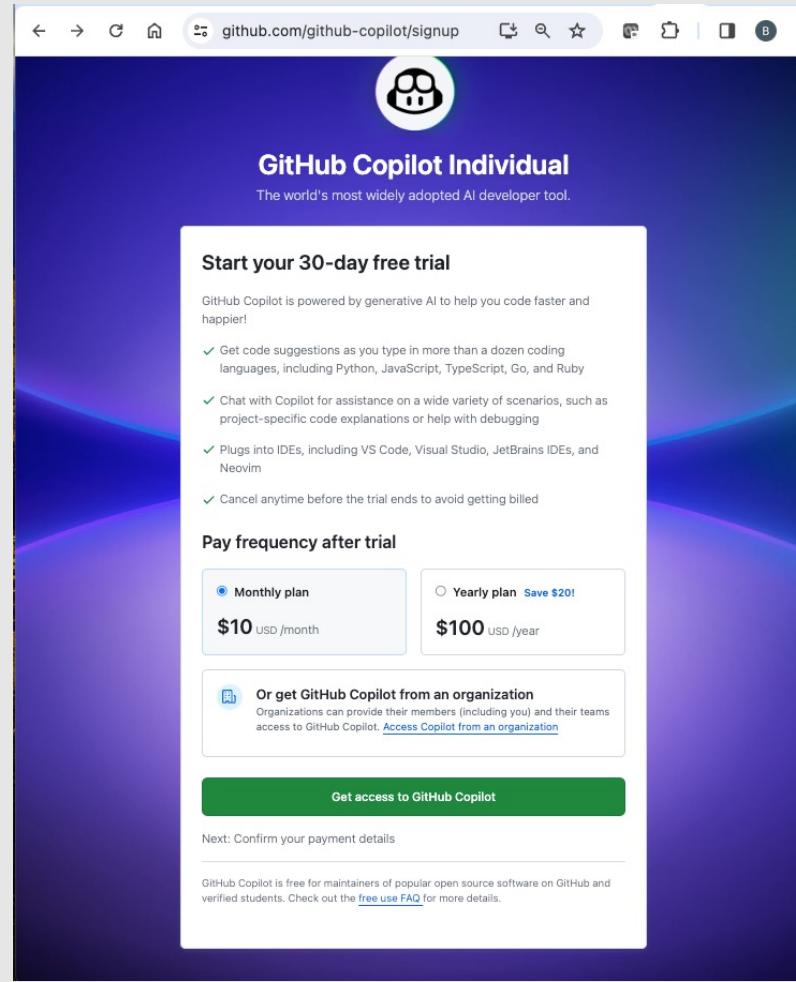




Important! Sign up for the free trial for Copilot

- To do the labs here, you must sign up for the GitHub Copilot trial (or already have it)
- Go to <https://github.com/github-copilot/signup> and follow steps
- If signed in, can also go to <https://github.com/settings/copilot>
- Cheapest plan is fine



The screenshot shows the GitHub Settings page under the 'Copilot' section. It includes a sidebar with account settings like Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, Enterprises, Moderation, Code, planning, and automation, Repositories, Codespaces, Packages, Copilot (which is selected), Pages, Saved replies, Security, Code security and analysis, Integrations, Applications, and Scheduled reminders.

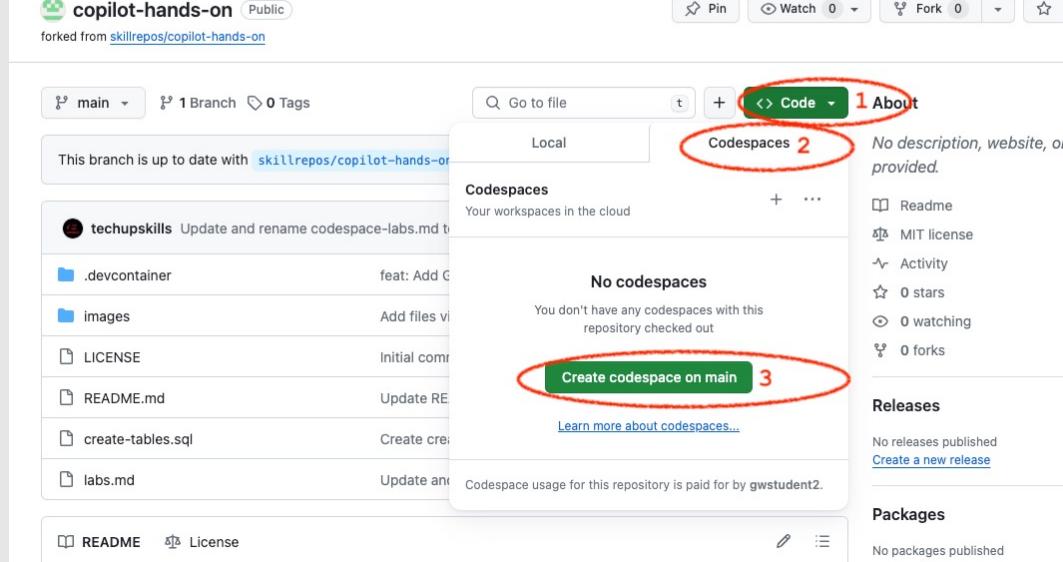


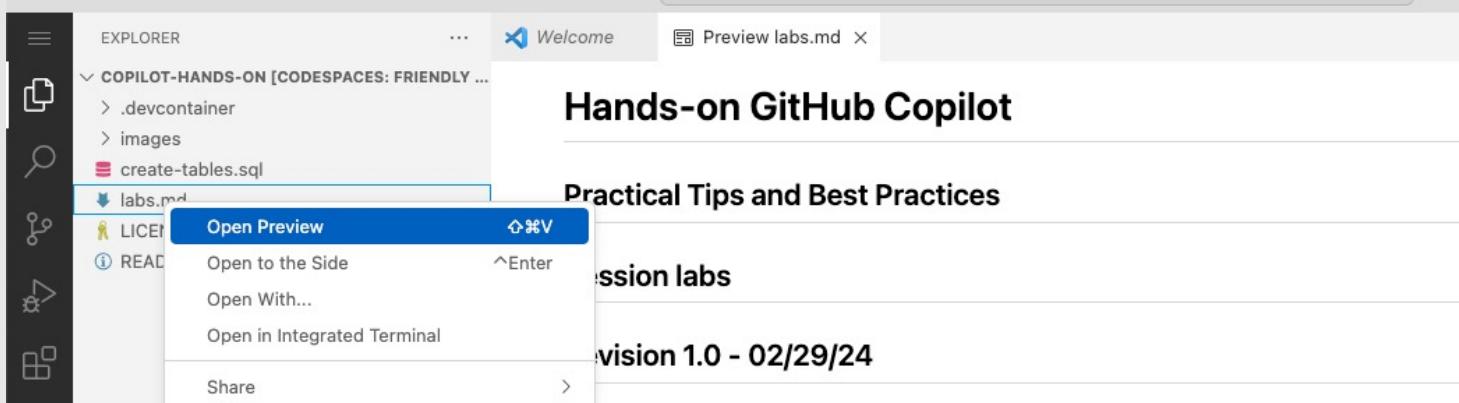
Lab prep

- See <https://github.com/skillrepos/copilot-hands-on>
- Follow instructions in README.md
- Startup codespace on your fork
- Open labs.md in preview mode

The screenshot shows the GitHub Copilot landing page. It features a large "GitHub Copilot" logo with the tagline "Your AI pair programmer". Below it is a prominent button for "Start your 30-day free trial". A detailed list of features follows, including suggestions for multiple languages, integration with various IDEs, and a free annual subscription option.

<https://github.com/skillrepos/copilot-hands-on/blob/main/images/cdd70.e60?raw=true>

The screenshot shows the GitHub repository "copilot-hands-on" with the "Code" dropdown menu open. The "Codespaces" option is highlighted with a red oval. A green button labeled "Create codespace on main" is also highlighted with a red oval and has a red number "3" above it, indicating it's the third step in the process.

The screenshot shows the GitHub Codespace interface for the "friendly-space-fishstick-96g6g755jrfp5qj.github.dev" repository. The "EXPLORER" sidebar shows files like ".devcontainer", "images", "create-tables.sql", "README.md", and "labs.md". A context menu is open over the "labs.md" file, with the "Open Preview" option highlighted with a blue bar and a red oval. The preview window displays the content of "labs.md" in a rich text editor.



Hands-on GitHub Copilot



Presented by Brent Laster &

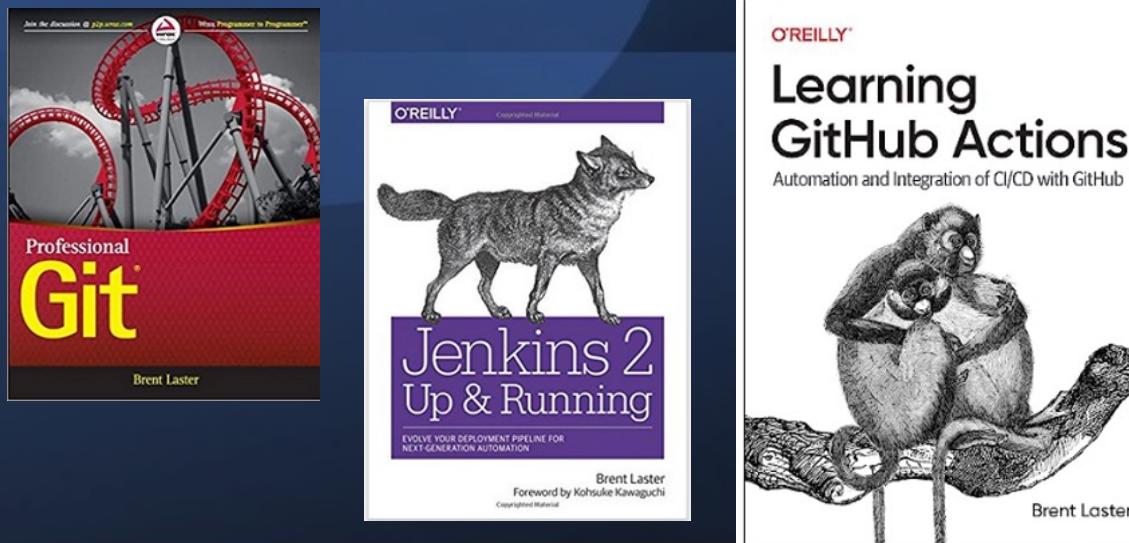
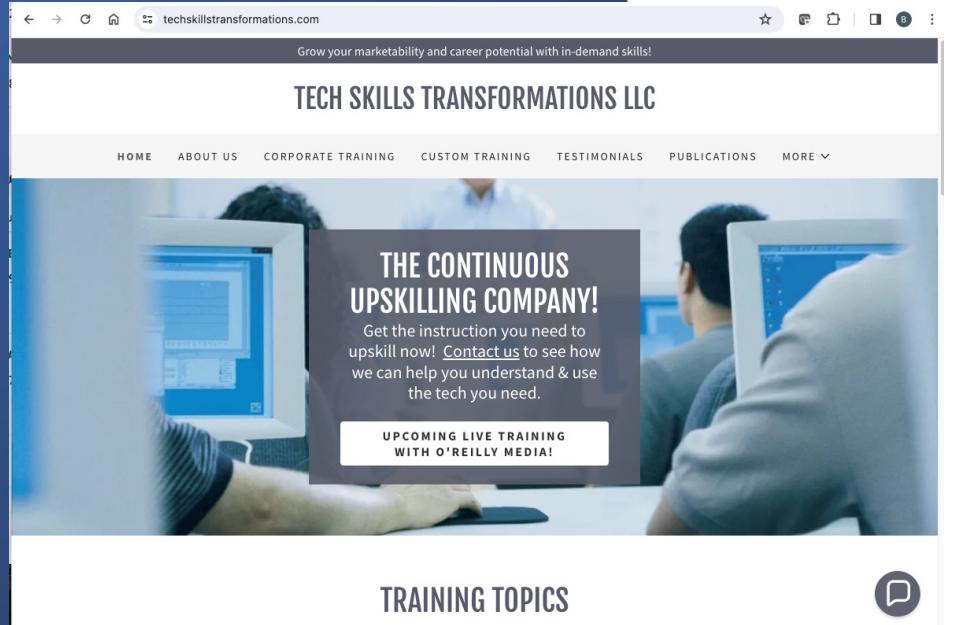
Tech Skills Transformations LLC

© 2024 Brent C. Laster & Tech Skills Transformations LLC



All rights reserved

About me



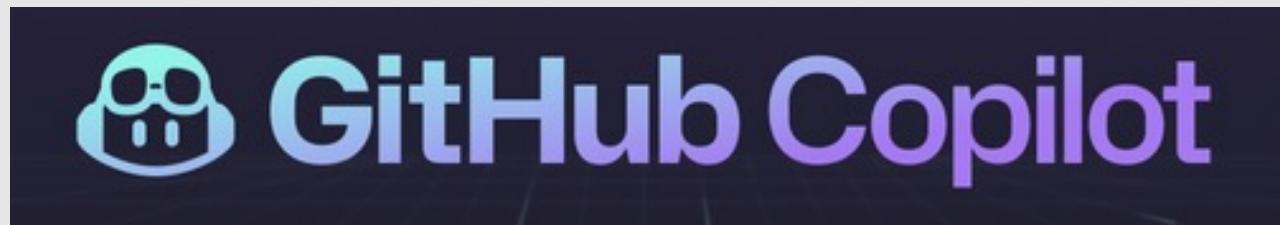
- Founder, Tech Skills Transformations LLC
- Global trainer – O'Reilly, NFJS, etc.
 - Git, GitHub, GitHub Actions
 - Jenkins, Gradle
 - Kubernetes, Tekton, Kustomize
 - Copilot, Codespaces
 - Etc.
- Author -
 - O'Reilly "reports"
 - Professional Git book
 - Jenkins 2 – Up and Running book
 - Learning GitHub Actions
- <https://www.linkedin.com/in/brentlaster>
- @BrentCLaster
- GitHub: brentlaster





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
- Leverage Copilot Chat interface (included) 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>



The Flip Side

New GitHub Copilot Research Finds 'Downward Pressure on Code Quality'

By David Ramel 01/25/2024

New research on the effect of AI-powered GitHub Copilot on software development cites some adverse results.

The "**Coding on Copilot**" whitepaper from GitClear sought to investigate the quality and maintainability of AI-assisted code compared to what would have been written by a human. In other words: "Is it more similar to the careful, refined contributions of a Senior Developer, or more akin to the disjointed work of a short-term contractor?"

The answer to that is summarized in this paragraph from the whitepaper's abstract:

"We find disconcerting trends for maintainability. Code churn -- the percentage of lines that are reverted or updated less than two weeks after being authored -- is projected to double in 2024 compared to its 2021, pre-AI baseline. We further find that the percentage of 'added code' and 'copy/pasted code' is increasing in proportion to 'updated,' 'deleted,' and 'moved' code. In this regard, AI-generated code resembles an itinerant contributor, prone to violate the DRY-ness [don't repeat yourself] of the repos visited."

Source: <https://visualstudiomagazine.com/Articles/2024/01/25/copilot-research.aspx>



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

The screenshot shows a dark-themed messaging interface for GitHub Copilot. At the top, there is a circular profile icon with a white robot head and the text "GitHub Copilot". Below this, a message from the AI says: "Hi @techupskills, how can I help you?". In the main body of the message, the AI provides a note about its nature: "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." The text is white on a black background.



Signup individual #1

The screenshot shows a web browser window with the URL github.com/github-copilot/signup. The main content is the GitHub Copilot sign-up page, featuring a section to start a 30-day free trial and payment options for monthly or yearly plans. On the right side, a sidebar menu is open, with the 'Try Copilot' option circled in red. At the bottom of the page, a note about free access for verified students is also circled in red.

github.com/github-copilot/signup

Paused

gwstudent

Type to search

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

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

- User preferences
 - Allow or block suggestions matching public code
 - Allow GitHub to use code snippets to improve Copilot suggestions
- 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

The screenshot shows a web browser window for github.com/github-copilot/signup/success. The page features a GitHub Copilot logo and the text "GitHub Copilot Individual" and "Your AI pair programmer". Below this, a section titled "Select your preferences" with the sub-instruction "You can change these at any time from your account settings." contains a "Policies" table. The table has one row for "Suggestions matching public code" with two options: "Allowed" (selected) and "Blocked". A red circle highlights the "Select an option" dropdown menu above the table. At the bottom is a green "Save and complete setup" button.

Policies	
Suggestions matching public code GitHub Copilot can allow or block suggestions matching public code. See the GitHub Copilot documentation to learn more.	Select an option ▾ <input checked="" type="checkbox"/> Allowed GitHub Copilot will show suggestions matching public code.
	<input type="checkbox"/> Blocked GitHub Copilot won't show suggestions matching public code.

[Save and complete setup](#)

© 2024 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



Signup Individual #3

The screenshot shows a web browser window with the URL github.com/github-copilot/signup/settings. The page has a dark background with a futuristic, geometric pattern of colored dots and lines. At the top center is a GitHub Copilot icon (a blue robot head wearing glasses). 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 large call-to-action button labeled "Install the GitHub Copilot extension" is centered. To its left, a list of supported code editors includes "Visual Studio Code", "Visual Studio", "JetBrains", and "Neovim", each with a small arrow icon. A pink callout bubble provides 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." At the bottom of the main content area, there's a note about unsupported editors and a link to "Billing and plans". The browser's address bar shows the full URL, and the top right corner displays a user profile picture and the status "Paused".



Post signup - billing info

The screenshot shows the GitHub Settings page for the user "Brent Laster (brentlaster)". The URL in the address bar is `github.com/settings/billing`. The left sidebar has a "Billing and plans" section selected. The main content area displays the "Billing Summary". It shows a "Your next payment" of \$10.00 due by October 12, 2023. A note states that this amount does not include spend on usage of metered services. Below this is a "Your recurring charges & discounts" section showing a GitHub Copilot plan of \$10.00/month starting on Oct 12, 2023. A yellow callout box contains a message about updates for Large File Storage (LFS) billing.

Click to go back, hold to see history

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

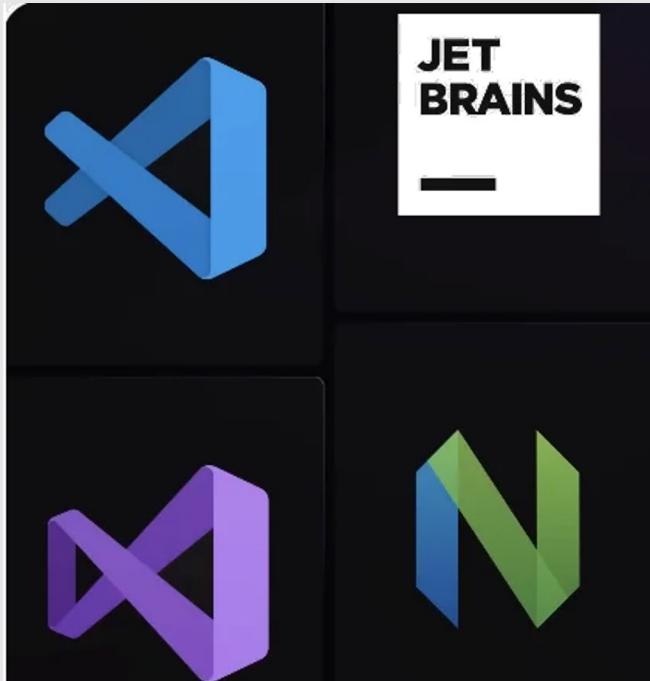
GitHub Copilot \$10.00/month On Oct 12, 2023



Installing GitHub Copilot into your IDE

- <https://docs.github.com/en/copilot/using-github-copilot/getting-started-with-github-copilot>

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



The screenshot shows a web browser displaying the GitHub Docs page at docs.github.com/en/copilot/using-github-copilot/getting-started-with-github-copilot. The page title is "Getting started with GitHub Copilot". The content explains that you can start using GitHub Copilot by installing the extension in your preferred environment. A section titled "Who can use this feature?" states that GitHub Copilot can be managed through personal accounts with GitHub Copilot Individual or through organization accounts with GitHub Copilot Business. It also notes that GitHub Copilot is free to use for verified students, teachers, and maintainers of popular open source projects, with a one-time 30-day trial available for others.

docs.github.com/en/copilot/using-github-copilot/getting-started-with-github-copilot

Github Docs Version: Free, Pro, & Team

GitHub Copilot / Use GitHub Copilot / Getting started

Getting started with GitHub Copilot

You can start using GitHub Copilot by installing the extension in your preferred environment.

Who can use this feature?

GitHub Copilot can be managed through personal accounts with GitHub Copilot Individual or through organization accounts with GitHub Copilot Business.

GitHub Copilot is free to use for verified students, teachers, and maintainers of popular open source projects. If you are not a student, teacher, or maintainer of a popular open source project, you can try GitHub Copilot for free with a one-time 30-day trial. After the free trial, you will need a paid subscription for continued use. For more information, see ["About billing for GitHub Copilot."](#)

Azure Data Studio JetBrains IDEs Vim/Neovim Visual Studio **Visual Studio Code**



Copilot Plans

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 >](#)

Copilot Enterprise

Available Feb 2024

Copilot personalized for your organization throughout the software development lifecycle

\$ 39 per user / month

- ← Everything in Copilot Business, plus...
- ✓ Chat personalized to your codebase
- ✓ Documentation search and summaries
- ✓ Pull request summaries
- ✓ Code review skills
- ✓ Fine-tuned models

*Requires GitHub Enterprise Cloud

• Enterprise

- chat connected to repos on github.com
- smart actions in GitHub, such as gen PR summaries



Organization Settings - Access

The screenshot shows the GitHub Organization Settings page for the 'Tech Skills Transformations LLC Public Trainings Docs' organization. The left sidebar has a 'Copilot' section expanded, with 'Access' selected. The main content area displays 'GitHub Copilot Business' information, including 1 seat purchased at \$19.00 monthly. It also shows that Copilot Business is active in the organization, controlled by 'Selected members'. A search bar at the bottom allows searching by name or email.

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

Getting Started New Tab

skilldocs

Overview Repositories Projects Packages Teams People Settings

Tech Skills Transformations LLC Public Trainings Docs

Organization Switch settings context Go to your organization profile

General

Access

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

Copilot seats

1 seat

Number of purchased seats in this organization.
[Learn more about licenses.](#)

Estimated monthly cost

\$19.00

Each purchased seat is \$19 USD/month.
[Review billing details.](#)

Copilot Business is active in your organization

Control which users or teams will have access to GitHub Copilot inside your organization. Invited users with approved access will receive an email with setup instructions. [View organization settings documentation](#)

Enabled for: Selected members

Access management

Get report Add seats

Search by name or email

https://github.com/organizations/skilldocs/settings/member_privileges



Organization Settings - Policies

The screenshot shows the GitHub Organization Settings page for the 'skilldocs' organization. The URL is <https://github.com/organizations/skilldocs/settings/copilot/policies>. The left sidebar has a tree view of organization settings categories: General, Access, Billing and plans, Repository roles, Member privileges, Import/Export, Moderation, Code, planning, and automation, Repository, Codespaces, Planning, Copilot, and Access. Under Copilot, 'Policies' is selected, indicated by a blue vertical bar. Other options include Content exclusion (Beta). The main content area is titled 'GitHub Copilot policies'. It explains that you can manage policies and grant access to Copilot features for all members with access. If beta features are enabled, you agree to [pre-release terms](#). Three policy sections are listed: 'Suggestions matching public code' (Blocked), 'Copilot Chat in the IDE' (Enabled), and 'Copilot in the CLI' (Beta). The 'Copilot in the CLI' section notes that if enabled, members will get [GitHub Copilot assistance in terminal](#).



Organization Settings - Content Exclusion

19

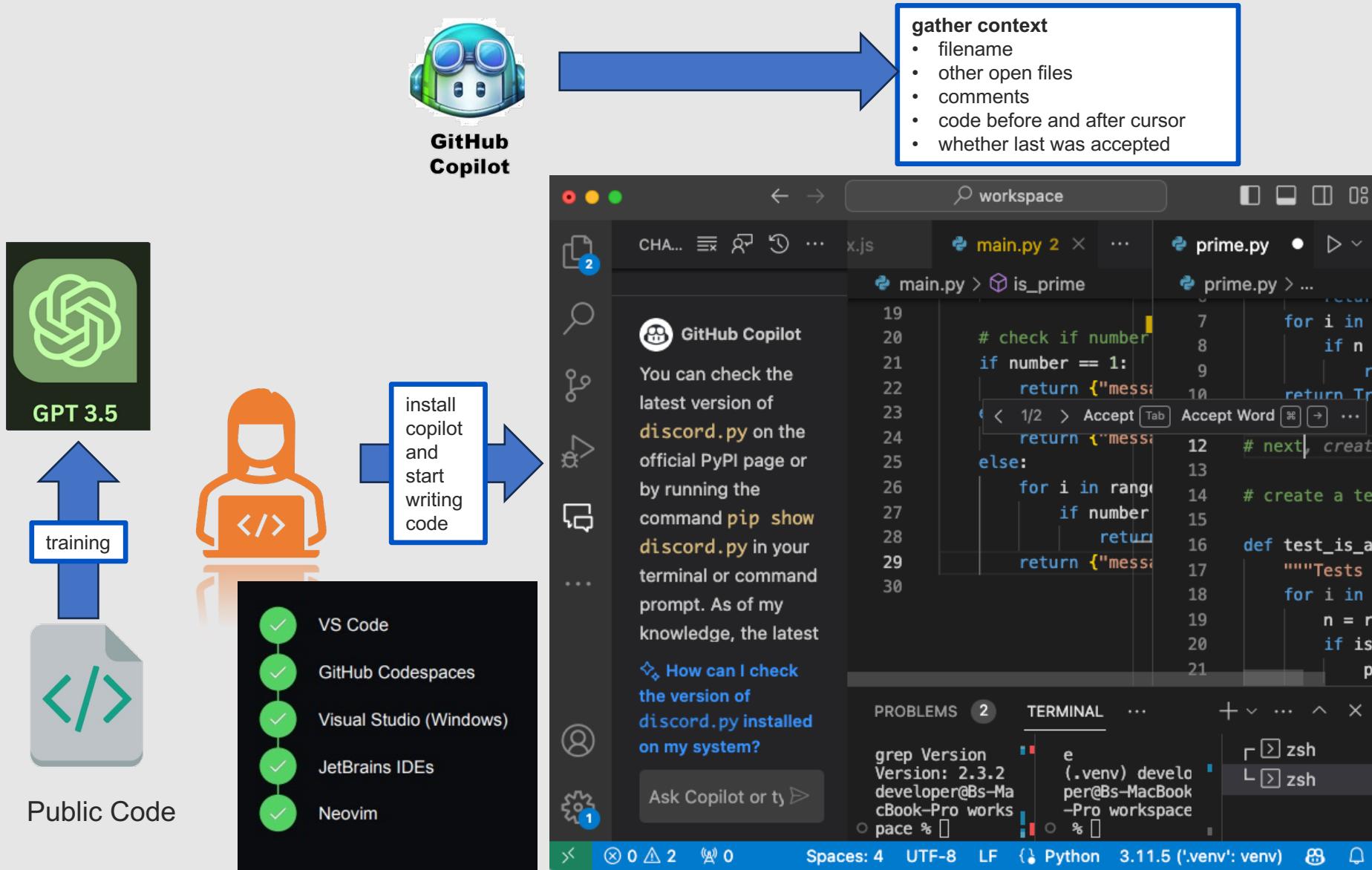
The screenshot shows the GitHub Organization Settings page for the "Tech Skills Transformations LLC Public Trainings Docs" organization. The left sidebar lists various settings categories like General, Access, Billing and plans, Repository roles, Member privileges, Import/Export, and Copilot. The "Content exclusion" section is currently selected and highlighted with a blue bar and a "Beta" badge. The main content area is titled "Content exclusion" and contains instructions about choosing repositories and paths to exclude. It states: "Choose the repositories and paths within your organization that GitHub Copilot should exclude. Copilot won't be able to access or utilize the contents located in those specified paths." Below this, it says: "All exclusions defined at the repository and organization level will apply to all members of your enterprise." A link to "Learn more about setup and usage" is provided. A code block shows example patterns for exclusion:

```
1 # Example patterns:  
  
smile:  
- /secrets/*  
  
git@internal.corp.net:my-team/my-repo:  
- /**/*.env  
- /*/releases/**/*
```

At the bottom, a note says: "Use separate lines to choose which folders and files will be excluded by GitHub Copilot".



How Copilot works (high-level)





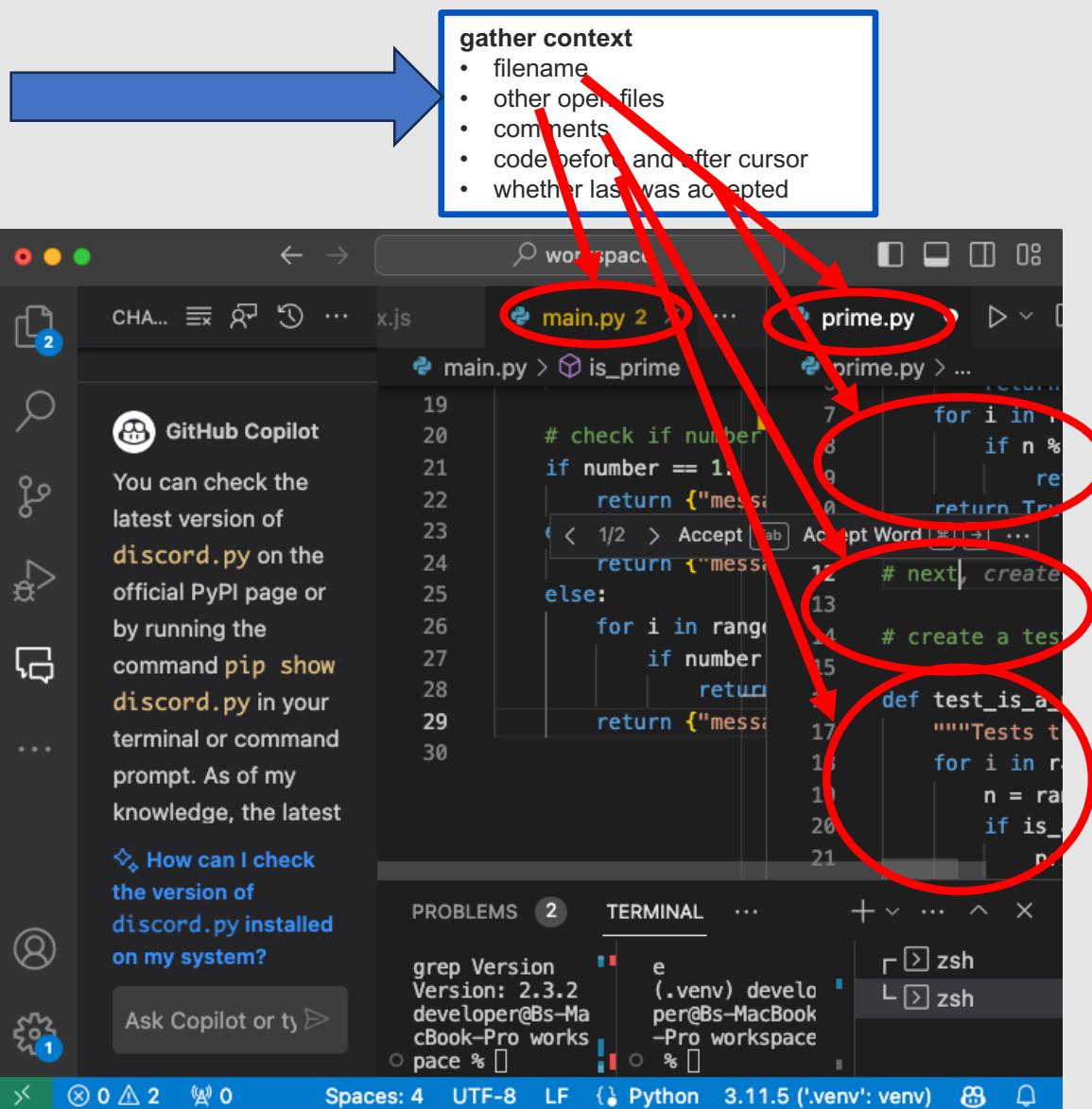
How Copilot works (high-level)



GPT 3.5

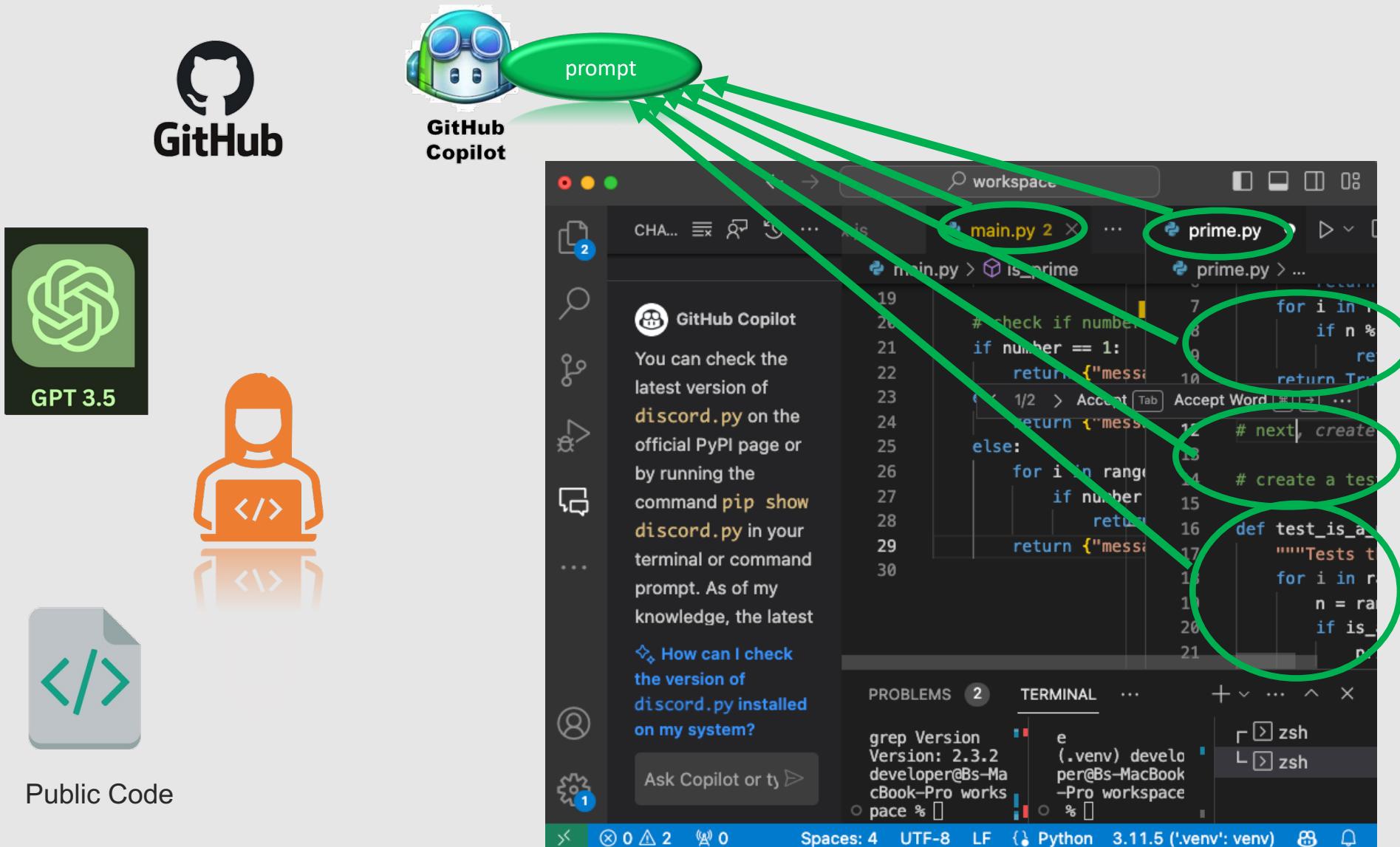


Public Code

GitHub
Copilot

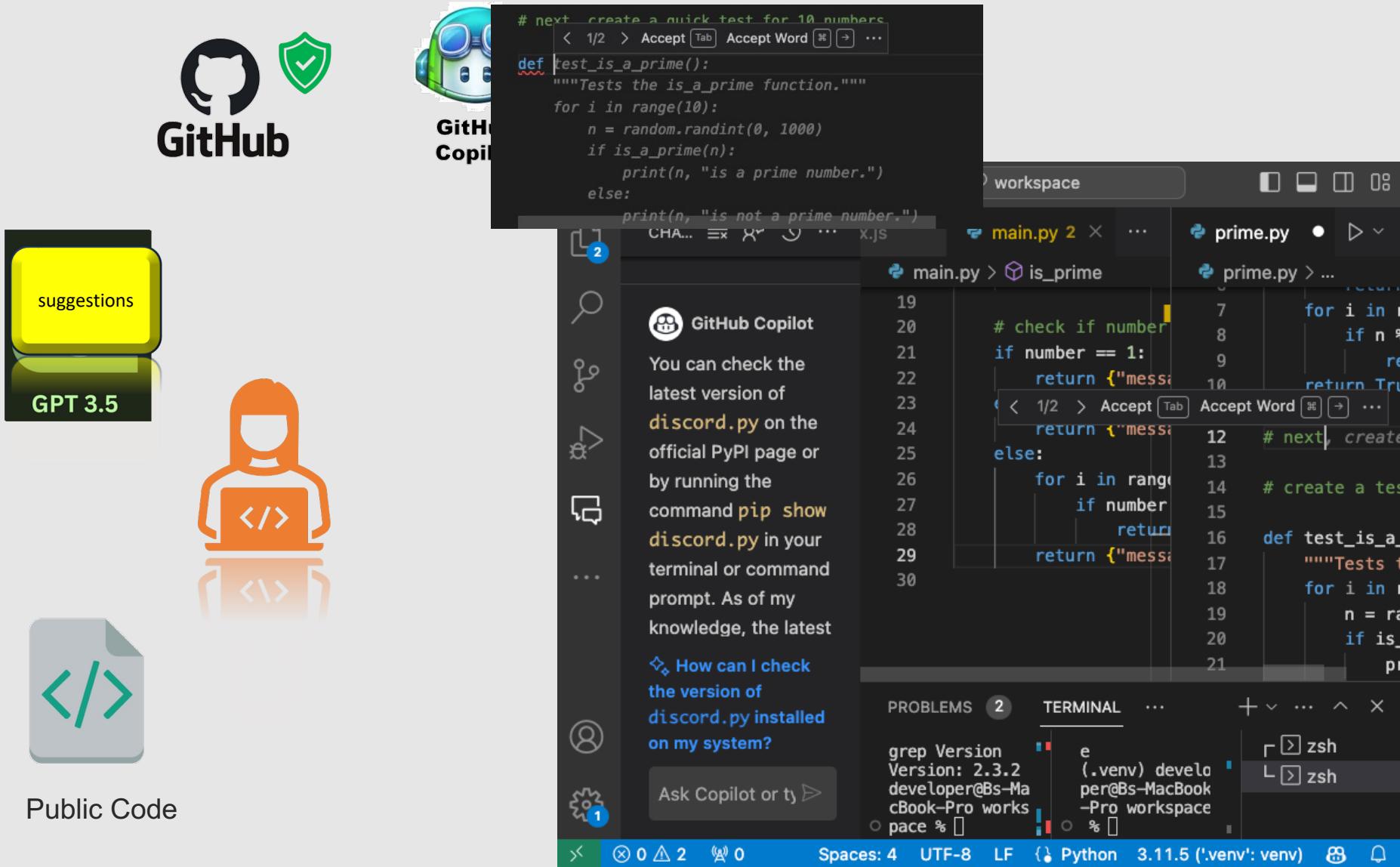


How Copilot works (high-level)



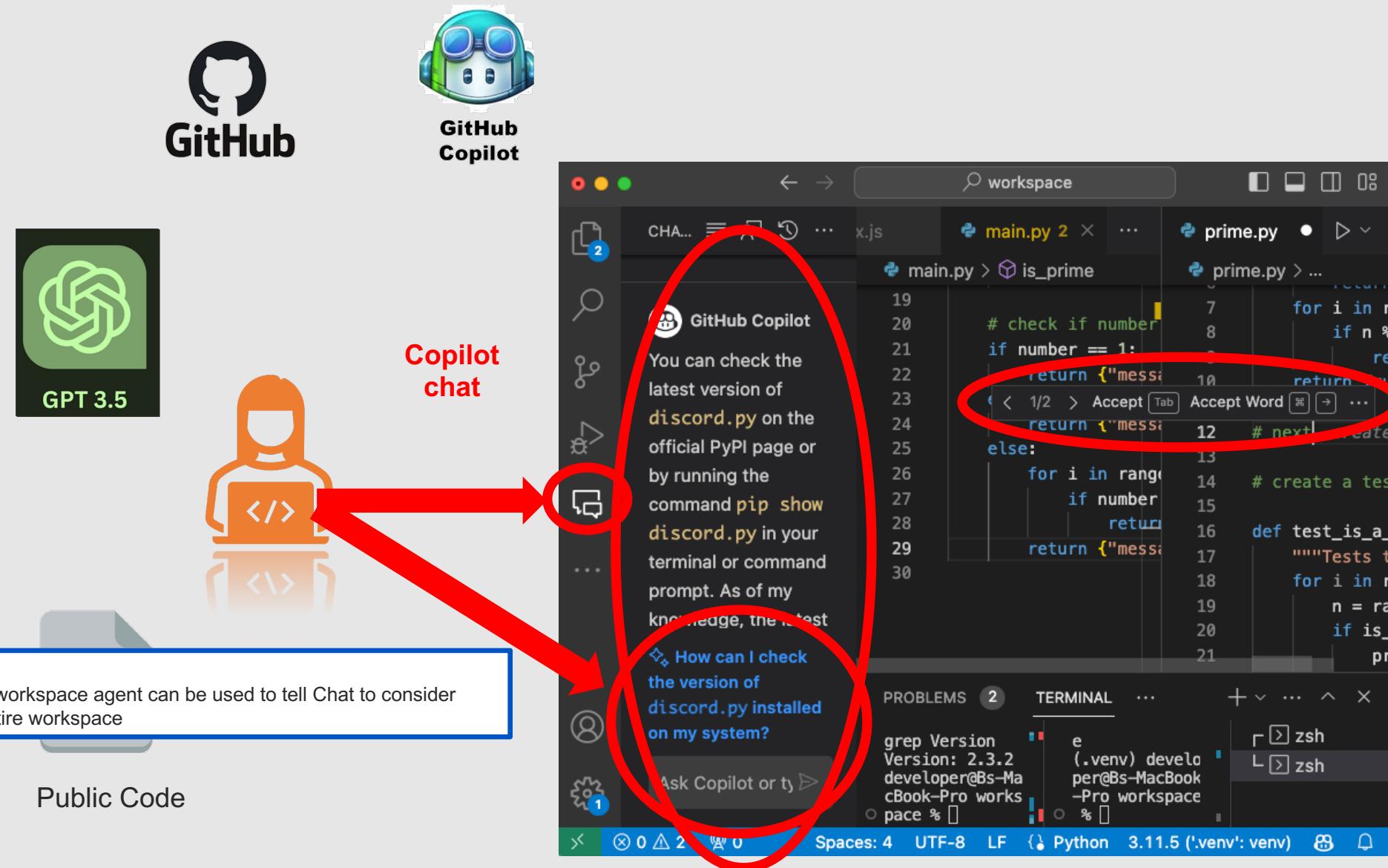


How Copilot works (high-level)



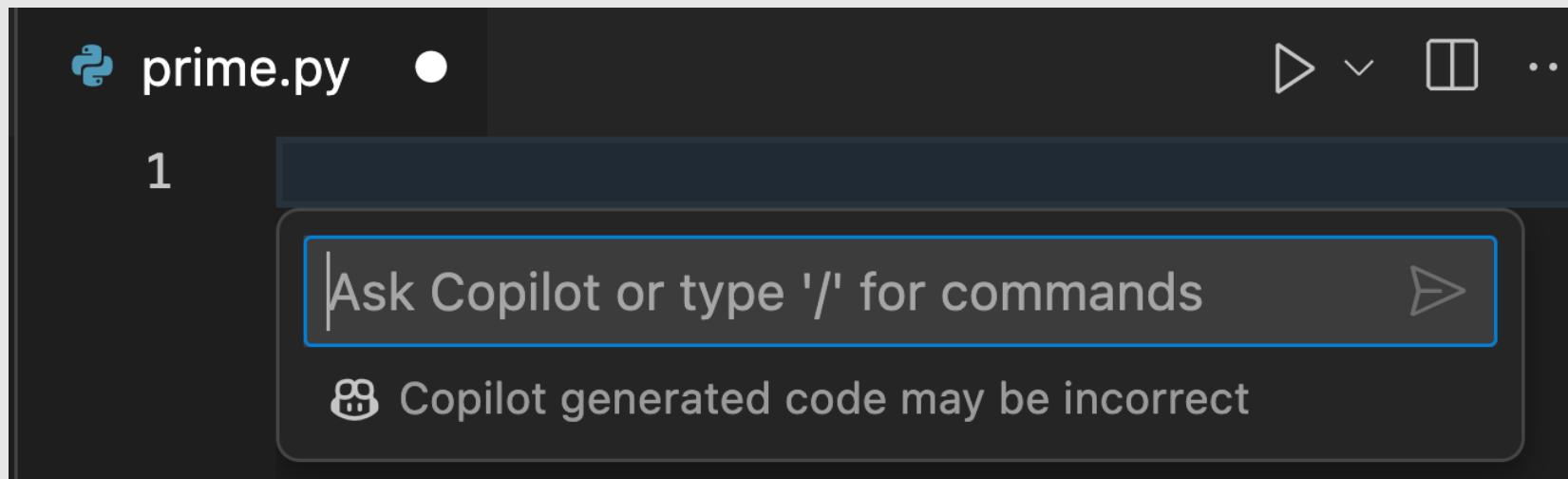
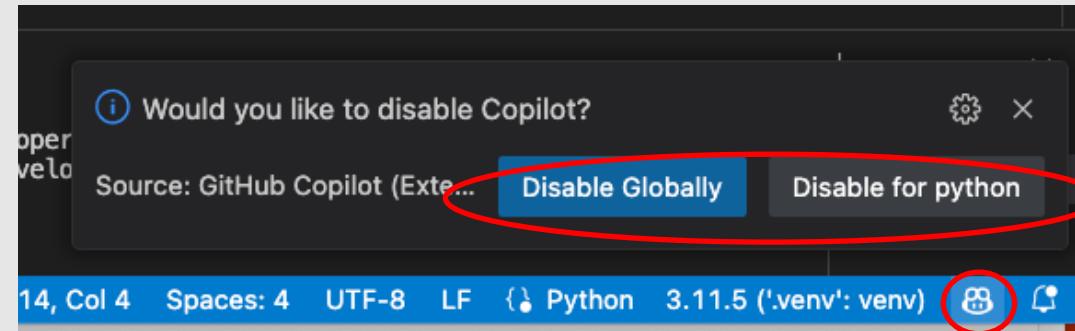
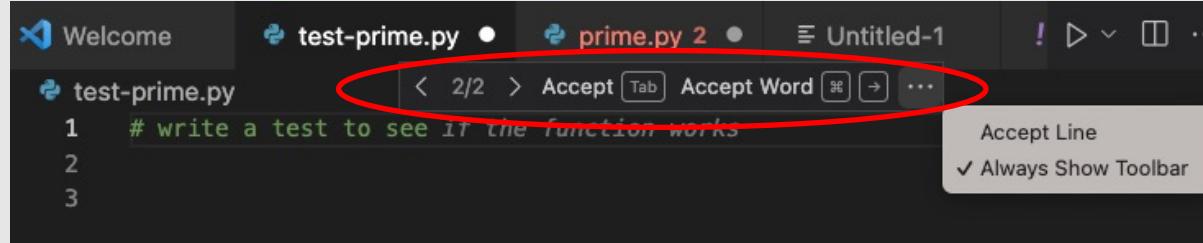


How Copilot works (high-level)



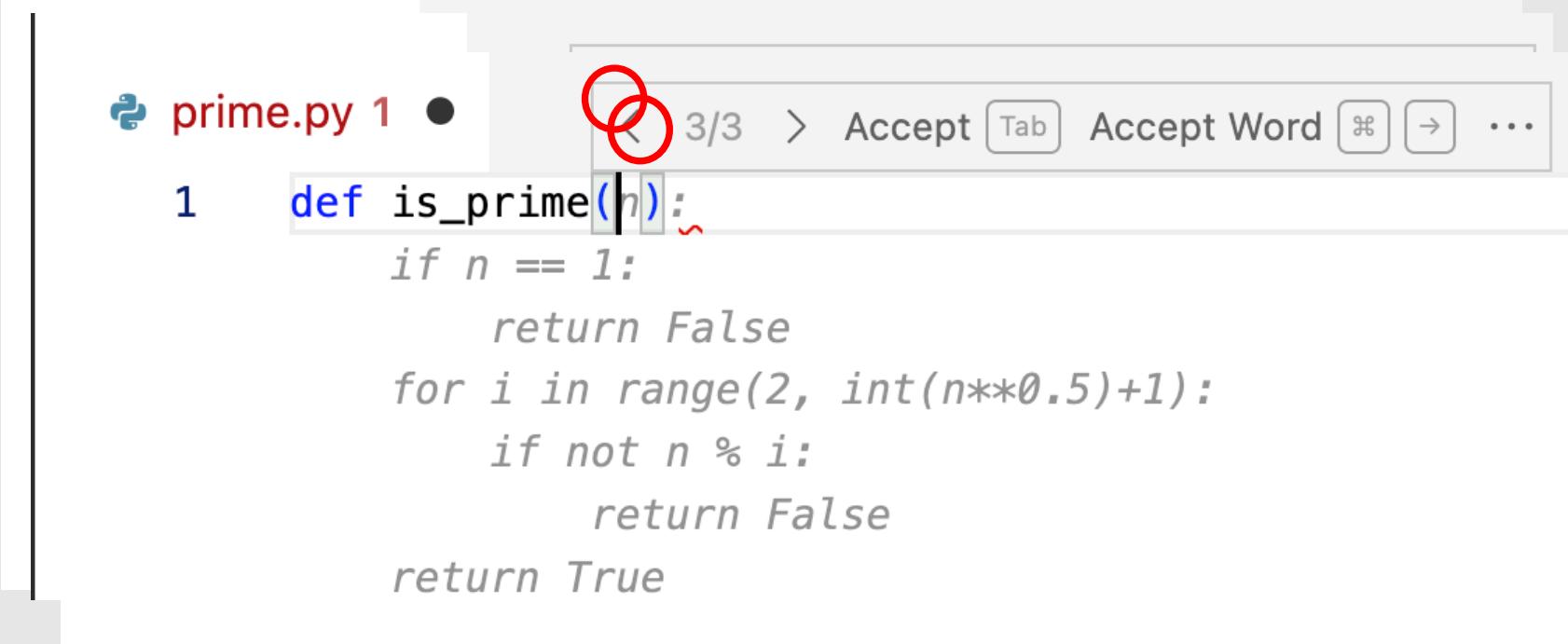


Copilot IDE shortcuts





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

28

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

```
JS index.js U X
JS index.js > ⚡ parseURL
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     // return parsed url
10    return parsedURL;
11 }
12 }
13 
```

```
JS GitHub Copilot 4 X
1 Synthesizing 9/10 solutions (Duplicates hidden)
2
3 Accept Solution
4 =====
5 Suggestion Replace code with this solution
6
7
8 // Path: index.js
9 // function to parse url
10 function parseURL(url) {
11     // parse url
12     var parsedURL = url.split('/');
13     // remove empty values
14     parsedURL = parsedURL.filter(function (n) {
15         return n != ''
16     });
17     // return parsed url
18     return parsedURL;
19 }
20
21
22 // Path: index.js
23 // function to parse url
24 function parseURL(url) {
```



About Labs

- Must be signed up for Copilot prior to doing these
 - <https://github.com/github-copilot/signup>
- 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.)

GitHub Copilot Individual
The world's most widely adopted AI developer tool.

Start your 30-day free trial

GitHub Copilot is powered by generative AI to help you code faster and happier!

- ✓ Get code suggestions as you type in more than a dozen coding languages, including Python, JavaScript, TypeScript, Go, and Ruby
- ✓ Chat with Copilot for assistance on a wide variety of scenarios, such as project-specific code explanations or help with debugging
- ✓ Plugs into IDEs, including VS Code, Visual Studio, JetBrains IDEs, and Neovim
- ✓ Cancel anytime before the trial ends to avoid getting billed

Pay frequency after trial

<input checked="" type="radio"/> Monthly plan \$10 USD /month	<input type="radio"/> Yearly plan Save \$20! \$100 USD /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.



Lab prep

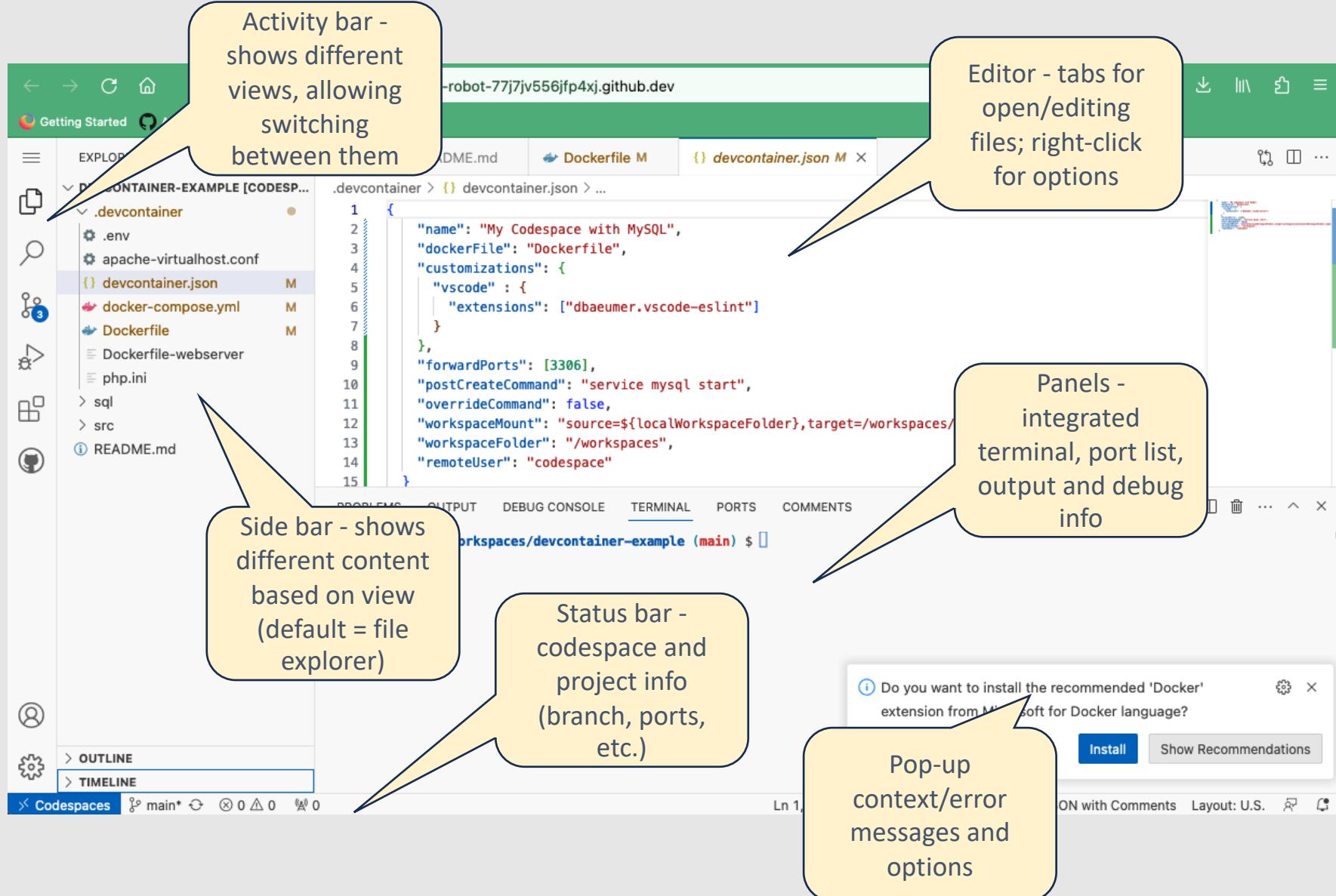
- See <https://github.com/skillrepos/copilot-hands-on>
- Follow instructions in README.md
- Startup codespace on your fork
- Open labs.md in preview mode

The screenshot shows the GitHub Copilot landing page. It features a prominent 'Start your 30-day free trial' button. Below the button, there's a list of benefits: '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!', and 'Cancel anytime before the trial ends to avoid getting billed'. At the bottom, there's a URL: <https://github.com/skillrepos/copilot-hands-on/blob/main/images/cdd70.e6n?raw=true>.

The screenshot shows the GitHub repository 'copilot-hands-on'. The 'Code' dropdown menu is open, with the 'Create codespace on main' option highlighted. Other options in the dropdown include 'Local' and 'Codespaces'. The repository page lists files like .devcontainer, images, LICENSE, README.md, create-tables.sql, and labs.md. On the right side, there's an 'About' section with a note about no description or website provided, and a 'Create codespace on main' button.

The screenshot shows a GitHub Codespace interface. The 'Preview labs.md' tab is active, displaying a guide titled 'Hands-on GitHub Copilot' with sections like 'Practical Tips and Best Practices' and 'Session labs'. The 'EXPLORER' sidebar shows files like .devcontainer, images, create-tables.sql, labs.md, LICENSE, and README. A context menu is open over the 'labs.md' file, with 'Open Preview' highlighted. Other options in the menu include 'Open to the Side', 'Open With...', 'Open in Integrated Terminal', and 'Share'.

Codespace in a browser - components



Lab 1 - Learning how to create good prompts for Copilot

Purpose: In this lab, we'll start to learn about Copilot and how it generates code based on the prompts we provide



Copilot Keyboard shortcuts

Keyboard shortcuts for Windows

Action	Shortcut
Accept an inline suggestion	Tab
Dismiss an inline suggestion	Esc
Show next inline suggestion	Alt +]
Show previous inline suggestion	Alt + [
Trigger inline suggestion	Alt + \
Open GitHub Copilot (additional suggestions in separate pane)	Alt + Enter

Keyboard shortcuts for Linux

Action	Shortcut
Accept an inline suggestion	Tab
Dismiss an inline suggestion	Esc
Show next inline suggestion	Alt +]
Show previous inline suggestion	Alt + [
Trigger inline suggestion	Alt + \
Open GitHub Copilot (additional suggestions in separate pane)	Alt + Enter

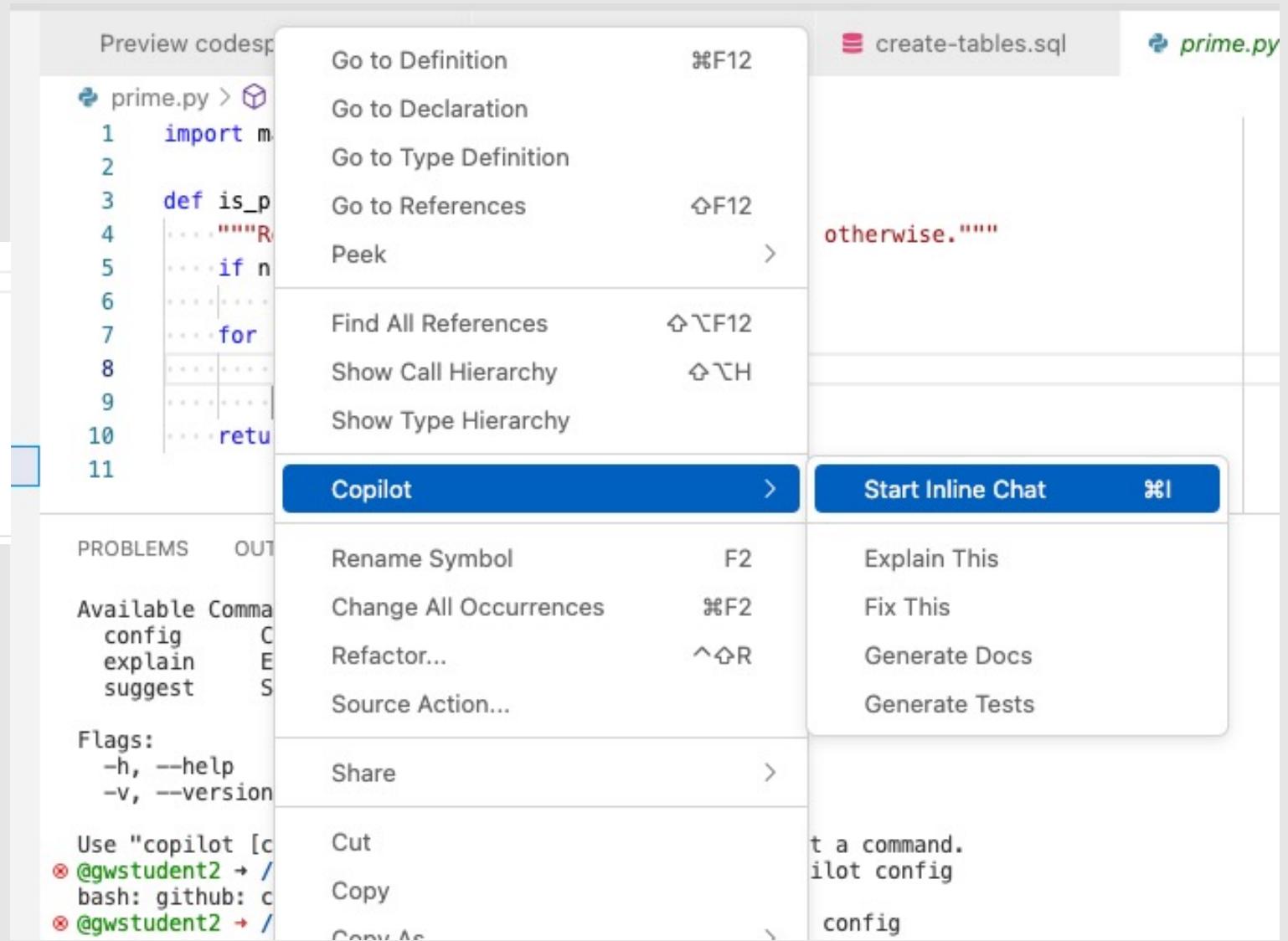
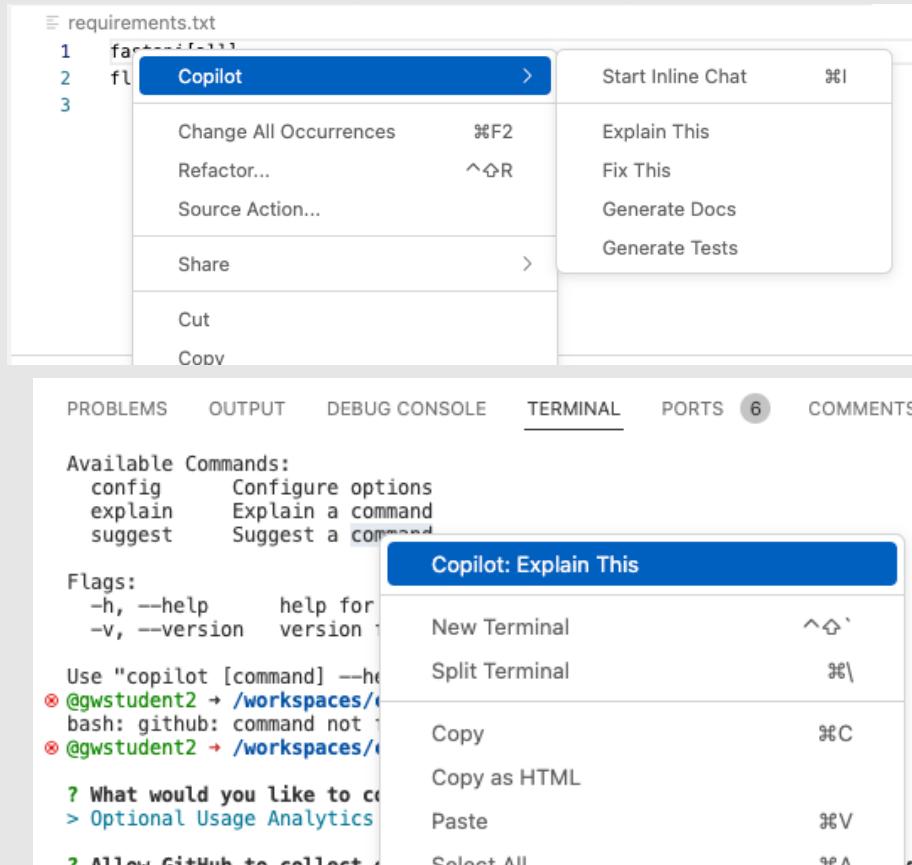
Keyboard shortcuts for macOS

Action	Shortcut
Accept an inline suggestion	Tab
Dismiss an inline suggestion	Esc
Show next inline suggestion	Option (⌘) or Alt +]
Show previous inline suggestion	Option (⌘) or Alt + [
Trigger inline suggestion	Option (⌘) + \
Open GitHub Copilot (additional suggestions in separate pane)	Option (⌘) or Alt + Return



Copilot context menu

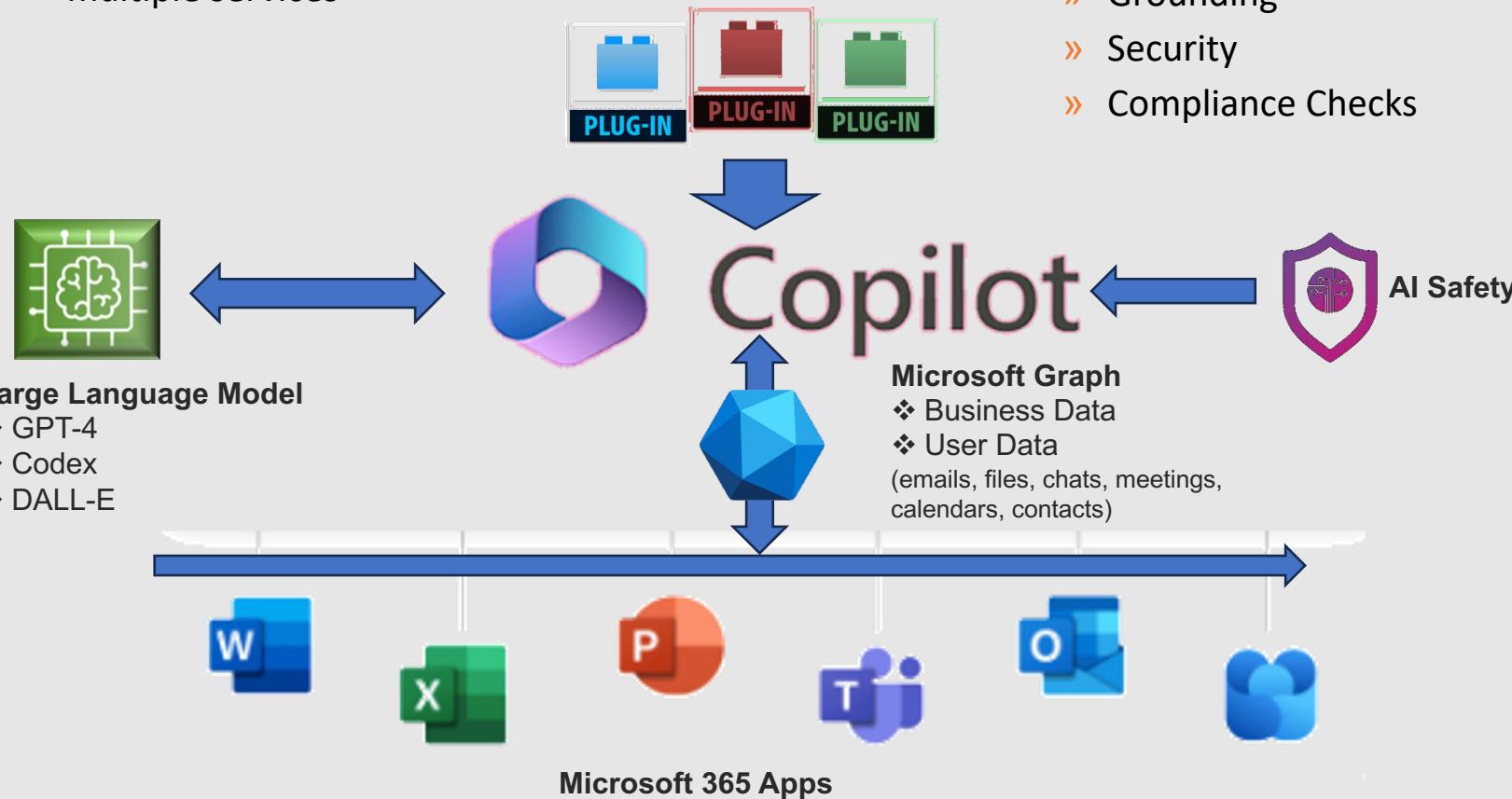
- Right-click
- Select "Copilot"





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

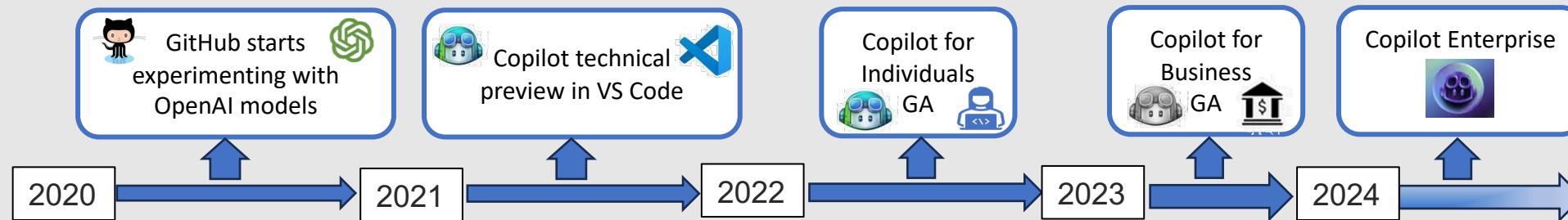


- Asking a question in app,
 - Copilot sends data to Graph to analyze / make sense of query
 - Data sent to GPT-4
 - Answer from ChatGPT sent back to Graph for
 - » Grounding
 - » Security
 - » Compliance Checks



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 GTP 3.5-turbo
 - » Originally trained in early 2022
 - » Model updated in March, 2023
 - » Chat updated to GTP 4 in late 2023
 - Models trained and provided by OpenAI
 - » OpenAI protects models as a trade secret





Chat benefits & commands

- 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

gwstudent2

how tall is the empire state building

GitHub Copilot

Used 1 reference >

Sorry, but I can only assist with programming related questions.

Only useful for programming context!

@terminal	Ask how to do something in the terminal
@vscode	Ask about VS Code
/api	@vscode
/search	@vscode
@workspace	Ask about your workspace
/explain	@workspace
/fix	@workspace
/new	@workspace
/newNotebook	@workspace
/tests	@workspace
/help	@workspace
/clear	

- "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
- @ items are agents - to be discussed later



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 in
@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.

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 `n % i == 0` instead of `not n % i` for clarity

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
```

insert into editor pane at cursor

copy

additional options

Insert Into New File

Run in Terminal ^Enter

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

```
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 Quick Fix
6     Fix using Copilot
7     Explain using Copilot
```

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             if n % i == 0:
7
    /fix "x" is not defined
    Accept Discard ⌂ Changed 1 line ↻
```

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

Can also highlight code and click on light bulb

```
1 function () {
2     const u
3     return f
4     protocol: urlObject.protocol,
5
6
7
8 Rewrite
9
10 Generate 'get' and 'set' accessors
11
12 ⚡ Modify using Copilot
13
14 Surr Enter to apply
15
16 Surround With: Do-While Statement
17
18 }
```

Lab 2 – Using Copilot to simplify and explain code

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



- 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

- Cmd+I
- Use / commands or type question
- Output may appear in chat window

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 Chat 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 1.

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
30  # test for a negative value
31  def test_is_not_prime_3():
32      response = client.get("/prime/-1")
33      assert response.status_code == 200
34      assert response.json() == {"number": -1, "is_prime": False}
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

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

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?

- Generally, 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 usually grab information from all files because of prompt sizes
 - working under constraints for system
 - can use @workspace to look at multiple files in workspace

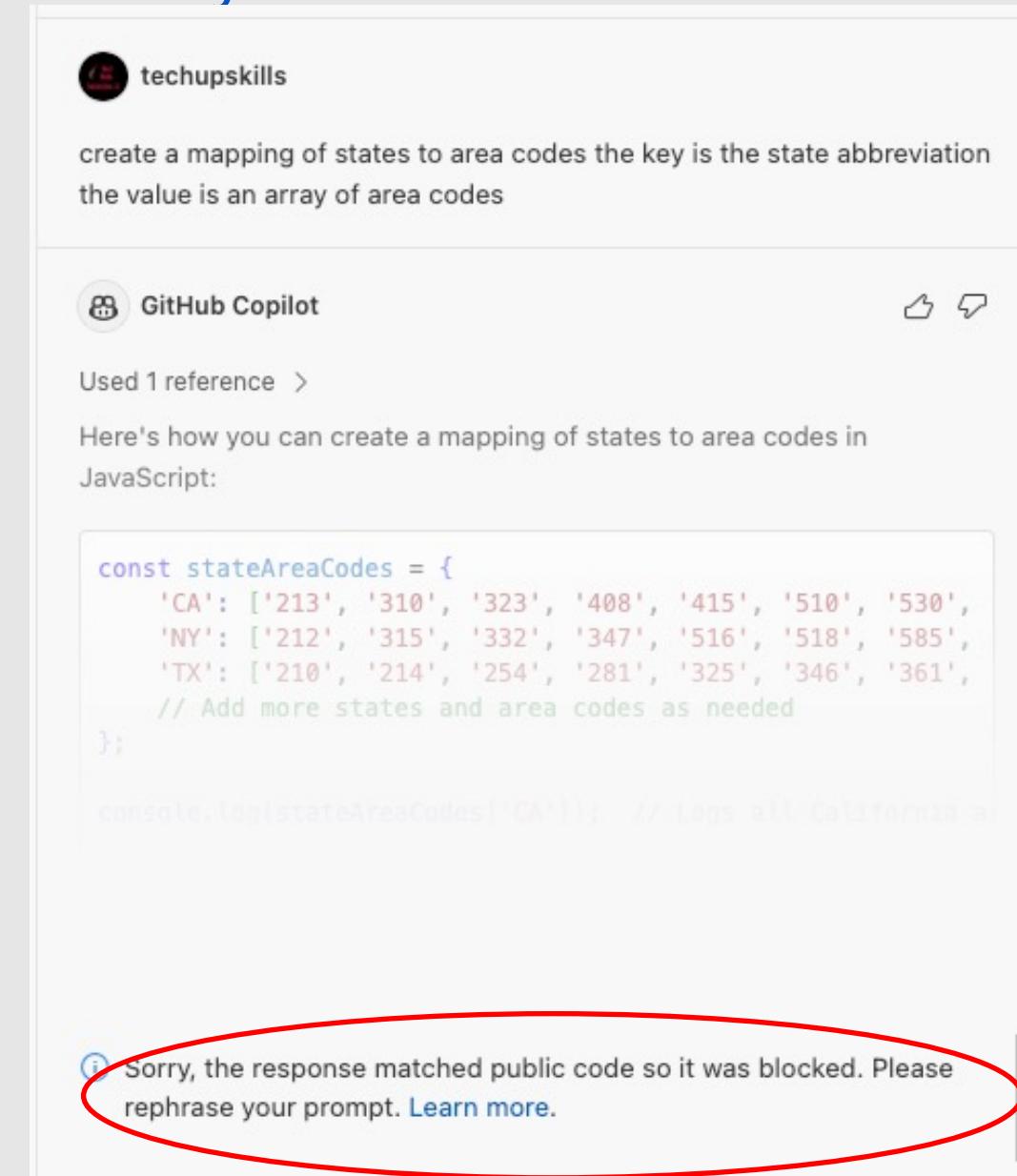
Does GitHub Copilot keep code?

"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>



The screenshot shows a GitHub discussion thread. The first post is by user 'techupskills' with the title 'create a mapping of states to area codes the key is the state abbreviation the value is an array of area codes'. The second post is by 'GitHub Copilot' with the title 'Used 1 reference >'. It contains sample JavaScript code for creating a state-area code mapping:

```
const stateAreaCodes = {
  'CA': ['213', '310', '323', '408', '415', '510', '530'],
  'NY': ['212', '315', '332', '347', '516', '518', '585'],
  'TX': ['210', '214', '254', '281', '325', '346', '361'],
  // Add more states and area codes as needed
};
```

Below this, there is a message from GitHub Copilot: "Sorry, the response matched public code so it was blocked. Please rephrase your prompt. [Learn more](#)".



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 a VS Code interface with a dark theme. On the left is a sidebar with icons for file operations, search, and other tools. The main area shows a Python file named `binary-search.py`. The code implements a binary search algorithm:`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,`On the right, a panel titled "Public code matches" is open, showing results for the current file. It includes a "Match" section with the original code at line 8 and a "Matched content" section with the entire function definition. A "License Summary" section indicates the snippet matches 4 references to public code.

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
18 < 1/1 > Accept Tab Accept Word ⌘ → ...
19 def test_is_prime():
20     assert is_prime(2) == True
21     assert is_prime(3) == True
22     assert is_prime(4) == False
23     assert is_prime(5) == True
24     assert is_prime(6) == False
```

of the code above



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
18 JS index.js 1 ● prime.py ●
19 103+
20 104+
21 105+
22 106+
23 107+
108+
109+
110+
111+
112+
113+
114+
115+
116+
117+
118
assert is_prime(84) == False
assert is_prime(85) == False
assert is_prime(86) == False
assert is_prime(87) == False
assert is_prime(88) == False
assert is_prime(90) == False
assert is_prime(91) == False
assert is_prime(92) == False
assert is_prime(93) == False
assert is_prime(94) == False
assert is_prime(95) == False
assert is_prime(96) == False
assert is_prime(98) == False
assert is_prime(99) == False
assert is_prime(100) == False
/fx "(" 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. On the left, there's a sidebar with a question icon (1 notification) and a gear icon. The main area has two tabs: 'CHAT: GITHUB COPILOT' and 'index.js 1'. The 'index.js 1' tab contains the following code: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`
A red circle highlights the 'prime.py' tab at the top right. Another red circle highlights the 'Insert Into New File' button in a tooltip. A third red circle highlights the '/tests' button in the bottom-left corner of the interface.

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

dev.sql

```

1  -- define a query to get all students enrolled in a course
2  SELECT students.student_id, students.first_name, students.
   last_name, students.email, students.phone, students.city,
   students.state, students.zip_code
3  FROM courses.students
4  INNER JOIN courses.registrations ON students.student_id =
   registrations.student_id
5  INNER JOIN courses.registration_items ON registrations.
   registration_id = registration_items.registration_id
6  INNER JOIN curriculum.subjects ON registration_items.course_id
   = curriculum.subjects.course_id
7  WHERE curriculum.subjects.course_id = 1;
8
9  -- write an index to improve the performance of the query
10 CREATE INDEX idx_registrations_course_id ON courses.
    registrations (course_id);
11
12 -- define a table for student attendance to capture attendance
   by class
13 CREATE TABLE courses.attendance (
14     attendance_id INT IDENTITY (1, 1) PRIMARY KEY,
15     registration_id INT NOT NULL,
16     attendance_date DATE NOT NULL,
17     attendance_status TINYINT NOT NULL

```

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

```

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

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 ...

```

Insert at Cursor

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

Ask Copilot or type '/' for commands

The screenshot shows a GitHub Copilot interface. On the left is a sidebar with icons for file, search, repository, and settings. The main area has a header "CHAT: GITHUB COPILOT". A message from "brentlaster" asks for a stored procedure definition. Below it, a message from "GitHub Copilot" provides a template for the stored procedure. A code editor window shows the beginning of the stored procedure creation. At the bottom, there's a prompt to ask Copilot or type a command.



Copilot and SQL

- Optimizing queries

A screenshot of a code editor showing a block of SQL code. The code includes several lines of comments and actual SQL statements. The SQL statements are color-coded: `SELECT`, `FROM`, and `WHERE` keywords are blue; table names (`courses.registration`, `courses.registrations`, `courses.students`) are orange; and column names (`registration_date`) are brown. Some parts of the code are highlighted with different colors: the `CREATE INDEX` statement and its target column are green, while the `SELECT` statement and its `WHERE` clause are pink. A tooltip at the bottom of the editor window displays the text `/optimize` and buttons for `Accept`, `Discard`, and a refresh icon. To the right of the buttons, it says `Changed 7 lines` with a thumbs-up and thumbs-down icon.

```
58      — select * from courses.registration where year(registration_date) = 2023 and
58+     — month(registration_date) = 9
58+     CREATE INDEX idx_registration_date ON courses.registrations
59+           (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 (
```

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

```

prime.py  dev.sql  create-tables.sql U ...
/workspace/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 join courses.registration_items ri on r.
10    registration_id = ri.registration_id
11
12
13 -- create tables
14 CREATE TABLE training.categories (
15     training_id INT IDENTITY (1, 1) PRIMARY KEY,
16     training_name VARCHAR (255) NOT
17         NULL
18 );
19
20 CREATE TABLE curriculum.vendors (
21     brand_id INT IDENTITY (1, 1) PRIMARY KEY,
22     brand_name VARCHAR (255) NOT NULL
23 );
24
25 CREATE TABLE curriculum.subjects (
26     course_id INT IDENTITY (1, 1) PRIMARY KEY,
27     product_name VARCHAR (255) NOT NULL,
28     branch_id INT NOT NULL,
29     training_id INT NOT NULL,
30     model_year SMALLINT NOT NULL,
31     class_hours DECIMAL (10,2) NOT NULL,
32     FOREIGN KEY (training_id) REFERENCES
33         curriculum.categories (training_id) ON
34             DELETE CASCADE ON UPDATE CASCADE,
35     FOREIGN KEY (brand_id) REFERENCES
36         curriculum.vendors (brand_id) ON DELETE
37             CASCADE
38 );

```

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 seed 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
```

- 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
7
8
```



Redo query with extra context

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

```
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)

66

- Let's ask...

The screenshot shows a GitHub Copilot interface. On the left, a user named 'brentlaster' asks 'what's the latest version of Go?'. Below it, GitHub Copilot responds with a message stating it doesn't have real-time capabilities but provides the latest stable version from September 2021 as Go 1.17. On the right, the interface shows two code snippets. The first snippet is a comment block starting with '11 // q: what's the latest version of go?'. The second snippet is a response block starting with '12 // a: 1.16'. A dropdown menu is open over the response, showing options like 'Accept', 'Accept Word', and '...', with 'Accept' highlighted. Below these, another code block starts with '13 // a: 1.13.4'. At the bottom, there's a text input field with placeholder 'Ask Copilot or type "/" for commands' and a right-pointing arrow button.

```
// q: what's the latest version  
of go?  
// a: 1.16
```

```
11  
12 // q: what's the latest version  
of go?  
13 // a: 1.13.4
```

Ask Copilot or type '/' for commands ➤



Copilot and Kubernetes

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

```
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





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



Examples: how to run frameworks and code to invoke APIs

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?

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

- How do I define my service in Go?

The screenshot shows the GitHub Copilot interface. On the left, under 'CHAT: GITHUB COPILOT', there is a code editor with the following Go code:

```
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,
        },
        ...
    },
}
```

On the right, under 'service.yaml' and 'create-t', there is a code editor with the following YAML configuration:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: my-app
  ports:
    - name: http
      port: 80
      targetPort: 8080
      nodePort: 30100
```

Below the code editors, a text box contains the following explanatory text:

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

 **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

 **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



Copilot agents

75

- Specialized contexts and flows to assist with specific domains
- Three currently
 - @workspace
 - @terminal
 - @vscode
- Also support slash commands
 - ex: @workspace /explain

current agents

@workspace - has knowledge about the code in your workspace; can help navigate and it by finding relevant files or classes

@terminal - get help on how to run a particular command; after response can "insert into terminal"

@vscode - knows about commands and features in the VS Code editor itself; can assist you in using them

gwstudent2
@workspace how can I test my code?

GitHub Copilot used @workspace
Used 7 references >
To test your Python code, you can use a unit testing framework like `unittest` or `pytest`.
If you're using `unittest`, you can create a test file with a structure similar to this:

```
import unittest
from prime import is_prime # assuming
```

gwstudent2
@terminal how do I commit my changes?

GitHub Copilot used @terminal
Used 2 references >
You can commit your changes using the following commands in your terminal:

```
git add .
git commit -m "{commit_message}"
```

- `{commit_message}`: Replace this with a brief description of the changes you made.

Please replace `{commit_message}` with your actual commit message.

gwstudent2
@vscode how do I debug

GitHub Copilot used @vscode

To start debugging in VS Code, you can use the **Debug: Start Debugging** command. This will start a debugging session for the currently active file.

If you want to select a specific configuration before starting the debug session, you can use the **Debug: Select and Start Debugging** command.

To view the debug pane, use the **View: Show Run and Debug** command.

Remember to have a `.vscode/launch.json` file in your workspace with the correct configuration for your debugging environment.

Show "Debug: Start Debugging"

Show "Debug: Select and Start Debugging"

Show "View: Show Run and Debug"



GitHub CLI and Copilot

- Provides a couple of subcommands for use with Copilot from CLI
 - config - configure options
 - » allow optional usage data collection
 - explain - explain a command
 - suggest - suggest a command
- Prereqs
 - GitHub CLI installed
 - Authenticated to GitHub
 - » gh auth
 - Copilot extension installed in command line
 - » gh extension install github/gh-copilot

● @gwstudent2 → /workspaces/copilot-dd (main) \$ gh copilot --help
Your AI command line copilot.

Usage:
copilot [command]

Examples:

```
$ gh copilot suggest "Install git"
$ gh copilot explain "traceroute github.com"
```

Available Commands:
config Configure options
explain Explain a command
suggest Suggest a command

Flags:
-h, --help help for copilot
-v, --version version for copilot

Use "copilot [command] --help" for more information about a command.



GitHub CLI and Copilot - examples

77

```
● @gwstudent2 → /workspaces/copilot-dd (main) $ gh copilot explain "ps -aux"
```

Welcome to GitHub Copilot in the CLI!
version 0.5.4-beta (2024-01-04)

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback
ation, see <https://gh.io/gh-copilot-transparency>

Explanation:

- `ps` is used to display information about active processes.
- `-aux` specifies the options for the `ps` command.
 - `-a` displays processes of all users.
 - `-u` displays a detailed format listing, including the user and other information.
 - `-x` displays processes without a controlling terminal.

```
● @gwstudent2 → /workspaces/copilot-dd (main) $ gh copilot suggest "install k9s"
```

Welcome to GitHub Copilot in the CLI!
version 0.5.4-beta (2024-01-04)

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback
ation, see <https://gh.io/gh-copilot-transparency>

? What kind of command can I help you with?
> generic shell command

Suggestion:

`brew install k9s`

? Select an option [Use arrows to move, type to filter]
Copy command to clipboard
Explain command
Revise command
Rate response
> Exit

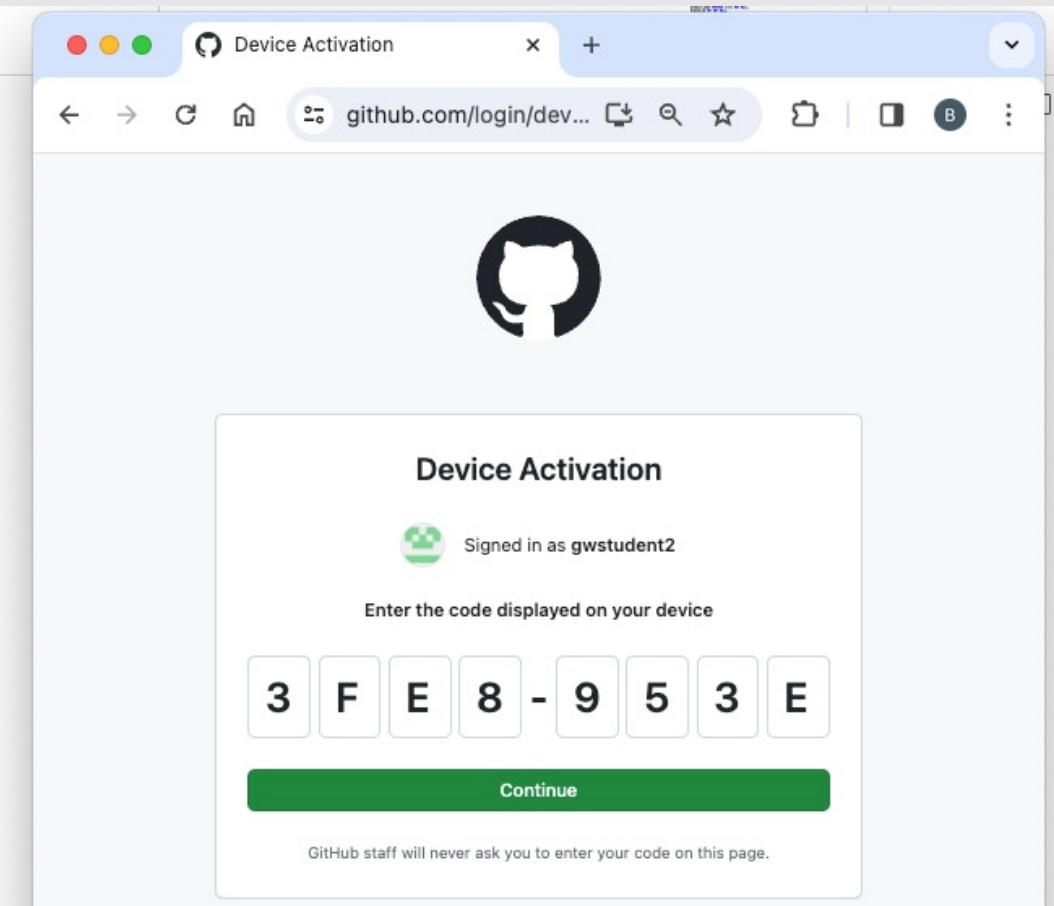


GitHub authentication from the CLI

78

- Certain CLI ops require auth to GitHub
- Done through `gh auth` command
- `--web` option to use browser
- May encounter token error
 - `export GITHUB_TOKEN=`
- Copy code for browser activation

```
2 CREATE TABLE training.categories (
3   + training_id INT IDENTITY (1, 1) PRIMARY KEY,
4   + training_name VARCHAR (255) NOT NULL
)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 6 COMMENTS
@gwstudent2 ~ /workspaces/copilot-dd (main) $ gh auth login --web
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
! First copy your one-time code: 3FE8-953E
Press Enter to open github.com in your browser...
```



Lab 9 – Agents and CLI

Purpose: In this lab, we'll get some practice using Copilot agents and the Copilot CLI



About Enterprise

- Available for enterprises that use GitHub Enterprise Cloud
- Includes
 - GitHub Copilot Chat in GitHub.com
 - Copilot pull request summaries
 - Copilot docset management
 - Additional features to come...



Future: 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>



That's all - thanks!

82

Contact: training@getskillsnow.com

techskillstransformations.com
getskillsnow.com

Professional Git 1st Edition

by Brent Laster (Author)

★★★★★ 7 customer reviews

[Look inside](#) ↴

