



GitHub Expert Services



GitHub Copilot Tips, Tricks and Best practices

Agenda



Beyond code completion



Context, Context, Context



Inline Chat



Copilot CLI



Workspace Agent



Threads



Slash Commands



Q&A



GitHub Copilot

Beyond code completion



It's no longer just a code completion tool in your editor—it now includes a chat interface that you can use in your IDE, a command line tool via a GitHub CLI extension, a summary tool in your pull requests, a helper tool in your terminals, and much, much more.

- Copilot

Context, Context, Context



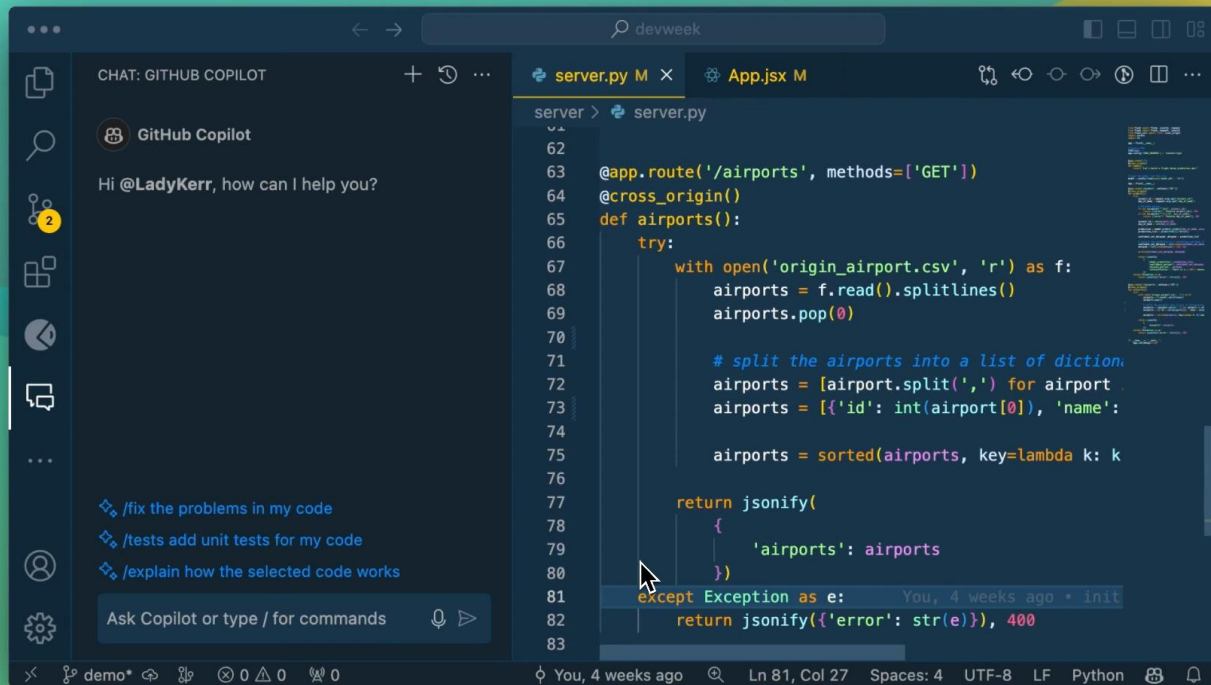
- LLMs are designed to make prediction based on the context provided
- Provide ample context is the key
- Copilot is able to infer from the code in IDE editor
- How about adding more context?

Context, Context, Context



- **Open Relevant files**
- **Provide Top level comment**
- **Set Includes and references**

Context, Context, Context



CHAT: GITHUB COPILOT

GitHub Copilot

Hi @LadyKerr, how can I help you?

- /fix the problems in my code
- /tests add unit tests for my code
- /explain how the selected code works

Ask Copilot or type / for commands

```

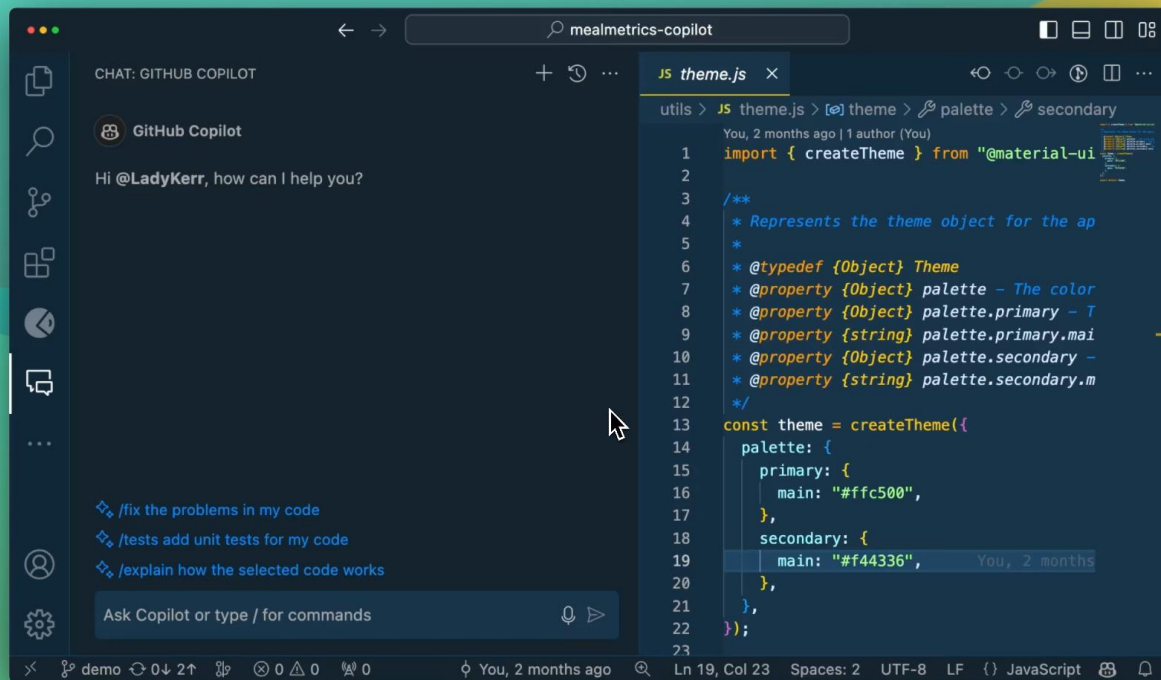
server > server.py
62
63 @app.route('/airports', methods=['GET'])
64 @cross_origin()
65 def airports():
66     try:
67         with open('origin_airport.csv', 'r') as f:
68             airports = f.read().splitlines()
69             airports.pop(0)
70
71         # split the airports into a list of diction
72         airports = [airport.split(',') for airport
73         airports = [{'id': int(airport[0]), 'name':
74
75         airports = sorted(airports, key=lambda k: k
76
77         return jsonify(
78             {
79                 'airports': airports
80             })
81     except Exception as e:
82         return jsonify({'error': str(e)}), 400
83

```

You, 4 weeks ago Ln 81, Col 27 Spaces: 4 UTF-8 LF Python



Context, Context, Context

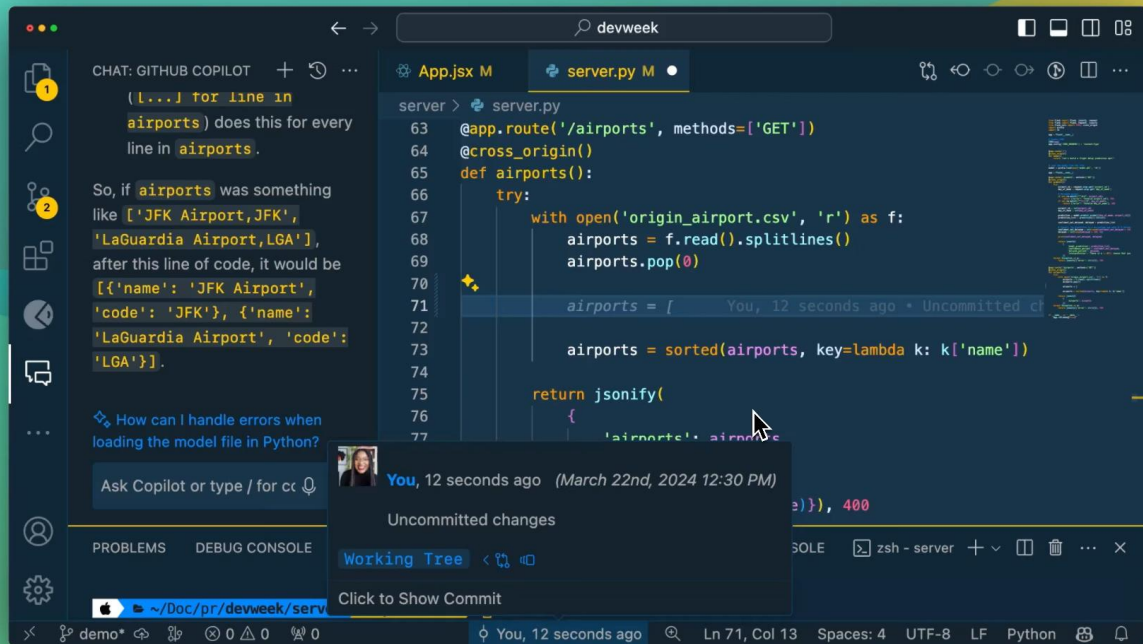


Context, Context, Context



- **Meaningful names**
- **Specific function comments**
- **Provide sample code**
- **Let's give them a try**

Context, Context, Context

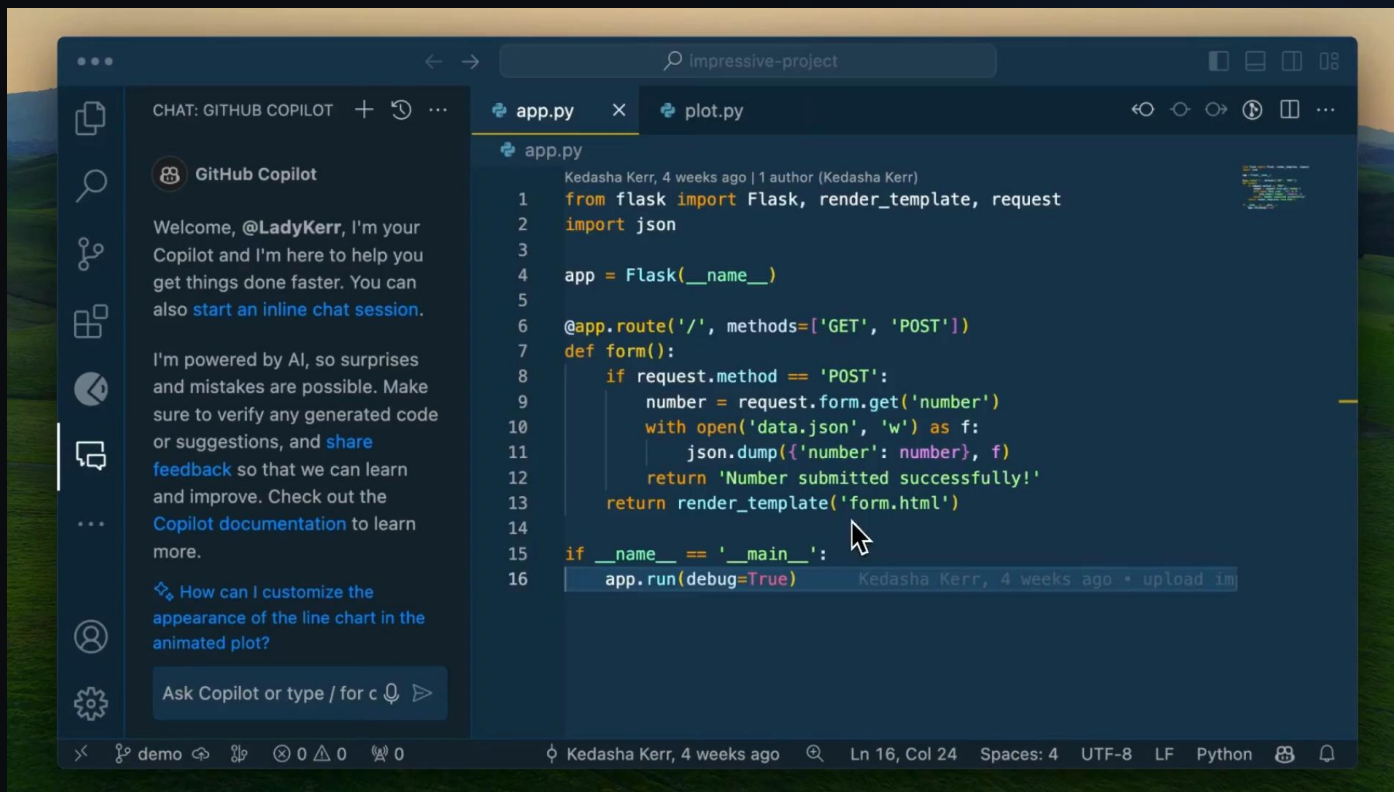


Inline Chat with GitHub Copilot



- CTRL + I on Windows
- Ask specific questions
- Inline code diffs
- Slash command support
- Let's give it a go

Inline Chat with GitHub Copilot

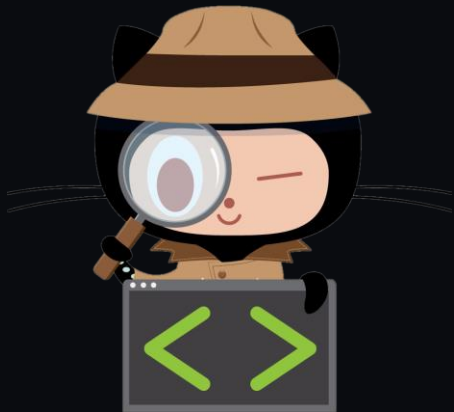


GitHub Copilot CLI



- You can use Copilot with the GitHub CLI to get suggestions and explanations for the command line.
- To ask Copilot in the CLI to explain a command, run `gh copilot explain` followed by the command that you want explained.
- To ask Copilot in the CLI to suggest a command, run `gh copilot suggest` followed by the command that you want

Copilot Chat



GitHub Copilot Chat provides an experience in your editor where you can have a conversation with the AI assistant. You can improve this experience by using built-in features to make the most out of it.

- Copilot

Copilot Chat



- Remove irrelevant requests
- Navigate through your conversation
- Use the @workspace agent
- Highlight relevant code
- Organize your conversations with threads

Slash Commands

Command	Description	Usage
<code>/explain</code>	Get code explanations	Open file with code or highlight code you want explained and type: <code>/explain what is the fetchPrediction method?</code>
<code>/fix</code>	Receive a proposed fix for the problems in the selected code	Highlight problematic code and type: <code>/fix propose a fix for the problems in fetchAirports route</code>
<code>/tests</code>	Generate unit tests for selected code	Open file with code or highlight code you want tests for and type: <code>/tests</code>
<code>/help</code>	Get help on using Copilot Chat	Type: <code>/help what can you do?</code>
<code>/clear</code>	Clear current conversation	Type: <code>/clear</code>
<code>/doc</code>	Add a documentation comment	Highlight code and type: <code>/doc</code>

You can also press CMD+I in your editor and type `/doc/` inline



Slash Commands

`/generate`

Generate code to answer your question

Type:

`/generate code that validates a phone number`

`/optimize`

Analyze and improve running time of the selected code

Highlight code and type:

`/optimize fetchPrediction method`

`/clear`

Clear current chat

Type:

`/clear`

`/new`

Scaffold code for a new workspace

Type:

`/new create a new django app`

`/simplify`

Simplify the selected code

Highlight code and type:

`/simplify`

`/feedback`

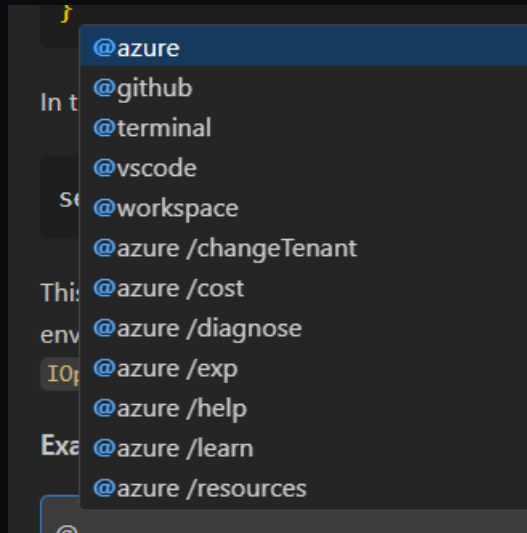
Provide feedback to the team

Type:

`/feedback`

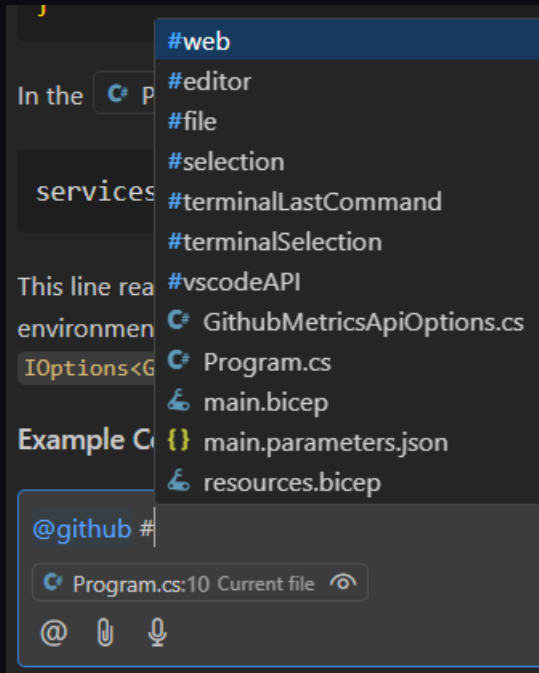


Workspace Agent



- Doesn't response straight away
- Analyzes the current workspace files and directories
- Creates a context from everything it was able to collect.

Chat Variables



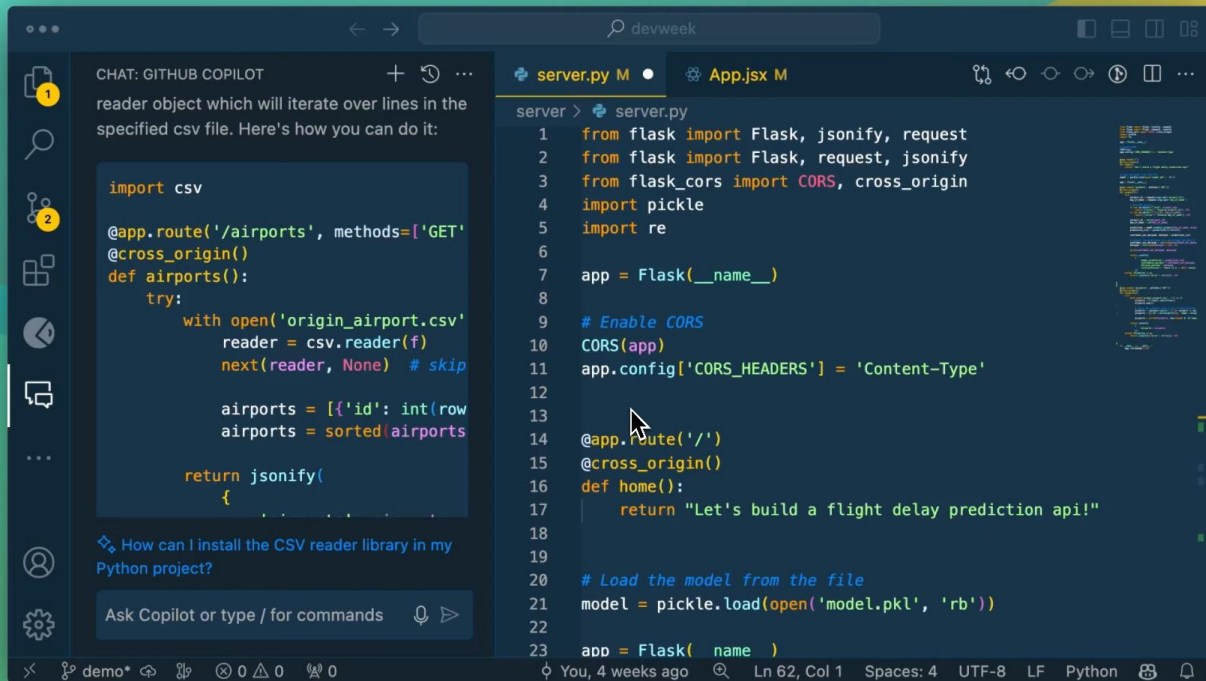
- Use chat variables to include specific context in your prompt.
- Use # in the chat prompt box, followed by a chat variable.
- #web is in preview that uses Bing web search engine

More tips



- **Attach relevant files for reference**
- **Start with GitHub Copilot Chat for faster debugging**
- **Be on the lookout for sparkles!**

More tips



The screenshot shows the VS Code interface with a chat window on the left and two code files open in the editor.

CHAT: GITHUB COPILOT

reader object which will iterate over lines in the specified csv file. Here's how you can do it:

```
import csv

@app.route('/airports', methods=['GET'])
@cross_origin()
def airports():
    try:
        with open('origin_airport.csv')
            reader = csv.reader(f)
            next(reader, None) # skip
            airports = [{ 'id': int(row
            airports = sorted(airports

    return jsonify(
        {


```

How can I install the CSV reader library in my Python project?

Ask Copilot or type / for commands

server.py

```
1 from flask import Flask, jsonify, request
2 from flask import Flask, request, jsonify
3 from flask_cors import CORS, cross_origin
4 import pickle
5 import re
6
7 app = Flask(__name__)
8
9 # Enable CORS
10 CORS(app)
11 app.config['CORS_HEADERS'] = 'Content-Type'
12
13
14 @app.route('/')
15 @cross_origin()
16 def home():
17     return "Let's build a flight delay prediction api!"
18
19
20 # Load the model from the file
21 model = pickle.load(open('model.pkl', 'rb'))
22
23 app = Flask( name )
```

App.jsx

```
1 import React, { useState } from 'react'
2 import './App.css'
3
4 function App() {
5   const [search, setSearch] = useState('')
6   const [results, setResults] = useState([])
7
8   const handleSearch = async () => {
9     const response = await fetch(`http://localhost:5000/airports?search=${search}`)
10    const data = await response.json()
11    setResults(data)
12  }
13
14   return (
15     <div>
16       <input type="text" value={search} onChange={e => setSearch(e.target.value)} />
17       <button onClick={handleSearch}>Search</button>
18       <div>
19         {results.map((airport) => (
20           <div>
21             <div>{airport.id}</div>
22             <div>{airport.name}</div>
23             <div>{airport.city}</div>
24             <div>{airport.country}</div>
25           </div>
26         ))}
27       </div>
28     </div>
29   )
30 }
31
32 export default App
```

Conclusion

- Context
- Prompt Crafting
- Stay in control





Q & A

Upcoming Sessions

- 1 Intro to Prompt Engineering
- 2 Demo of Use Cases
- 3 Q&A Session



Thank you!