err)
}

// Change the service type to ClusterIP
service.Spec.Type = "ClusterIP"

// Update the service
_, err = clientset.CoreV1().Services("default").Update(context.TODO(), &service, metav1.UpdateOptions{})
if err != nil {
    panic(err)
}
```

❖ How can I expose a Kubernetes service to the internet?

What is the K8s API call in Go to change a service from type Nodeport to type ClusterIP?





# Getting help with implementation in a language

73

- How do I define my service in Go?

The screenshot shows the GitHub Copilot interface. On the left, a code editor displays Go code for defining a Kubernetes Service object. On the right, a file browser shows the corresponding YAML configuration. The Go code is auto-generated to match the YAML structure.

```
CHAT: GITHUB COPILOT
spec: v1.ServiceSpec{
    Type: v1.ServiceTypeNodePort,
    Selector: map[string]string{
        "app": "my-app",
    },
    Ports: []v1.ServicePort{
        {
            Name:      "http",
            Port:      80,
            TargetPort: intstr.FromInt(8080),
            NodePort:   30100,
        },
        ,
    },
}
}

! service.yaml ● create-t □ ...
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: my-service
5 spec:
6   type: NodePort
7   selector:
8     app: my-app
9   ports:
10    - name: http
11      port: 80
12      targetPort: 8080
13      nodePort: 30100
14
```

This Go code creates a Kubernetes service object that matches the YAML you provided. Note that this code just defines the service. To actually create the

❖ [How can I create a Kubernetes service using the Go client?](#)

How do I define my service in Go?



## Lab 7 –YAML generation, API use, translating code and the 2021 problem

Purpose: In this lab, we'll learn about how to have Copilot help with YAML generation, API usage, translating code and explore the 2021 problem



# Copilot and regular expressions



CHAT: GITHUB COPILOT



create a regular expression to validate an SSN



GitHub Copilot

Here's a regular expression that can be used to validate a Social Security Number (SSN) in the format of XXX-XX-XXXX:

```
const regex = /^[^\d{3}-\d{2}-\d{4}]/;
```



This regular expression matches a string that starts with three digits, followed by a hyphen, then two digits, another hyphen, and finally four digits. The **^** and **\$** characters at the beginning and end of the

**↳ How can I validate an email address using regular expressions in JavaScript?**

Ask Copilot or type '/' for commands



CHAT: GITHUB COPILOT



create a regular expression to validate driver's license numbers

GitHub Copilot



Sure! Here's a regular expression that can be used to validate driver's license numbers:

```
const regex = /^[^\d{3}-\d{2}-\d{4}]/;
```

This regular expression matches a string that starts with one or two uppercase letters, followed by one to six digits, an optional uppercase letter, and zero to two digits. The **^** and **\$** characters at the beginning and end of the regular expression,

**↳ What is the difference between kubectl apply and kubectl create?**

Ask Copilot or type '/' for commands





# Automatic data generation

CHAT: GITHUB COPILOT

GITHUB Copilot

Here's an example mapping of states to area codes in JavaScript:

```
const stateToAreaCodes = {  
    "AL": ["205", "251", "256", "334"],  
    "AK": ["907"],  
    "AZ": ["480", "520", "602", "623", "928"],  
    "AR": ["479", "501", "870"],  
    "CA": ["209", "213", "310", "323", "408", "415", "510", "530", "559", "5  
    // and so on for other states  
};
```

In this example, each state abbreviation is a key in the `stateToAreaCodes` object, and the value for each key is an array of area codes associated with that state. You can add or remove states and area codes as needed for your specific use case.

❖ How can I validate an email address using regular expressions?

create a mapping of states to area codes  
the key is the state abbreviation  
the value is an array of area codes



## Lab 8 –Exploring Javascript, regex generation, and pattern data generation

Purpose: In this lab, we'll learn how to have Copilot generate regex expressions and pattern data



# GitHub Copilot vs ChatGPT

Feature	GitHub Copilot	ChatGPT
Primary Purpose	Code generation and assistance	Natural language understanding and generation
Developed By	GitHub in collaboration with OpenAI	OpenAI
Use Case	Software development, code suggestions	Natural language conversation and tasks
AI Model	GPT-3.5 architecture	GPT-3.5 architecture
Integration	Integrated with code editors (e.g., VS Code)	Typically used in chatbots and applications
Programming Languages	Supports multiple programming languages	Language-agnostic
Assistance Level	Provides code suggestions and auto-completion	Conversational responses and text generation
Training Data	Trained on code repositories, documentation	Trained on a diverse dataset of text
Learning Approach	Supervised learning with code examples	Unsupervised learning from internet text
Customization	Limited customization for code generation	Can be fine-tuned for specific tasks
Common Use Cases	Writing code, refactoring, code documentation	Text generation, chatbots, language tasks
User Interaction	Works within code editors, responds to code	Text input and output via APIs or platforms
API Availability	Limited API access for specific use cases	Available through OpenAI API for various
Pricing Model	Subscription-based pricing	Usage-based pricing (API calls)
Privacy Concerns	May involve sharing code with GitHub	Privacy concerns related to text content
Community Support	GitHub community, documentation	OpenAI community, developer resources



# Copilot X

- Copilot X will include additional features and use GPT-4

Feature	Copilot	Copilot X
Code completion	Yes	Yes
Chat and voice	No	Yes
Pull requests support	No	Yes
Code doc questions	No	Yes
GPT level	GPT-3	GPT-4



# Copilot X: Chat

- Same as available now for early preview
- Integration with voice activation (GitHub Copilot Voice)
- Knows about code, error messages
- Deeply embedded in IDE
- Can provide deep analysis and explanations, gen unit tests, and propose fixes

The screenshot shows the GitHub Copilot X interface integrated into the Visual Studio Code (VS Code) environment. On the left, there's a sidebar with various icons for file operations like copy, paste, search, and refresh. A message from "martinwoodward" says: "Propose a fix for the bugs in my code." Below it, "GitHub Copilot" responds: "There are a couple of issues in the provided code:" followed by two numbered bullet points. The first point states: "1. The split() method is called on the wrong delimiter # instead of the correct delimiter (space).". The second point states: "2. The order of the elements in the tuple is incorrect. It should be (date, value, currency) instead of (value, currency, date).". Underneath, "GitHub Copilot" provides the corrected code:

```

import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples
    (date, value, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2023-01-02 -34.01 USD
        2023-01-03 2.59 DKK
        2023-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((float(value),
                         currency,
                         datetime.datetime.strptime(date,
                                                     "%Y-%m-%d")))
    return expenses

```

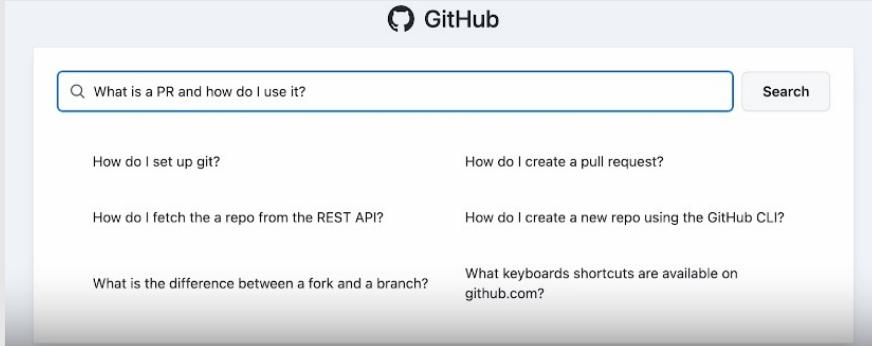
The code editor tab is titled "add\_elements.py M X" and shows the same Python script with syntax highlighting. The status bar at the bottom indicates "PROBLEMS", "OUTPUT", "TERMINAL", and "DEBUG CONSOLE".

Source: <https://github.blog/2023-03-22-github-copilot-x-the-ai-powered-developer-experience/>



# Copilot X: Copilot for Docs

- Uses chat interface to provide users with AI-generated responses to questions about doc

 GitHub

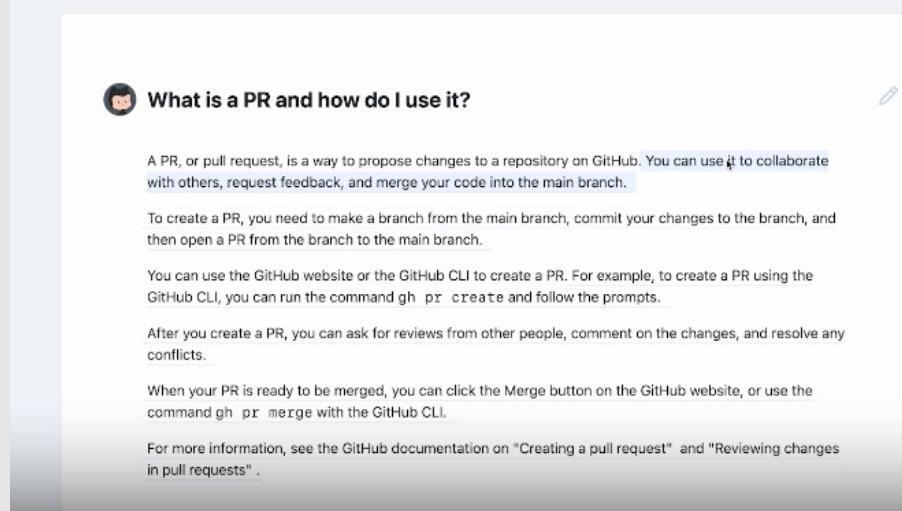
What is a PR and how do I use it? Search

How do I set up git? How do I create a pull request?

How do I fetch the a repo from the REST API? How do I create a new repo using the GitHub CLI?

What is the difference between a fork and a branch? What keyboards shortcuts are available on github.com?

Enable sharing Only you can view this thread



What is a PR and how do I use it?

A PR, or pull request, is a way to propose changes to a repository on GitHub. You can use it to collaborate with others, request feedback, and merge your code into the main branch.

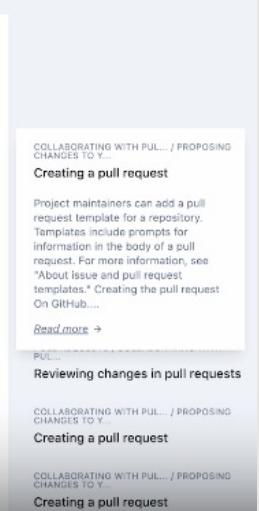
To create a PR, you need to make a branch from the main branch, commit your changes to the branch, and then open a PR from the branch to the main branch.

You can use the GitHub website or the GitHub CLI to create a PR. For example, to create a PR using the GitHub CLI, you can run the command `gh pr create` and follow the prompts.

After you create a PR, you can ask for reviews from other people, comment on the changes, and resolve any conflicts.

When your PR is ready to be merged, you can click the Merge button on the GitHub website, or use the command `gh pr merge` with the GitHub CLI.

For more information, see the GitHub documentation on "Creating a pull request" and "Reviewing changes in pull requests".



COLLABORATING WITH PULL REQUESTS / PROPOSING CHANGES TO YOUR REPOSITORY  
Creating a pull request

Project maintainers can add a pull request template for a repository. Templates include prompts for information in the body of a pull request. For more information, see "About issue and pull request templates." Creating the pull request On GitHub...  
[Read more →](#)

PULL REQUESTS  
Reviewing changes in pull requests

COLLABORATING WITH PULL REQUESTS / PROPOSING CHANGES TO YOUR REPOSITORY  
Creating a pull request

COLLABORATING WITH PULL REQUESTS / PROPOSING CHANGES TO YOUR REPOSITORY  
Creating a pull request



# Copilot X: GitHub Copilot CLI

- Use Copilot to compose commands, figure out arguments, flags, etc.

```
Downloads %
Downloads % ?? use ffmpeg to add a watermark to input.mp4
```

Command

```
ffmpeg -i input.mp4 -i watermark.png -filter_complex "overlay=10:10" output.mp4
```

Explanation

- ffmpeg is used to process video and audio files.
  - ◆ -i input.mp4 specifies the input file.
  - ◆ -i watermark.png specifies the input file.
  - ◆ -filter\_complex "overlay=10:10" specifies the filter to apply.
    - ◊ overlay=10:10 specifies that we want to overlay the second input file at position (10, 10).
  - ◆ output.mp4 specifies the output file.

>  Run this command  
 Revise query  
 Cancel



# Copilot X : Copilot for PRs - markers

- ***Available for tech preview now***
- ***copilot: all*** - showcases all the different kinds of content at once
- ***copilot: summary*** - expands to a 1 paragraph summary of changes in the PR
- ***copilot: walkthrough*** - expands to a detailed list of changes, including links to relevant code
- ***copilot:poem*** - expands to a poem about changes in the pull request

The screenshot shows a GitHub pull request comment from a user named mattrothenberg. The comment is timestamped as "commented 1 minute ago". Below the timestamp, there are three sections of placeholder text, each preceded by a double hash symbol (##) and followed by a descriptive note in parentheses:

- ## Description  
<!--- Describe your changes in detail -->
- ## Related Issue  
<!--- Please link to the issue here: -->
- ## Explanation of Changes  
<!--- Provide a walkthrough of your changes: -->

At the bottom right of the comment area, there is a green button labeled "Update Comment".

Source: <https://githubnext.com/projects/copilot-for-pull-requests>



# Copilot X for PRs : Under consideration

## Resolving issues with AI

Someone just filed an issue on your repository: "Replace TensorFlow with PyTorch". Seems like a big job? Maybe you are not sure where to start? We think that AI can help.

We are prototyping functionality to automatically describe how to solve an issue and even to automatically suggest the changes you need to make. In this video we file the issue about moving to PyTorch and use our AI to example how it might be done, to generate code suggestions and raise a new pull request.

### AI-powered PR completion

Some PRs flow almost automatically from one or two lines of edits. Add a comment to a function? AI can "complete the job" and add comments to all your functions, throughout all your code. Change your packages.json from Express to Koa? Today's AI models can guide you how to adjust your code.

We're working to empower our AI bot with the capability to complete pull requests given a few indicative edits. We've mocked this up in VS Code and are looking to integrate this into our bot and other delivery channels.

### AI-powered review responses

Is someone nit-picking on your changes? What if we could auto-generate changes in response to their requests to add documentation or even to add another test?

### AI-powered PR repair

We know the PR process can damage your flow. There is inevitably a delay while you wait for a review, or you wait for Actions to run. And many PRs go back and forth a few times. Each time you switch in and then switch out again is disruptive. We want to know if we can use artificial intelligence to fill in the gaps and reduce the number of switches.

How many times have you submitted a change and forgot to update the unit tests? Or the documentation? Or introduced linter errors. Perhaps we can fix that for you....watch this space!

## Reviewing pull-requests with AI

Considerable amounts of developer time are spent on code review and on preparing a PR for code review. We think that AI can help. In this video we use our AI to describe the changes in the pull request and to review the code. See how it makes actionable suggestions for improvement that you can just click to accept. Maybe this won't replace human reviewers, but we think it can cut down the time spent in the review cycle.

Source: <https://githubnext.com/projects/copilot-for-pull-requests>



# GitHub Copilot Labs

- VS Code-only extension
- For experimental features
- <https://githubnext.com/projects/copilot-labs/>

The screenshot shows the GitHub Copilot Labs extension page in the VS Code Marketplace. The search bar at the top contains "copilot labs". The main listing for "GitHub Copilot Labs" shows it has 766K installs and a 4-star rating. A modal window is open, displaying a warning message: "⚠ To use GitHub Copilot Labs, you need to accept the GitHub Next Pre-Release License Terms. Sign up by visiting <https://githubnext.com/projects/copilot-labs/>, then restart Visual Studio Code." Below the modal, there are two buttons: "Sign up" and "Dismiss". At the bottom of the extension card, there's a link to "Source: GitHub Copilot Labs (Extension)".

The screenshot shows the GitHub Copilot Labs landing page. It features a large header with the title "GitHub Copilot Labs" and a subtext "A VS Code extension for experimental applications of GitHub Copilot.". Below the header, there's a section titled "WHAT'S IT FOR?" with the text "ML-powered editor features for developers." A "STAGE" badge indicates it's a "USABLE PROTOTYPE". On the right, there's a "WHO MADE IT?" section featuring profiles for Irene Alvarado, Idan Gazit, and Amelia Wattenberger. A "SHARE" button is located below the stage badge. At the bottom right, there's a "Sign up for Copilot Labs" button.

The screenshot shows a permission request from "Next Waitlist by GitHub Next". It asks for permission to "Verify your GitHub identity (techupskills)", "Know which resources you can access", and "Act on your behalf". There's a "Learn more" link for the "Act on your behalf" permission. Below these, there's a section for "Resources on your account" with a "Email addresses (read)" option. At the bottom, there are "Cancel" and "Authorize Next Waitlist" buttons. A note says "Authorizing will redirect to https://githubnext.com". At the very bottom, there are status indicators for "Not owned or operated by GitHub", "Created 11 months ago", and "More than 1K GitHub users".



# Signing up for Copilot Labs

- Go to  
<https://githubnext.com/projects/copilot-labs/>
- Not a waitlist
- Have to accept terms
- Install extension
- Copilot installed prereq

The screenshot shows a web browser window with the URL [githubnext.com/projects/copilot-labs/](https://githubnext.com/projects/copilot-labs/). The page has a light blue gradient background. At the top, it says "GitHub Copilot Labs". Below that, it describes the project as "A VS Code extension for experimental applications of GitHub Copilot." It includes sections for "WHAT'S IT FOR?", "STAGE", and "WHO MADE IT?". The "STAGE" section is highlighted with a purple box and says "USABLE PROTOTYPE". The "WHO MADE IT?" section lists three people: Irene Alvarado, Idan Gazit, and Amelia Wattenberger, each with a small profile picture. There are "SHARE" buttons for Facebook, Twitter, and LinkedIn. A large "Sign up for Copilot Labs >" button is at the bottom. A note at the bottom of the page says: "GitHub Copilot is a groundbreaking AI pair programmer for developers, but we're just beginning to explore its potential. Sign up now to help us build the future of development together."



# GitHub Copilot Labs Extension

marketplace.visualstudio.com/items?itemName=GitHub.copilot-labs

Visual Studio | Marketplace

Sign in

New to Visual Studio Code? [Get it now.](#)

**GitHub Copilot Labs** Preview

GitHub [github.com](https://github.com) | 901,794 installs | ★★★★★ (22) | Free

Experimental features for GitHub Copilot

[Install](#) [Trouble Installing?](#)

---

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

Copilot Labs is a place for GitHub to experiment with new ideas before we take them into production. These experiments may never become part of the official product, or may change significantly before they become part of the supported product. Because they are experimental, features may break, not work as expected, or otherwise fail to meet our quality standards for production-ready code. All Copilot Labs features are provided "as-is" with no guarantees of support.

Finally, Copilot Labs is governed by separate terms. GitHub may collect more/different telemetry in Copilot Labs because, again, the experiments are designed for learning and not production.

### GitHub Copilot Labs

GitHub Copilot Labs is a companion extension to [GitHub Copilot](#) that houses experimental and up-and-coming features in a sidebar.

To learn more about GitHub Copilot Labs, [read our post](#).

**Categories**

- [Other](#)

**Tags**

- [keybindings](#)

**Works with**

- Universal

**Resources**

- [Issues](#)
- [Repository](#)
- [Homepage](#)
- [Download Extension](#)

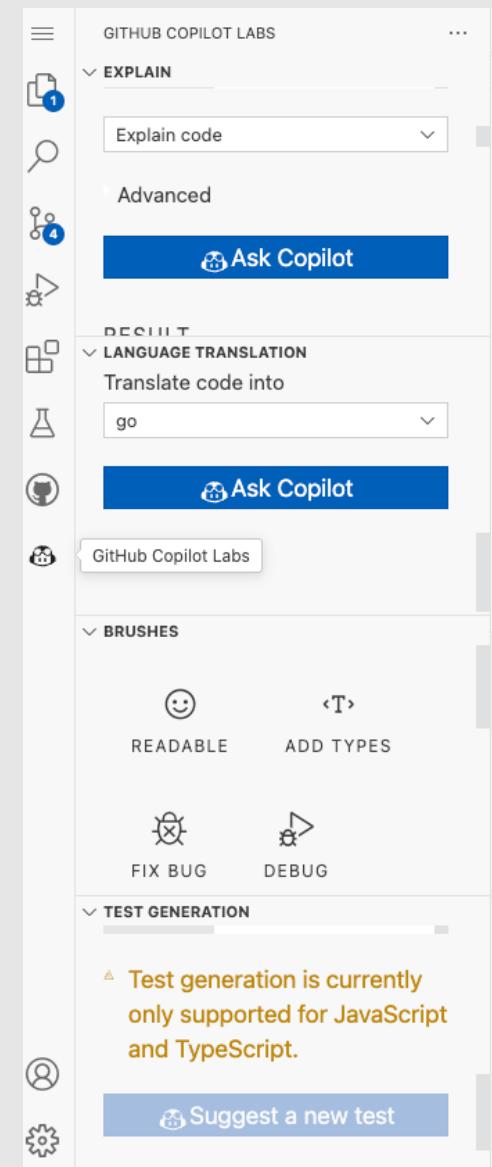
**Project Details**

- [`github/copilot-docs`](#)



# Features of Copilot Labs

- Explain this code
  - Highlight a block of code and ask Copilot to explain it in plain language
- Translate this code
  - Highlight a block of code, select the language you'd like to translate it into
- IDE Brushes
  - Allows you to select code and "paint" it with different kinds of ML transformations
- Test Generation
  - Uses a tool called TestPilot (currently only for JavaScript and TypeScript) to create more meaningful tests based on code and doc





# That's all - thanks!

89

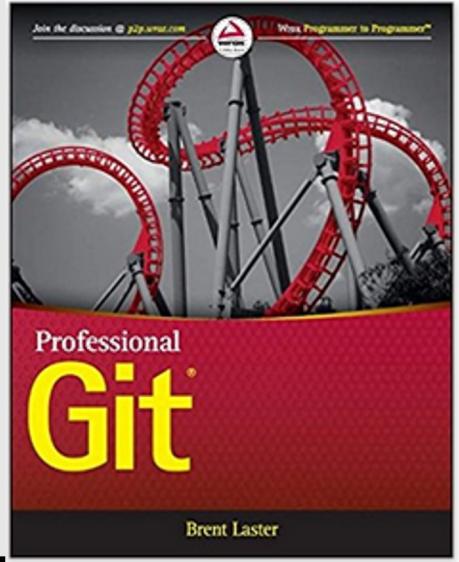
[techskillstransformations.com](https://techskillstransformations.com)  
[getskillsnow.com](https://getskillsnow.com)

## Professional Git 1st Edition

by Brent Laster (Author)

4.5 stars 7 customer reviews

[Look inside](#) ↓



@techupskills

[techupskills.com](https://techupskills.com) | [techskillstransformations.com](https://techskillstransformations.com)

The collage consists of three screenshots from Brent Laster's websites:

- Top Screenshot:** A screenshot of the website [techskillstransformations.com](https://techskillstransformations.com). It features a dark header with the site name and a navigation bar with "Home" and "Contact Us". Below the header is a large image of a woman sitting at a desk, looking down at a laptop and writing in a notebook. To the left of the image is a text box with the heading "TECH LEARNING MADE EASY" and a subtext about upcoming trainings.
- Middle Left Screenshot:** A screenshot of the book cover for "Learning GitHub Actions" by Brent Laster, published by O'Reilly. The cover features a black and white illustration of a monkey holding a smartphone. The title is "Learning GitHub Actions: Automation and Integration of CI/CD with GitHub" by Brent Laster.
- Middle Right Screenshot:** A screenshot of the book cover for "Jenkins 2 Up & Running" by Brent Laster, published by O'Reilly. The cover features a detailed illustration of a fox standing on a purple banner. The title is "Jenkins 2 Up & Running: Evolve Your Deployment Pipeline for Next Generation Automation" by Brent Laster.