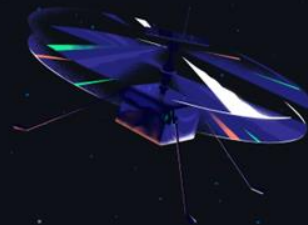




# GitHub Copilot Developer Training

Level: Intermediate

Let's build from here



## AGENDA

**GitHub Copilot - Introduction**

**Coding**

**Best practices & Prompt Engineering**

**Workshop (1 - 2 hours long)**

**Secure coding**

**Wrap-up, Q&A**



# Outcome of this training

You will achieve...

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



# Covered in Copilot Fundamental

We will not talk about...

- ✗ Statistics around Copilot usage and satisfaction
- ✗ Successful customer case study
- ✗ Enterprise & Organization administrator interface



# GitHub Copilot

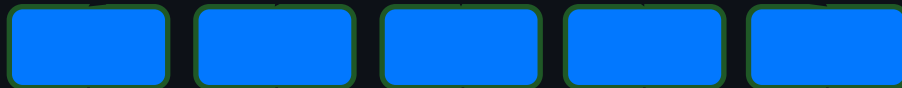
- An intelligent pair programmer
- Draws context from comments & code to suggest individual lines and whole functions
- Powered by OpenAI Codex
  - Copilot uses a transformative model
  - Trained on large datasets to ensure accuracy
- Available as extensions to popular IDEs
- Programming Languages and Technology available in Public code base all are supported

```
TS sentiment.ts  -GO write_sql.go  parse_expense

1  #!/usr/bin/env ts-node
2
3  import { fetch } from "fetch-h2";
4
5  // Determine whether the sentiment of
6  // Use a web service
7  async function isPositive(text: string) {
8      const response = await fetch(`http://
9          method: "POST",
10         body: `text=${text}`,
11         headers: {
12             "Content-Type": "application/x-www
13         },
14     });
15     const json = await response.json();
16     return json.label === "pos";
17 }
```

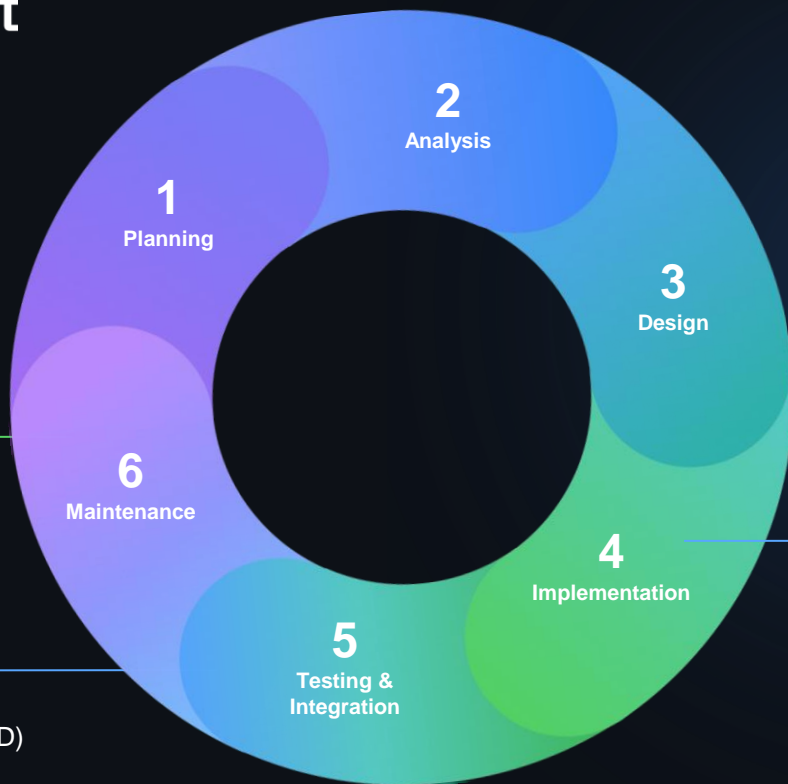


# Design



## IMPLEMENTATION





Refactoring code (code translate)  
 Reviewing code (code explain)  
 Documentation

Unit testing (TDD and BDD)  
 Finding code errors  
 Debugging  
 Code review  
 AI Pull Requests

Convert comments to code  
 Autofill for repetitive code  
 Show alternatives

# GitHub Copilot

Once enabled...



OpenAI  
Codex



Context

Suggestions



Visual Studio



Neovim



VS Code



JetBrains IDEs

runtime.go course.rb time.js IsPrimeTest.java

```
1 package main
2
3 type Run struct {
4     Time int // in milliseconds
5     Results string
6     Failed bool
7 }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

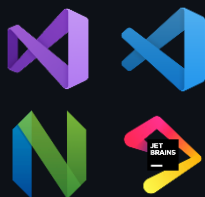
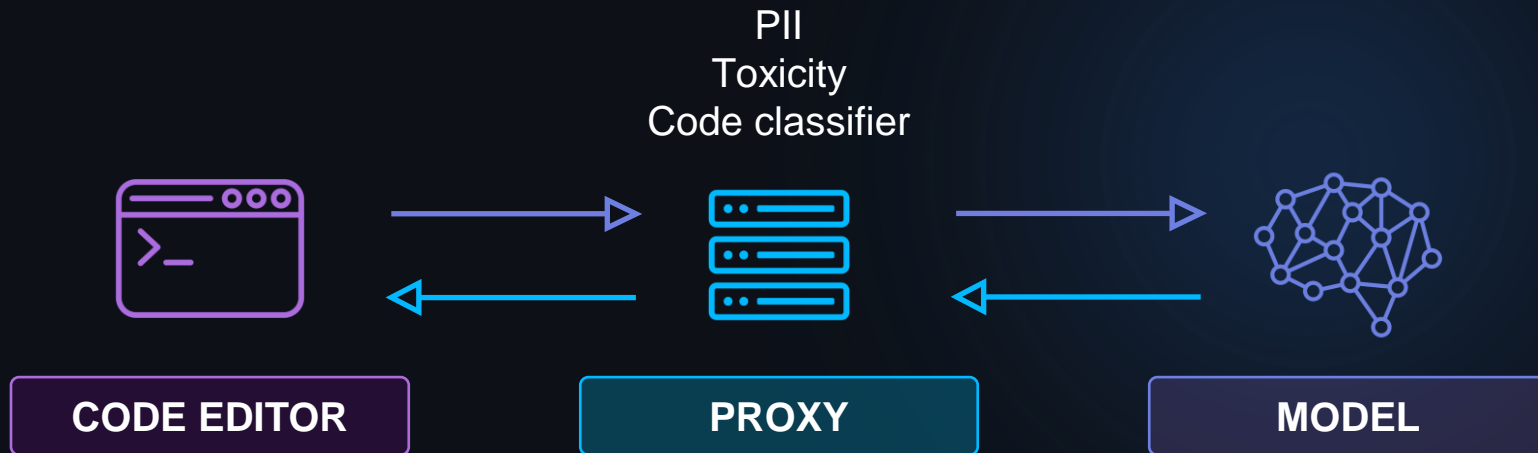


# GitHub Copilot works something like this

- 1) Code starts in the user's editor. The Copilot extension gathers context from open files. That context is called the "prompt."
- 2) The prompt is sent with SSL encryption to a proxy server living in a GitHub-owned Azure subscription.
- 3) At the proxy server, we apply filters to scrub PII, tokens, vulnerabilities etc.
- 4) The proxy server sends the scrubbed prompt to the model.
- 5) The model produces 'n' suggestions based on the prompt, then deletes the prompt from memory.
- 6) The suggestions produced by the model are retained for a period of up to 30 days
- 7) The responses are sent back to the proxy server for another scrubbing.
- 8) The proxy server sends the suggestions back to the editor via HTTPS where they are displayed.



# Data flow through the Copilot ecosystem



PII  
Toxicity  
Code classifier  
Code quality  
Duplicate detection



# Coding

**GH Copilot vs GH Copilot Chat**

**Slash commands**

**Data Generation & Testing**

**Code Refining & Refactoring**

**Block Files & Debugging**

**GH Rest APIs**

# When to use Copilot vs Copilot Chat

## Copilot

Direct Code Writing

Seamless IDE Integration

Solo Development

## Copilot Chat

In-Depth Assistance

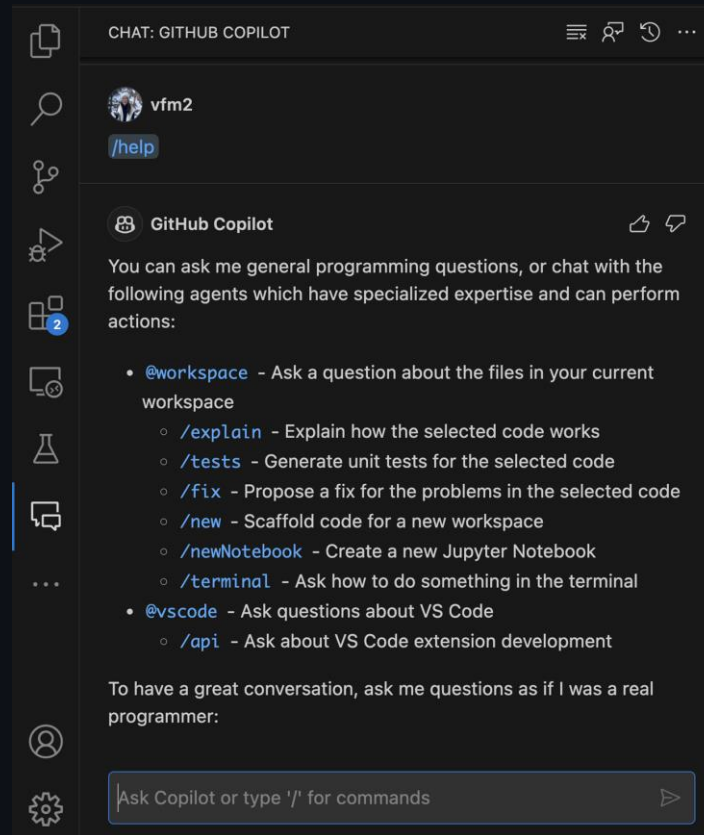
Learning & Teaching

Collaborative Scenarios



# Slash Commands


`/help` to find  
available  
commands in your  
IDE




# 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
9. Q: The software was deployed successfully. A: Positive
10. Q: The software was deployed successfully. A: Positive

🔗 How can I integrate sentiment analysis into my application?



# Unit Testing Generation

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

CHAT: GITHUB COPILOT

bryantson

Can you help me to generate 3 unit testing methods for different possible case scenarios for all functions in selected Puzzle.java?

GitHub Copilot

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.concatenateStringsIfI
        Assertions.assertEquals(expectedConcatenation, actualConc

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

JavaDemo > J Puzzle.java

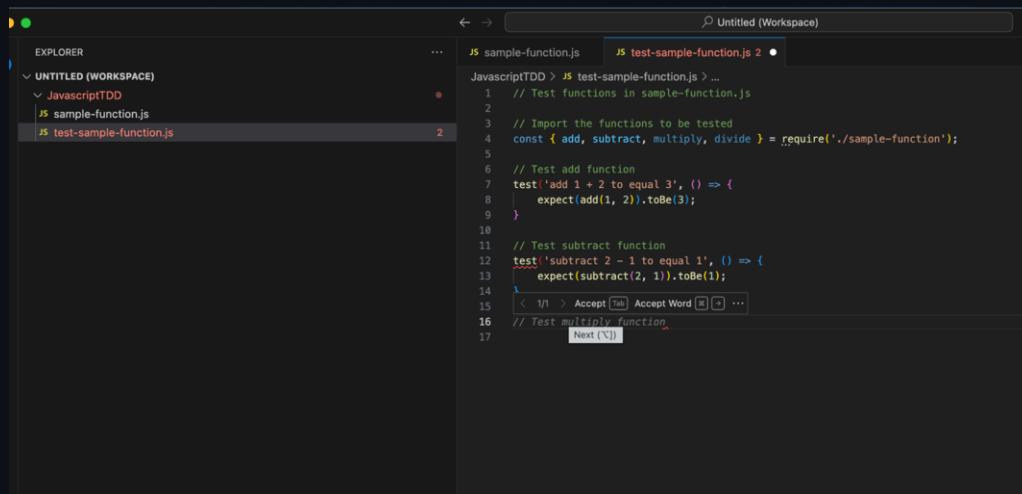
```
1 // This code sums the numbers from 1 to 100 and prints the result.
2
3 public class Puzzle {
4
5     private int sum_everything_odd_number(int[] list) { // sum_everything_od
6         int sum = 0; // sum is an integer variable with the value 0
7         for (int i = 0; i < list.length; i++) { // for loop that iterates th
8             if (list[i] % 2 == 1) { // if the number is odd
9                 sum += list[i]; // add the number to the sum
10            }
11        }
12        return sum; // return the sum
13    }
14
15     private String concatenateStrings(String first, String second) {
16         return first + second;
17     }
18
19     public static void main(String[] args) {
20         Puzzle puzzle = new Puzzle();
21         int[] list = {1, 2, 3, 4, 5};
22         System.out.println(puzzle.sum_everything_odd_number(list));
23         System.out.println(puzzle.concatenateStrings("Hello", "World"));
24     }
25 }
26
27 }
```

Can you explain how to run these unit tests in Visual Studio Code?

Ask Copilot or type "?" for commands

# Test Driven Development (TDD)

Test Driven Development (TDD) practice is possible



The screenshot shows a code editor with a file named `test-sample-function.js` open. The file contains JavaScript code for testing functions. The Explorer panel on the left shows the file structure under 'UNTITLED (WORKSPACE)'.

```

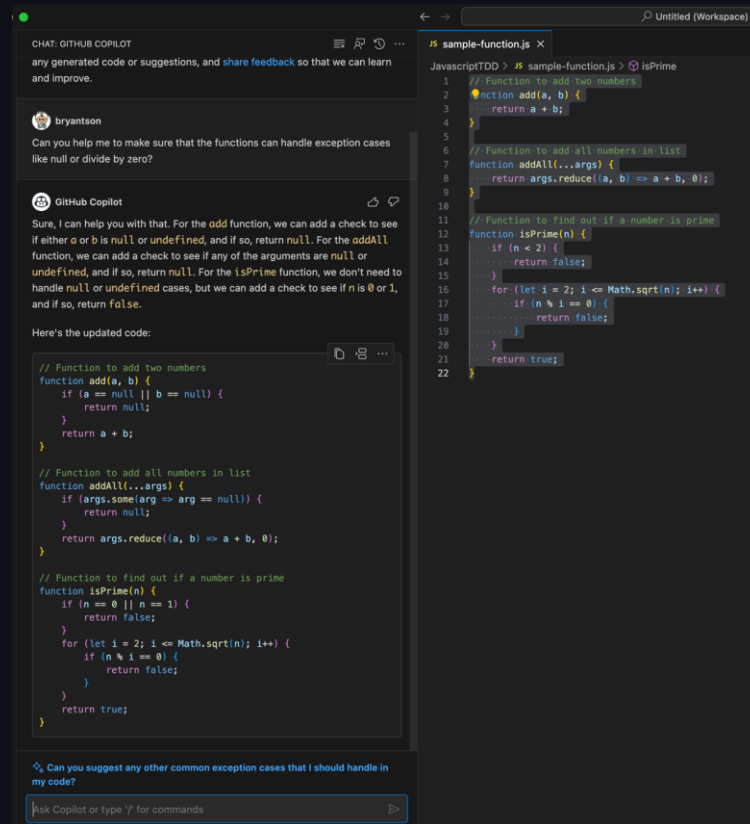
1 // Test functions in sample-function.js
2
3 // Import the functions to be tested
4 const { add, subtract, multiply, divide } = require('./sample-function');
5
6 // Test add function
7 test('add 1 + 2 to equal 3', () => {
8   expect(add(1, 2)).toBe(3);
9 })
10
11 // Test subtract function
12 test('subtract 2 - 1 to equal 1', () => {
13   expect(subtract(2, 1)).toBe(1);
14 })
15
16 // Test multiply function
17

```



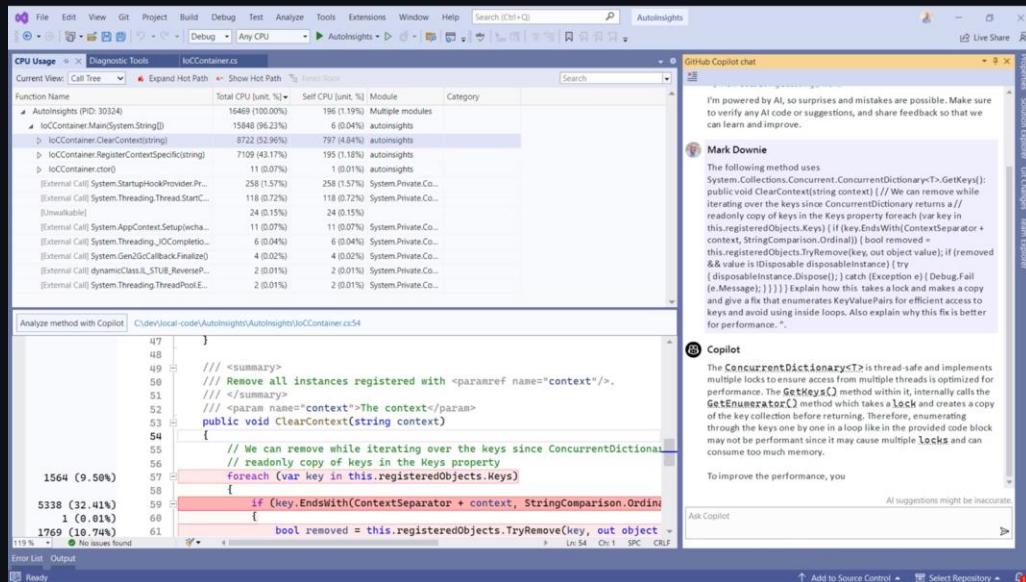
# Code Refactoring

Refactoring is possible through  
**GitHub Copilot Chat**



# Code Refining

## GH Copilot CPU usage tool



The screenshot displays the Visual Studio IDE with the CPU Usage tool open. The tool shows a list of functions and their CPU usage percentages. The top function is 'Autoinsights (PID: 30324)' with a total CPU usage of 100.00% and a self CPU usage of 1.19%.

Function Name	Total CPU (unit, %)	Self CPU (unit, %)	Module	Category
Autoinsights (PID: 30324)	16469 (100.00%)	196 (1.19%)	Multiple modules	
IoContainer.MainSystem.String()	15848 (96.23%)	6 (0.04%)	autoinsights	
IoContainer.ClearContext(string)	8722 (52.96%)	797 (4.84%)	autoinsights	
IoContainer.RegisterContextSpec(string)	7109 (42.17%)	195 (1.18%)	autoinsights	
IoContainer.Clear()	11 (0.07%)	1 (0.01%)	autoinsights	
[Internal Call] System.StartupHookProvider.Pr...	258 (1.57%)	258 (1.57%)	System.Private.Co...	
[Internal Call] System.Threading.Thread.Start...	118 (0.72%)	118 (0.72%)	System.Private.Co...	
[Internal Call] System.AppContext.Setup(vecha...	24 (0.15%)	24 (0.15%)	System.Private.Co...	
[Internal Call] System.Threading.IOCompleto...	11 (0.07%)	11 (0.07%)	System.Private.Co...	
[Internal Call] System.Threading.IOCompleto...	6 (0.04%)	6 (0.04%)	System.Private.Co...	
[Internal Call] System.GenGcCallback.Finalize...	4 (0.02%)	4 (0.02%)	System.Private.Co...	
[Internal Call] dynamicClass_IL_STUB.ReverseP...	2 (0.01%)	2 (0.01%)	System.Private.Co...	
[Internal Call] System.Threading.ThreadPool.E...	2 (0.01%)	2 (0.01%)	System.Private.Co...	

The GitHub Copilot chat window on the right provides context and suggestions for the code. It mentions that the code is powered by AI and suggests verifying AI code or suggestions. It also provides a 'Mark Down' button and a 'Copilot' button.

The code being analyzed is in the file 'C:\dev\local-code\Autoinsights\IoContainer.cs:54'. The code snippet shows a method 'ClearContext' that removes all instances registered with a context. The code is as follows:

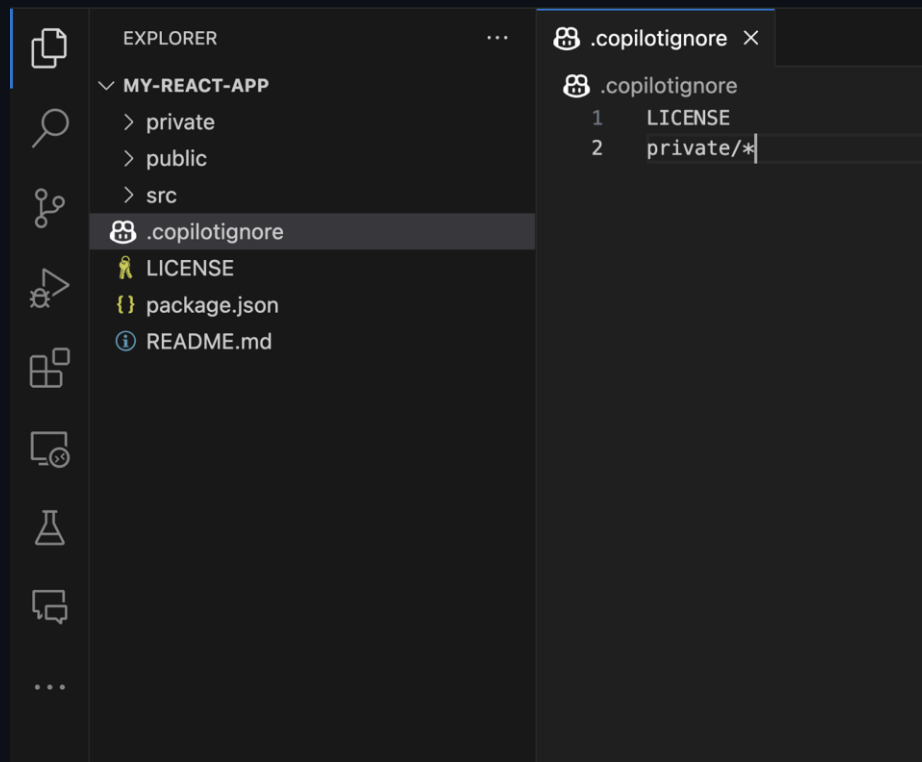
```

47     }
48
49     /// <summary>
50     /// Remove all instances registered with <paramref name="context"/>.
51     /// </summary>
52     /// <param name="context">The context</param>
53     public void ClearContext(string context)
54     {
55         // We can remove while iterating over the keys since ConcurrentDictionary
56         // readonly copy of keys in the Keys property
57         foreach (var key in this.registeredObjects.Keys)
58         {
59             if (key.EndsWith(ContextSeparator + context, StringComparison.Ordinal)
60             {
61                 bool removed = this.registeredObjects.TryRemove(key, out object

```

# Block files from Copilot

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



# Code Debugging

## Debugging with GitHub Copilot

```
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':

/fix Expected ":"
Accept Discard | v ↺
Changed 1 line

47         sprite.kill()
48         self.active = False
49         self.plane.kill()
50
```



# GitHub Rest API

## GitHub Copilot Rest API

GET /orgs/{org}/copilot/billing

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" \
  /orgs/ORG/copilot/billing
```

Example response

Response schema

Status: 200

```
{
  "seat_breakdown": {
    "total": 12,
    "added_this_cycle": 9,
    "pending_invitation": 0,
    "pending_cancellation": 0,
    "active_this_cycle": 12,
    "inactive_this_cycle": 11
  },
  "seat_management_setting": "assign_selected",
  "public_code_suggestions": "block"
}
```





# Demo: Coding practices



# Best practices

Getting accuracy closer to the expectation

- Prompt Engineering





Working at a project(s) level

Leveraging Copilot to increase code quality

Addressing concerns around security exploit

# 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

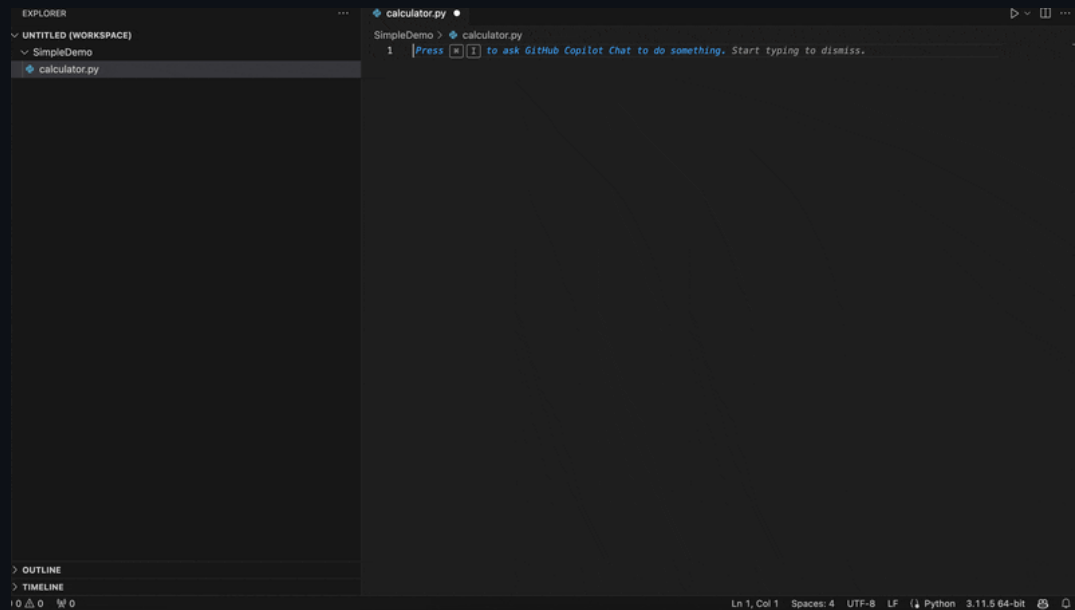


# Problems #1:

Copilot fails to produce answers or will keep repeating

## Some problems

- Fails to produce answer
- Hallucination - Keeps repeating

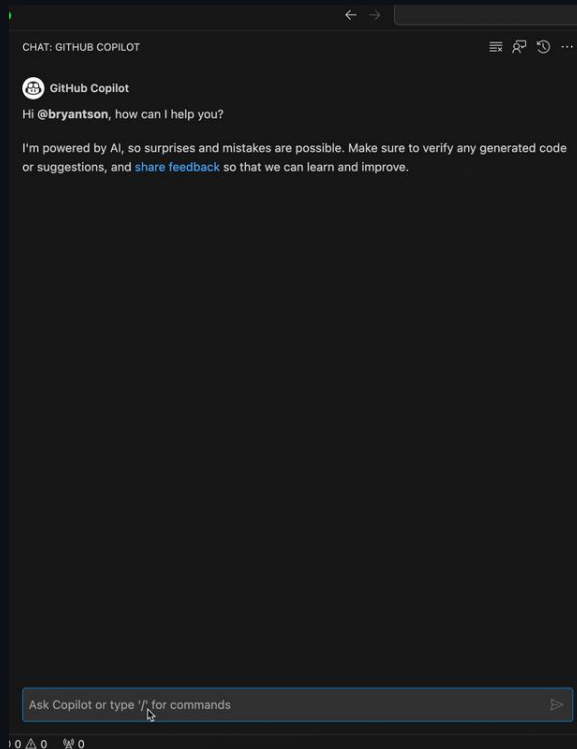


# Problems #2:

## Copilot generates incorrect result

## Reasons why...

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



# What is a Prompt?



*In the context of Copilot, a prompt is a piece of code or natural language description that is used to generate code suggestions. It is the input that Copilot uses to generate its output.*

- Copilot

# What is Prompt Engineering?



*Prompt engineering is the process of designing and creating high-quality prompts that can be used to generate accurate and useful code suggestions with Copilot.*

- Copilot

# Why is Prompt Engineering Important?



- Maximizes utility of Copilot
- Minimizes repeated iterations
- Determines code suggestion quality
- Skill prompts guide Copilot to understand context & nuances
- Refined interactions reduces misunderstandings

# Techniques



**Zero-Shot  
Prompting**



**One-Shot  
Prompting**




**Few-Shot  
Prompting**

# Techniques

## Zero-Shot Programming with Copilot

 abelberhane

Write me unit tests for all of the operations in my calculator.js file.

 GitHub Copilot

Sure! Here's an example of a test suite that covers all of the operations in a `calculator.js` file:

```
const { calculate } = require('./calculator');

describe('calculator', () => {
  describe('addition', () => {
    it('adds 1 + 2 to equal 3', () => {
      expect(calculate(1, 2, '+')).toBe(3);
    });

    it('adds -5 + 7 to equal 2', () => {
      expect(calculate(-5, 7, '+')).toBe(2);
    });

    it('adds 0.1 + 0.2 to equal 0.3', () => {
      expect(calculate(0.1, 0.2, '+')).toBeCloseTo(0.3);
    });
  });

  describe('subtraction', () => {
    it('subtracts 5 - 3 to equal 2', () => {
      expect(calculate(5, 3, '-')).toBe(2);
    });
  });
});
```



# Techniques

## One or few shot Programming with Copilot



abelberhane

Write me unit tests for all of the operations in my calculator.js file.

Use the following example: `test('adds 1 + 2 to equal 3', () => { expect(calculate(1, 2, '+')).toBe(3); });`



GitHub Copilot

Here are some examples of unit tests for all of the operations in a `calculator.js` file:

```
const { calculate } = require('./calculator');

test('adds 1 + 2 to equal 3', () => {
  const result = calculate(1, 2, '+');
  expect(result).toBe(3);
});

test('subtracts 5 - 3 to equal 2', () => {
  const result = calculate(5, 3, '-');
  expect(result).toBe(2);
});

test('multiplies 2 * 4 to equal 8', () => {
  const result = calculate(2, 4, '*');
  expect(result).toBe(8);
});

test('divides 10 / 5 to equal 2', () => {
  const result = calculate(10, 5, '/');
  expect(result).toBe(2);
});

test('throws an error for an invalid operation', () => {
  expect(() => calculate(1, 2, '%')).toThrow('Invalid operation');
});

test('throws an error for division by zero', () => {
  expect(() => calculate(1, 0, '/')).toThrow('Division by zero');
});
```







# Demo: Prompt Engineering

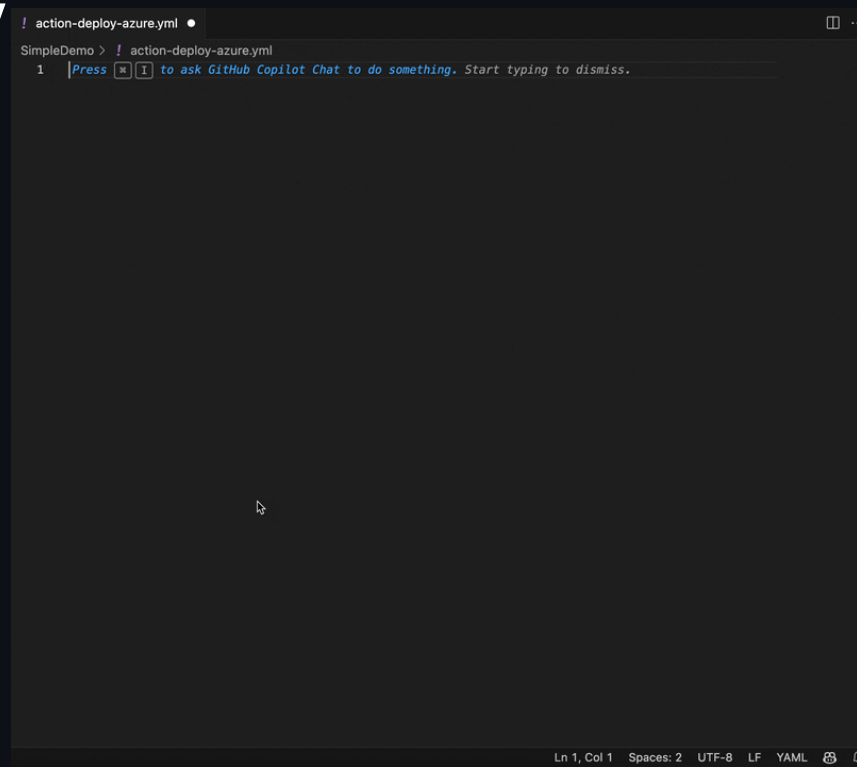


# Problems #3:

## Library/module version discrepancy

### Old trained data

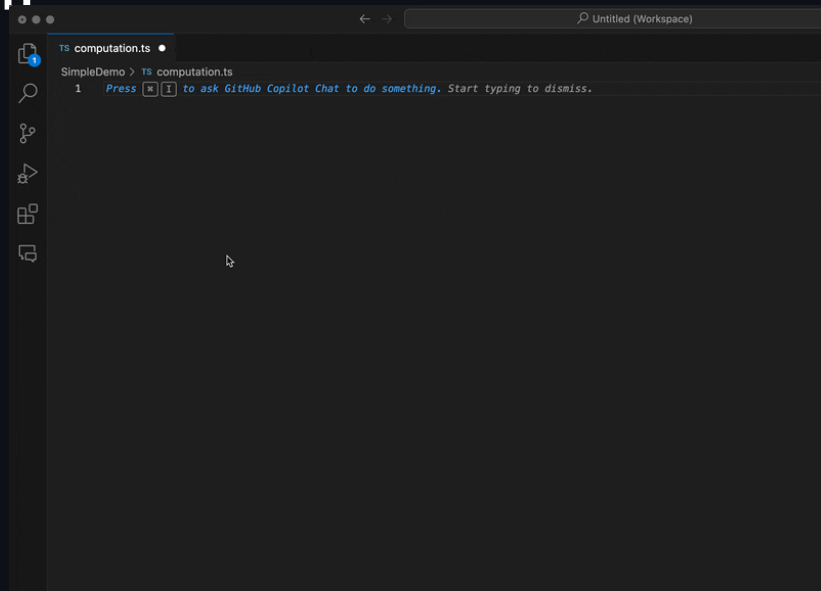
- While packages go through frequent updates, Copilot does not use latest data



# Problems #4:

Copilot suggests non-optimal solution

Although solution works...  
Suggested solution is not  
optimal because Quick  
Sort can be implemented  
in  $O(1)$  complexity,  
meaning no space  
required

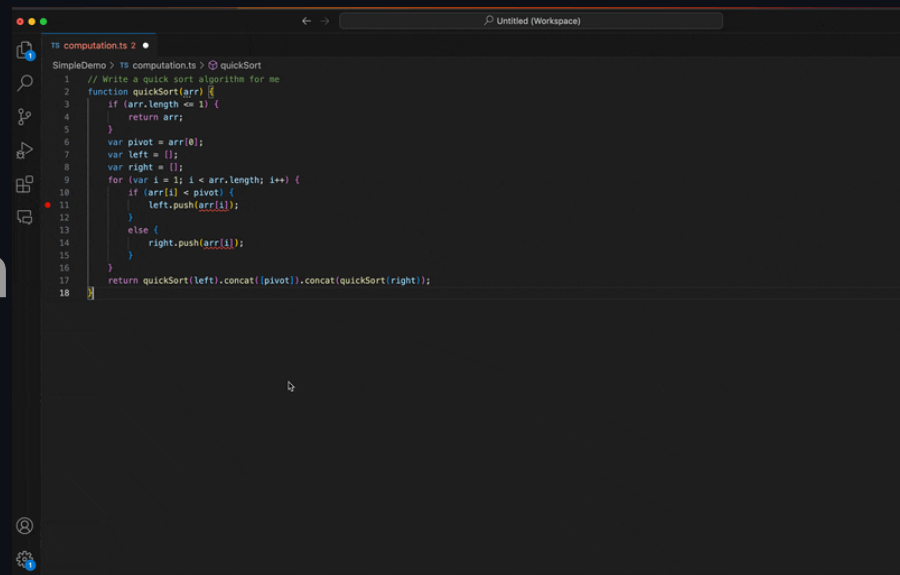


# How to improve Problem #4

## Copilot suggests non-optimal solution

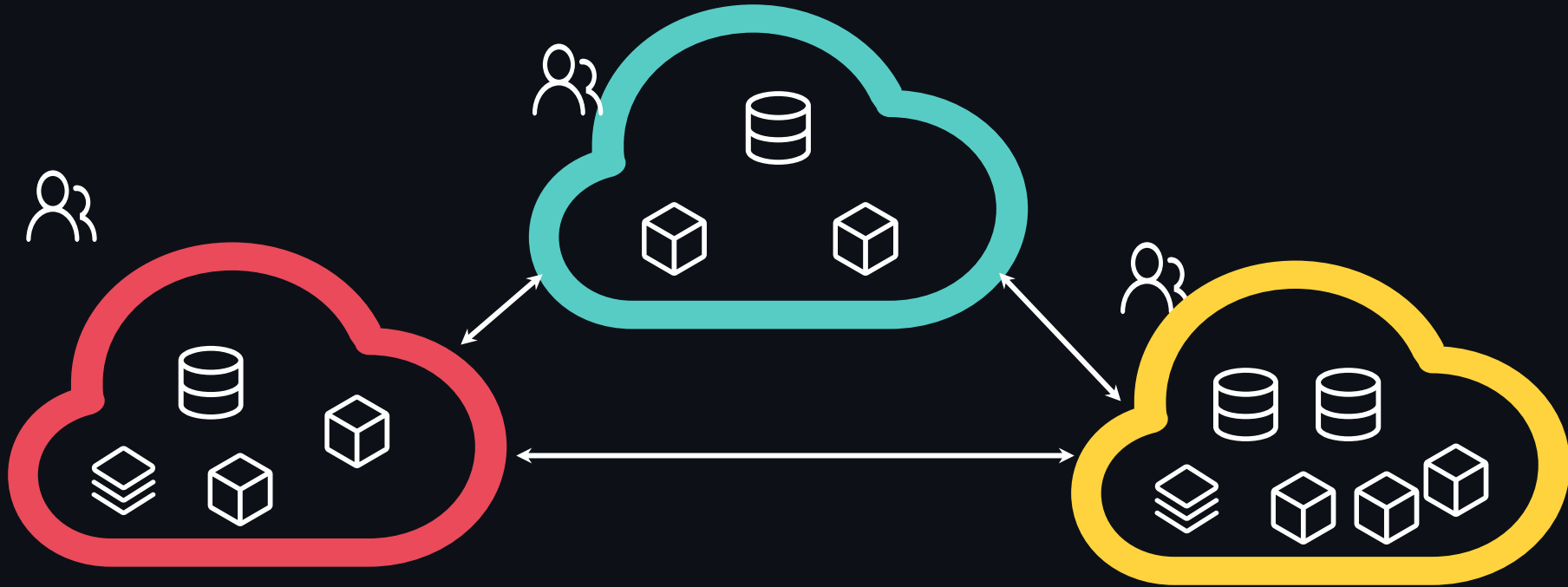
### How to improve

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



```
TS computation.ts 2
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 }
```

# How to work at project(s) level



# Neighboring Tabs

TS sentiments.ts

write\_sql.go

parse\_expenses.py

addresses.rb

```

1 #!/usr/bin/env ts-node
2
3 import { fetch } from "fetch-h2";
4
5 // Determine whether the sentiment of text is positive
6 // Use a web service
7 async function isPositive(text: string): Promise<boolean> {
8     const response = await fetch(`http://text-processing.com/api/sentiment/`, {
9         method: "POST",
10        body: `text=${text}`,
11        headers: {
12            "Content-Type": "application/x-www-form-urlencoded",
13        },
14    });
15    const json = await response.json();
16    return json.label === "pos";
17 }

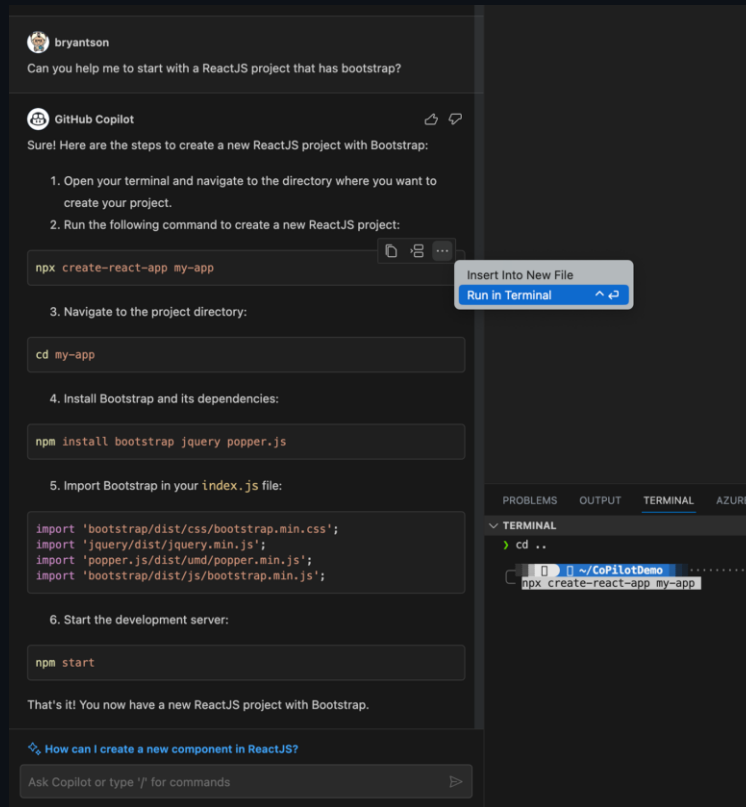
```



# Working in a project level

Ask in **GitHub Copilot Chat** to get started

GitHub Copilot Chat (default)



The screenshot shows the GitHub Copilot Chat interface. On the left, a user named 'bryantson' asks: "Can you help me to start with a ReactJS project that has bootstrap?". The Copilot response provides a step-by-step guide:

1. Open your terminal and navigate to the directory where you want to create your project.
2. Run the following command to create a new ReactJS project:

```
npx create-react-app my-app
```

3. Navigate to the project directory:

```
cd my-app
```

4. Install Bootstrap and its dependencies:

```
npm install bootstrap jquery popper.js
```

5. Import Bootstrap in your index.js file:

```
import 'bootstrap/dist/css/bootstrap.min.css';
import 'jquery/dist/jquery.min.js';
import 'popper.js/dist/umd/popper.min.js';
import 'bootstrap/dist/js/bootstrap.min.js';
```

6. Start the development server:

```
npm start
```

That's it! You now have a new ReactJS project with Bootstrap.

At the bottom, there is a suggestion: "How can I create a new component in ReactJS?". Below this is a search bar with the placeholder text "Ask Copilot or type '/' for commands".

On the right side of the chat window, there is a "Run in Terminal" button. A tooltip above it says "Insert Into New File" and "Run in Terminal".

At the bottom right, the "TERMINAL" tab is active, showing the command: `npx create-react-app my-app`.



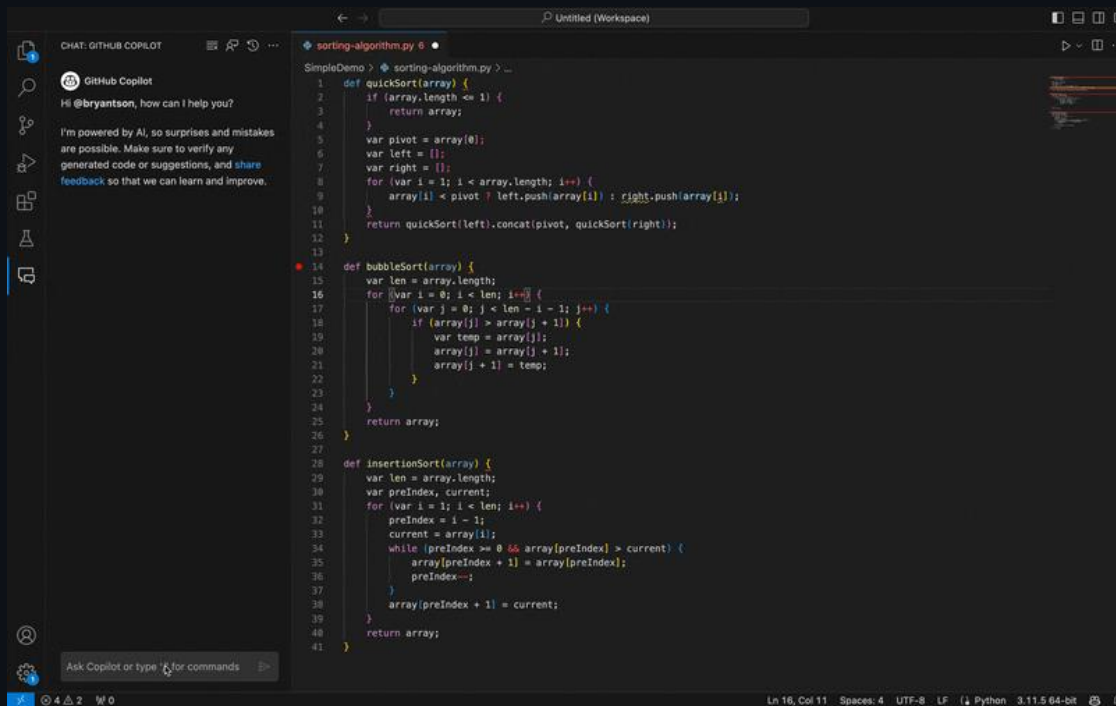
# Using Copilot to improve code quality





# Copilot Chat: Ask to Refactor

Ask GitHub  
Copilot Chat to  
refactor your  
code




CHAT: GITHUB COPILOT

GitHub Copilot

Hi @bryantson, how can I help you?

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

Ask Copilot or type  for commands

```

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         array[i] < pivot ? left.push(array[i]) : right.push(array[i]);
10    }
11    return quickSort(left).concat(pivot, quickSort(right));
12 }
13
14 def bubbleSort(array):
15     var len = array.length;
16     for (var i = 0; i < len; i++) {
17         for (var j = 0; j < len - i - 1; j++) {
18             if (array[j] > array[j + 1]) {
19                 var temp = array[j];
20                 array[j] = array[j + 1];
21                 array[j + 1] = temp;
22             }
23         }
24     }
25     return array;
26 }
27
28 def insertionSort(array):
29     var len = array.length;
30     var preIndex, current;
31     for (var i = 1; i < len; i++) {
32         preIndex = i - 1;
33         current = array[i];
34         while (preIndex >= 0 && array[preIndex] > current) {
35             array[preIndex + 1] = array[preIndex];
36             preIndex--;
37         }
38         array[preIndex + 1] = current;
39     }
40     return array;
41 }

```

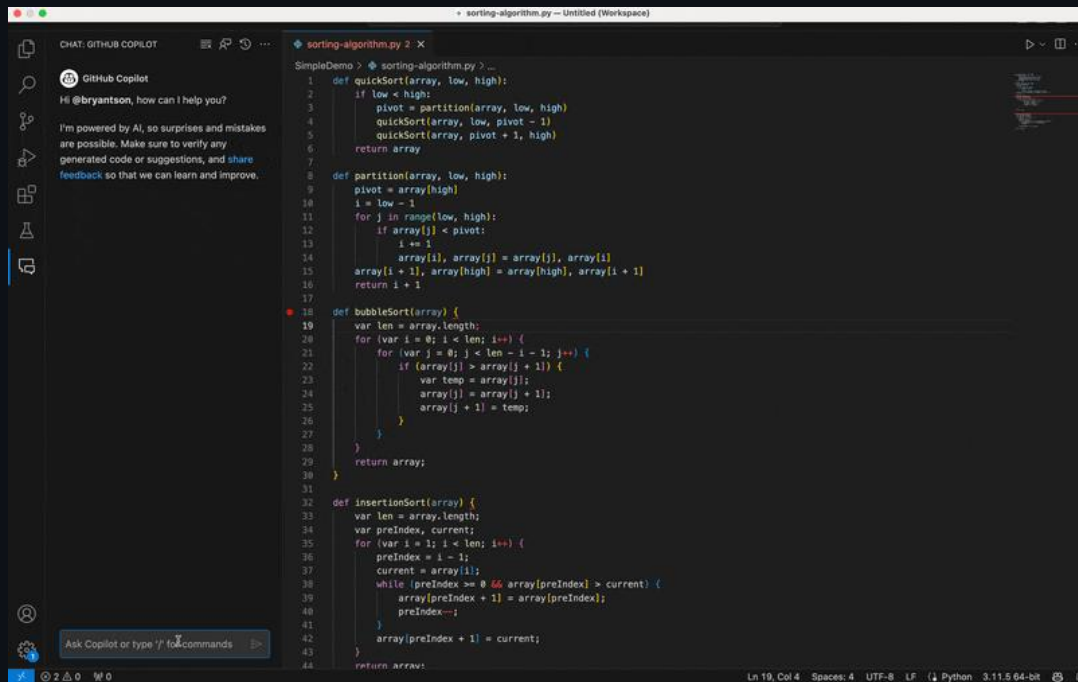
Ln 16, Col 11 Spaces: 4 UTF-8 LF Python 3.11.5 64-bit

GitHub Copilot Chat (default)



# Copilot Chat: Ask to generate tests

Ask GitHub  
Copilot Chat to  
generate tests



CHAT: GITHUB COPILOT

GitHub Copilot

Hi @bryantson, how can I help you?

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

Ask Copilot or type `/?` for commands

```

SimpleDemo > sorting-algorithm.py 2 X
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;

```

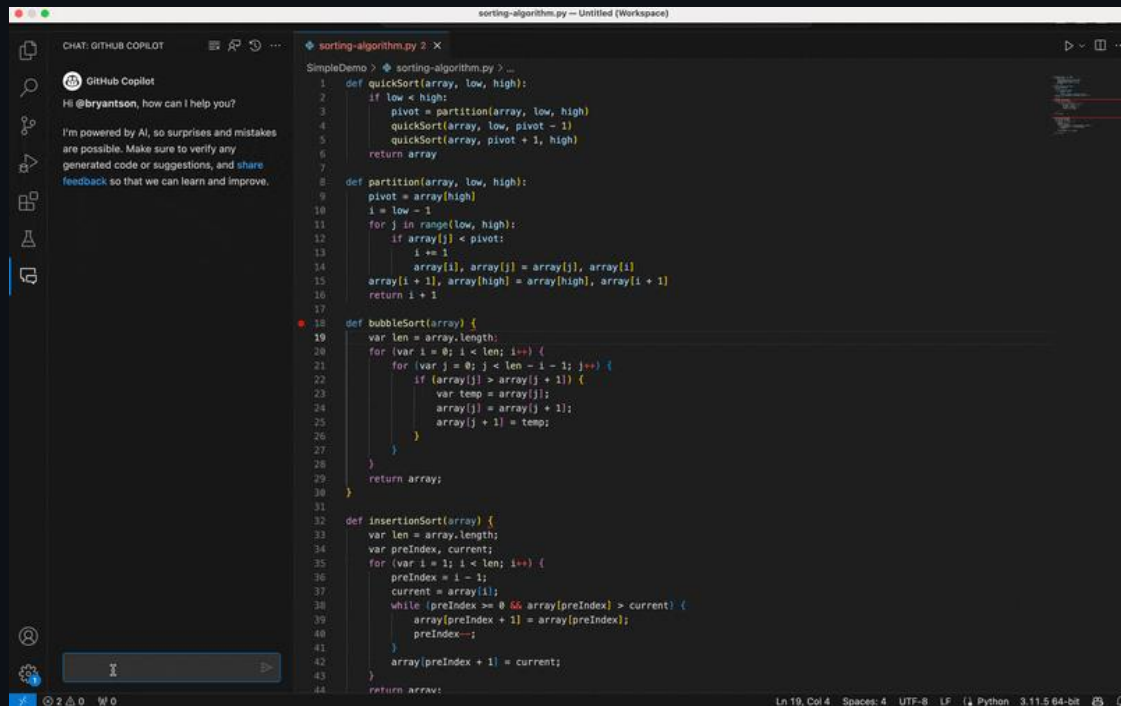
Ln 19, Col 4 Spaces: 4 UTF-8 LF (4 Python 3.11.5 64-bit)

GitHub Copilot Chat (default)



# /fix, /tests and other / command options

Leverage /  
command  
options that  
can help to  
improve your  
code



CHAT: GITHUB COPILOT

GitHub Copilot

Hi @bryantson, how can I help you?

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

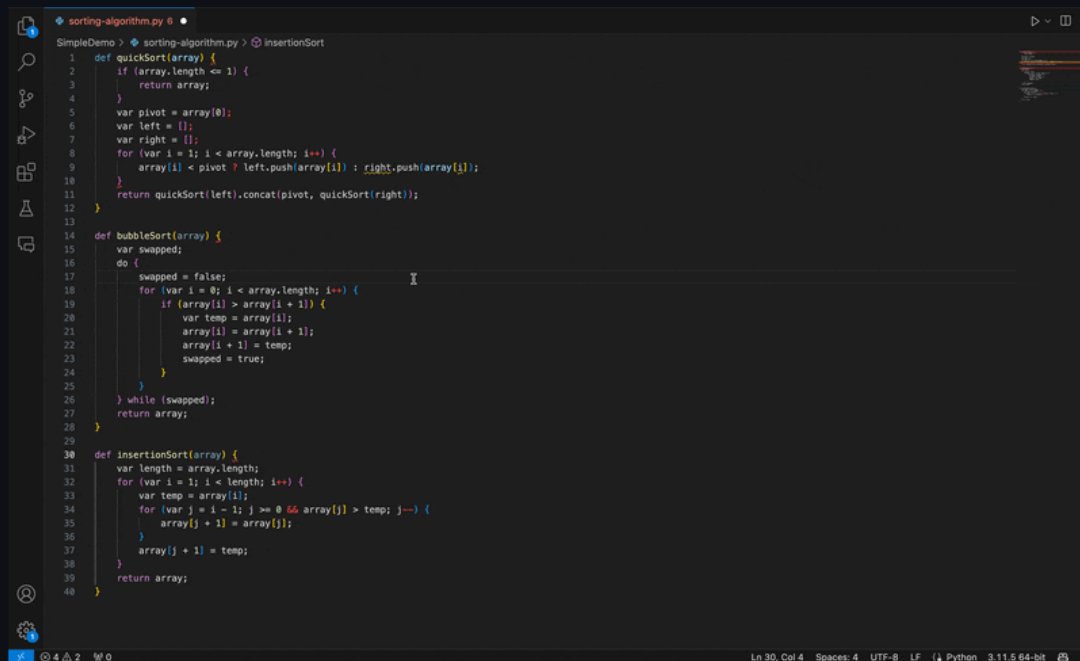
```
SimpleDemo > def sorting-algorithm.py > ...
1 def quickSort(array, low, high):
2     if low < high:
3         pivot = partition(array, low, high)
4         quickSort(array, low, pivot - 1)
5         quickSort(array, pivot + 1, high)
6     return array
7
8 def partition(array, low, high):
9     pivot = 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 }
```

Ln 19, Col 4 Spaces: 4 UTF-8 LF Python 3.11.5 64-bit



# In-file Copilot options

Copilot now offers **in-file Copilot** feature to selectively improve



The screenshot shows a code editor with a file named 'sorting-algorithm.py'. The code contains three sorting functions: quickSort, bubbleSort, and insertionSort. The quickSort function is partially completed, and the bubbleSort function is also partially completed. The insertionSort function is fully visible. The editor interface includes a sidebar on the left with icons for file explorer, search, and other tools. The status bar at the bottom indicates the current line and column (Ln 30, Col 4), the number of spaces (4), the encoding (UTF-8), the line format (LF), the language (Python), and the bitness (3.11.5 64-bit).

```
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     array[i] < pivot ? left.push(array[i]) : right.push(array[i]);
10  }
11  return quickSort(left).concat(pivot, quickSort(right));
12 }
13
14 def bubbleSort(array) {
15   var swapped;
16   do {
17     swapped = false;
18     for (var i = 0; i < array.length; i++) {
19       if (array[i] > array[i + 1]) {
20         var temp = array[i];
21         array[i] = array[i + 1];
22         array[i + 1] = temp;
23         swapped = true;
24       }
25     }
26   } while (swapped);
27   return array;
28 }
29
30 def insertionSort(array) {
31   var length = array.length;
32   for (var i = 1; i < length; i++) {
33     var temp = array[i];
34     for (var j = i - 1; j >= 0 && array[j] > temp; j--) {
35       array[j + 1] = array[j];
36     }
37     array[j + 1] = temp;
38   }
39   return array;
40 }
```

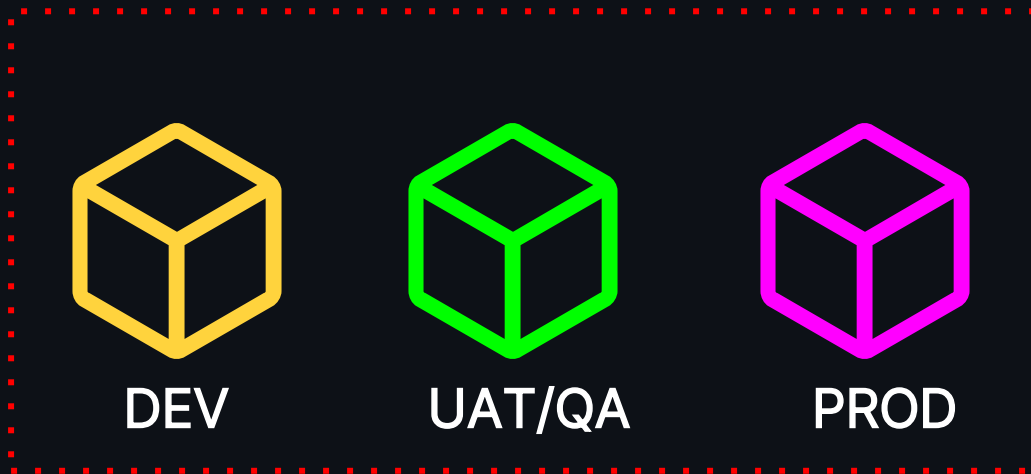


# What about my production layer?

Local



Cloud/Server



Current Copilot

“Copilot X”



# Copilot Security Exploit?

## Prompt Injection

Carefully crafted prompts to make the model to ignore its original instruction or perform unintended actions





# Workshop



# Choose Workshop Track

## Copilot Workshop



Workshop for those who want to see how to build web application

Primary IDE: VS Code  
Primary OS: Independent

## Copilot Workshop



Workshop for those who want to see how to build Java applications

Primary IDE: IntelliJ  
Primary OS: Independent

## Copilot Workshop



Workshop for those who want to see how to build .NET applications

Primary IDE: Visual Studio  
IDE  
Primary OS: Windows





# Secure coding

**Copilot and secure coding**

**Copilot + GitHub Advanced Security**

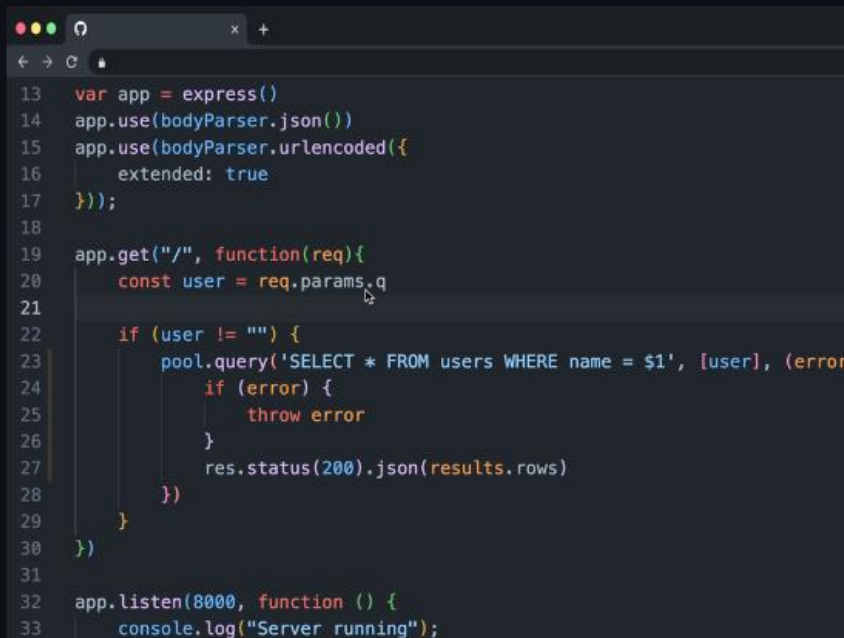
**Vulnerability detection**

**Vulnerability remediation**

**Copilot Trust Center**

# Copilot and secure coding

- AI-based vulnerability system that helps prevent insecure coding patterns (e.g. SQL script injection)
- Vulnerability filters are applied only to the Copilot-generated suggestion itself
  - ⊗ It cannot detect downstream vulnerabilities introduced by the code e.g. on deployment infrastructure
  - ⓘ We recommend taking the **same precautions** you take with code written by engineers (linting, code scanning, etc.)
- Copilot Chat can be used to query code for known vulnerabilities



```

13 var app = express()
14 app.use(bodyParser.json())
15 app.use(bodyParser.urlencoded({
16   extended: true
17 }));
18
19 app.get("/", function(req){
20   const user = req.params.q
21
22   if (user != "") {
23     pool.query('SELECT * FROM users WHERE name = $1', [user], (error
24       if (error) {
25         throw error
26       }
27       res.status(200).json(results.rows)
28     })
29   }
30 })
31
32 app.listen(8000, function () {
33   console.log("Server running");

```



# Copilot + GHAS

- Copilot is not a replacement of GHAS features
- Copilot can be used in tandem with GHAS features to detect and remediate vulnerabilities earlier during the SDLC
  - GHAS Code scanning results
  - GHAS Secret scanning



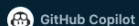
# Vulnerability detection

- Prompt for insecure coding patterns
- Use GHAS Code Scanning results
- Write custom CodeQL queries
- Increase your knowledge of secure coding patterns
- Create custom Secret Scanning patterns



stebje

It looks like this code is insecure. Help me understand what the issue is and how to resolve it.



Yes, this code is vulnerable to SQL injection attacks. The issue is that the `search` parameter is directly concatenated into the SQL query string without any sanitization or validation. This means that an attacker could potentially inject malicious SQL code into the query, which could lead to data theft or destruction.

To fix this issue, you should use parameterized queries instead of concatenating strings. Parameterized queries use placeholders for the input values, which are then bound to the query separately. This ensures that the input values are properly sanitized and validated before being used in the query.

Here's an example of how to use parameterized queries with the `pg` library:

```
const search = req.params.q;

if (search !== '') {
  const query = "SELECT * FROM users WHERE name = $1";
  const values = [search];
  pool.query(query, values, (err, res) => {
    console.log(err, res);
    pool.end();
  });
}
```

In this example, the `$1` placeholder is used for the `search` parameter, and the actual value is passed in as an array of values to the `pool.query()` function. This ensures that the `search` parameter is properly sanitized and validated before being used in the query.

```
4
5 const pool = new Pool( config: {
6   user: 'postgres',
7   host: 'localhost',
8   database: 'postgres',
9   password: 'process.env.POSTGRES_PW',
10  port: 5432,
11 });
12
13 var app = express();
14 app.use(bodyParser.json());
15 app.use(bodyParser.urlencoded({ extended: true }));
16
17 app.get('/', function (req, res) {
18   const search = req.params.q;
19
20   if (search !== '') {
21     var query = "SELECT * FROM users WHERE name = \"\" + search + \"\"";
22     pool.query( queryTextOrConfig: query, callback: (err, res) => {
23       console.log( message: err, optionalParams[0]: res);
24       pool.end();
25     });
26   }
27 });
28
29 app.listen( port: 8000, callback: function () {
30   console.log( message: 'Example app listening on port 8000!');
31 });
```



# Vulnerability remediation

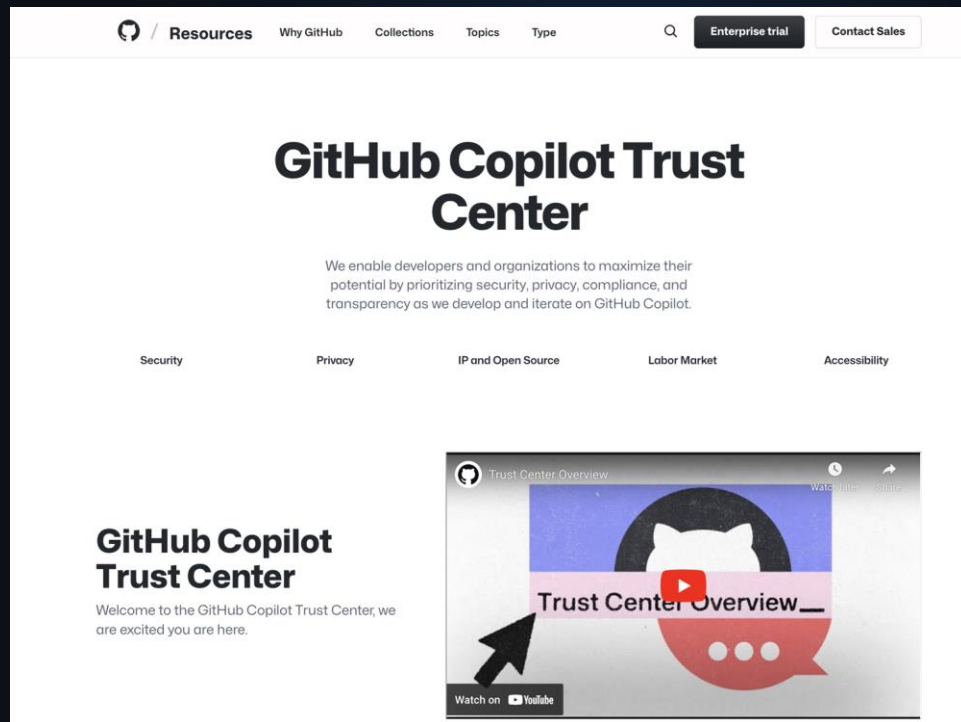
- Copilot helps prevent suggestions that contain insecure coding patterns in real-time
- Automatically fix insecure code based on Copilot suggestions
- Validate and improve existing CodeQL queries



# Security & Trust

## Copilot Trust Center

- Security
- Privacy
- Data flow
- Copyright
- Labor market
- Accessibility
- Contracting



# Wrap Up



# Thank you

