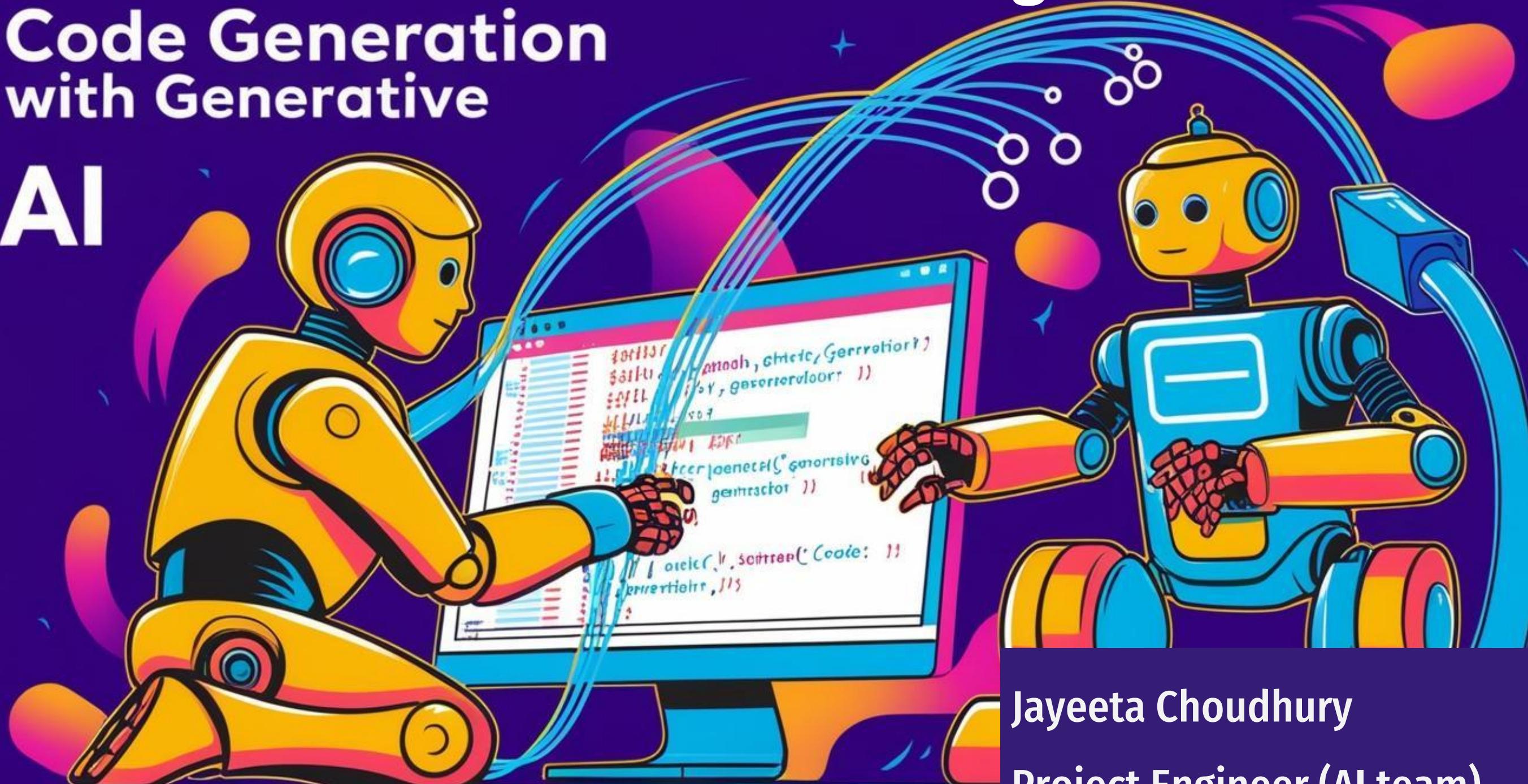


# Code Generation with Generative

AI

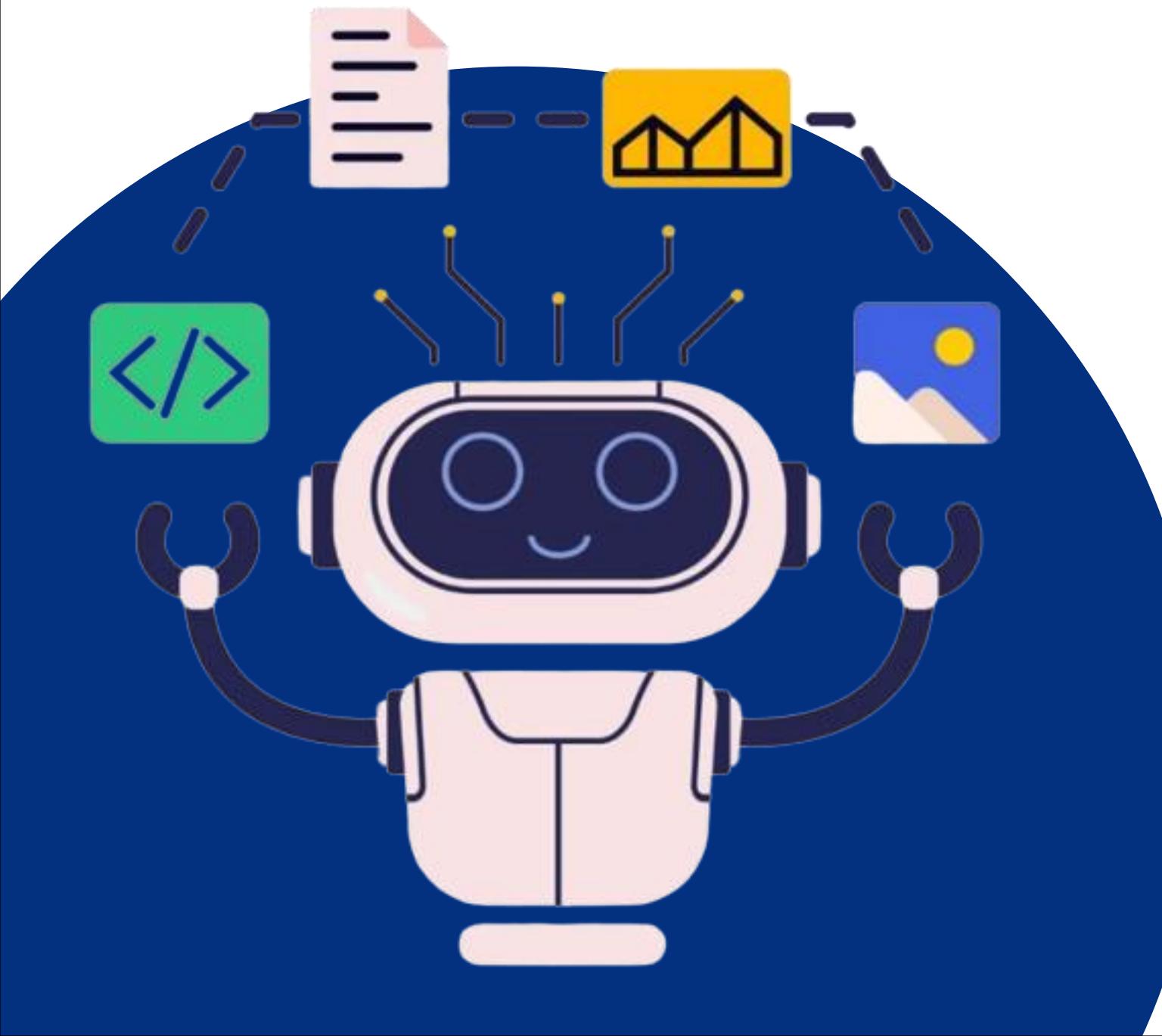
# AI-Assisted Coding



Jayeeta Choudhury  
Project Engineer (AI team)  
C-DAC, Bangalore



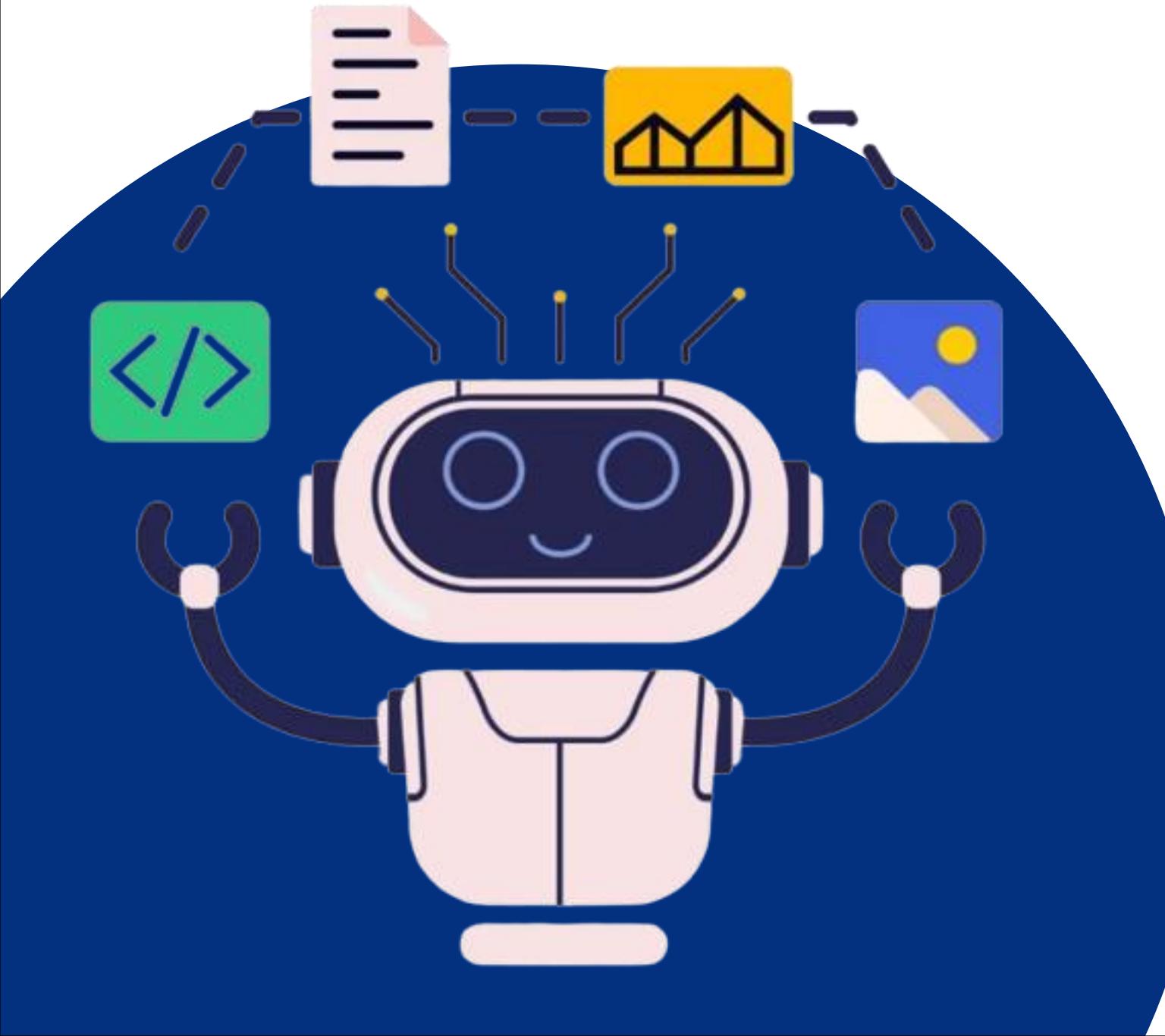
# Outline



01. How Software Development Evolved
02. Github Survey and Experiment on Developers
03. Benefits of Code Generation with Gen AI
04. LLMs for Code Generation and Datasets used for their Training
05. AI tools for Code Generation



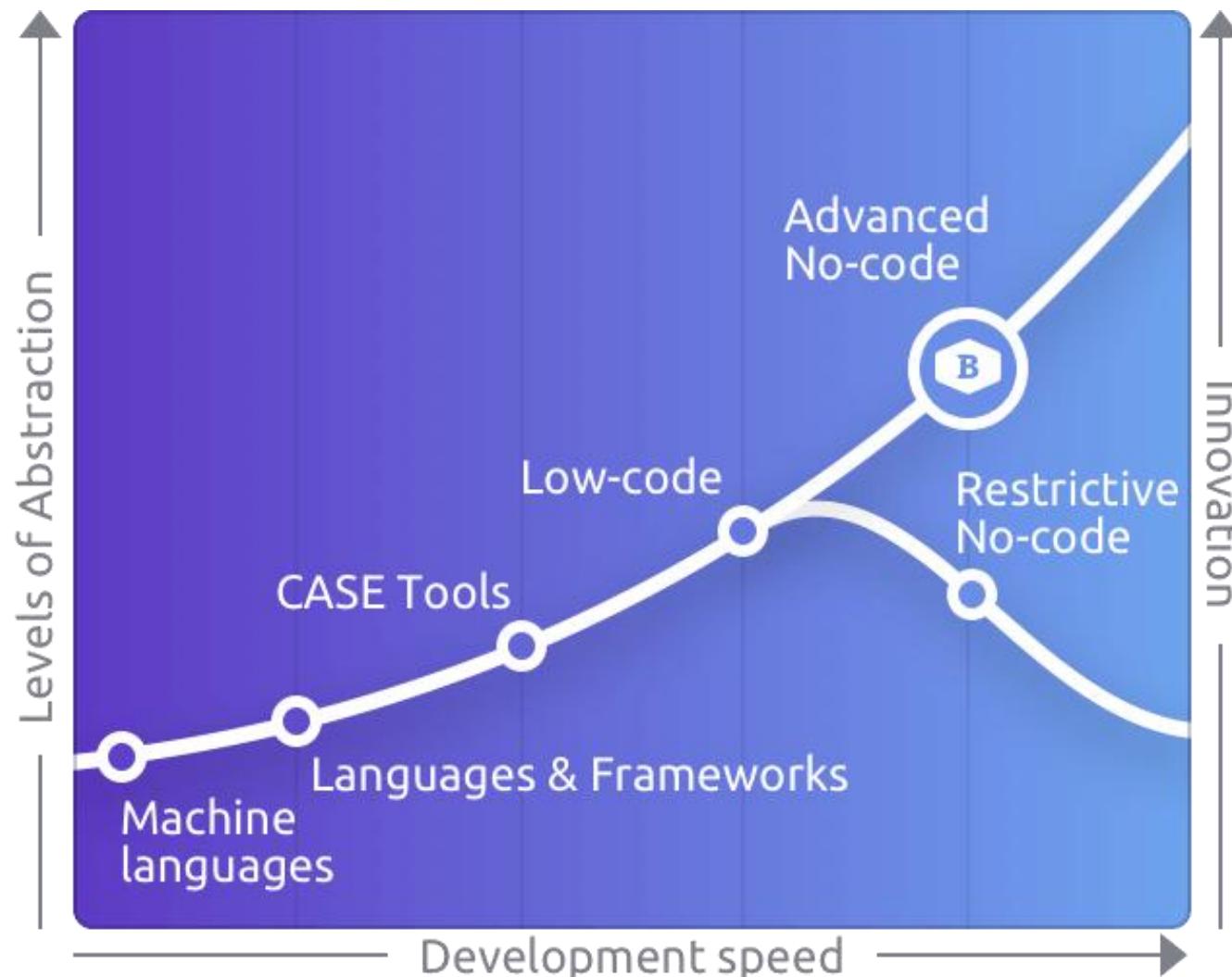
# Outline



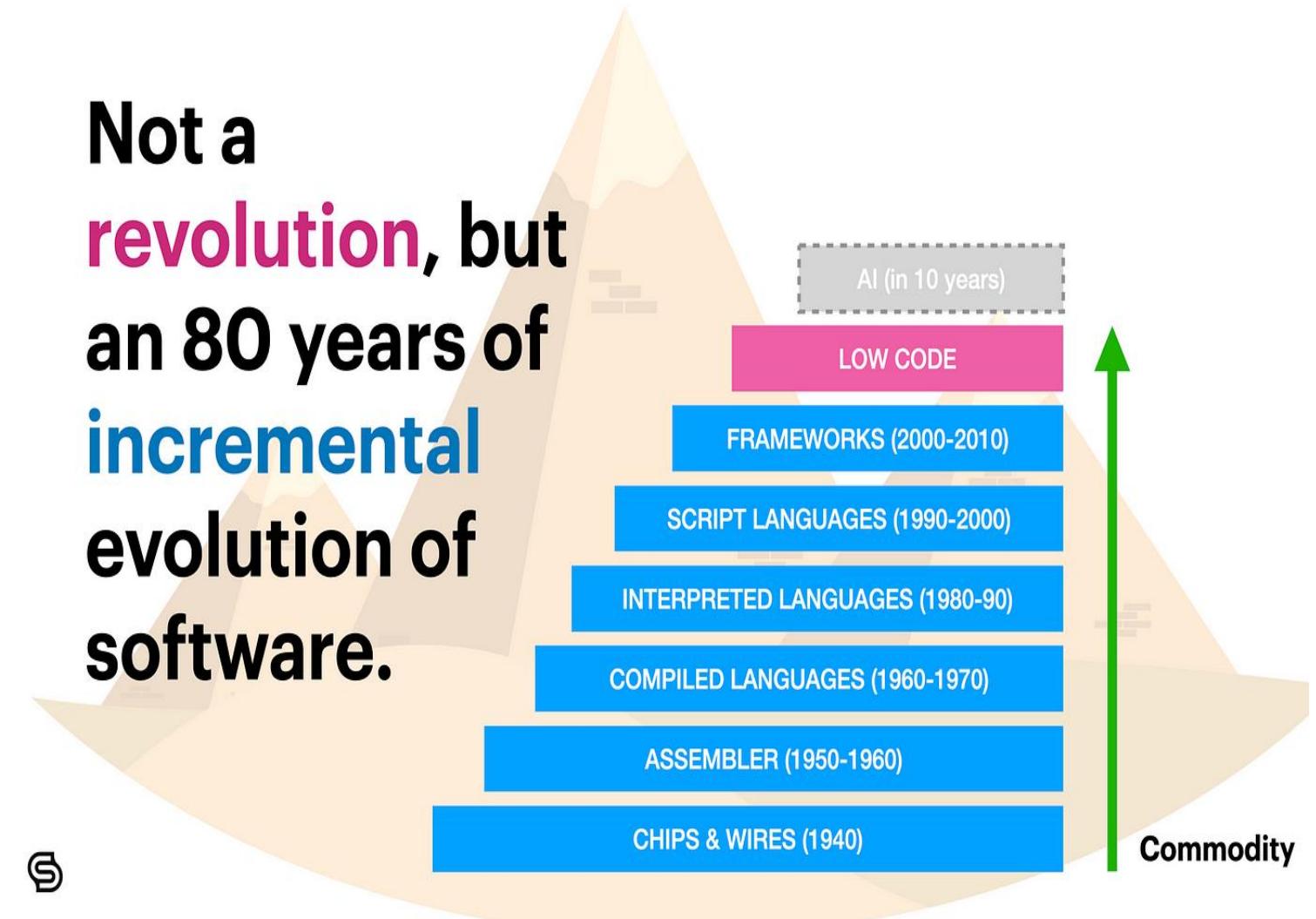
06. Challenges of Code Generation with AI
07. Prompt Engineering Tips for Code Generation
08. Github Copilot Setup and Usage
09. Use Case Demo
10. Conclusion

# How Software Development Evolved

- Coding Evolution: From manual binary coding to high-level programming languages, complexity has reduced significantly over time.
- Efficiency Growth: Each phase introduced tools that made writing, debugging, and scaling code faster and easier.
- AI's Role: Today, AI tools simplify coding further, automating tasks and enhancing productivity.

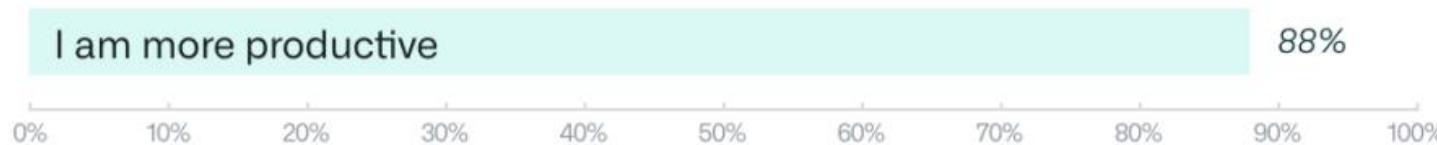


**Not a revolution, but an 80 years of incremental evolution of software.**

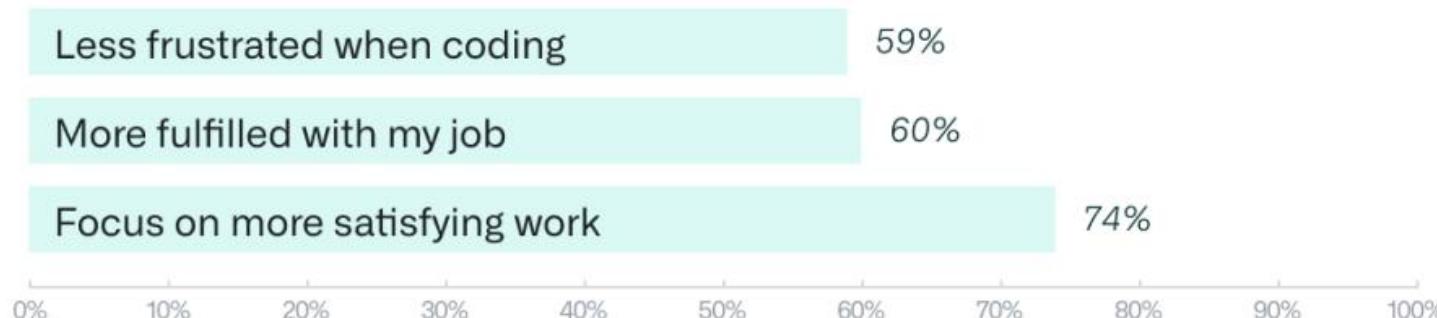


# When using GitHub Copilot...

## Perceived Productivity



## Satisfaction and Well-being\*



## Efficiency and Flow\*



**Github surveyed more than 2,000 developers to learn at scale about their experience using GitHub Copilot and found out**

Github Recruited



developers, and split them randomly into two groups.

gave them the task of writing a web server in JavaScript

### **45 Used**

GitHub Copilot

**78%**  
finished

**1 hour, 11 minutes**

average to complete the task

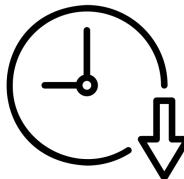


Results are statistically significant ( $P=.0017$ ) and the 95% confidence interval is [21%, 89%]

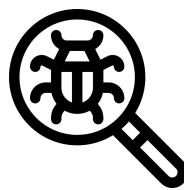
In Another Experiment Done by Github they observed

- The group that used GitHub Copilot had a higher rate of completing the task.
- The striking difference was that developers who used GitHub Copilot completed the task significantly faster—**55% faster** than the developers who didn't use GitHub Copilot.

# Benefits of AI Driven Code Generation



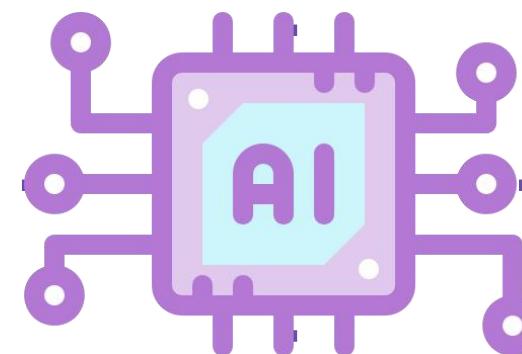
Reduced Development Time



Simplified Debugging



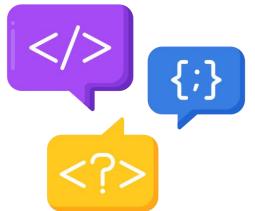
Improved Code Quality



Learning Assistance



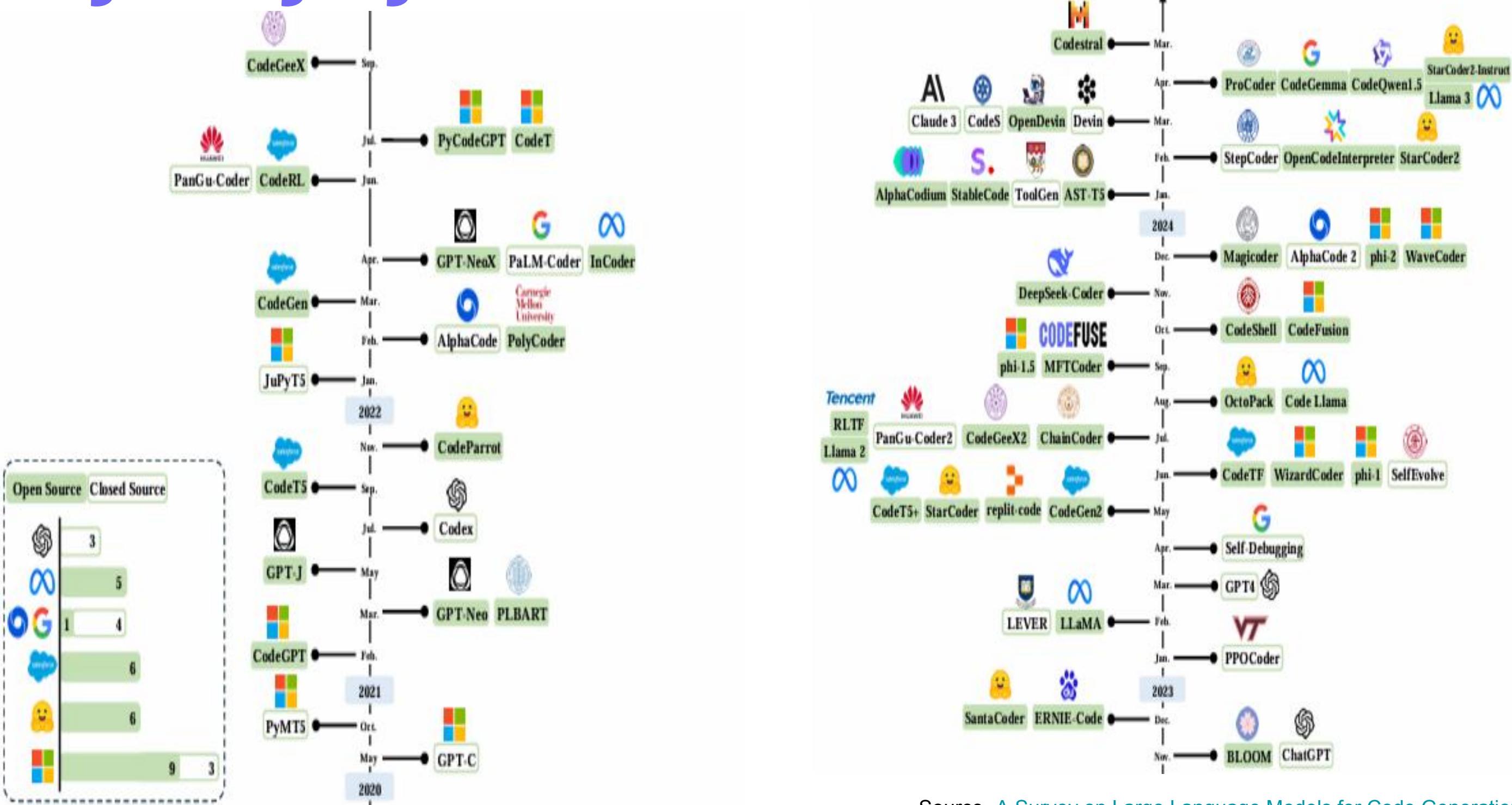
Cross Language Flexibility



Task Specific Outputs



# Large Language Models Trained for Code Generation



Source- A Survey on Large Language Models for Code Generation

# Datasets used for Training LLMs for Code Generation

<b>Dataset</b>	<b>Size (GB)</b>	<b>Files (M)</b>	<b>#PL</b>
CodeSearchNet [110]	20	6.5	6
Google BigQuery[96]	-	-	-
The Pile [78]	95	19	-
CodeParrot [254]	180	22	1
GitHub Code[254]	1,024	115	32
ROOTS [137]	163	15	13
The Stack [132]	3,136	317	30
The Stack v2 [170]	32K	3K	619

Datasets used for pre-training LLMs

#PL = Number of programming languages included in the dataset

<b>Dataset</b>	<b>Size</b>	<b>#PL</b>
CodeAlpaca-20K [43]	20k	-
CommitPackFT [187]	2GB	277
Evol-Instruct-Code-80k [225]	80k	-
evol-codealpaca-v1 [251]	110K	-
Magicoder-OSS-Instruct-75k [278]	75k	Python, Shell, TypeScript, C++, Rust, PHP, Java, Swift, C#
Self-OSS-Instruct-SC2-Exec-Filter-50k [304]	50k	Python

Datasets used for instruction tuning LLMs

# AI Tools For Code Generation

	 GitHub Copilot	 ChatGPT	 tabnine	 Amazon Q Developer	 Devin
<b>Supported Languages</b>	All Languages	All Popular Languages	20 Programming Languages	15 Programming Languages	All Popular Languages
<b>Integration</b>	Works with Visual Studio Code, JetBrains IDEs, and Neovim	Works through a chat interface	Works with popular IDEs like VS Code, JetBrains, Sublime Text	Integrated into IDEs such as VS Code, JetBrains, and AWS Cloud9	Works with IDEs like VS Code and JetBrains
<b>Underlying Model</b>	OpenAI Codex	GPT-4o, o1	Tabnine Model	Unknown	Diverse
<b>Price</b>	Starting at \$10 Per Month	Pro Plan Starts \$20/Month	Pro Plan Costs \$12/Month	Pro Version Costs \$19/Month	Basic Plan Starts at \$25/Month
<b>Code Completion</b>	✓	✓	✓	✓	✓
<b>Code Generation</b>	✓	✓	✓	✓	✓
<b>Debugging Assistance</b>	✓	✓	✓	✓	✓
<b>Documentation Generation</b>	✓	✓	✓	✗	✗

# Comparing Relevant Code Generation Tools

Features	ChatGPT	Amazon CodeWhisperer	GitHub Copilot
IDE Support	No IDE Support	JetBrains, Visual Studio Code, AWS Cloud9, or the AWS Lambda console	IntelliJ IDEA, Android Studio, AppCode, CLion, Code With Me Guest, DataGrip, DataSpell, GoLand, JetBrains Client, MPS, PhpStorm, PyCharm, Rider, RubyMine, WebStorm
First Release Time	Nov-30-2022	June-23-2022	Oct-29-2021
Developer	OpenAI	AWS	OpenAI-Microsoft
Providing References to Suggestions	NO	YES	NO
Explanation of Suggestions	YES	NO	NO
Providing Multiple Suggestions	NO (Theoretically user can manually ask for another suggestion.)	YES (Up to 5)	YES (Up to 10)
Training Data Source	GitHub Repositories, OpenAI Codex Dataset, other code repositories such as GitLab, Bitbucket, and SourceForge	"Vast amounts of publicly available code"	"...trained on all languages that appear in public repositories" (Fine-tuned)
Programming Languages work best with (according to the vendor)	N/A	C#, Java, JavaScript, Python, and TypeScript	C, C++, C#, Go, Java, JavaScript, PHP, Python, Ruby, Scala, and TypeScript
Multipurpose (other than programming)	YES	NO	NO
Subscription	ChatGPT Free ChatGPT Plus (\$20 per month)	Free Preview	Copilot for Students (Free) Copilot for Individuals (\$10 per month) Copilot for Business (\$19 per user, per month)
Can be Used Offline?	NO	NO	NO
Can it Access Local Files?	NO	YES	YES

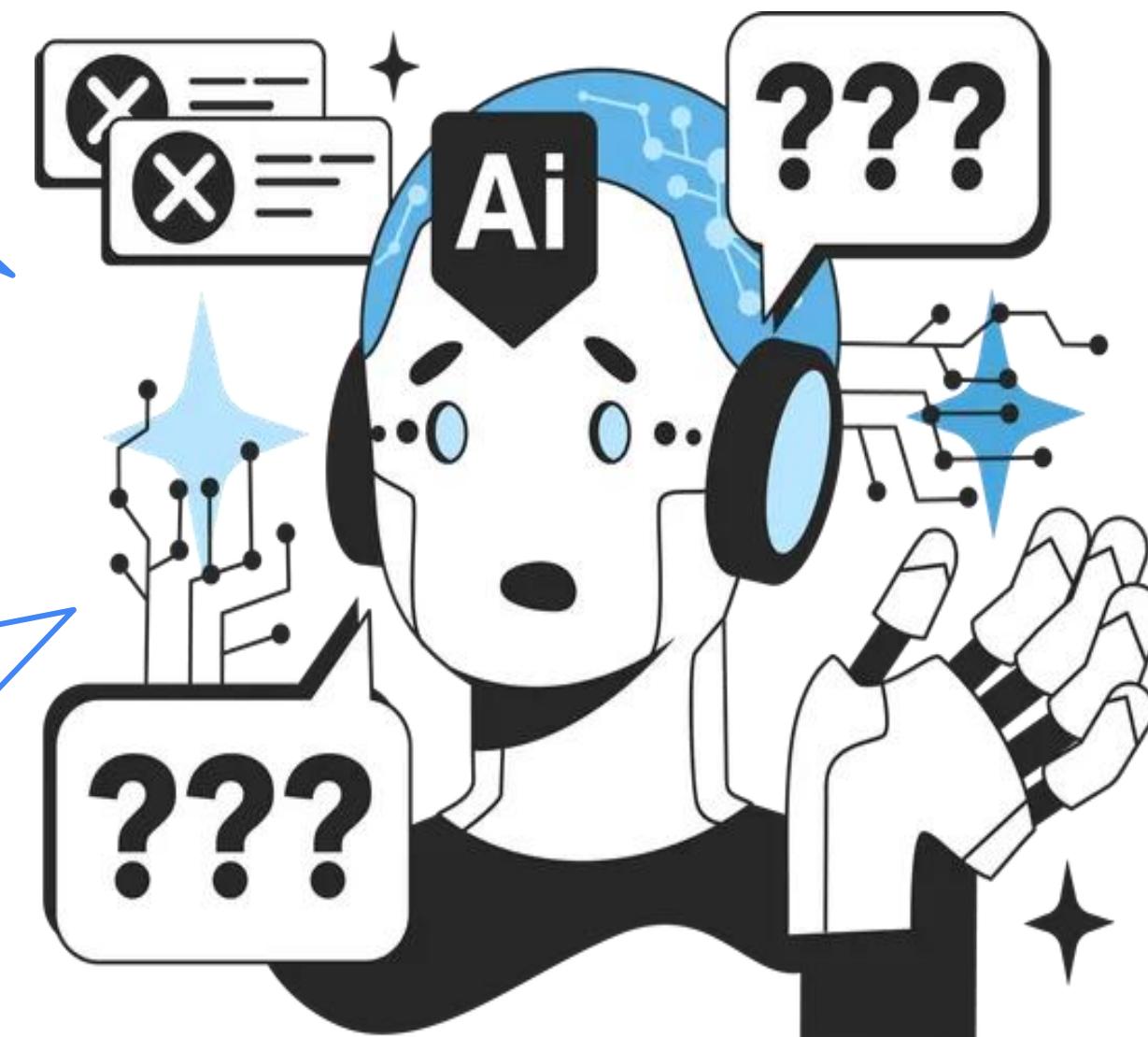
# Challenges of Code Generation with Generative AI

Understanding Natural and Programming Languages Both Which can be confusing

Balancing Readability, Efficiency, Correctness, and Completeness

Handling Diverse Input/Output Requirements with varying Complexities

Due to the inherent complexity of programming tasks LLMs can Hallucinate



# Prompt Engineering to the Rescue

**Be specific:** Clearly define task requirements and expected functionality.

**Specify language:** Mention the programming language in the prompt explicitly.

**Include examples:** Provide input-output examples to clarify expected behavior.

**State libraries:** List required libraries or frameworks for the task.

**Ask for comments:** Request code with inline comments for better understanding.

**Limit complexity:** Break complex tasks into smaller, manageable subtasks for clarity.

**Focus on constraints:** Specify performance, scalability, or formatting requirements.

**Iterate prompts:** Refine and retry prompts to improve the generated code.

**Use templates:** Provide partial code or templates for the model to follow.

**Request explanations:** Ask for explanations alongside code to verify logic and intent



Getting started with



Free Version

# What features are included in Copilot Free?

- **Code completion** in Visual Studio Code, Visual Studio, JetBrains IDES, Vim/Neovim, Xcode, and Azure Data Studio
- **Copilot Edits** to make changes across multiple files (only in Visual Studio Code and Visual Studio)
- **Copilot Chat** in Visual Studio Code, Visual Studio, JetBrains IDES, and GitHub.com
- **Block suggestions** matching public code
- Access to **Claude 3.5 Sonnet** models

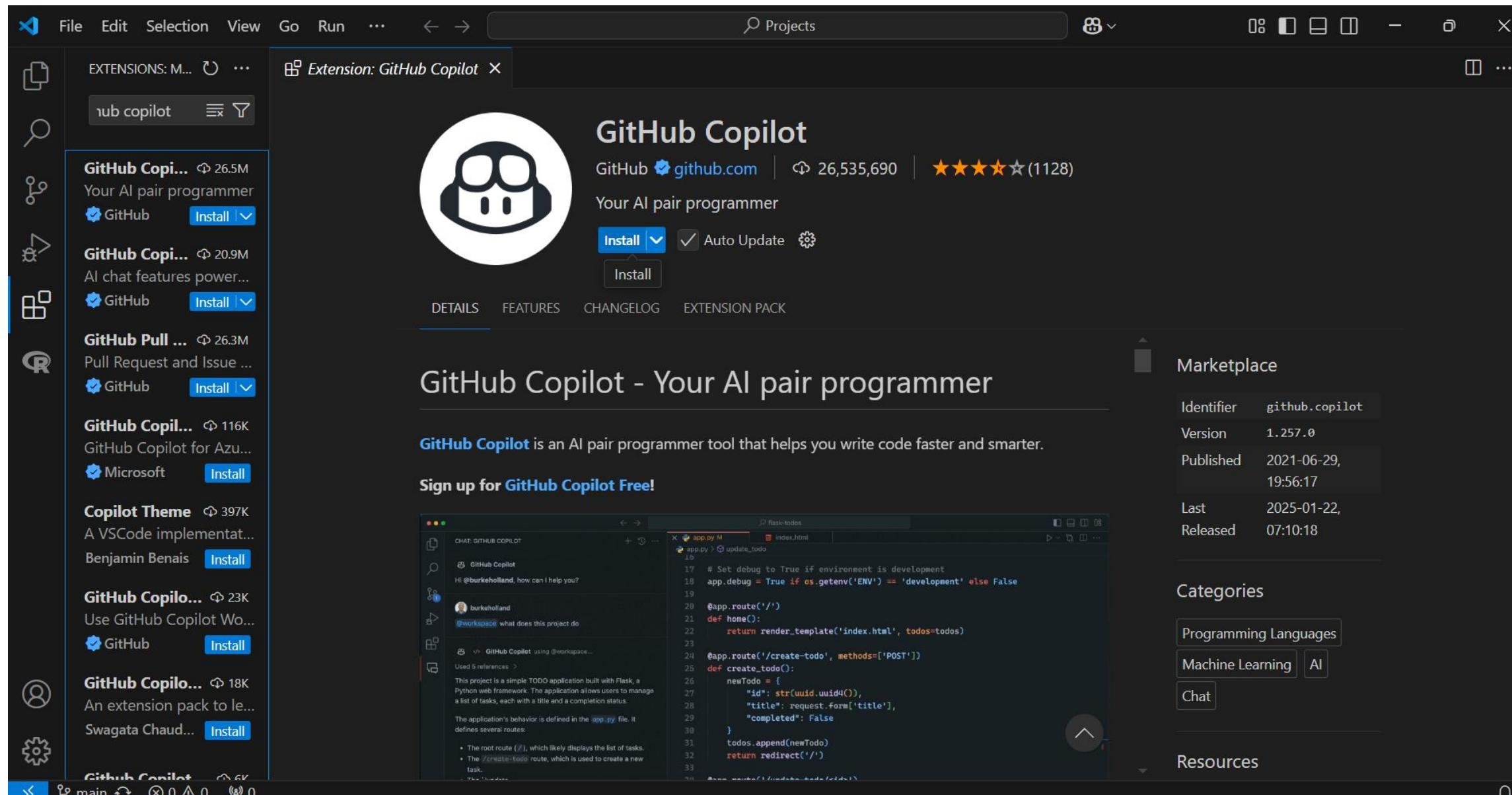
# What are the limitations of Copilot Free?

- **Code completions** are limited to **2000** completions per month.
- **Copilot Chat** is limited to **50 chat messages per month**. This limit includes both standard chats and multi-file editing chats in VS Code and Visual Studio.
- When you reach these limits, you can upgrade to Copilot Pro to continue using Copilot.

# Set up Visual Studio Code with Copilot

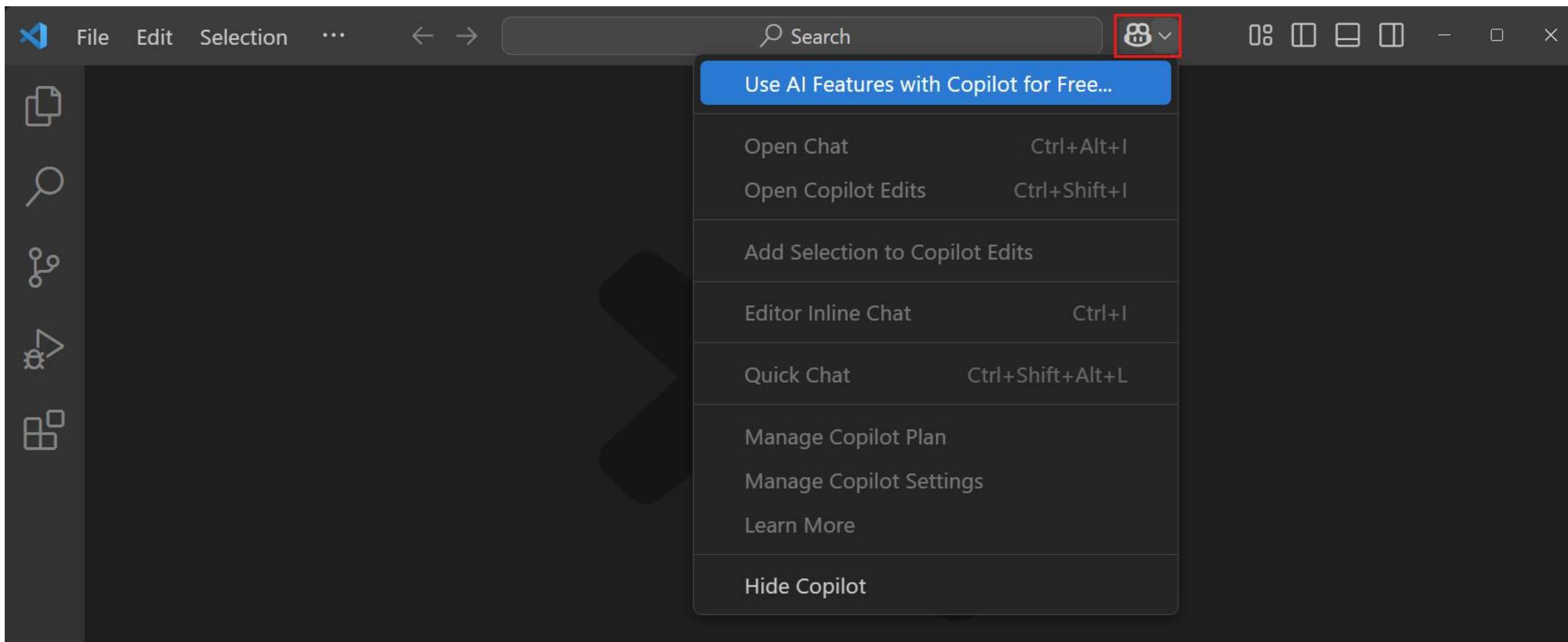
To get started with Copilot Free in Visual Studio, you need:

1. Ensure you have a recent version of VS Code
2. GitHub Copilot in Visual Studio



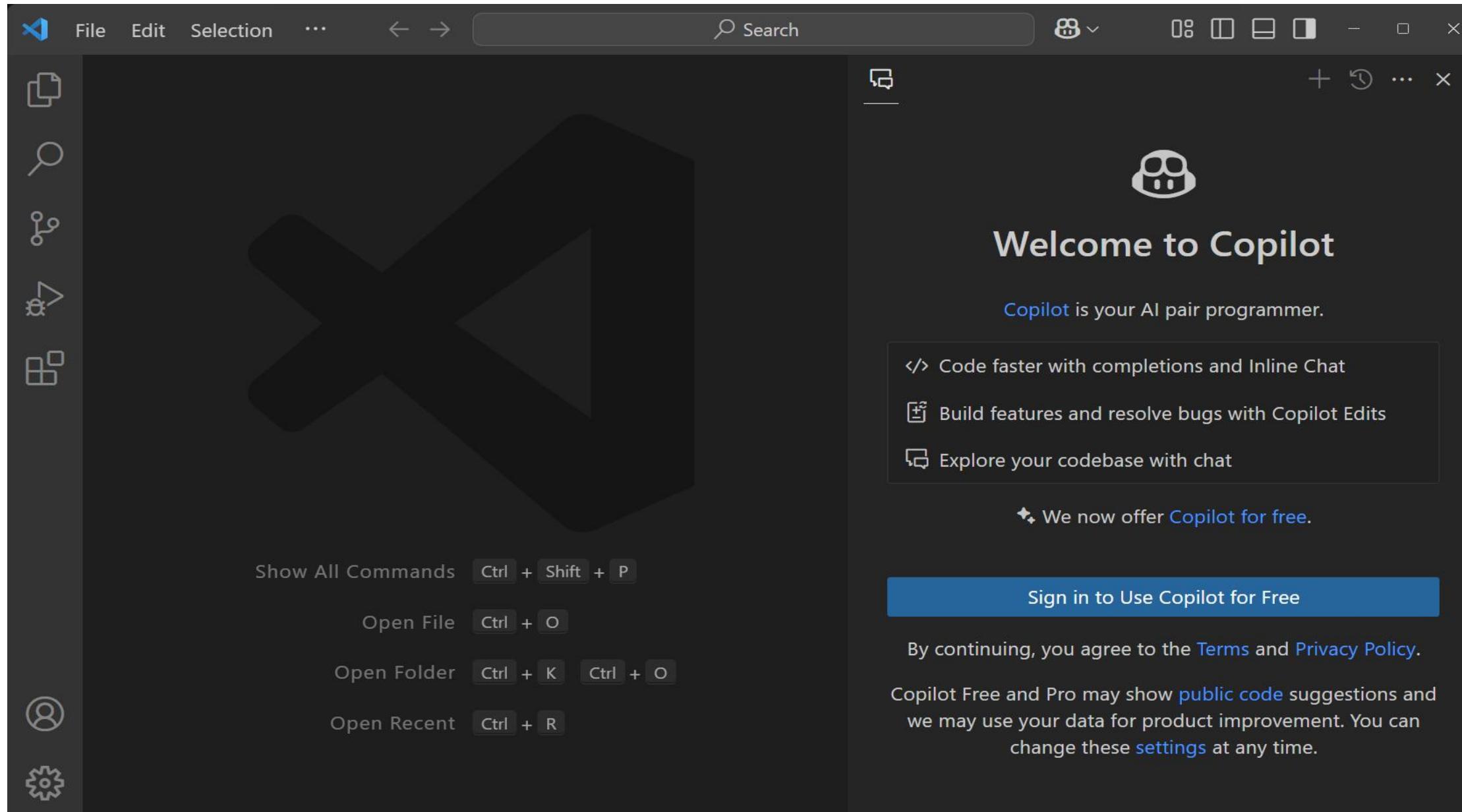
# Set up Visual Studio Code with Copilot

3. Select Use AI Features with Copilot for Free... from the Copilot menu in the title bar or from the Command Palette (Ctrl+Shift+P)



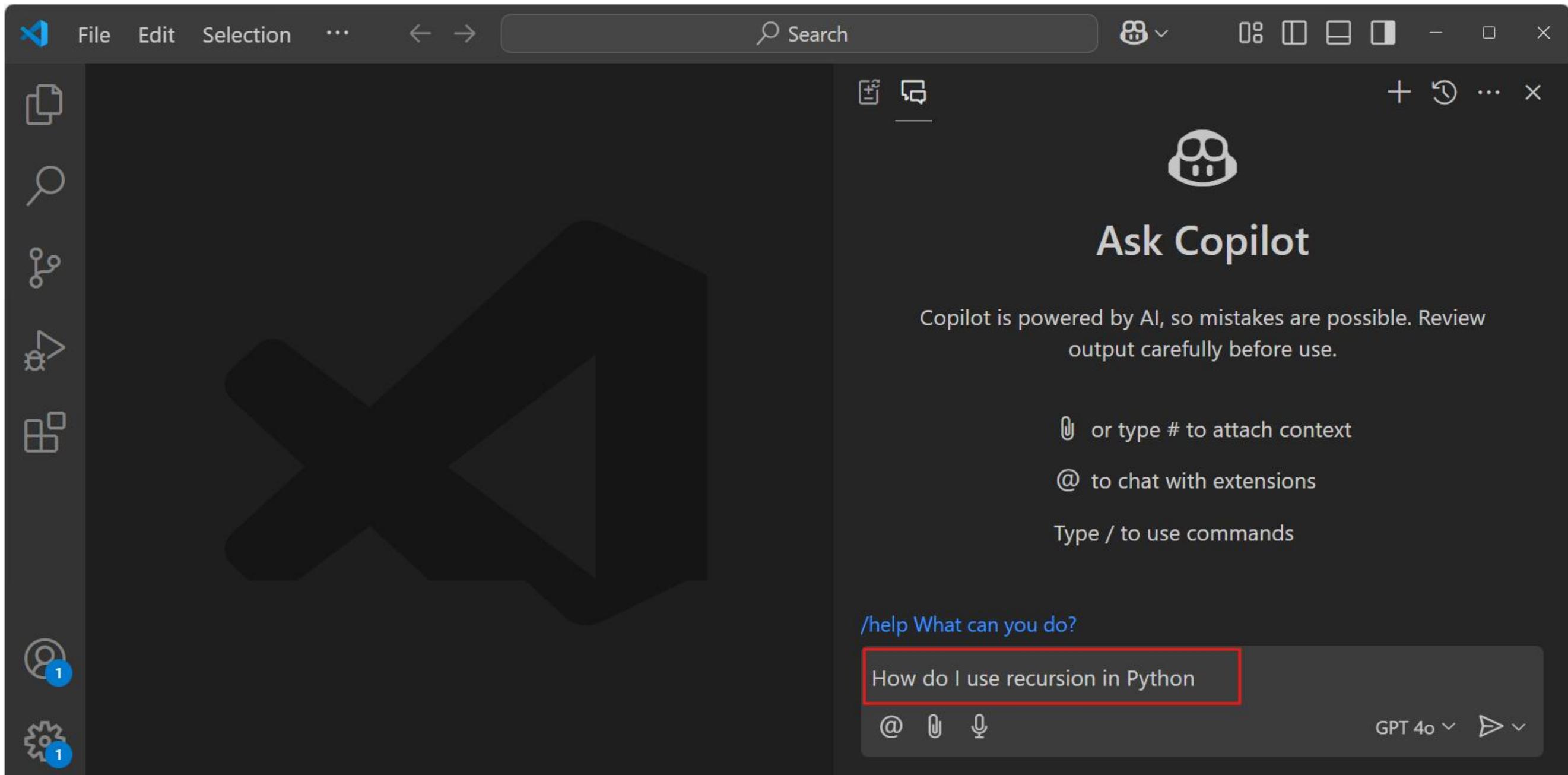
# Set up Visual Studio Code with Copilot

4. Select Sign in to Use Copilot for Free to sign in to your GitHub account and sign up for Copilot Free  
If you already have a Copilot subscription associated with your GitHub account, VS Code uses that one after you sign in.



# Set up Visual Studio Code with Copilot

5. Get started by entering a prompt in the chat input field



# Using GitHub Copilot method 1

Start **typing code** and accept suggestions with the Tab key  
Writing code for evaluating a model with help of copilot

The screenshot shows a Jupyter Notebook interface with the title bar "try-github-copilot". The left sidebar lists two notebooks: "BikeSharingDataAnalysis.ipynb" and "BikeSharingDataAnalysis.ipynb > ...". The main area displays Python code for training a linear regression model:

```
# Train a linear regression model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, y_train)
```

A tooltip from GitHub Copilot is visible, showing the suggestion "LinearRegression()", which is highlighted with a blue border. The tooltip text reads: "Press [ctrl]+[space] to ask GitHub Copilot to do something, start typing to dismiss." The status bar at the bottom right indicates "Python".

## Using GitHub Copilot method 2 -

Use // and write your requirements Accept suggestions with the Tab key

Generating a function that retrieves DB items by priority

```
0 references | 0 changes | 0 authors, 0 changes
39     public static void CreateTables()
40     {
41         using (var context = new TaskContext())
42         {
43             context.Database.ExecuteSqlRaw("CREATE TABLE tasks (id INT PRIMARY KEY, title VARCHAR(50), priority INT)");
44         }
45     }
46
47     I
48
49
50
51
52
53
54
55
56
57
58
59     }
60 }
```

## Using GitHub Copilot method 3 -

Use **ctrl + I** and write your requirements and click on accept to accept changes  
Asking copilot to make code readable in javascript

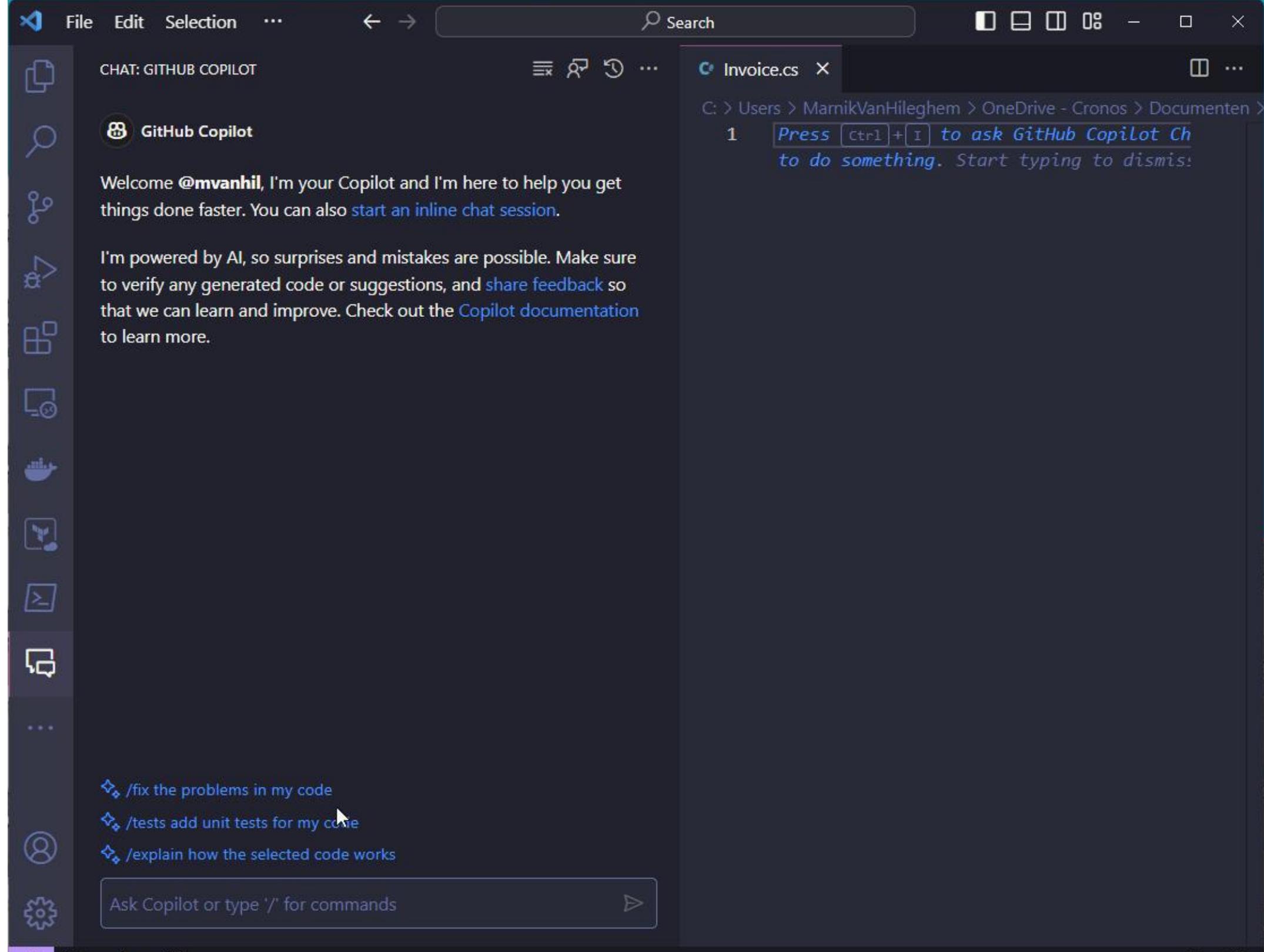
The screenshot shows a Visual Studio Code window with a dark theme. The title bar reads "utils.js - Untitled (Workspace) - Visual Studio Code - Insiders". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar has icons for file operations like Open, Save, Find, and Run. The main editor area contains the following JavaScript code:

```
97 }
98
99 function resetFontWeight(defaultFontWeight) {
100     $('#note').css('font-weight', defaultFontWeight);
101     $('#fontWeight').val(defaultFontWeight);
102 }
103
104 function resetShowWordCountPill(defaultShowWordCountPill) {
105     defaultShowWordCountPill === 'Yes' ? $('.word-count-container').show() : $('.word-count-container').hide();
106     $('#showWordCountPill').val(defaultShowWordCountPill);
107 }
108
109 function countWords(str) {
110     return str.trim().split(/\s+/).length;
111 }
112
113 function calculateCharactersAndWords(str) {
114     const characterCount = str.length;
115     const wordCount = str !== '' ? countWords(str) : 0;
116     const wordCountText = `${characterCount} character(s), ${wordCount} word(s)`;
117
118     return wordCountText;
119 }
```

The status bar at the bottom shows "Ln 102, Col 2" and "Spaces: 4" along with other file and encoding details.

**Using GitHub  
Copilot method 4 -**  
**Use Github copilot  
chat to generate code  
and apply changes  
using apply to editor  
button**

Asking copilot to create  
a realistic invoice model  
with line items in C-  
sharp



The screenshot shows the GitHub Copilot interface within the Visual Studio Code editor. The top bar includes File, Edit, Selection, ..., Back, Forward, Search, and window control buttons. The left sidebar has a dark theme with icons for file operations like Open, Save, Find, and others. The main area is titled "CHAT: GITHUB COPILOT" and "GitHub Copilot". It displays a welcome message for the user @mvanhil, stating: "Welcome @mvanhil, I'm your Copilot and I'm here to help you get things done faster. You can also start an inline chat session." Below this, it says: "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. Check out the Copilot documentation to learn more." At the bottom of the main area, there's a list of commands starting with a star icon: "/fix the problems in my code", "/tests add unit tests for my code", and "/explain how the selected code works". A text input field at the bottom says "Ask Copilot or type '/' for commands". The status bar at the bottom shows line 1, column 1, spaces: 4, UTF-8, CRLF, C#, and other icons.

CHAT: GITHUB COPILOT

**GitHub Copilot**

Welcome @mvanhil, I'm your Copilot and I'm here to help you get things done faster. You can also [start an inline chat session](#).

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. Check out the [Copilot documentation](#) to learn more.

Ask Copilot or type '/' for commands

Invoice.cs

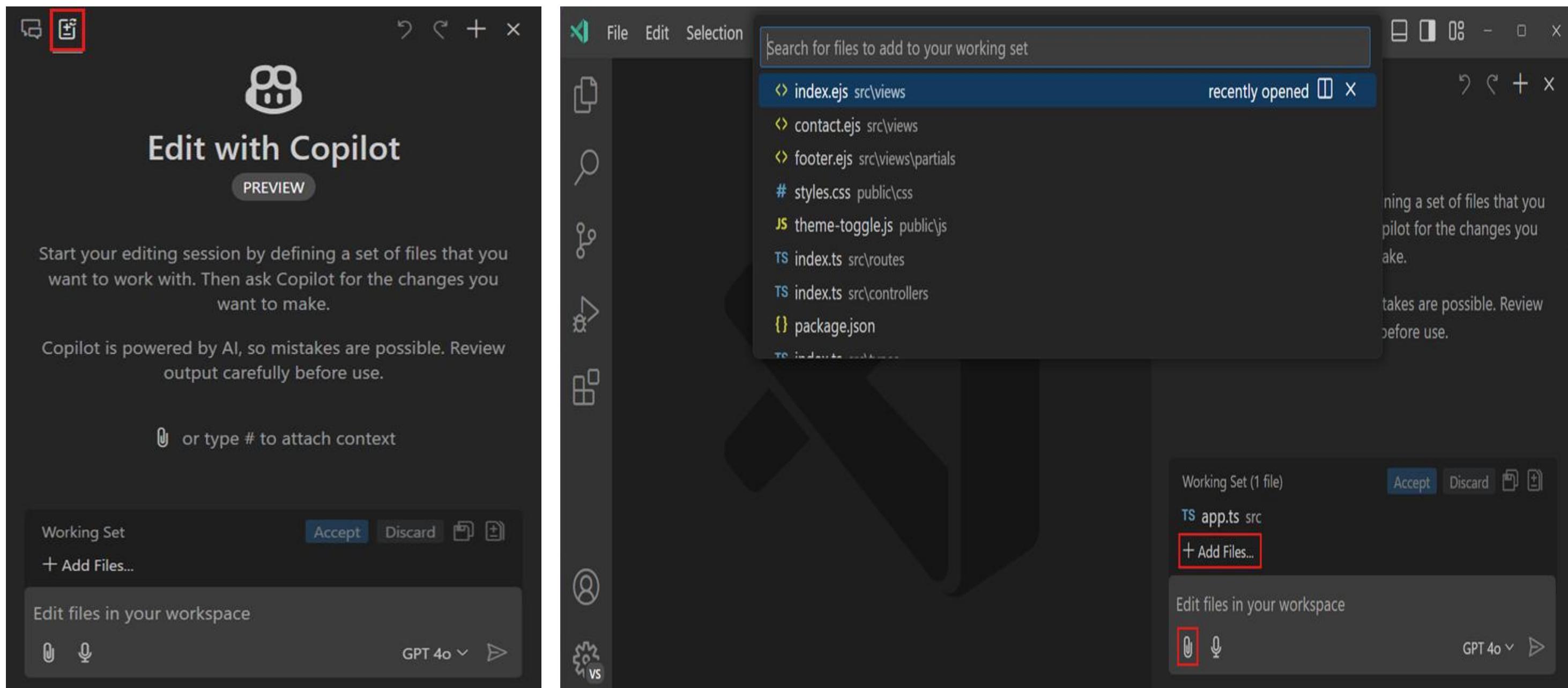
C:\Users\.../OneDrive - Cronos\Documenten>

1 Press [Ctrl]+[I] to ask GitHub Copilot Ch to do something. Start typing to dismiss.

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C#

## Using GitHub Copilot method 5 -

**Using Copilot edits** - Copilot Edits proposes code changes across multiple files in your workspace. These edits are applied directly in the editor, so you can quickly review them in-place, with the full context of the surrounding code.



## Using GitHub Copilot method 5 -

**Using Copilot edits** - Copilot Edits proposes code changes across multiple files in your workspace. These edits are applied directly in the editor, so you can quickly review them in-place, with the full context of the surrounding code.

The screenshot shows a VS Code workspace for a 'flask-chatbot' project. The Explorer sidebar shows files like `__init__.py`, `routes.py`, `index.html`, `style.css`, and `requirements.txt`. The `index.html` file is open in the editor, displaying HTML code for a chatbot interface. A floating panel titled 'Copilot Edits (Ctrl+Shift+I)' contains suggestions for modifying the `style.css` file:

- Suggestion 1: 'make chatbot interface more beautiful fit it to whole screen add icon for sender and receiver and make send button green'
- Suggestion 2: 'GitHub Copilot style.css' (with a detailed description: 'Update the CSS to make the chatbot interface fit the whole screen, add icons for sender and receiver, and make the send button green.'
- Suggestion 3: 'routes.py' (with a message: 'No changes needed in this file for the requested modifications.'

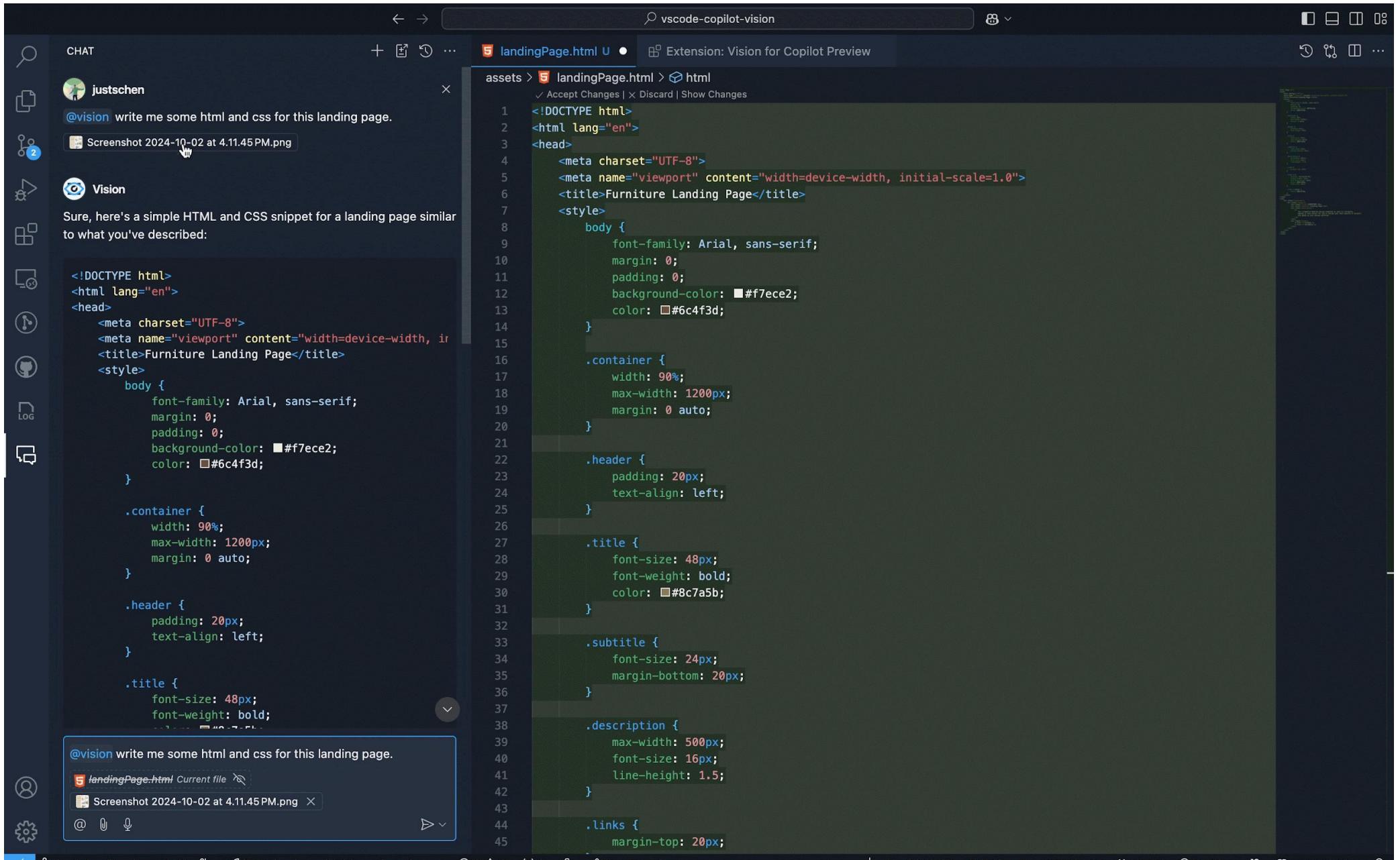
The bottom right corner of the floating panel shows a 'Working Set (5 files)' list with items: `# style.css`, `__init__.py`, `index.html`, `__init__.py`, and `routes.py`. Buttons for 'Accept' and 'Discard' are also visible.

The terminal at the bottom shows the command: `(myenv) C:\Users\Charu\Desktop\Projects\flask-chatbot>`.

# Added Extensions - Vision for Copilot



**Vision for Copilot**  
 Preview is an **extension** that enhances chat interactions by enabling users to **leverage advanced vision capabilities**. Allows users to attach images directly as contextual input, enriching conversations and enabling more dynamic, visually-supported responses.



```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Furniture Landing Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f7ece2;
      color: #6c4f3d;
    }

    .container {
      width: 90%;
      max-width: 1200px;
      margin: 0 auto;
    }

    .header {
      padding: 20px;
      text-align: left;
    }

    .title {
      font-size: 48px;
      font-weight: bold;
      color: #8c7a5b;
    }

    .subtitle {
      font-size: 24px;
      margin-bottom: 20px;
    }

    .description {
      max-width: 500px;
      font-size: 16px;
      line-height: 1.5;
    }

    .links {
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="header">
      <h1>Furniture Landing Page</h1>
    </div>
    <div class="title">
      <h2>Explore Our Furniture Range</h2>
    </div>
    <div class="subtitle">
      <p>Discover the perfect furniture pieces for your home or office. Our range includes a variety of styles and materials to suit every taste and budget.</p>
    </div>
    <div class="description">
      <p>Our furniture is made from high-quality materials and is designed to be both functional and aesthetically pleasing. We offer a wide range of products, including chairs, desks, sofas, beds, and more. Whether you're looking for a new sofa for your living room or a desk for your office, we have something to meet your needs.</p>
    </div>
    <div class="links">
      <a href="#">View All Products</a>
      <a href="#">About Us</a>
      <a href="#">Contact Us</a>
    </div>
  </div>
</body>

```

# Application 1 - Adding Comments

The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the following details:

- File Explorer:** Shows a project structure under "FACE\_RECOGNITION" containing files: Data, model.py, pairs.py, README.md, test.py, and train.py.
- Editor:** Displays a Python script named "model.py". The code defines a SiameseNetwork class that inherits from nn.Module. It contains methods for initializing the base model, performing forward passes on one input or two inputs, calculating the L1 distance between outputs, and applying a fully connected layer to the distance.
- GitHub Copilot Panel:** A floating panel on the right side of the editor interface. It has a header for "charurathour" and a section titled "Add comments" which lists "model.py:1-26". This section is highlighted with a red rectangle.
- Copilot Suggestion:** Below the "Add comments" section, there is a "GitHub Copilot" section stating "Used 1 reference". It shows a snippet of code with "Apply in Editor" and "Edit" buttons.
- Commenting Area:** The main body of the panel contains a partially generated code block for the "forward" method, starting with "import torch" and "import torch.nn as nn". It includes a "What comments should I add?" section and an "Ask Copilot" button.
- Status Bar:** At the bottom, the status bar shows the current file is "main\*", the line and column are "Ln 22, Col 43", and the encoding is "UTF-8". Other details include "Spaces: 4", "CRLF", "Python", "3.11.5 ('base': conda)", and the GitHub Copilot version "GPT 4o".

# Application 2 - Generating Explanation of Existing Code

EXPLORER ...

**FACE RECOGNITION**

- > Data
- model.py U
- pairs.py U
- 2K+
- README.md
- test.py 1
- train.py U

pairs.py X model.py U test.py D v ⏪ ⏴ ...

pairs.py > split\_and\_preprocess\_pairs > preprocess\_pairs

```

1 import os
2 import random
3 import numpy as np
4 from PIL import Image
5 from sklearn.model_selection import train_
6
7 def load_images_from_folder(folder):
8     images = []
9     for filename in os.listdir(folder):
10         img = Image.open(os.path.join(folder,
11                           filename))
12         if img is not None:
13             images.append(img)
14     return images
15
16 def create_pairs(main_directory):
17     positive_pairs = []
18     negative_pairs = []
19     labels = []
20
21     subfolders = [f.path for f in os.scandir(
22         main_directory) if f.is_dir()]
23     all_images = {os.path.basename(subfold
24
25     for person, images in all_images.items():
26         for i in range(len(images)):
27             for j in range(i + 1, len(imag
28                 positive_pairs.append((ima
29                 labels.append(1)
30
31     persons = list(all_images.keys())
32
33     pairs_train, pairs_temp, labels_train, labels_temp = train_test_split(
34         positive_pairs, labels, test_size=0.2, random_state=42)
35     pairs_val, pairs_test, labels_val, labels_test = train_test_split(pairs_
36
37     def preprocess_pairs(pairs):
38         . . .
39
40
41     def split_and_preprocess_pairs(pairs, labels):
42         pairs_train, pairs_temp, labels_train, labels_temp = train_test_split(
43             pairs, labels, test_size=0.2, random_state=42)
44         pairs_val, pairs_test, labels_val, labels_test = train_test_split(pairs_
45
46         . . .
47
48         return pairs_train, pairs_val, pairs_test, labels_train, labels_val, labels_
49
50
51     def train_model(pairs_train, labels_train):
52         . . .
53
54     def validate_model(pairs_val, labels_val):
55         . . .
56
57     def test_model(pairs_test, labels_test):
58         . . .
59
60
61     def evaluate_model(model, pairs_val, labels_val):
62         . . .
63
64
65     def predict(model, pairs):
66         . . .
67
68
69     def save_model(model, file_name):
70         . . .
71
72
73     def load_model(file_name):
74         . . .
75
76
77     def main():
78         . . .
79
80
81     if __name__ == "__main__":
82         main()
83
84
85     def preprocess_image(img):
86         . . .
87
88
89     def calculate_accuracy(labels, predictions):
90         . . .
91
92
93     def calculate_precision_recall_f1(labels, predictions):
94         . . .
95
96
97     def calculate_confusion_matrix(labels, predictions):
98         . . .
99
100
101     def calculate_roc_auc(labels, predictions):
102         . . .
103
104
105     def calculate_f1_score(labels, predictions):
106         . . .
107
108
109     def calculate_recall(labels, predictions):
110         . . .
111
112
113     def calculate_precision(labels, predictions):
114         . . .
115
116
117     def calculate_f1_score(labels, predictions):
118         . . .
119
120
121     def calculate_recall(labels, predictions):
122         . . .
123
124
125     def calculate_precision(labels, predictions):
126         . . .
127
128
129     def calculate_f1_score(labels, predictions):
130         . . .
131
132
133     def calculate_recall(labels, predictions):
134         . . .
135
136
137     def calculate_precision(labels, predictions):
138         . . .
139
140
141     def calculate_f1_score(labels, predictions):
142         . . .
143
144
145     def calculate_recall(labels, predictions):
146         . . .
147
148
149     def calculate_precision(labels, predictions):
150         . . .
151
152
153     def calculate_f1_score(labels, predictions):
154         . . .
155
156
157     def calculate_recall(labels, predictions):
158         . . .
159
160
161     def calculate_precision(labels, predictions):
162         . . .
163
164
165     def calculate_f1_score(labels, predictions):
166         . . .
167
168
169     def calculate_recall(labels, predictions):
170         . . .
171
172
173     def calculate_precision(labels, predictions):
174         . . .
175
176
177     def calculate_f1_score(labels, predictions):
178         . . .
179
180
181     def calculate_recall(labels, predictions):
182         . . .
183
184
185     def calculate_precision(labels, predictions):
186         . . .
187
188
189     def calculate_f1_score(labels, predictions):
190         . . .
191
192
193     def calculate_recall(labels, predictions):
194         . . .
195
196
197     def calculate_precision(labels, predictions):
198         . . .
199
200
201     def calculate_f1_score(labels, predictions):
202         . . .
203
204
205     def calculate_recall(labels, predictions):
206         . . .
207
208
209     def calculate_precision(labels, predictions):
210         . . .
211
212
213     def calculate_f1_score(labels, predictions):
214         . . .
215
216
217     def calculate_recall(labels, predictions):
218         . . .
219
220
221     def calculate_precision(labels, predictions):
222         . . .
223
224
225     def calculate_f1_score(labels, predictions):
226         . . .
227
228
229     def calculate_recall(labels, predictions):
230         . . .
231
232
233     def calculate_precision(labels, predictions):
234         . . .
235
236
237     def calculate_f1_score(labels, predictions):
238         . . .
239
240
241     def calculate_recall(labels, predictions):
242         . . .
243
244
245     def calculate_precision(labels, predictions):
246         . . .
247
248
249     def calculate_f1_score(labels, predictions):
250         . . .
251
252
253     def calculate_recall(labels, predictions):
254         . . .
255
256
257     def calculate_precision(labels, predictions):
258         . . .
259
260
261     def calculate_f1_score(labels, predictions):
262         . . .
263
264
265     def calculate_recall(labels, predictions):
266         . . .
267
268
269     def calculate_precision(labels, predictions):
270         . . .
271
272
273     def calculate_f1_score(labels, predictions):
274         . . .
275
276
277     def calculate_recall(labels, predictions):
278         . . .
279
280
281     def calculate_precision(labels, predictions):
282         . . .
283
284
285     def calculate_f1_score(labels, predictions):
286         . . .
287
288
289     def calculate_recall(labels, predictions):
290         . . .
291
292
293     def calculate_precision(labels, predictions):
294         . . .
295
296
297     def calculate_f1_score(labels, predictions):
298         . . .
299
300
301     def calculate_recall(labels, predictions):
302         . . .
303
304
305     def calculate_precision(labels, predictions):
306         . . .
307
308
309     def calculate_f1_score(labels, predictions):
310         . . .
311
312
313     def calculate_recall(labels, predictions):
314         . . .
315
316
317     def calculate_precision(labels, predictions):
318         . . .
319
320
321     def calculate_f1_score(labels, predictions):
322         . . .
323
324
325     def calculate_recall(labels, predictions):
326         . . .
327
328
329     def calculate_precision(labels, predictions):
330         . . .
331
332
333     def calculate_f1_score(labels, predictions):
334         . . .
335
336
337     def calculate_recall(labels, predictions):
338         . . .
339
340
341     def calculate_precision(labels, predictions):
342         . . .
343
344
345     def calculate_f1_score(labels, predictions):
346         . . .
347
348
349     def calculate_recall(labels, predictions):
350         . . .
351
352
353     def calculate_precision(labels, predictions):
354         . . .
355
356
357     def calculate_f1_score(labels, predictions):
358         . . .
359
360
361     def calculate_recall(labels, predictions):
362         . . .
363
364
365     def calculate_precision(labels, predictions):
366         . . .
367
368
369     def calculate_f1_score(labels, predictions):
370         . . .
371
372
373     def calculate_recall(labels, predictions):
374         . . .
375
376
377     def calculate_precision(labels, predictions):
378         . . .
379
380
381     def calculate_f1_score(labels, predictions):
382         . . .
383
384
385     def calculate_recall(labels, predictions):
386         . . .
387
388
389     def calculate_precision(labels, predictions):
390         . . .
391
392
393     def calculate_f1_score(labels, predictions):
394         . . .
395
396
397     def calculate_recall(labels, predictions):
398         . . .
399
400
401     def calculate_precision(labels, predictions):
402         . . .
403
404
405     def calculate_f1_score(labels, predictions):
406         . . .
407
408
409     def calculate_recall(labels, predictions):
410         . . .
411
412
413     def calculate_precision(labels, predictions):
414         . . .
415
416
417     def calculate_f1_score(labels, predictions):
418         . . .
419
420
421     def calculate_recall(labels, predictions):
422         . . .
423
424
425     def calculate_precision(labels, predictions):
426         . . .
427
428
429     def calculate_f1_score(labels, predictions):
430         . . .
431
432
433     def calculate_recall(labels, predictions):
434         . . .
435
436
437     def calculate_precision(labels, predictions):
438         . . .
439
440
441     def calculate_f1_score(labels, predictions):
442         . . .
443
444
445     def calculate_recall(labels, predictions):
446         . . .
447
448
449     def calculate_precision(labels, predictions):
450         . . .
451
452
453     def calculate_f1_score(labels, predictions):
454         . . .
455
456
457     def calculate_recall(labels, predictions):
458         . . .
459
460
461     def calculate_precision(labels, predictions):
462         . . .
463
464
465     def calculate_f1_score(labels, predictions):
466         . . .
467
468
469     def calculate_recall(labels, predictions):
470         . . .
471
472
473     def calculate_precision(labels, predictions):
474         . . .
475
476
477     def calculate_f1_score(labels, predictions):
478         . . .
479
480
481     def calculate_recall(labels, predictions):
482         . . .
483
484
485     def calculate_precision(labels, predictions):
486         . . .
487
488
489     def calculate_f1_score(labels, predictions):
490         . . .
491
492
493     def calculate_recall(labels, predictions):
494         . . .
495
496
497     def calculate_precision(labels, predictions):
498         . . .
499
500
501     def calculate_f1_score(labels, predictions):
502         . . .
503
504
505     def calculate_recall(labels, predictions):
506         . . .
507
508
509     def calculate_precision(labels, predictions):
510         . . .
511
512
513     def calculate_f1_score(labels, predictions):
514         . . .
515
516
517     def calculate_recall(labels, predictions):
518         . . .
519
520
521     def calculate_precision(labels, predictions):
522         . . .
523
524
525     def calculate_f1_score(labels, predictions):
526         . . .
527
528
529     def calculate_recall(labels, predictions):
530         . . .
531
532
533     def calculate_precision(labels, predictions):
534         . . .
535
536
537     def calculate_f1_score(labels, predictions):
538         . . .
539
540
541     def calculate_recall(labels, predictions):
542         . . .
543
544
545     def calculate_precision(labels, predictions):
546         . . .
547
548
549     def calculate_f1_score(labels, predictions):
550         . . .
551
552
553     def calculate_recall(labels, predictions):
554         . . .
555
556
557     def calculate_precision(labels, predictions):
558         . . .
559
560
561     def calculate_f1_score(labels, predictions):
562         . . .
563
564
565     def calculate_recall(labels, predictions):
566         . . .
567
568
569     def calculate_precision(labels, predictions):
570         . . .
571
572
573     def calculate_f1_score(labels, predictions):
574         . . .
575
576
577     def calculate_recall(labels, predictions):
578         . . .
579
580
581     def calculate_precision(labels, predictions):
582         . . .
583
584
585     def calculate_f1_score(labels, predictions):
586         . . .
587
588
589     def calculate_recall(labels, predictions):
590         . . .
591
592
593     def calculate_precision(labels, predictions):
594         . . .
595
596
597     def calculate_f1_score(labels, predictions):
598         . . .
599
600
601     def calculate_recall(labels, predictions):
602         . . .
603
604
605     def calculate_precision(labels, predictions):
606         . . .
607
608
609     def calculate_f1_score(labels, predictions):
610         . . .
611
612
613     def calculate_recall(labels, predictions):
614         . . .
615
616
617     def calculate_precision(labels, predictions):
618         . . .
619
620
621     def calculate_f1_score(labels, predictions):
622         . . .
623
624
625     def calculate_recall(labels, predictions):
626         . . .
627
628
629     def calculate_precision(labels, predictions):
630         . . .
631
632
633     def calculate_f1_score(labels, predictions):
634         . . .
635
636
637     def calculate_recall(labels, predictions):
638         . . .
639
640
641     def calculate_precision(labels, predictions):
642         . . .
643
644
645     def calculate_f1_score(labels, predictions):
646         . . .
647
648
649     def calculate_recall(labels, predictions):
650         . . .
651
652
653     def calculate_precision(labels, predictions):
654         . . .
655
656
657     def calculate_f1_score(labels, predictions):
658         . . .
659
660
661     def calculate_recall(labels, predictions):
662         . . .
663
664
665     def calculate_precision(labels, predictions):
666         . . .
667
668
669     def calculate_f1_score(labels, predictions):
670         . . .
671
672
673     def calculate_recall(labels, predictions):
674         . . .
675
676
677     def calculate_precision(labels, predictions):
678         . . .
679
680
681     def calculate_f1_score(labels, predictions):
682         . . .
683
684
685     def calculate_recall(labels, predictions):
686         . . .
687
688
689     def calculate_precision(labels, predictions):
690         . . .
691
692
693     def calculate_f1_score(labels, predictions):
694         . . .
695
696
697     def calculate_recall(labels, predictions):
698         . . .
699
700
701     def calculate_precision(labels, predictions):
702         . . .
703
704
705     def calculate_f1_score(labels, predictions):
706         . . .
707
708
709     def calculate_recall(labels, predictions):
710         . . .
711
712
713     def calculate_precision(labels, predictions):
714         . . .
715
716
717     def calculate_f1_score(labels, predictions):
718         . . .
719
720
721     def calculate_recall(labels, predictions):
722         . . .
723
724
725     def calculate_precision(labels, predictions):
726         . . .
727
728
729     def calculate_f1_score(labels, predictions):
730         . . .
731
732
733     def calculate_recall(labels, predictions):
734         . . .
735
736
737     def calculate_precision(labels, predictions):
738         . . .
739
740
741     def calculate_f1_score(labels, predictions):
742         . . .
743
744
745     def calculate_recall(labels, predictions):
746         . . .
747
748
749     def calculate_precision(labels, predictions):
750         . . .
751
752
753     def calculate_f1_score(labels, predictions):
754         . . .
755
756
757     def calculate_recall(labels, predictions):
758         . . .
759
760
761     def calculate_precision(labels, predictions):
762         . . .
763
764
765     def calculate_f1_score(labels, predictions):
766         . . .
767
768
769     def calculate_recall(labels, predictions):
770         . . .
771
772
773     def calculate_precision(labels, predictions):
774         . . .
775
776
777     def calculate_f1_score(labels, predictions):
778         . . .
779
780
781     def calculate_recall(labels, predictions):
782         . . .
783
784
785     def calculate_precision(labels, predictions):
786         . . .
787
788
789     def calculate_f1_score(labels, predictions):
790         . . .
791
792
793     def calculate_recall(labels, predictions):
794         . . .
795
796
797     def calculate_precision(labels, predictions):
798         . . .
799
800
801     def calculate_f1_score(labels, predictions):
802         . . .
803
804
805     def calculate_recall(labels, predictions):
806         . . .
807
808
809     def calculate_precision(labels, predictions):
810         . . .
811
812
813     def calculate_f1_score(labels, predictions):
814         . . .
815
816
817     def calculate_recall(labels, predictions):
818         . . .
819
820
821     def calculate_precision(labels, predictions):
822         . . .
823
824
825     def calculate_f1_score(labels, predictions):
826         . . .
827
828
829     def calculate_recall(labels, predictions):
830         . . .
831
832
833     def calculate_precision(labels, predictions):
834         . . .
835
836
837     def calculate_f1_score(labels, predictions):
838         . . .
839
840
841     def calculate_recall(labels, predictions):
842         . . .
843
844
845     def calculate_precision(labels, predictions):
846         . . .
847
848
849     def calculate_f1_score(labels, predictions):
850         . . .
851
852
853     def calculate_recall(labels, predictions):
854         . . .
855
856
857     def calculate_precision(labels, predictions):
858         . . .
859
860
861     def calculate_f1_score(labels, predictions):
862         . . .
863
864
865     def calculate_recall(labels, predictions):
866         . . .
867
868
869     def calculate_precision(labels, predictions):
870         . . .
871
872
873     def calculate_f1_score(labels, predictions):
874         . . .
875
876
877     def calculate_recall(labels, predictions):
878         . . .
879
880
881     def calculate_precision(labels, predictions):
882         . . .
883
884
885     def calculate_f1_score(labels, predictions):
886         . . .
887
888
889     def calculate_recall(labels, predictions):
890         . . .
891
892
893     def calculate_precision(labels, predictions):
894         . . .
895
896
897     def calculate_f1_score(labels, predictions):
898         . . .
899
900
901     def calculate_recall(labels, predictions):
902         . . .
903
904
905     def calculate_precision(labels, predictions):
906         . . .
907
908
909     def calculate_f1_score(labels, predictions):
910         . . .
911
912
913     def calculate_recall(labels, predictions):
914         . . .
915
916
917     def calculate_precision(labels, predictions):
918         . . .
919
920
921     def calculate_f1_score(labels, predictions):
922         . . .
923
924
925     def calculate_recall(labels, predictions):
926         . . .
927
928
929     def calculate_precision(labels, predictions):
930         . . .
931
932
933     def calculate_f1_score(labels, predictions):
934         . . .
935
936
937     def calculate_recall(labels, predictions):
938         . . .
939
940
941     def calculate_precision(labels, predictions):
942         . . .
943
944
945     def calculate_f1_score(labels, predictions):
946         . . .
947
948
949     def calculate_recall(labels, predictions):
950         . . .
951
952
953     def calculate_precision(labels, predictions):
954         . . .
955
956
957     def calculate_f1_score(labels, predictions):
958         . . .
959
960
961     def calculate_recall(labels, predictions):
962         . . .
963
964
965     def calculate_precision(labels, predictions):
966         . . .
967
968
969     def calculate_f1_score(labels, predictions):
970         . . .
971
972
973     def calculate_recall(labels, predictions):
974         . . .
975
976
977     def calculate_precision(labels, predictions):
978         . . .
979
980
981     def calculate_f1_score(labels, predictions):
982         . . .
983
984
985     def calculate_recall(labels, predictions):
986         . . .
987
988
989     def calculate_precision(labels, predictions):
990         . . .
991
992
993     def calculate_f1_score(labels, predictions):
994         . . .
995
996
997     def calculate_recall(labels, predictions):
998         . . .
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1
```

# Application 3 - Code Review

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Explorer View:** Shows a project structure under "PROJECTS" with files like test.py, routes.py, \_init\_.py, etc.
- Code Editor:** The file routes.py is open, displaying Python code for a Flask application. The code includes imports for flask, app, transformers, and torch, and defines routes for index and chat.
- Terminal:** The terminal shows a new environment (new\_env) at C:\Users\Charu\Desktop\Projects>.
- GitHub Copilot Panel:** A sidebar on the right contains a message from user charurathour: "review this code according to PEP 8". Below it, GitHub Copilot provides a review of the code according to PEP 8 guidelines, listing four sections: Imports, Line Length, Blank Lines, and Comments.
- Bottom Right Panel:** A "Ask Copilot" panel is open, showing the current file is routes.py. It also displays a GPT 4o interface with a message input field and other controls.

# Application 4 - Debugging

The screenshot shows a VS Code interface with the following details:

- Explorer:** Shows a project structure with a file tree. The `requirements.txt` file is selected.
- Terminal:** Displays the following error message:
 

```
cli.py", line 260, in locate_app
    __import__(module_name)
File "C:\Users\Charu\Desktop\Projects\flask-chatbot\app\__init__.py", line
ne 7, in <module>
    from app import routes
File "C:\Users\Charu\Desktop\Projects\flask-chatbot\app\routes.py", line
3, in <module>
    from transformers import AutoModelForCausalLM, AutoTokenizer
ModuleNotFoundError: No module named 'transformers'
```
- GitHub Copilot:** A sidebar panel provides a solution:
 

```
"cli.py", line 260, in locate_app __import__(module_name) File "C:\Users\Charu\Desktop\Projects\flask-chatbot\app\__init__.py", line 7, in <module> from app import routes File "C:\Users\Charu\Desktop\Projects\flask-chatbot\app\routes.py", line 3, in <module> from transformers import AutoModelForCausalLM, AutoTokenizer ModuleNotFoundError: No module named 'transformers'
```

Explain this error and how it can be solved
- Bottom Status Bar:** Shows the current environment as `(chatbot_env) C:\Users\Charu\Desktop\Projects\flask-chatbot>`, file `requirements.txt`, and other status indicators like line 5, column 6, and encoding.

# Application 5 - Language Conversion

The screenshot shows a Visual Studio Code (VS Code) interface with two code editors and a terminal.

**Left Editor (Python):**

```

1  class AddTwoNumbers:
2
3      # Function to add two numbers
4      @staticmethod
5      def add_numbers(num1, num2):
6          return num1 + num2
7
8      # Main method to test the function
9      if __name__ == "__main__":
10         number1 = 10 # First number
11         number2 = 20 # Second number
12
13     # Call the add_numbers function and print the result
14     sum = AddTwoNumbers.add_numbers(number1, number2)
15     print(f"The sum of {number1} and {number2} is: {sum}")
16

```

**Right Editor (Java):**

```

public class AddTwoNumbers {
    // Function to add two numbers
    public static int addNumbers(int num1, int num2) {
        return num1 + num2;
    }

    // Main method to test the function
    public static void main(String[] args) {
        int number1 = 10; // First number
        int number2 = 20; // Second number

        // Call the addNumbers function and print the result
        int sum = addNumbers(number1, number2);
        System.out.println("The sum of " + number1 + " and " + number2 + " is: " + sum);
    }
}

```

A red box highlights the Java code and the AI-generated conversion instructions below it:

- convert this function in python
- test.py:1-2
- How do I run this code?
- Ask Copilot
- test.py Current file
- GPT 4o

**Terminal:**

```
(chatbot_env) C:\Users\Charu\Desktop\Projects\flask-chatbot>
```

**Sidebar:**

- EXPLORER
- PROJECTS
  - > flask-calculator-app
  - < flask-chatbot
    - < app
    - > \_pycache\_
    - > static
    - > templates
    - < \_init\_.py
    - < routes.py
- README.md
- requirements.txt
- test.py

**Bottom Bar:**

- PROBLEMS
- OUTPUT
- DEBUG CONSOLE
- TERMINAL
- ...
- + < > ^ X
- powershell
- cmd
- powershell

# Application 6 - Creation of Workspace with Copilot

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Projects
- Sidebar:** EXPLORER, PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, OUTLINE (1), TIMELINE.
- Terminal:** (base) C:\Users\Charu\Desktop\Projects>
- Copilot Panel:** Shows a message from user "charurathour": "i want to create a simple chatbot interface with flask".
  - Workspace Suggestion:** "Workspace used /new (rerun without)"
  - Directory Structure Suggestion:** flask-chatbot (app, static, templates, venv, requirements.txt, README.md)
  - Create Workspace...** button
  - Ask Copilot:** What should I include in requirements.txt?
  - Input Fields:** @, U, Q
  - Bottom Right:** GPT 4o, >, <

# Use Case - Building a Chatbot Application with GitHub Copilot

## Step 1: Setting Goals and Structuring Your Workflow

### Set Your Goal

- Define what you want to build.
- Example: "I want to create a chatbot for answering basic user queries."
- Be specific about the problem your application will solve.

### Identify Requirements

- What does your application need?
- Input: What will the user provide?
  - Example: Text queries in a chat window.
- Output: What will the app deliver?
  - Example: AI-generated text responses.

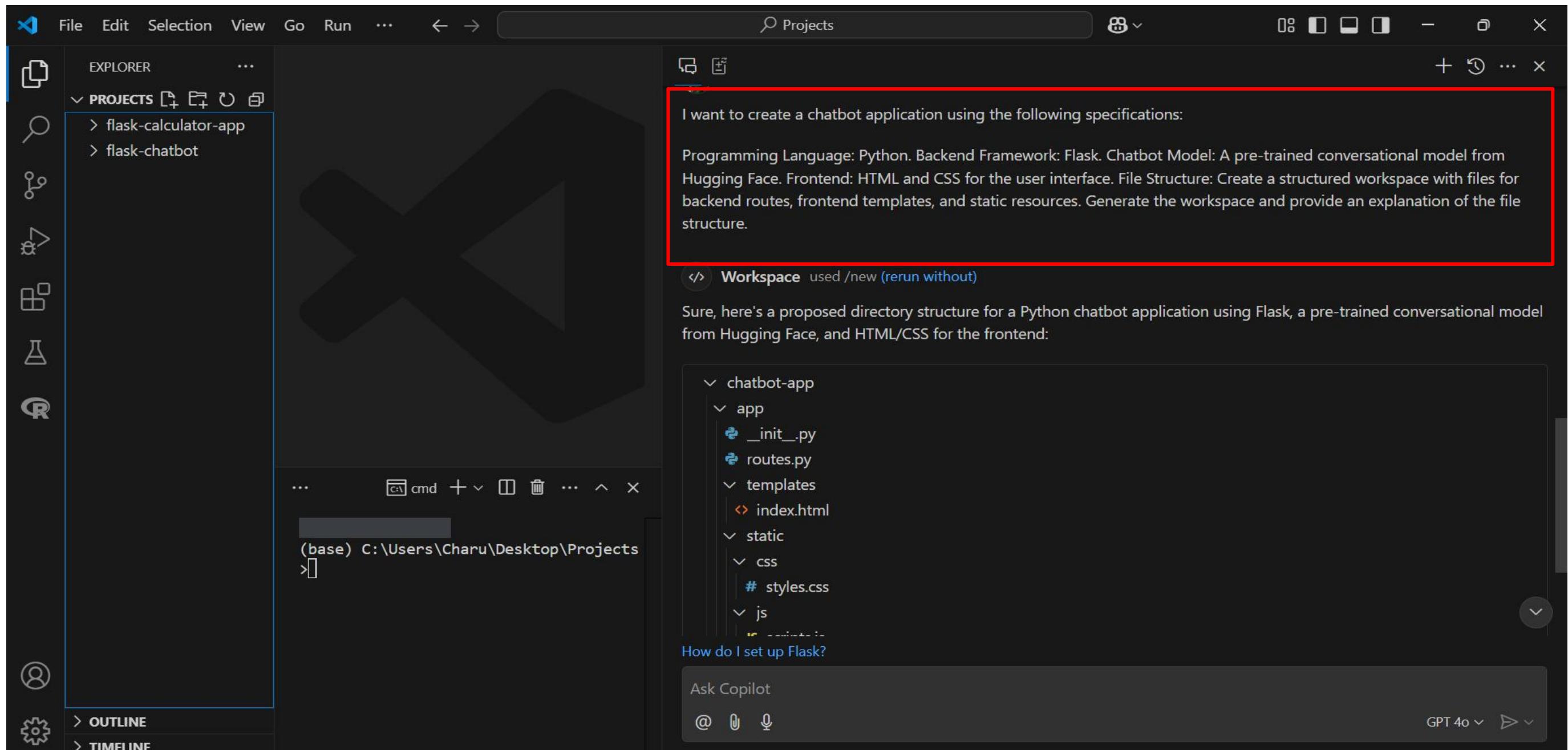
### Decide on the Tech Stack

- Language: Choose the programming language.  
Example: Python for backend and AI models.
- Libraries and Frameworks: List what you'll need.  
Example: Flask for backend, Transformers from Hugging Face for chatbot logic, HTML/JavaScript for frontend.
- Backend: Will you use Flask/Django/Node.js, or something else?
- Frontend: Web-based interface (HTML/JS) or other frameworks like React.

You can use Copilot chat to research and ideate for setting up goal

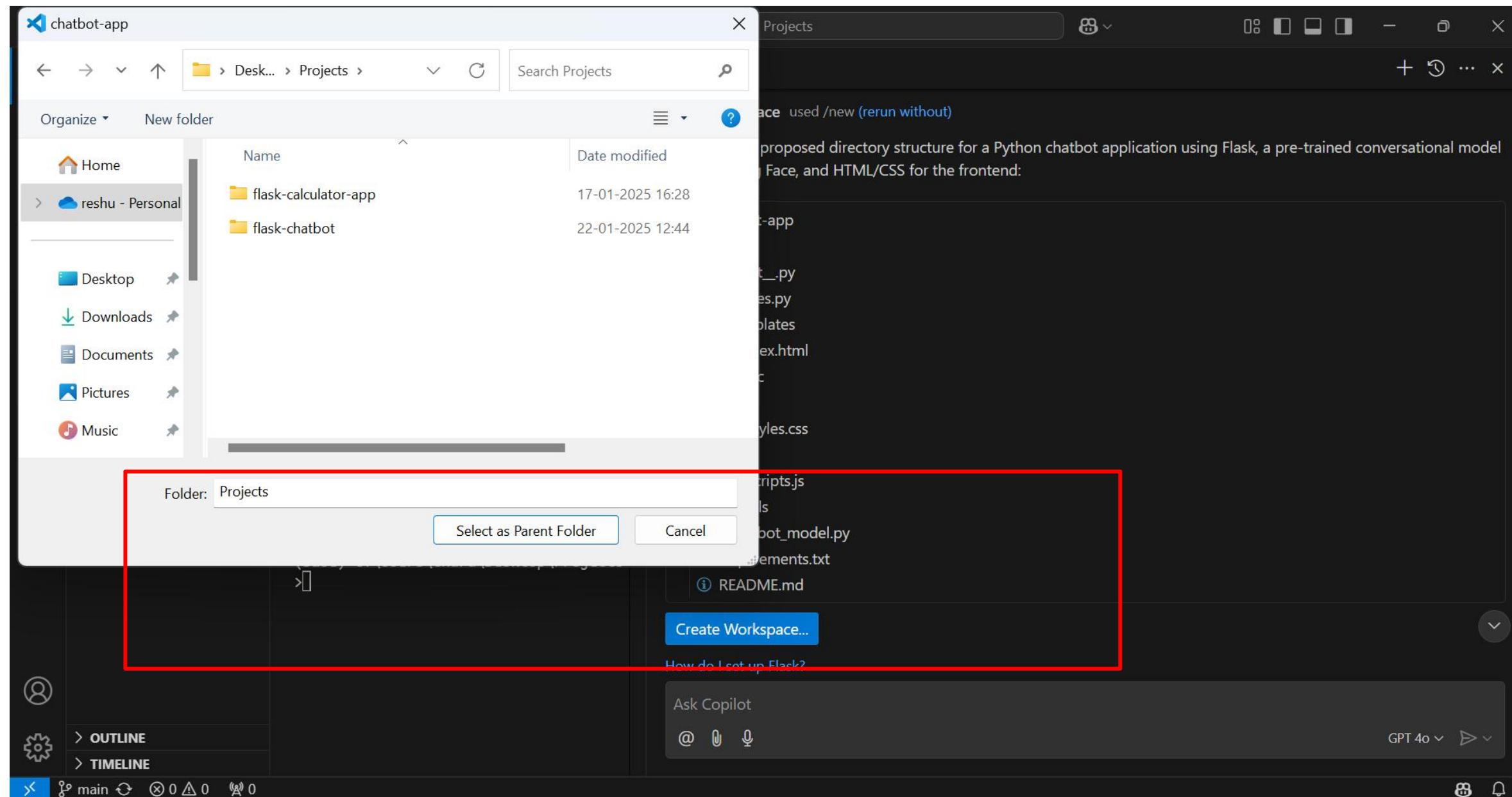
# Use Case - Building a Chatbot Application with GitHub Copilot

Step 2 : Use Copilot to create workspace by defining your goal and file structure in copilot chat area



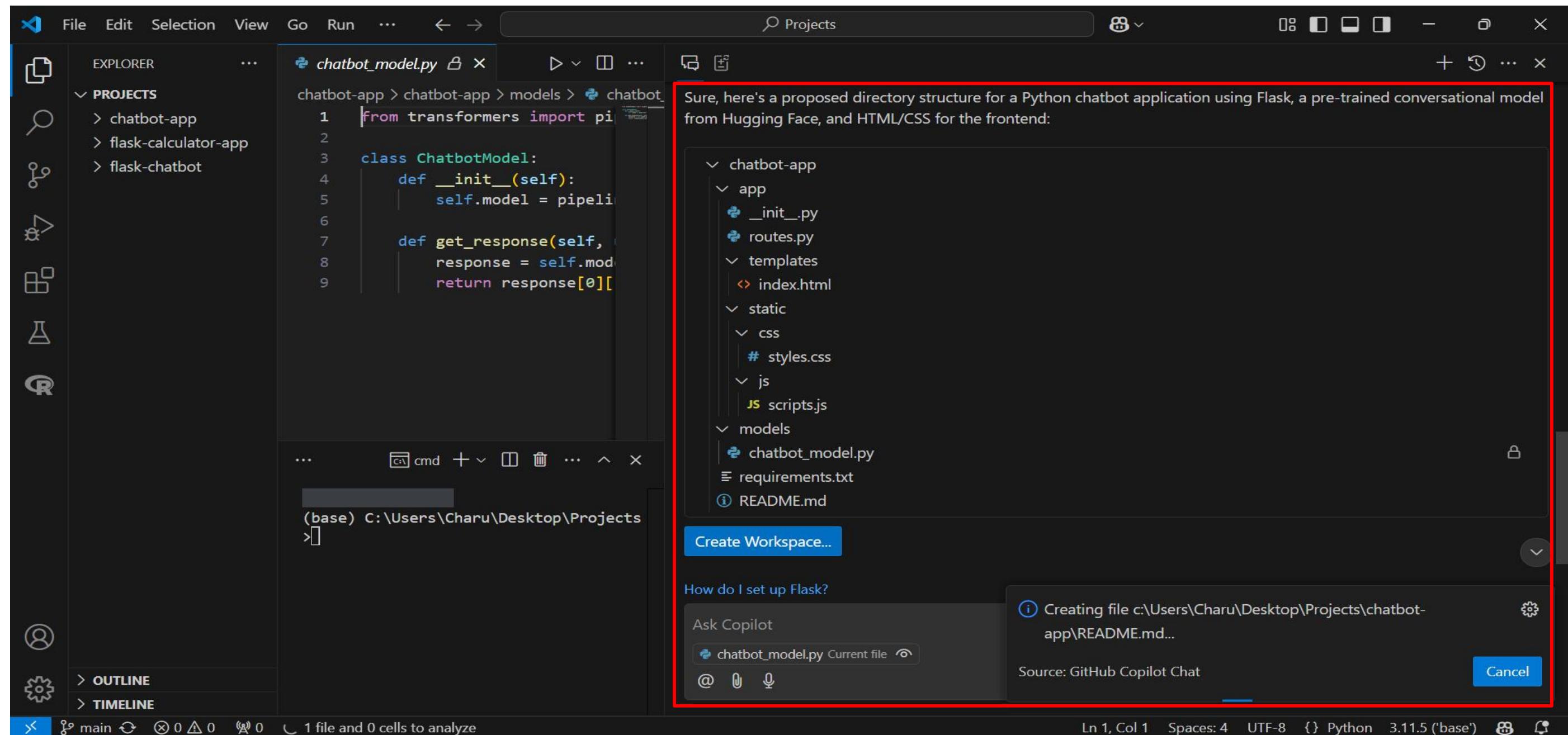
# Use Case - Building a Chatbot Application with GitHub Copilot

Step 2 : Use Copilot to create workspace by defining your goal and file structure in copilot chat area



# Use Case - Building a Chatbot Application with GitHub Copilot

Step 2 : Use Copilot to create workspace by defining your goal and file structure in copilot chat area



# Use Case - Building a Chatbot Application with GitHub Copilot

## Step 3 : Create Environment and install dependencies from requirements file generated by copilot

The screenshot shows a Visual Studio Code (VS Code) interface with a dark theme. On the left is the Explorer sidebar, which lists a project structure under 'chatbot-app'. The 'requirements.txt' file is selected in the Explorer. The main area shows the contents of 'requirements.txt':

```
1 Flask==2.2.2
2 transformers==4.21.1
3 torch==1.12.1
4 gunicorn==20.1.0
5 requests==2.26.0
```

Below the code editor is the Terminal tab, which displays the command-line output of the pip installation process:

```
(base) C:\Users\Charu\Desktop\Projects\chatbot-app>conda activate chatbot_demo_env
(chatbot_demo_env) C:\Users\Charu\Desktop\Projects\chatbot-app>pip install -r requirements.txt
Collecting Flask==2.2.2 (from -r requirements.txt (line 1))
  Using cached Flask-2.2.2-py3-none-any.whl.metadata (3.9 kB)
Collecting transformers==4.21.1 (from -r requirements.txt (line 2))
  Using cached transformers-4.21.1-py3-none-any.whl.metadata (81 kB)
Collecting torch==1.12.1 (from -r requirements.txt (line 3))
  Downloading torch-1.12.1-cp39-cp39-win_amd64.whl.metadata (22 kB)
Collecting gunicorn==20.1.0 (from -r requirements.txt (line 4))
  Downloading gunicorn-20.1.0-py3-none-any.whl.metadata (3.8 kB)
Collecting requests==2.26.0 (from -r requirements.txt (line 5))
  Using cached requests-2.26.0-py2.py3-none-any.whl.metadata (4.8 kB)
Collecting Werkzeug>=2.2.2 (from Flask==2.2.2->-r requirements.txt (line 1))
  Using cached werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.0 (from Flask==2.2.2->-r requirements.txt (line 1))
  Using cached jinja2-3.1.5-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.0 (from Flask==2.2.2->-r requirements.txt (line 1))
  Using cached itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.0 (from Flask==2.2.2->-r requirements.txt (line 1))
```

A red box highlights the terminal output. In the bottom right corner of the terminal, there is a 'Create Workspace...' button and a 'How do I set up Flask?' help section. A 'Ask Copilot' input field contains the text 'requirements.txt Current file'.

# Use Case - Building a Chatbot Application with GitHub Copilot

## Step 4 : Review code and make required changes with copilot

The screenshot shows a Visual Studio Code (VS Code) interface with a dark theme. On the left is the Explorer sidebar showing project files like `chatbot-app`, `app`, `models`, and `run.py`. The `routes.py` file is selected in the Explorer and is currently open in the main editor area. A red box highlights the code in the editor.

```
from flask import Flask, request, jsonify, render_template, Blueprint
from transformers import pipeline

app = Flask(__name__)
main = Blueprint('main', __name__)

# Load the pre-trained conversational model
chatbot = pipeline('conversational', model='microsoft/DialoGPT-large')

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.json.get('message')
    response = chatbot(user_input)
    return jsonify({'response': response[0]['generated_text']})
```

To the right of the editor, there is a panel titled "2. Set up routes: Create the `routes.py` file to define the routes for the Flask app." It contains two snippets of code, one above the other, which are identical to the code in the editor. Below this panel is another section titled "Ask Copilot" with a text input field containing "`routes.py Current file`".

# Use Case - Building a Chatbot Application with GitHub Copilot

Step 5 : Run code and Debug with help of copilot if any errors come

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a project structure with files like `__init__.py`, `index.html`, `run.py`, and `styles.css`.
- Code Editor:** Displays `run.py` content:

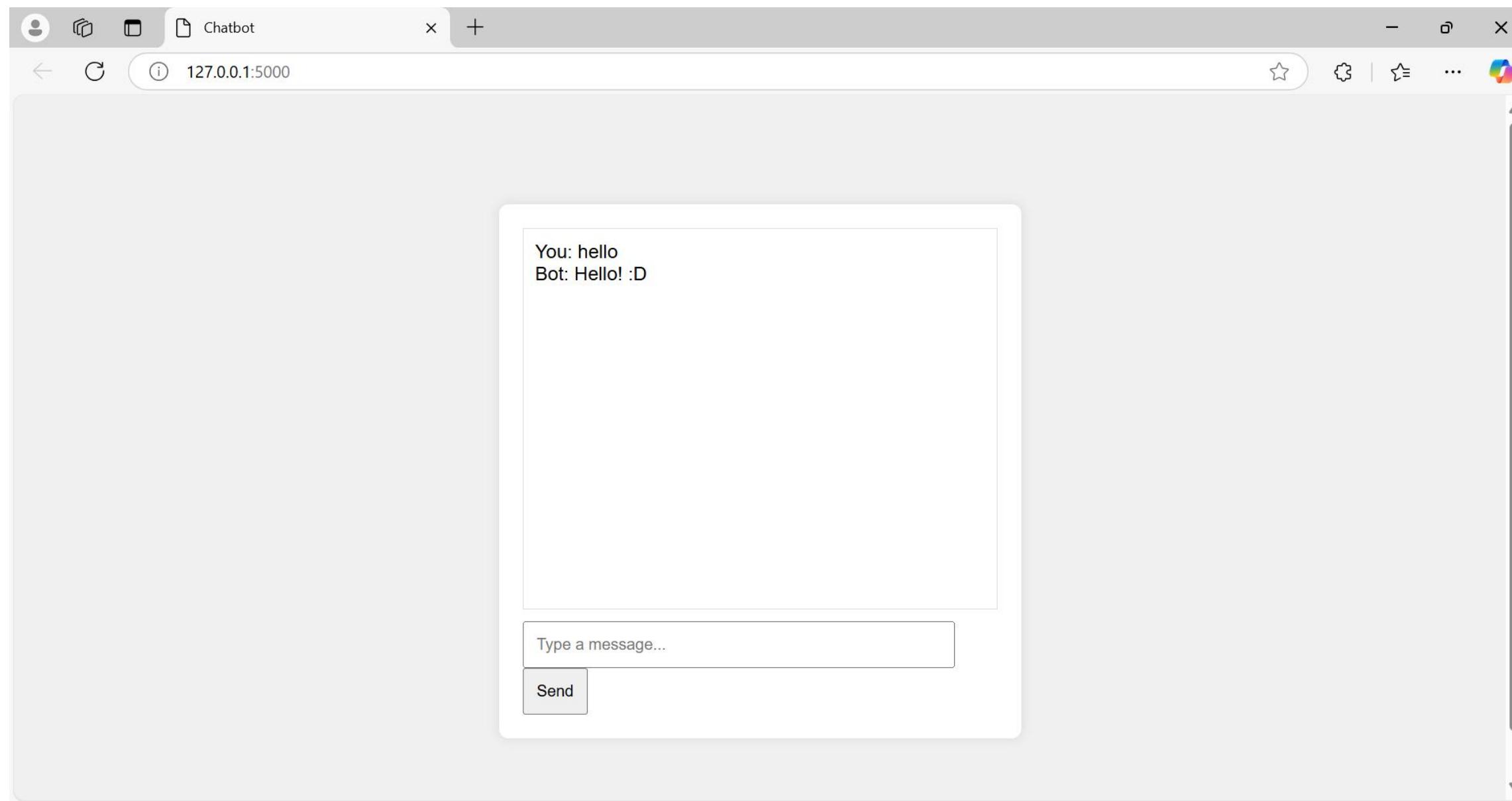
```
# filepath: /C:/Users/Charu/Desktop/Projects/chatbot-app/run.py
from app import create_app

app = create_app()

if __name__ == '__main__':
    app.run(debug=True)
```
- Terminal:** Shows a stack trace of an `ImportError` related to `url_quote` from `werkzeug.urls`.
- GitHub Copilot Panel:** A red box highlights the right-hand panel which provides a solution for the error. It includes:
  - The error message: `(C:\Users\Charu\anaconda3\envs\chatbot_demo_env\lib\site-packages\werkzeug\urls.py)`
  - A link to explain the error and suggest a solution.
  - A section titled "GitHub Copilot" with a note about using 1 reference.
  - An explanation of the issue: "The error you're encountering is due to an incompatibility between the versions of Flask and Werkzeug. The `url_quote` function has been removed in recent versions of Werkzeug."
  - A "Solution 1: Downgrade Werkzeug" section suggesting to downgrade Werkzeug or update Flask.
  - A note to update `requirements.txt` to specify a compatible Werkzeug version.
  - An "Ask Copilot" input field where the user can type questions.

# Use Case - Building a Chatbot Application with GitHub Copilot

Step 6 : Your basic interface is ready now you can iterate and make changes and update with copilot



# Use Case - Building a Chatbot Application with GitHub Copilot

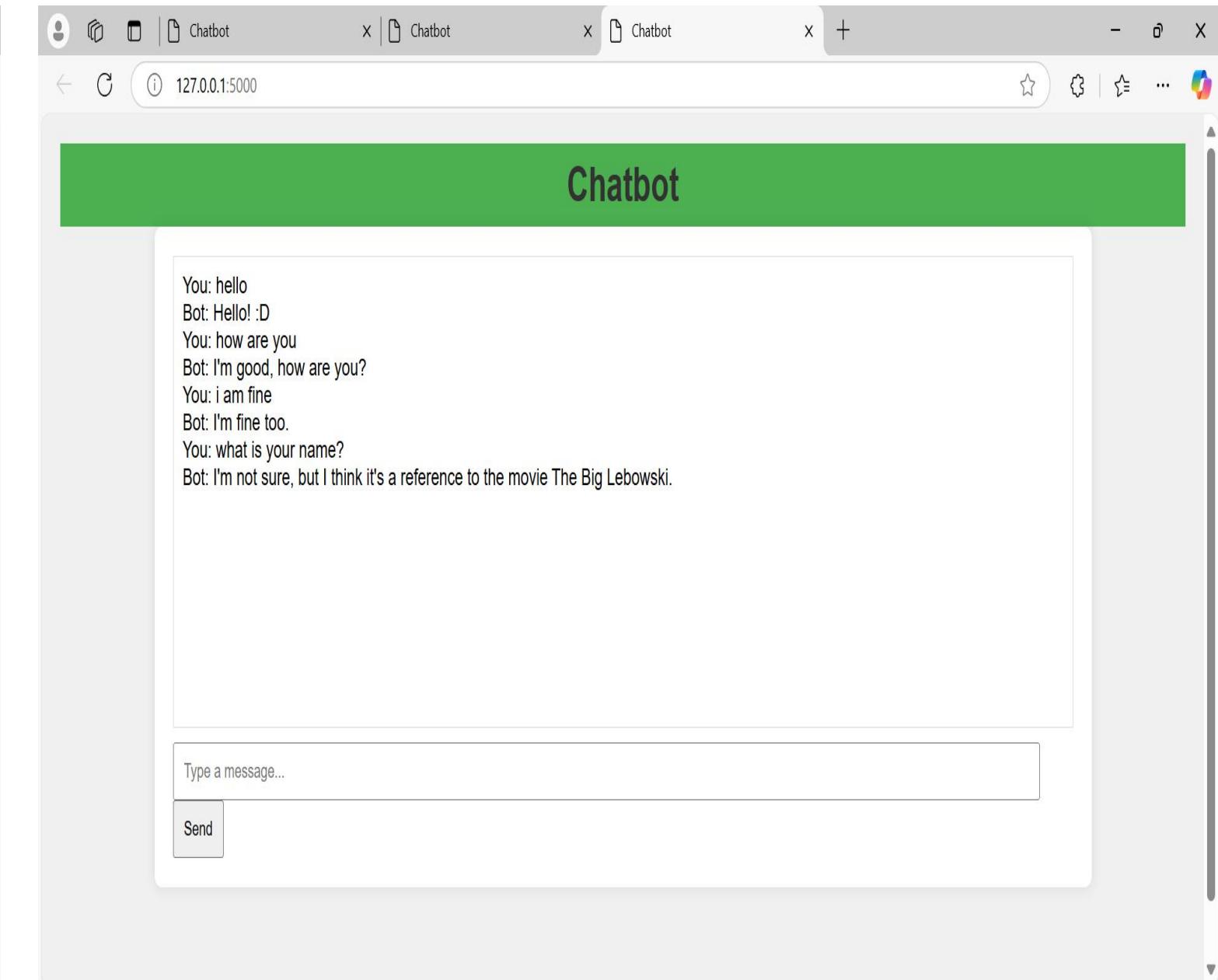
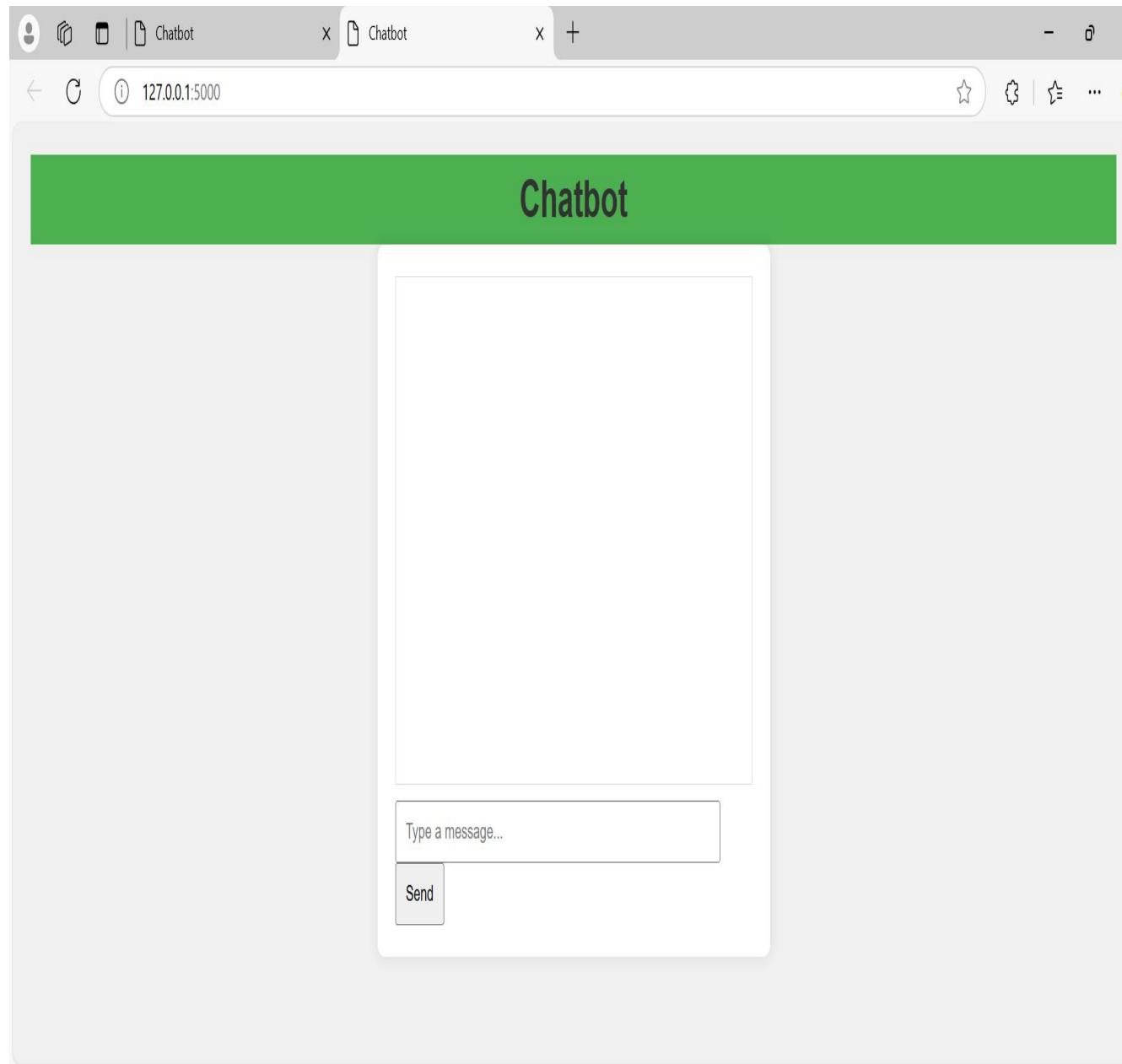
Step 6 : Iterate and make changes and upgrade your application with copilot

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a project structure for "chatbot-app" with files like `__init__.py`, `routes.py`, `index.html`, `styles.css`, and `chatbot_model.py`.
- Code Editor:** Displays Python code for a Flask application. A specific line of code, `response = chatbot.get_`, is highlighted with a tooltip from GitHub Copilot suggesting to "Accept" or "Discard" the completion.
- Terminal:** Shows the command `(chatbot_demo_env) C:\Users\Charu\Desktop\Projects\chatbot-app>`.
- GitHub Copilot Panel:** On the right, there are three suggestions:
  - "update interface by adding header of green color and heading Chatbot" (targeting `index.html`)
  - "GitHub Copilot" (targeting `index.html`)
  - "Add a header with a green background color and a heading "Chatbot"."
- Working Set:** A sidebar lists files with their status: `index.html` (Accept All Edits), `__init__.py`, `routes.py`, `scripts.js`, `run.py`, and `chatbot_model.py`.
- Bottom Status Bar:** Shows file information: Line 7, Column 27, Spaces: 4, UTF-8, LF, Python 3.11.5 ('base').

# Use Case - Building a Chatbot Application with GitHub Copilot

Step 6 : Iterate and make changes and update with copilot and update your code to Github side by side



# Conclusion: AI Empowering Developers

- **Time-Saving:** AI automates repetitive tasks like code generation, debugging, and documentation, reducing development time significantly.
- **Focus on Creativity:** With routine tasks handled by AI, developers can dedicate more time to innovative and complex problem-solving.
- **Enhanced Productivity:** By streamlining workflows, AI helps developers build faster, more reliable, and scalable software solutions.
- **Future of Coding:** AI tools are transforming development into a more creative and efficient process.





The future of coding isn't just about AI writing code—  
it's about how we, as developers, guide it.

Embrace AI to focus on what truly matters—innovative  
solutions and impactful applications.