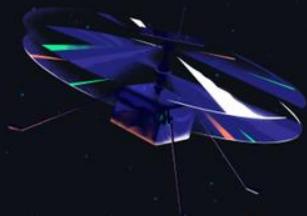




GitHub Copilot for Developers



AGENDA

Demo/Workshop

Coding

Improving code quality

Common Problems

Q & A

Outcome of this training

At the end of training, you will achieve the following results...

- ✓ Get answers to specific use case scenario questions
- ✓ Increase existing Copilot skills by following a specific workshop tutorial
- ✓ Learn in-depth tips and tricks and best practices on how to best utilise GitHub Copilot

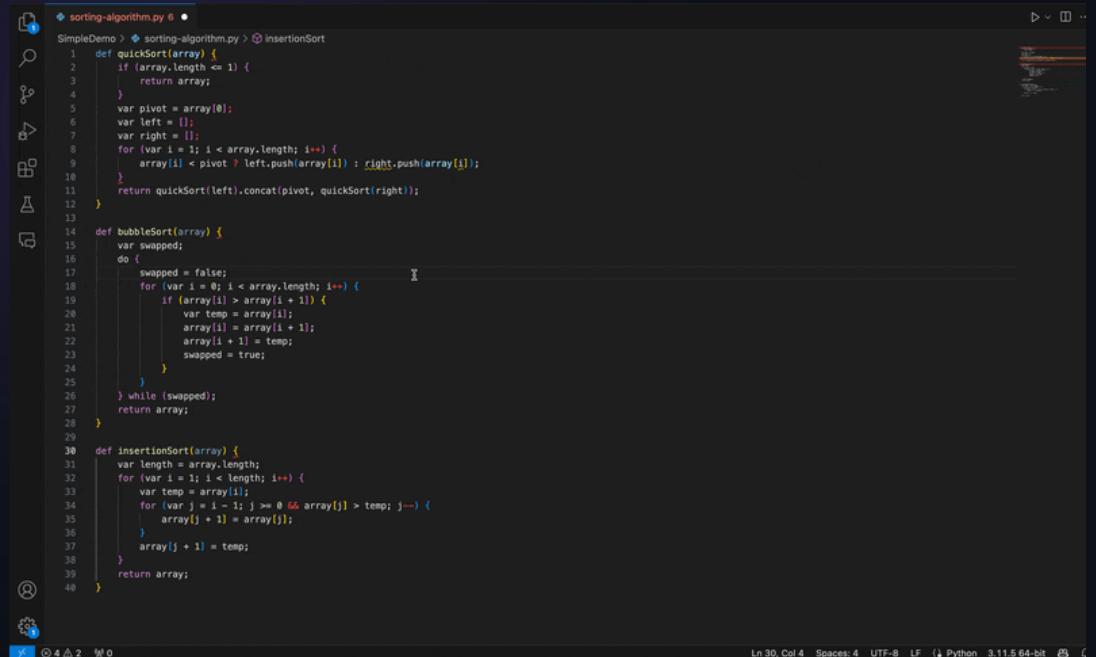


Workshop

Coding

In-file Copilot options

Copilot now
offers **in-file**
Copilot feature



```
sorting-algorithm.py 6
SimpleDemo > sorting-algorithm.py > insertionSort
1 def quickSort(array) {
2     if (array.length <= 1) {
3         return array;
4     }
5     var pivot = array[0];
6     var left = [];
7     var right = [];
8     for (var i = 1; i < array.length; i++) {
9         if (array[i] < pivot) {
10             left.push(array[i]);
11         } else {
12             right.push(array[i]);
13         }
14     }
15     return quickSort(left).concat(pivot, quickSort(right));
16 }
17
18 def bubbleSort(array) {
19     var swapped;
20     do {
21         swapped = false;
22         for (var i = 0; i < array.length; i++) {
23             if (array[i] > array[i + 1]) {
24                 var temp = array[i];
25                 array[i] = array[i + 1];
26                 array[i + 1] = temp;
27                 swapped = true;
28             }
29         }
30     } while (swapped);
31     return array;
32 }
33
34 def insertionSort(array) {
35     var length = array.length;
36     for (var i = 1; i < length; i++) {
37         var temp = array[i];
38         for (var j = i - 1; j >= 0 && array[j] > temp; j--) {
39             array[j + 1] = array[j];
40         }
41         array[j + 1] = temp;
42     }
43     return array;
44 }
```

Data Generation

Ask Copilot to generate data

CHAT: GITHUB COPILOT

bryantson Produce 10 exemplars for sentiment analysis. Examples are categorized as either positive or negative. Produce 2 negative examples and 8 positive examples. Use this format for the examples:
Q: A:

 GitHub Copilot Sure, here are 10 examples for sentiment analysis:

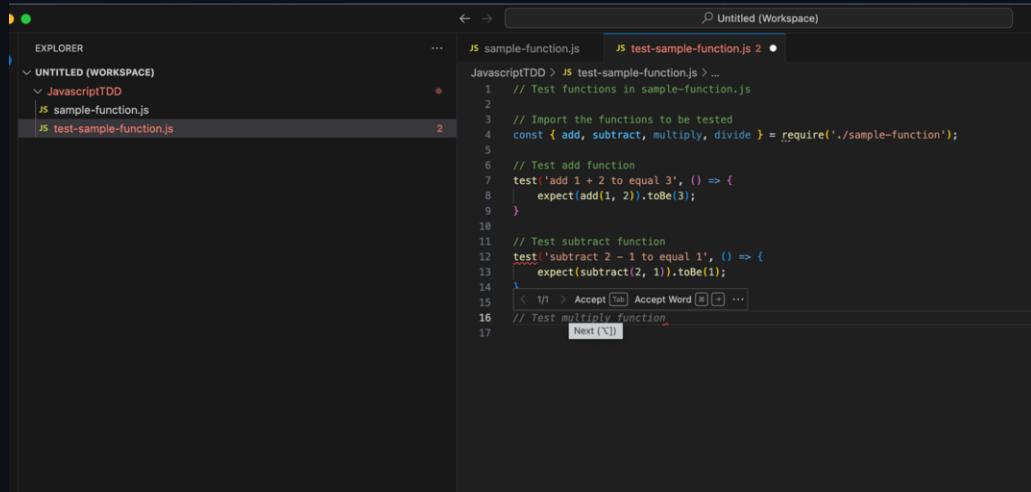
1. Q: The code runs smoothly without any errors. A: Positive
2. Q: The application's user interface is intuitive and user-friendly. A: Positive
3. Q: The new feature implemented in the software significantly improves its performance. A: Positive
4. Q: The documentation for the project is comprehensive and easy to understand. A: Positive
5. Q: The code is well-structured and adheres to the coding standards. A: Positive
6. Q: The application crashes frequently. A: Negative
7. Q: The software update has made the application slower. A: Negative
8. Q: The team resolved the software bugs quickly. A: Positive

How can I integrate sentiment analysis into my application?

Ask Copilot or type '/' for commands >

Test Driven Development (TDD)

Test Driven
Development
(TDD) practice is
possible

A screenshot of a code editor interface, likely Visual Studio Code, demonstrating Test Driven Development (TDD).

The interface shows:

- EXPLORER** sidebar: UNTITLED (WORKSPACE) > JavascriptTDD > sample-function.js > test-sample-function.js
- Editor Area:** A file named "test-sample-function.js" is open, showing a test script for a "sample-function.js" module.

```
// Test functions in sample-function.js
const { add, subtract, multiply, divide } = require('./sample-function');

// Test add function
test('add 1 + 2 to equal 3', () => {
  expect(add(1, 2)).toBe(3);
})

// Test subtract function
test('subtract 2 - 1 to equal 1', () => {
  expect(subtract(2, 1)).toBe(1);
})

// Test multiply function
test('multiply 2 * 3 to equal 6', () => {
  expect(multiply(2, 3)).toBe(6);
})
```
- Status Bar:** Shows "Untitled (Workspace)" and file navigation controls.

Test Driven Development (TDD)

Sources > PeopleKit > Person.swift

1

Test Driven Development (TDD)

```
Tests > PeopleKitTests > PersonTests.swift > PersonTests
3 @testable import PeopleKit
2 import XCTest
1
4 class PersonTests: XCTestCase {
1 }
2
```

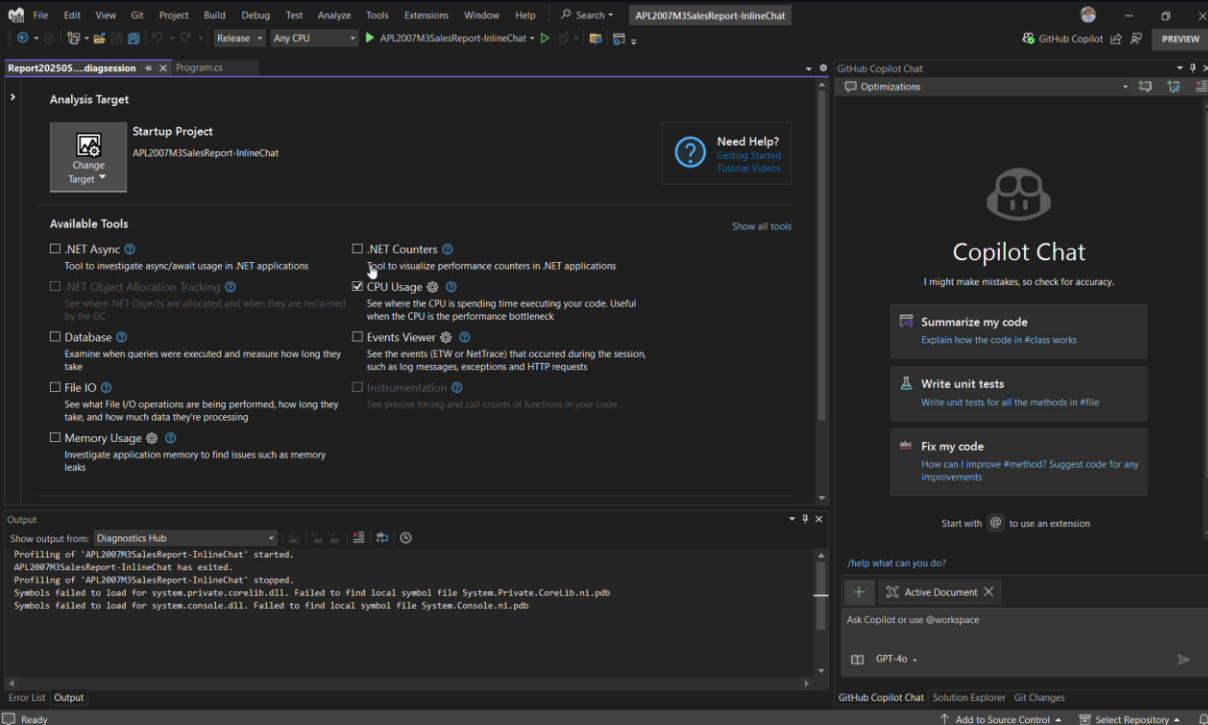
Test Driven Development (TDD)

Sources > PeopleKit > Person.swift > Person

```
1 struct Person {
2     let givenName: String
1     let familyName: String
2 }
3
```

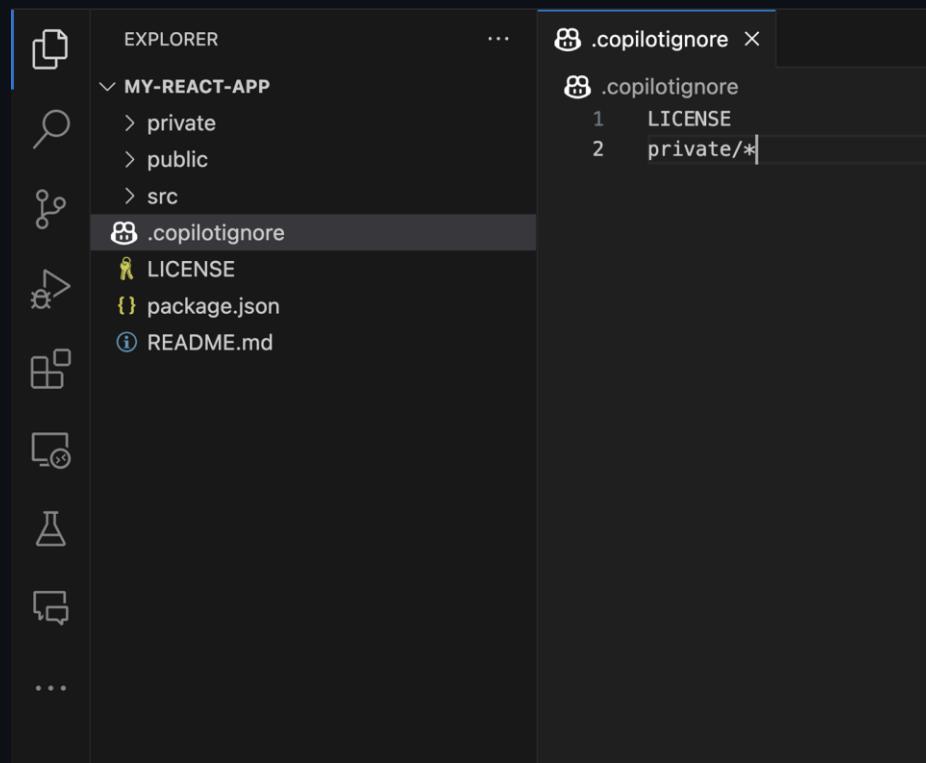
Code Profiling

GitHub Copilot CPU usage tool



Block files from Copilot

Use `.copilotignore` to block files and folders from being used by Github Copilot



Block files from Copilot

Ignore specific
repositories and
paths in Restricted
Content

The screenshot shows the GitHub organization settings for 'evil-copilot'. The left sidebar lists various settings categories like General, Features, Access, and Copilot. The 'Copilot' section is expanded, showing 'Restricted content' (Beta). The main area displays a YAML configuration for ignoring specific repositories and paths. A tooltip at the bottom provides keyboard navigation instructions.

```
1  /**: ["**/.env"]  
2    
3  marais-testing-repo:  
4    - "/src/kernel.rs"  
5    
6  git@github.com:prime/react.git:  
7    - "e2e/*"  
8    
9  git@gitlab.com:openity-labs/surreal-react.git:  
10   - "fly.toml"  
11   - "**/surrealdbjs/*"
```

Use `Control + Shift + m` to toggle the `tab` key moving focus. Alternatively, use `esc` then `tab` to move to the next interactive element on the page.
Use separate lines to choose what folders and files will be ignored by GitHub Copilot

maraistr last edited 1 minute ago

[Discard changes](#) [Save changes](#)

Code Debugging

Debugging with GitHub Copilot

A screenshot of a code editor showing a Python script for a game. The script contains a function named `collisions` which checks if the player's plane has collided with any obstacles. A tooltip from GitHub Copilot suggests fixing a syntax error by adding a colon at the end of a line. The code editor shows the current state of the code with some lines highlighted.

```
+1
42     def collisions(self):
43         if pygame.sprite.spritecollide(self.plane, self.collision_sprites,
44             or self.plane.rect.top <= 0:
45             for sprite in self.collision_sprites.sprites():
46 -                 if sprite.sprite_type = 'obstacle':
46+                 if sprite.sprite_type == 'obstacle':
47                     sprite.kill()
48                     self.active = False
49                     self.plane.kill()
50
```

/fix Expected ":"

Accept Discard ⌂ Changed 1 line ↗ ↘

GitHub Rest API

GitHub Copilot Rest API

GET

/enterprises/{enterprise}/copilot/usage

cURL

JavaScript

GitHub CLI

```
# GitHub CLI api
# https://cli.github.com/manual/gh_api

gh api \
  -H "Accept: application/vnd.github+json" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  /enterprises/ENTERPRISE/copilot/usage
```

Example response

Response schema

Status: 200

```
[  
  {  
    "day": "2023-10-15",  
    "totalSuggestionsCount": 5000,  
    "totalAcceptancesCount": 3000,  
    "totalLinesSuggested": 7000,  
    "totalLinesAccepted": 3500,  
    "totalActiveUsers": 15,  
    "totalChatAcceptances": 45,  
    "totalChatTurns": 350,  
    "totalActiveChatUsers": 8,  
    "breakdown": [  
      {  
        "language": "python",  
        "count": 1000  
      },  
      {  
        "language": "JavaScript",  
        "count": 800  
      },  
      {  
        "language": "Python",  
        "count": 700  
      },  
      {  
        "language": "Java",  
        "count": 600  
      },  
      {  
        "language": "C/C++",  
        "count": 500  
      },  
      {  
        "language": "TypeScript",  
        "count": 400  
      },  
      {  
        "language": "Go",  
        "count": 300  
      },  
      {  
        "language": "C#",  
        "count": 200  
      },  
      {  
        "language": "Ruby",  
        "count": 150  
      },  
      {  
        "language": "PHP",  
        "count": 100  
      }  
    ]  
  }]
```

Copilot Enterprise

Utilise the full context
of an **enterprise's**
codebase.

Copilot Enterprise

Copilot personalised for your organisation throughout the software development lifecycle

- Everything in Copilot Business, plus...
- Chat personalised to your codebase
- Answers based on your organisation's knowledge base
- Pull request summaries
- Fine-tuned models (coming soon as an add-on)

**Creating pull requests that
tell a story...**

The screenshot shows a GitHub pull request interface with a dark theme. On the left, there's a sidebar with various icons for repository management. The main area displays a pull request titled "feature-book-search-pr". A message box at the top says "Enter to commit on 'feature-book-search-pr'". Below it is a "Commit" button. The "Changes" section lists four files with modifications:

- J BookDatabase.java** src/main/java/com/github/demo/service M
- J BookDatabaseImpl.java** src/main/java/com/github/demo/service M
- J BookService.java** src/main/java/com/github/demo/service M
- J BookServlet.java** src/main/java/com/github/demo/servlet M

The right side shows the code for **BookDatabaseImpl.java** with line numbers from 107+ to 141+. The code implements a method to search for books by title. It uses a PreparedStatement to execute a SELECT query and adds found books to a list.

```
107+     @Override
108+     public List<Book> getBooksByTitle(String name) throws BookServiceException {
109+         List<Book> books = new ArrayList<Book>();
110+
111+         if (!isValid()) {
112+             throw new BookServiceException(message:"Database connection is
113+                                         invalid");
114+         }
115+
116+         try {
117+             stmt = connection.createStatement();
118+             String query = "SELECT * FROM books WHERE title LIKE '%" + name +
119+                           "'";
120+
121+             ResultSet results = stmt.executeQuery(query);
122+
123+             while (results.next()) {
124+                 Book book = new Book(
125+                     results.getString(columnLabel:"author"),
126+                     results.getString(columnLabel:"title"),
127+                     results.getString(columnLabel:"image")
128+                 );
129+                 books.add(book);
130+             }
131+         } catch (SQLException e) {
132+             logger.error("Failed while searching for '{}', name");
133+             throw new BookServiceException(e);
134+         } finally {
135+             try {
136+                 if (stmt != null) {
137+                     stmt.close();
138+                 }
139+             } catch (SQLException se) {
140+                 // Do nothing
141+             }
142+             stmt = null;
143+         }
144+     }
145+ }
```

Below the code editor are tabs for OUTPUT, TERMINAL, DEBUG CONSOLE, and COMMENTS. The TERMINAL tab is selected, showing a terminal session in zsh with the command `main *1`. The bottom status bar indicates the line and column (Ln 115, Col 1), the number of selected spaces (Spaces: 4), the encoding (UTF-8), the file format (LF), the programming language (Java), and the current quote search results (CodeQuote - No Matches).

Helping you become a shell
superhero...

Cmder

C:\sources\AZ2007LabAppM3

λ copilot

Right=Insert Suggestion

GitHub Copilot Knowledge Bases

Create knowledge bases,
bringing together Markdown
documentation across one or
more repositories.

GitHub Copilot can then search
that knowledge base as context
and synthesise a response.

Customize / Add knowledge base

Knowledge bases are composed of Markdown files ('.md' and '.mdx') from a single repository or a group of repositories. By assembling knowledge bases focused on specific domains, you can enhance Copilot's ability to provide more precise and relevant responses.

Name *

e.g. Internal docs

Description

A brief description of this knowledge base's contents and purpose.

Content

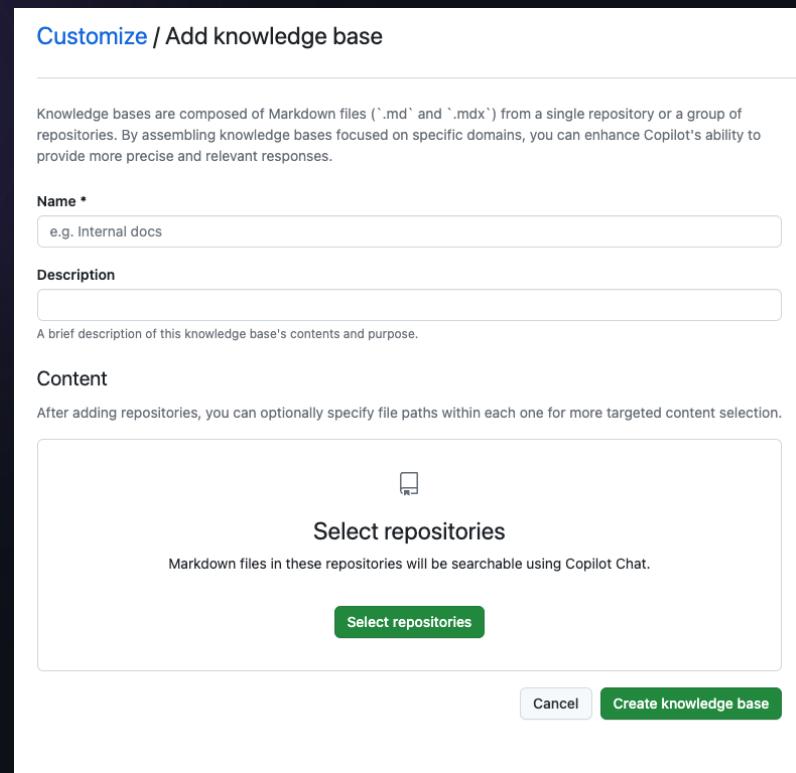
After adding repositories, you can optionally specify file paths within each one for more targeted content selection.

Select repositories

Markdown files in these repositories will be searchable using Copilot Chat.

Select repositories

Cancel Create knowledge base



GitHub Copilot Workspace

Technical Preview

Copilot Workspace
builds a full editable
plan in natural language
to help you get started

Run your code directly
in Copilot Workspace,
tweak all code changes
until you are happy with
the result

The screenshot shows the GitHub Copilot Workspace interface. On the left, there's a sidebar titled "octocademy / utils-library" with a "main" branch selected. It lists two files: "src/utils.py" and "README.md". Under "src/utils.py", there are several items in a checklist:

- Add a function to validate secure URLs
- Add a function to validate GitHub URLs
- + Add item

Under "README.md", there are also several items in a checklist:

- Update the README to include a summary of the repository
- Add a section summarizing the key validation methods with examples
- Include a section about the license linking to the LICENSE file
- Add a section explaining how to contribute
- + Add item

On the right, there's a "Files changed" section showing the diff for "src/utils.py". The diff highlights several regex definitions:1 import re
2
3 # Regular expression for validating email addresses
4 email_regex = r'^[\\w.\\+\\-]+@[a-zA-Z0-9\\-\\.][a-zA-Z0-9\\-\\.]*\$'
5
6 # Regular expression for validating phone numbers
7 phone_regex = r'^\\(\\+?\\d{1}\\)?\\d{9,15}\$'
8+
9+# Regular expression for validating secure URLs
10+secure_url_regex = r'^https:\\/'

Below the code editor is a "Terminal" window showing a command-line session on Ubuntu 20.04.6 LTS. The user runs "python -m unittest utils_test.py" and sees the output:Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 6.2.0-1019-azure x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
Last login: Thu Apr 11 18:42:15 2024 from ::1
@chrisreddingt0n ~ /workspaces/utils-library (chrisreddingt0n/add-url-validations) \$ cd src
@chrisreddingt0n ~ /workspaces/utils-library/src (chrisreddingt0n/add-url-validations) \$ python -m unittest utils_test.py
..

Ran 2 tests in 0.004s

OK
@chrisreddingt0n ~ /workspaces/utils-library/src (chrisreddingt0n/add-url-validations) \$

GitHub Copilot Workspace

Preview

The screenshot shows a GitHub repository page for 'shin-akuma / AVD-Session-Host-Auto-Shutdown'. The repository is public and contains 2 branches and 0 tags. It has 4 commits, with the latest being a merge pull request from 'shin-akuma/add-readme'. The README file has been updated, and a README.md file has been added. The repository has 0 stars, 1 watching, and 0 forks. There are no releases or packages published. The Languages section shows PowerShell at 100.0%. The repository description states it can be used to shut down Azure Virtual Desktop Session Hosts which have no active sessions.

shin-akuma / AVD-Session-Host-Auto-Shutdown

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

AVD-Session-Host-Auto-Shutdown Public

Pin Unwatch 1 Fork 0 Star 0

main 2 Branches 0 Tags Go to file Add file Code

shin-akuma Merge pull request #1 from shin-akuma/add-readme e479cb0 · 4 minutes ago 4 Commits

Update AVD-Session-Host-Auto-Shutdown.ps1 3 months ago

Add README file for AVD-Session-Host-Auto-Shutdown 4 minutes ago

README

AVD Session Host Auto Shutdown

Description

The script can be used to shut down Azure Virtual Desktop Session Hosts which have no active sessions.

Prerequisites

Required Powershell modules:

- 'Az.Accounts'
- 'Az.Compute'

github.com/shin-akuma/.../AVD-Session-Host-Auto-Shutdown.ps1

About

No description, website, or topics provided.

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Languages

PowerShell 100.0%

GitHub Copilot Coding Agent

Preview

The screenshot illustrates the GitHub Copilot Coding Agent interface across three main sections: a pull request view, an agent task interface, and a summary overview.

Pull Request View: Shows a pull request titled "Consolidate FabricPS-PBIP.ps1 and Sample.ps1 into unified sample script". The pull request is currently draft and wants to merge 3 commits from the "copilot/merge-fabric-pbip-sample" branch into the "main" branch. A Copilot comment suggests merging the files into a new "Sample.ps1" file for better coverage and code quality. The changes section details the consolidation of functionality from "FabricPS-PBIP.ps1" to "Sample.ps1" and the removal of "TMDL.ps1".

Agents Task View: Displays a task titled "Merge the Fabric-PBIP.ps1 file and Sample.ps1 file into a new Sample.ps1 file with better coverage and code quality". The task is assigned to the "shin-akuma/tmdl-ps" agent. The task status shows "You started a task" and "Copilot started work". The task details include steps like "Start 'playwright' MCP server" and "Start 'github-mcp-server' MCP server". It also provides links to "View FabricPS-PBIP.ps1", "View Sample.ps1", and "View TMDL.ps1".

Summary Overview: Provides a high-level summary of the task. It includes the task title "[WIP] Merge Fabric-PBIP.ps1 and Sample.ps1 for better coverage #1", completion status (Completed 9 minutes ago), and a summary message thanking the user for asking Copilot to work on the task. It also includes a link to "Copilot coding agent tips" for customization.

What type of tasks are for Copilot Coding Agent

- ✓ GitHub Copilot provides better results when assigned **clear, well-scoped tasks**. An ideal task includes:
- ✓ A **clear description** of the problem to be solved or the work required.
- ✓ Complete **acceptance criteria** on what a good solution looks like (for example, should there be unit tests?).
- ✓ Directions about which **files** need to be changed.

What type of tasks are not

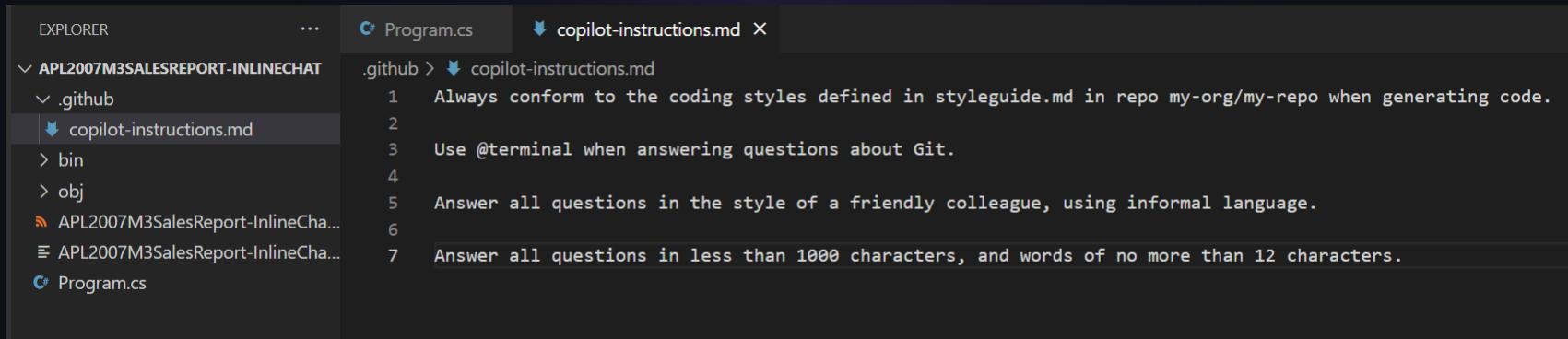
- ✗ Complex and broadly scoped tasks
- ✗ Sensitive and critical tasks
- ✗ Ambiguous tasks
- ✗ Learning tasks

Custom instructions for GitHub Copilot

Repository custom instructions allow you to specify repository-wide instructions and preferences, in a single file, that automatically apply to any conversation held in the context of the repository, instead of repeatedly adding this contextual detail to your chat questions

Custom instructions for GitHub Copilot

This example of a `.github/copilot-instructions.md` file contains four instructions that will be added to all chat questions.

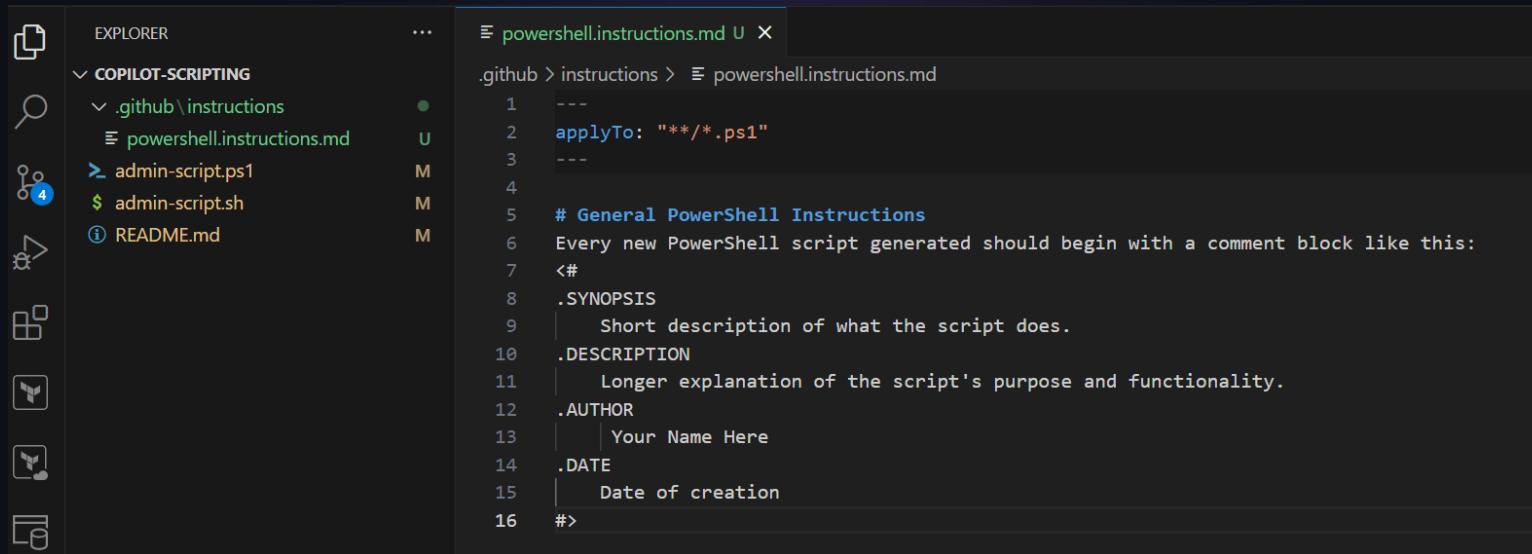


The screenshot shows a dark-themed code editor interface. At the top, there's a navigation bar with tabs for 'EXPLORER', '...', 'Program.cs', and 'copilot-instructions.md'. The 'copilot-instructions.md' tab is active. Below the navigation bar is a tree view of the repository structure under the '.github' folder. The 'copilot-instructions.md' file is selected. The main area displays the contents of the file:

```
.github > copilot-instructions.md
1 Always conform to the coding styles defined in styleguide.md in repo my-org/my-repo when generating code.
2
3 Use @terminal when answering questions about Git.
4
5 Answer all questions in the style of a friendly colleague, using informal language.
6
7 Answer all questions in less than 1000 characters, and words of no more than 12 characters.
```

Custom instructions for GitHub Copilot

This example of using .instructions.md files to store custom instructions for specific tasks.

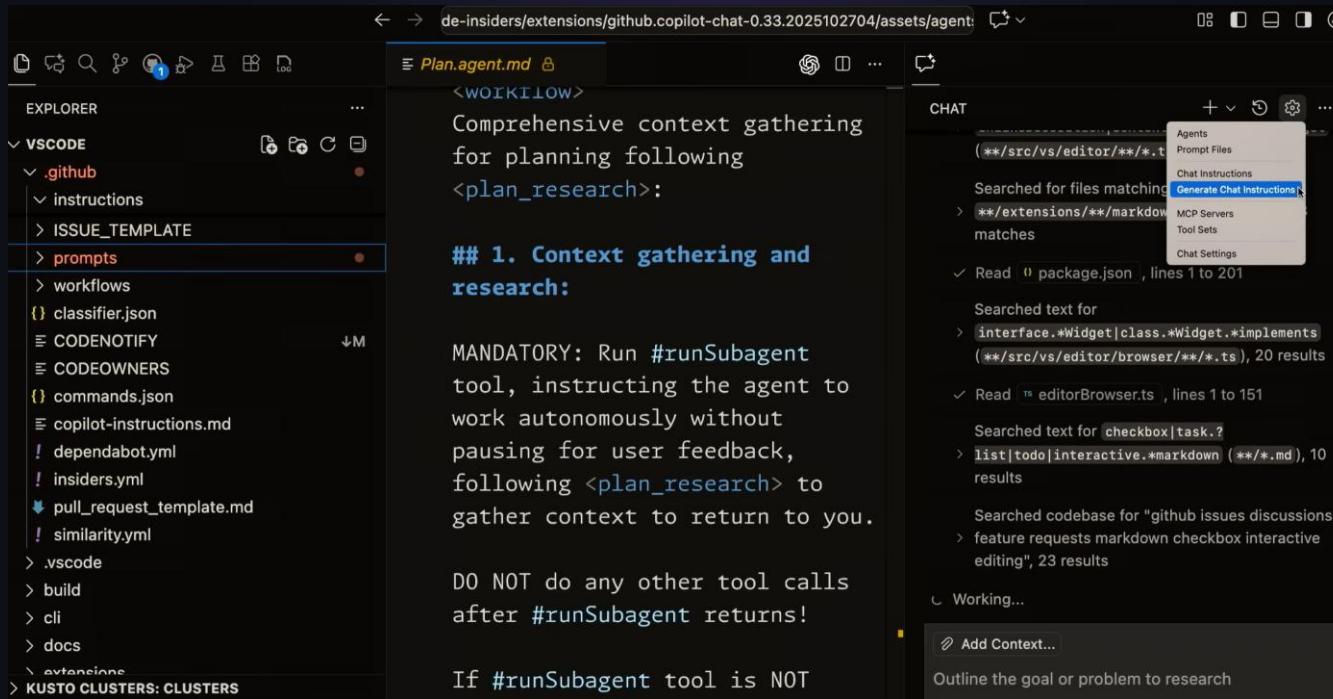


The screenshot shows the VS Code interface with the Explorer and Editor panes visible. The Explorer pane on the left displays a project structure under 'COPilot-SCRIPTING'. It includes a '.github\instructions' folder containing a 'powershell.instructions.md' file (marked as 'U' for updated), and other files like 'admin-script.ps1' and 'admin-script.sh' (both marked as 'M' for modified). The Editor pane on the right shows the contents of the 'powershell.instructions.md' file:

```
.github > instructions > powershell.instructions.md

1  ---
2  applyTo: "**/*.ps1"
3  ---
4
5  # General PowerShell Instructions
6  Every new PowerShell script generated should begin with a comment block like this:
7  <#
8  .SYNOPSIS
9      Short description of what the script does.
10 .DESCRIPTION
11     Longer explanation of the script's purpose and functionality.
12 .AUTHOR
13     Your Name Here
14 .DATE
15     Date of creation
16 #>
```

Custom instructions for GitHub Copilot



Custom instructions for GitHub Copilot

Workspace instructions:

Typically in a `.github/copilot-instructions.md` file

VS Code saves the `.md` file for **Workspace instructions** in the workspace folder located at `.github/instructions`.

User instructions:

On a Mac, user instructions are stored in this folder:`~/Library/Application`

`Support/Code/User/profiles/`

On Windows, you can find the user instructions here:`%APPDATA%\Code\User\profiles`

Prompt files (experimental)

The screenshot shows a dark-themed GitHub Copilot interface. At the top, there are tabs for 'instructions.md' (marked with a 'U'), 'react.prompts.md' (marked with a 'U'), 'MyForm.tsx', and 'myform.schema.ts'. The 'react.prompts.md' tab is active.

In the Explorer sidebar on the left, there's a tree view showing a folder structure under '.github': 'COPIL...', '.github', 'instructions', 'powershell.in...', 'prompts', and 'react.prompts...'. The 'react.prompts...' folder is selected.

The main area displays the content of 'react.prompts.md':

```
1 ---  
2 mode: 'agent'  
3 tools: ['githubRepo', 'codebase']  
4 description: 'Generate a new React form component'  
5 ---  
6 Your goal is to generate a new React form component based on the  
7 templates in #githubRepo contoso/react-templates.  
8 Ask for the form name and fields if not provided.  
9  
10 Requirements for the form:  
11 * Use form design system components: [design-system/Form.md](../docs/_  
design-system/Form.md)  
12 * Use `react-hook-form` for form state management:  
13 * Always define TypeScript types for your form data  
14 * Prefer *uncontrolled* components using register  
15 * Use `defaultValues` to prevent unnecessary renders  
16 * Use `yup` for validation:  
17 * Create reusable validation schemas in separate files  
18 * Use TypeScript types to ensure type safety  
19 * Customize UX-friendly validation rules
```

Below the code editor, the 'TERMINAL' tab is active, showing a PowerShell prompt: PS C:\sources\copilot-scripting>.

The 'CHAT' pane on the right shows a conversation between 'shin-akuma' and GitHub Copilot:

- shin-akuma: new react form
- GitHub Copilot: <react.prompts.md:1-19>
- GitHub Copilot: > Used 1 reference
- GitHub Copilot: To generate a new React form component, I'll need the form name and the fields you want in the form.
- shin-akuma: Could you please provide:
- GitHub Copilot: • The name of the form (e.g., "UserProfileForm")
• The fields you want (e.g., name, email, age, etc.), including their types and any validation requirements
- GitHub Copilot: Once you provide these details, I will generate the form component, TypeScript types, and validation schema according to your requirements.
- shin-akuma: <react.prompts.md:1-19>
- GitHub Copilot: > Used 1 reference
- GitHub Copilot: ✓ Searched for files matching **/design-system/Form.md, no matches

The bottom right corner shows status information: '1 file changed', 'myform.schema.ts', 'Keep', 'Undo', 'Add Context...', 'react.prompts.md Current file', 'Edit files in your workspace in agent mode', 'Agent', 'GPT-4.1', and a right-pointing arrow.

Prompt files (experimental)

The screenshot shows a dark-themed instance of Visual Studio Code with several open files and a prominent experimental feature overlay.

Explorer: Shows a tree view with a node for `react.prompts...` currently selected. Other items include `.github`, `instructions`, `powershell.in...`, `prompts`, `admin-script.ps1`, `admin-script.sh`, `myform.schema...`, `MyForm.tsx`, and `README.md`.

Editor: Displays the file `react.prompts.md` with the following content:

```
1 ---  
2 mode: 'agent'  
3 tools: ['githubRepo', 'codebase']  
4 description: 'Generate a new React form component'  
5 ---  
6 Your goal is to generate a new React form component based on the  
7 templates in #githubRepo contoso/react-templates.  
8 Ask for the form name and fields if not provided.  
9  
10 Requirements for the form:  
11 * Use form design system components: [design-system/Form.md](.../docs/_  
design-system/Form.md)  
12 * Use `react-hook-form` for form state management:  
13 * Always define TypeScript types for your form data  
14 * Prefer *uncontrolled* components using register  
15 * Use `defaultValue` to prevent unnecessary rerenders  
16 * Use `yup` for validation:  
17 * Create reusable validation schemas in separate files  
18 * Use TypeScript types to ensure type safety  
19 * Customize UX-friendly validation rules
```

Terminal: Shows a PowerShell terminal window with the command `PS C:\sources\copilot-scripting> []`.

GitHub Copilot Chat: An experimental feature overlay on the right side of the interface. It includes a message from `shin-akuma` asking for a new React form with name `myform` and fields `myfield`. Below this, GitHub Copilot provides a summary of its search and creation actions:

- Used 1 reference
- Searched for files matching `**/design-system/Form.md`, no matches
- Searched text for `yup` (`**/*.ts`), no results
- Searched text for `react-hook-form` (`**/*.tsx`), no results
- Created `myform.schema.ts`
- Read `myform.schema.ts`, lines 1 to 1

A command input field contains `npm install yup`, with a note below it: "Install the yup validation library required for the form schema." Buttons for `Continue` and `Cancel` are present.

Status Bar: Shows "1 file changed" and "myform.schema.ts". Buttons for "Keep" and "Undo" are available. A message at the bottom says "Edit files in your workspace in agent mode". The status bar also includes "Agent" and "GPT-4.1" dropdowns, and a "34" icon in the bottom right corner.

Context Engineering Primitives

Custom Instructions

*.instructions.md

Rules of engagement

Primary purpose

Always-on guidelines that influence all AI interactions

- 1. Coding standards
- 2. Language-specific rules
- 3. Documentation guidelines

Usage

Automatically applied to all requests

Reusable Prompts

*.prompts.md

Executable commands

Best for

Specific tasks you run when needed

- 1. Generate components
- 2. Increase test coverage
- 3. Atomic commit

Custom Agents

*.agents.md

Personas and workflows

Different working contexts with specialized tools and instructions

- 1. API Architect
- 2. SRE Debugger
- 3. Security Review

User switches agent (soon: auto-suggested)

Context Engineering Primitives

Custom Instructions

*.instructions.md

Rules of engagement

Reusable Prompts

*.prompts.md

Executable commands

Custom Agents

*.agents.md

Personas and workflows

Primary purpose

Always-on guidelines that influence all AI interactions

Specific tasks you run when needed

Different working contexts with specialized tools and instructions

Best for

1. Coding standards
2. Language-specific rules
3. Documentation guidelines

1. Generate components
2. Increase test coverage
3. Atomic commit

1. API Architect
2. SRE Debugger
3. Security Review

Usage

Automatically applied to all requests

User runs slash command (soon: auto-run)

User switches agent (soon: auto-suggested)

Get Started

START HERE!

1. Bootstrap minimal rules
2. Fix agent mistakes
3. Domain-specific instruction

AUTOMATE CHORES

4. Automate your own tasks
5. Share domain-specific repetitive tasks

SPECIALIZE WORKFLOWS

6. BYO-Planning workflows
7. Modular MCP-powered specialists

Improving code quality

Copilot Chat: Ask to refactor code

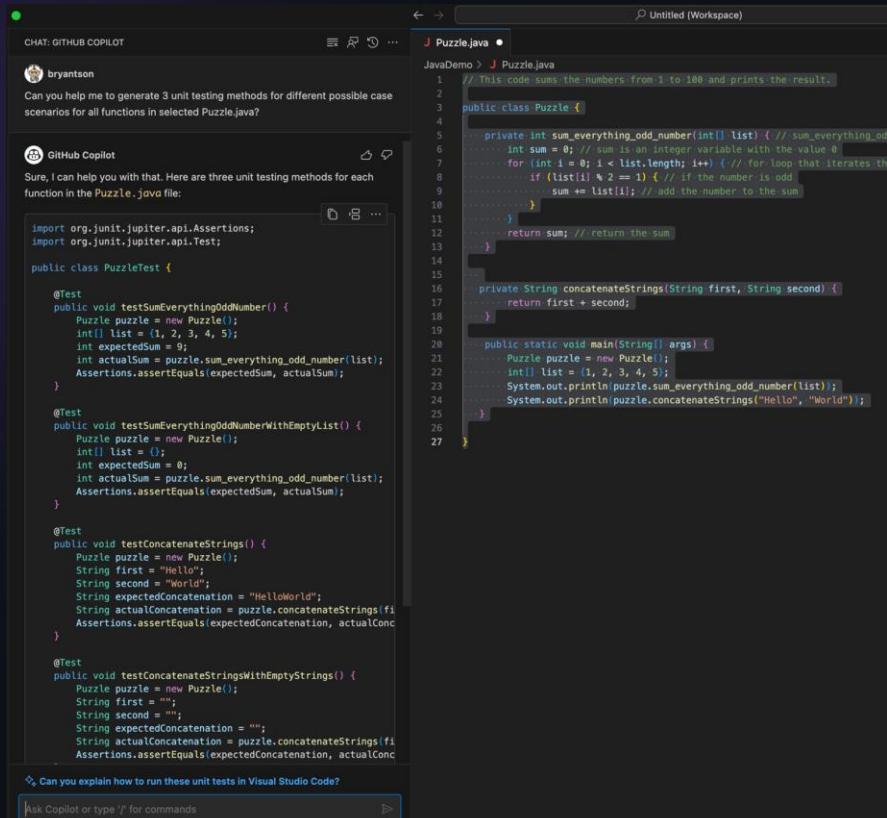
Ask GitHub
Copilot Chat to
refactor your
code

The screenshot shows a code editor window with a dark theme. On the left, there's a sidebar for 'CHAT: GITHUB COPilot' with a message from GitHub Copilot asking for feedback. The main area displays a Python script named 'sorting-algorithm.py'. The code contains three sorting functions: quickSort, bubbleSort, and insertionSort. The 'bubbleSort' function has several red error markers. Below the code, a message says 'Ask Copilot or type ⌘/ for commands'.

```
SimpleDemo > # sorting-algorithm.py 6
SimpleDemo > # sorting-algorithm.py > ...
1 def quickSort(array) {
2     if (array.length <= 1) {
3         return array;
4     }
5     var pivot = array[0];
6     var left = [];
7     var right = [];
8     for (var i = 1; i < array.length; i++) {
9         if (array[i] < pivot) {
10             left.push(array[i]);
11         } else {
12             right.push(array[i]);
13         }
14     }
15     return quickSort(left).concat(pivot, quickSort(right));
16 }
17
18 def bubbleSort(array) {
19     var len = array.length;
20     for (var i = 0; i < len; i++) {
21         for (var j = 0; j < len - i - 1; j++) {
22             if (array[j] > array[j + 1]) {
23                 var temp = array[j];
24                 array[j] = array[j + 1];
25                 array[j + 1] = temp;
26             }
27         }
28     }
29     return array;
30 }
31
32 def insertionSort(array) {
33     var len = array.length;
34     var preIndex, current;
35     for (var i = 1; i < len; i++) {
36         preIndex = i - 1;
37         current = array[i];
38         while (preIndex >= 0 && array[preIndex] > current) {
39             array[preIndex + 1] = array[preIndex];
40             preIndex--;
41         }
42         array[preIndex + 1] = current;
43     }
44     return array;
45 }
```

Unit Testing Generation - Copilot Chat

Works best for
general **unit**
testing case
generation



The screenshot shows a code editor interface with a dark theme. At the top, there's a header bar with icons for file operations and a tab labeled "Untitled (Workspace)". Below the header, a "CHAT: GITHUB COPILOT" section shows a message from "bryantson": "Can you help me to generate 3 unit testing methods for different possible case scenarios for all functions in selected Puzzle.java?". A response from "GitHub Copilot" follows: "Sure, I can help you with that. Here are three unit testing methods for each function in the Puzzle.java file:".

```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class PuzzleTest {

    @Test
    public void testSumEverythingOddNumber() {
        Puzzle puzzle = new Puzzle();
        int[] list = {1, 2, 3, 4, 5};
        int expectedSum = 9;
        int actualSum = puzzle.sum_everything_odd_number(list);
        Assertions.assertEquals(expectedSum, actualSum);
    }

    @Test
    public void testSumEverythingOddNumberWithEmptyList() {
        Puzzle puzzle = new Puzzle();
        int[] list = {};
        int expectedSum = 0;
        int actualSum = puzzle.sum_everything_odd_number(list);
        Assertions.assertEquals(expectedSum, actualSum);
    }

    @Test
    public void testConcatenateStrings() {
        Puzzle puzzle = new Puzzle();
        String first = "Hello";
        String second = "World";
        String expectedConcatenation = "HelloWorld";
        String actualConcatenation = puzzle.concatenateStrings(first, second);
        Assertions.assertEquals(expectedConcatenation, actualConcatenation);
    }

    @Test
    public void testConcatenateStringsWithEmptyStrings() {
        Puzzle puzzle = new Puzzle();
        String first = "";
        String second = "";
        String expectedConcatenation = "";
        String actualConcatenation = puzzle.concatenateStrings(first, second);
        Assertions.assertEquals(expectedConcatenation, actualConcatenation);
    }
}
```

At the bottom of the editor, a message box asks: "Can you explain how to run these unit tests in Visual Studio Code?". A footer bar at the very bottom contains the text "Ask Copilot or type '?' for commands".

Copilot Chat: Ask to generate tests

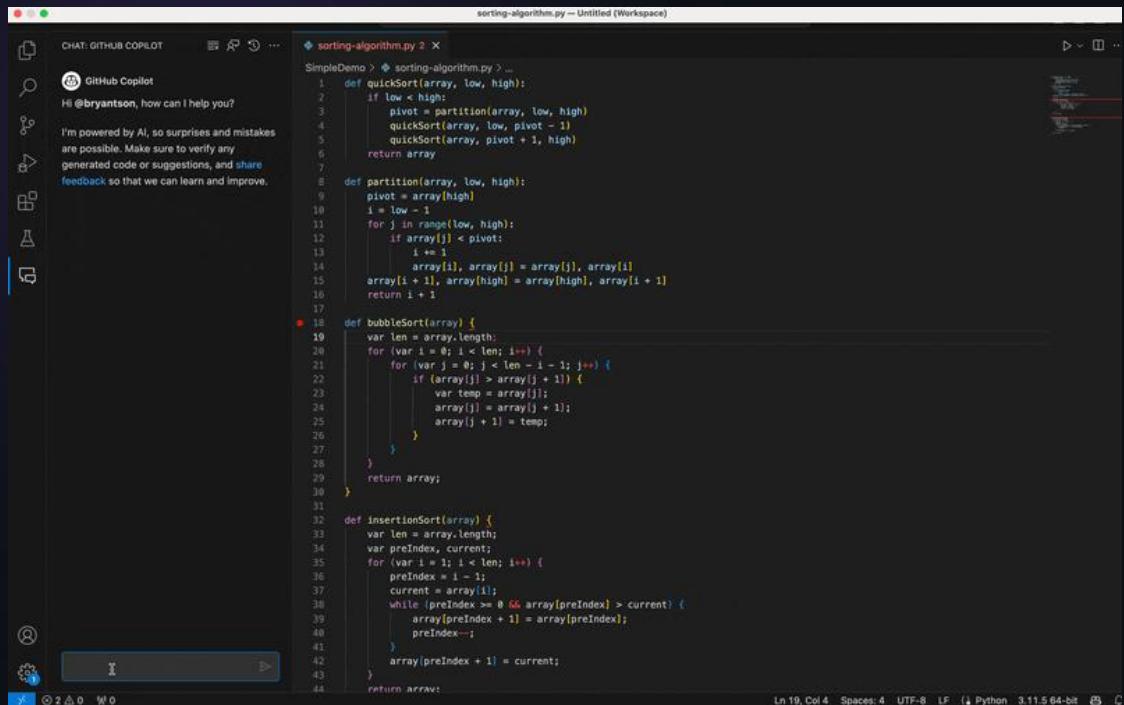
Ask GitHub
Copilot Chat to
generate tests

```
SimpleDemo > sorting-algorithm.py 2 ...
1 def quickSort(array, low, high):
2     if low < high:
3         pivot = partition(array, low, high)
4         quickSort(array, low, pivot - 1)
5         quickSort(array, pivot + 1, high)
6     return array
7
8 def partition(array, low, high):
9     pivot = array[high]
10    i = low - 1
11    for j in range(low, high):
12        if array[j] < pivot:
13            i += 1
14            array[i], array[j] = array[j], array[i]
15    array[i + 1], array[high] = array[high], array[i + 1]
16    return i + 1
17
18 def bubbleSort(array):
19     var len = array.length;
20     for (var i = 0; i < len; i++) {
21         for (var j = 0; j < len - i - 1; j++) {
22             if (array[j] > array[j + 1]) {
23                 var temp = array[j];
24                 array[j] = array[j + 1];
25                 array[j + 1] = temp;
26             }
27         }
28     }
29     return array;
30
31
32 def insertionSort(array) {
33     var len = array.length;
34     var preIndex, current;
35     for (var i = 1; i < len; i++) {
36         preIndex = i - 1;
37         current = array[i];
38         while (preIndex >= 0 && array[preIndex] > current) {
39             array[preIndex + 1] = array[preIndex];
40             preIndex--;
41         }
42         array[preIndex + 1] = current;
43     }
44     return array;
}

```

/fix, /tests and other / command options

Leverage / command options that can help to improve your code



The screenshot shows a code editor interface with a dark theme. On the left, there is a sidebar titled "CHAT: GITHUB COPILOT" which says "Hi @bryantson, how can I help you?". It also includes a note: "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 main area displays three Python scripts: "sorting-algorithm.py", "SimpleDemo.py", and "BubbleSort.py". The "sorting-algorithm.py" file contains implementations of quickSort, partition, bubbleSort, and insertionSort. The "SimpleDemo.py" file contains a quickSort call. The "BubbleSort.py" file contains a bubbleSort implementation. The status bar at the bottom right shows "Ln 19, Col 4" and "Python 3.11.5 64-bit".

```
sorting-algorithm.py 2: X
SimpleDemo > sorting-algorithm.py ...
1 def quickSort(array, low, high):
2     if low < high:
3         pivot = partition(array, low, high)
4         quickSort(array, low, pivot - 1)
5         quickSort(array, pivot + 1, high)
6     return array
7
8 def partition(array, low, high):
9     pivot = array[high]
10    i = low - 1
11    for j in range(low, high):
12        if array[j] < pivot:
13            i += 1
14            array[i], array[j] = array[j], array[i]
15    array[i + 1], array[high] = array[high], array[i + 1]
16    return i + 1
17
18 def bubbleSort(array):
19     var len = array.length;
20     for (var i = 0; i < len; i++) {
21         for (var j = 0; j < len - i - 1; j++) {
22             if (array[j] > array[j + 1]) {
23                 var temp = array[j];
24                 array[j] = array[j + 1];
25                 array[j + 1] = temp;
26             }
27         }
28     }
29     return array;
30
31 def insertionSort(array) {
32     var len = array.length;
33     var preIndex, current;
34     for (var i = 1; i < len; i++) {
35         preIndex = i - 1;
36         current = array[i];
37         while (preIndex >= 0 && array[preIndex] > current) {
38             array[preIndex + 1] = array[preIndex];
39             preIndex--;
40         }
41         array[preIndex + 1] = current;
42     }
43     return array;
44 }
```



Demo

Common Problems

Getting accuracy closer to expectation

Problems

- ⚠ Copilot fails to produce answer or to keep repeating
- ⚠ Copilot generates incorrect result
- ⚠ Library/module version discrepancies issue
- ⚠ Copilot suggests non-optimal solution

Problem #1:

Copilot fails to produce answer or keep repeating

Some problems

- Fails to produce answer
- Hallucination - Keeps repeating

The screenshot shows a dark-themed code editor interface. On the left, the Explorer sidebar displays a workspace named 'SimpleDemo' containing a file named 'calculator.py'. The main editor area shows the following code:

```
1 | Press ⌘ I to ask GitHub Copilot Chat to do something. Start typing to dismiss.
```

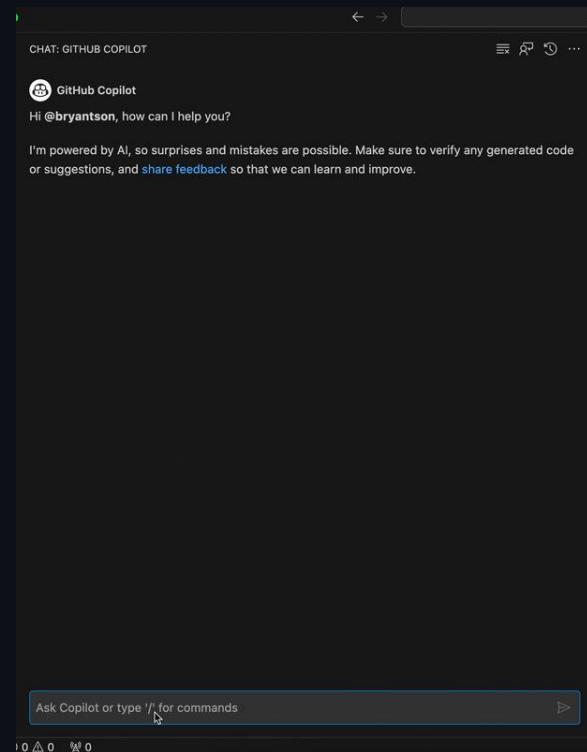
The status bar at the bottom right indicates the current line and column as 'Ln 1, Col 1', and provides other file metadata: 'Spaces: 4', 'UTF-8', 'LF', 'Python 3.11.5 64-bit', and a refresh icon.

Problem #2:

Copilot generates incorrect result

Reasons why...

- Not enough context
- Old trained data
- Generative in nature

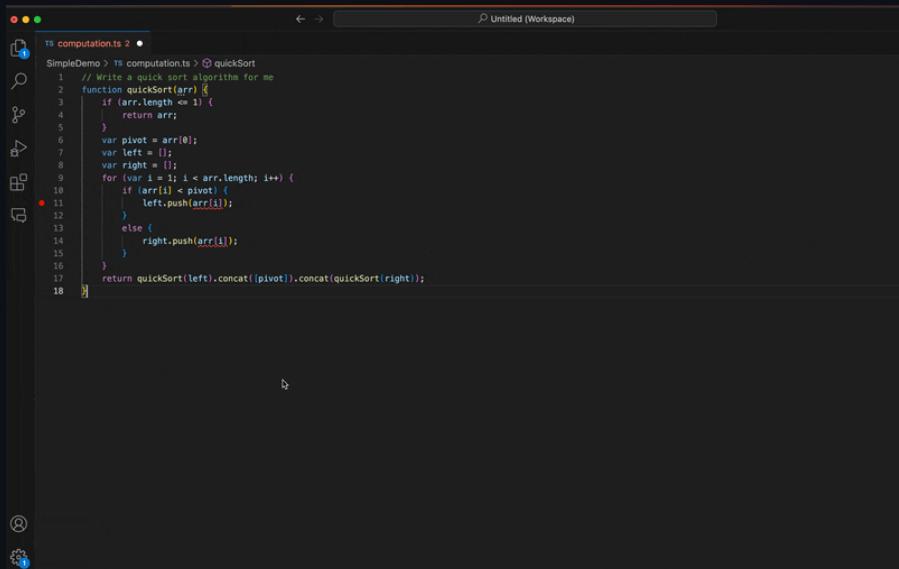


Problem #3:

Copilot suggests non-optimal solution

How to improve

- Ask in Copilot Chat
- Use in-line suggestion from Copilot



A screenshot of a code editor window titled "TS computation.ts". The code is a quicksort algorithm. A red dot on the status bar indicates a warning or error. The code is as follows:

```
SimpleDemo > ts computation.ts ⚡ quickSort
1 // Write a quick sort algorithm for me
2 function quickSort(arr) {
3     if (arr.length <= 1) {
4         return arr;
5     }
6     var pivot = arr[0];
7     var left = [];
8     var right = [];
9     for (var i = 1; i < arr.length; i++) {
10        if (arr[i] < pivot) {
11            left.push(arr[i]);
12        }
13        else {
14            right.push(arr[i]);
15        }
16    }
17    return quickSort(left).concat([pivot]).concat(quickSort(right));
18 }
```

Develop new code features using GitHub Copilot

- [Develop new code features using GitHub Copilot | mslearn-github-copilot-dev](#)
- <https://github.com/MicrosoftLearning/mslearn-github-copilot-dev/raw/refs/heads/main/DownloadableCodeProjects/Downloads/AZ2007LabAppM3.zip>

Create unit tests by using GitHub Copilot Chat

- [Develop unit tests using GitHub Copilot | mslearn-github-copilot-dev](#)
- [https://github.com/MicrosoftLearning/mslearn-github-copilot-dev/raw/refs/heads/main/DownloadableCodeProjects/Downloads/AZ2007LabAppM4.zip](#)

Challenge 1

- C#
- <https://learn.microsoft.com/en-gb/training/modules/generate-documentation-using-github-copilot-tools/6-exercise-complete-code-documentation-challenge>
- Python
- <https://experience.cloudlabs.ai/#/labguidepreview/f9fd80ac-fc1a-4609-a3f1-06650aec389e> (Page 9)

Challenge 2

- C#
- <https://experience.cloudlabs.ai/#/labguidepreview/f9fd80ac-fc1a-4609-a3f1-06650aec389e> (Page 6)
- Python
- <https://experience.cloudlabs.ai/#/labguidepreview/f9fd80ac-fc1a-4609-a3f1-06650aec389e> (Page 5)

Q & A

GitHub Copilot for Developers Survey



<https://forms.office.com/r/jAjHSesAVa>

Upcoming Sessions

- 1 GitHub Copilot for DevOps
- 2 GitHub Copilot for ML and Data Ops
- 3 Q&A Session



Thank you