

GitHub Copilot



<https://copilot.github.com/>

Powered by



Trained on billions of lines of public code, GitHub Copilot puts the knowledge you need at your fingertips, saving you time and helping you stay focused.



Extends your editor

GitHub Copilot is available today as a Visual Studio Code extension. It works wherever Visual Studio Code works — on your machine or in the cloud on [GitHub Codespaces](#). And it's fast enough to use as you type.



Speaks all the languages you love

GitHub Copilot works with a broad set of frameworks and languages. The technical preview does especially well for Python, JavaScript, TypeScript, Ruby, and Go, but it understands dozens of languages and can help you find your way around almost anything.

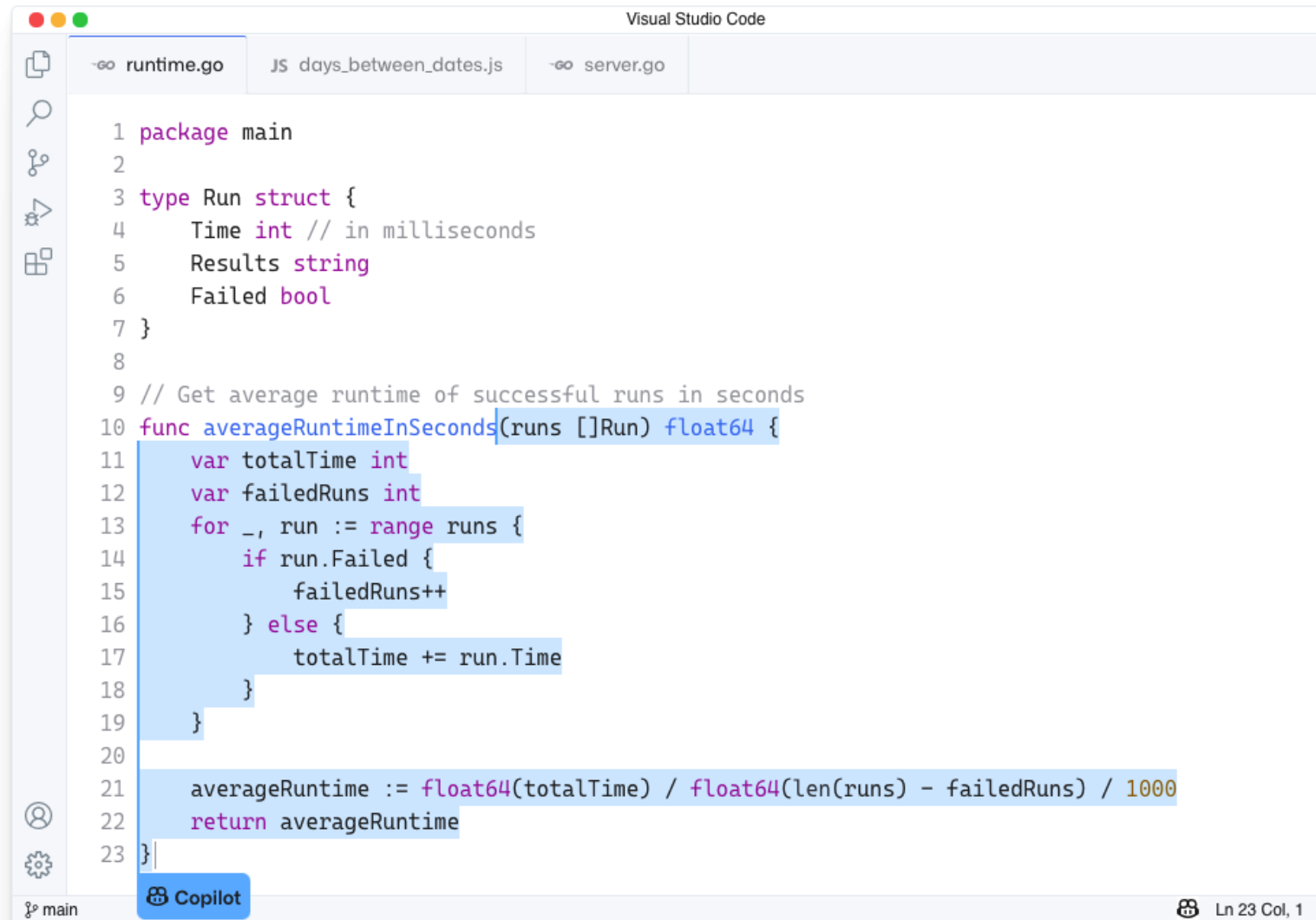


You're the pilot

With GitHub Copilot, you're always in charge. You can cycle through alternative suggestions, choose which to accept or reject, and manually edit suggested code. GitHub Copilot adapts to the edits you make, matching your coding style.

More than autocomplete

GitHub Copilot is powered by Codex, the new AI system created by OpenAI. GitHub Copilot understands significantly more context than most code assistants. So, whether it's in a docstring, comment, function name, or the code itself, GitHub Copilot uses the context you've provided and synthesizes code to match. Together with OpenAI, we're designing GitHub Copilot to get smarter at producing safe and effective code as developers use it.



Convert comments to code. Write a comment describing the logic you want, and let GitHub Copilot assemble the code for you.

```
collaborators.ts  get_repositories.py  JS non_alt_images.js

1 /**
2  * json schema:
3  * [
4  *   { name: "John Doe",
5  *     collaborators: ["Jane Doe", "Herbert Frapp", "Elsie McEwan"]
6  * },
7  *   { name: "Jane Doe",
8  *     collaborators: ["John Doe", "Karen Smith"]
9  * },
10 *   { name: "Skittles the Cat",
11 *     collaborators: []
12 * }
13 * ]
14 */
15 function collaborators_map(json: any): Map<string, Set<string>> {
16   const map = new Map<string, Set<string>>();
17   for (const item of json) {
18     const name = item.name;
19     const collaborators = item.collaborators;
20     const set = new Set<string>(collaborators);
21     map.set(name, set);
22   }
23   return map;
24 }
```

comment_marker.ts

course.rb

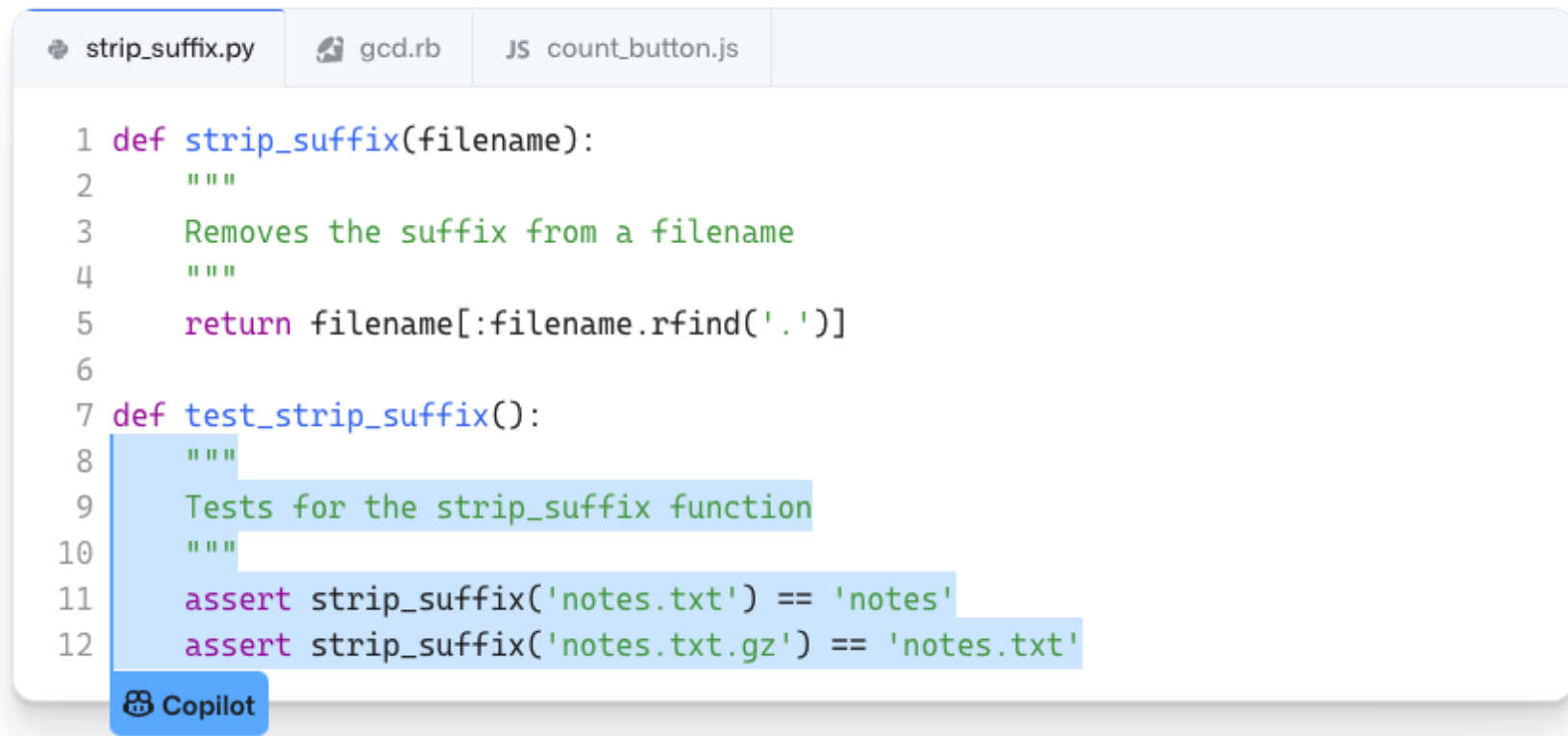
JS time.js

```
1 interface CommentMarker {
2   start: string;
3   end: string;
4 }
5
6 const markers: { [language:string]: CommentMarker } = {
7   javascript: { start: '//', end: '' },
8   python:      { start: '#', end: '' },
9   css:         { start: '/*', end: '*/' },
10  html:        { start: '<!--', end: '-->' },
11  bash:        { start: '#', end: '' },
12  postscript:  { start: '%', end: '' },
13  cobol:       { start: '*', end: '' },
14  asm:         { start: ';', end: '' },
15  haskell:     { start: '--', end: '' }
16 }
```

Copilot


Autofill for repetitive code.
GitHub Copilot works great for quickly producing boilerplate and repetitive code patterns. Feed it a few examples and let it generate the rest!

Tests without the toil. Tests are the backbone of any robust software engineering project. Import a unit test package, and let GitHub Copilot suggest tests that match your implementation code.



```
strip_suffix.py gcd.rb JS count_button.js

1 def strip_suffix(filename):
2     """
3     Removes the suffix from a filename
4     """
5     return filename[:filename.rfind('.')]
6
7 def test_strip_suffix():
8     """
9     Tests for the strip_suffix function
10    """
11    assert strip_suffix('notes.txt') == 'notes'
12    assert strip_suffix('notes.txt.gz') == 'notes.txt'
```

 Copilot

Show me alternatives. Want to evaluate a few different approaches? GitHub Copilot can show you a list of solutions. Use the code as provided, or edit it to meet your needs.



The screenshot shows the Visual Studio Code editor interface. The title bar at the top reads "Visual Studio Code". On the left is a sidebar with icons for Explorer, Search, Source Control, Run and Debug, Extensions, Accounts, and Settings. The main editor area displays a Python function:

```
1 def max_sum_slice(xs):  
2     max_ending = max_so_far = 0  
3     for x in xs:  
4         max_ending = max(0, max_ending + x)  
5         max_so_far = max(max_so_far, max_ending)  
6     return max_so_far
```

At the top of the editor, a GitHub Copilot suggestion bar is visible with the following controls: "Next" (with a right arrow icon), "Previous" (with a left arrow icon), and "Accept" (with a Tab key icon). The status bar at the bottom left shows "main" with a branch icon, and the bottom right shows the GitHub logo and "Ln 6 Col, 21".

Code confidently in unfamiliar territory

Whether you're working in a new language or framework, or just learning to code, GitHub Copilot can help you find your way. Tackle a bug, or learn how to use a new framework without spending most of your time spelunking through the docs or searching the web.

 Fetch tweets


 Draw a scatterplot


 Memoization


 Goodreads rating

JS fetch_tweets.js

 fetch_tweets.py

 fetch_tweets.rb

 fetch_tweets.ts

 fetch_tweets.go

```
1 import tweepy, os # secrets in environment variables
2
3 def fetch_tweets_from_user(user_name):
4     # authentication
5     auth = tweepy.OAuthHandler(os.environ['TWITTER_KEY'], os.environ['TWITTER_SECRET'])
6     auth.set_access_token(os.environ['TWITTER_TOKEN'], os.environ['TWITTER_TOKEN_SECRET'])
7     api = tweepy.API(auth)
8
9     # fetch tweets
10    tweets = api.user_timeline(screen_name=user, count=200, include_rts=False)
11    return tweets
```

 Copilot

Flight reports

Hundreds of engineers, including many of our own, have been using GitHub Copilot every day. It's transformed the way they work – here's what they have to say:

“

Trying to code in an unfamiliar language by googling everything is like navigating a foreign country with just a phrase book. Using GitHub Copilot is like hiring an interpreter.

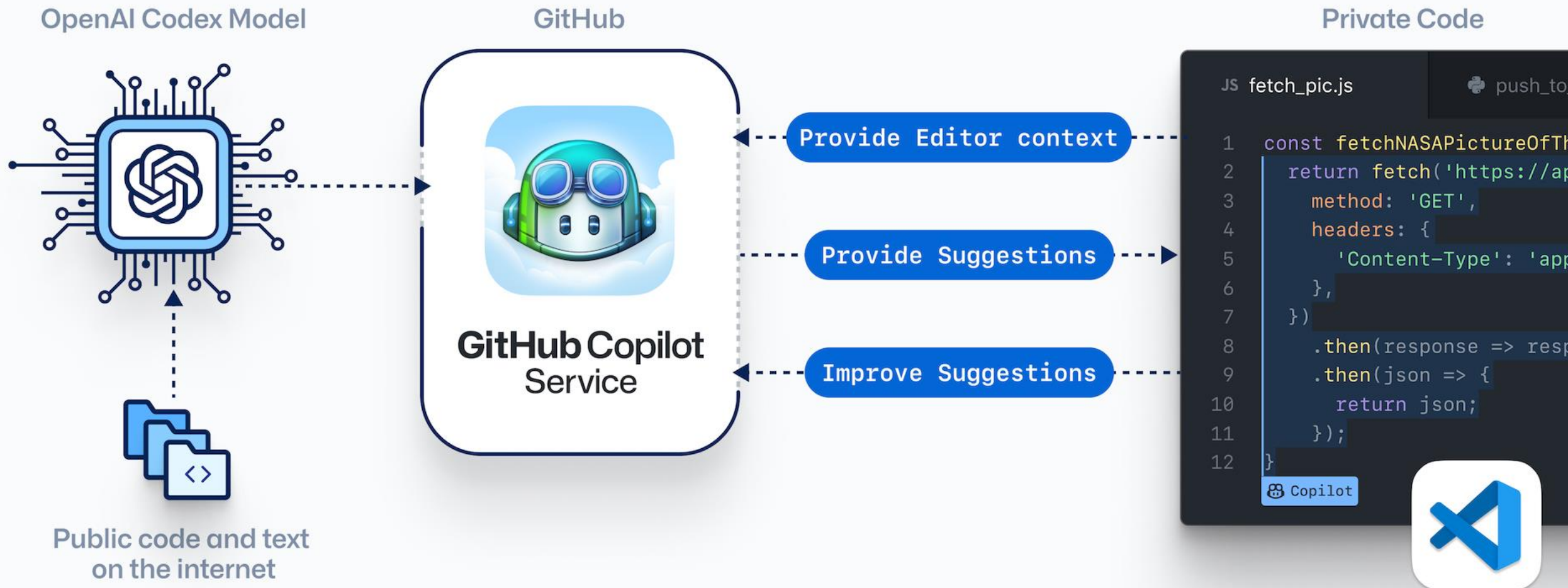
Harri Edwards // OpenAI

“

It surprised me with how precisely it understood my comment and generated accurate suggestions. The ability to choose from 10 different suggestions was the cherry on top.

Gunnika Batra

How does it work?





Join the GitHub Copilot waitlist

Access is limited to a small group of testers during the technical preview of [GitHub Copilot](#). Sign up today for your chance to try it out and help us improve.

[Sign in to join waitlist](#)



Join the GitHub Copilot waitlist

Access is limited to a small group of testers during the technical preview of [GitHub Copilot](#). Sign up today for your chance to try it out and help us improve.

How often do you use Visual Studio Code?

- ☐ Every day
- ☐ At least once a week
- ☐ Rarely

☐ I agree to these [additional telemetry terms](#) as part of the technical preview

[Join the waitlist](#)



GitHub Copilot

github.copilot

Preview

GitHub

113,872

★★★★☆

Repository

v1.1.1959

Your AI pair programmer

Disable

Uninstall



This extension is enabled globally.

[Details](#)[Feature Contributions](#)[Changelog](#)

GitHub Copilot

GitHub Copilot is an AI pair programmer which suggests line completions and entire function bodies as you type. GitHub Copilot is powered by the OpenAI Codex AI system, trained on public Internet text and billions of lines of code.

To learn more about GitHub Copilot, visit copilot.github.com.

Technical Preview

Access to GitHub Copilot is limited to a small group of testers during the technical preview of GitHub Copilot. If you don't have access to the technical preview, you will see an error when you try to use this extension.

Don't have access yet? [Sign up for the waitlist](#) for your chance to try it out. GitHub will notify you once you have access.

This technical preview is a Beta Preview under the [GitHub Terms of Service](#).

Usage

To get started with GitHub Copilot, visit the [Getting Started guide](#). Visit the [Preview repo](#) for more resources. This guide and repo is only available to users in the technical preview.

Troubleshooting

We'd love to get your help in making GitHub Copilot better! If you have feedback or encounter any problems, please reach out on the [Discussions forum](#).

1. Create a new JavaScript (.js) file.
2. Type the following function header:

```
function calculateDaysBetweenDates(begin, end) {
```

3. GitHub Copilot will automatically suggest an entire function body in grayed text, as shown below. The exact suggestion may vary.



The screenshot shows a code editor with a dark theme. At the top, a tab is labeled 'JS test.js' with a line number '1' and a white dot. Below the tab, the prompt 'JS test.js > calculateDaysBetweenDates' is visible. The first line of code is 'function calculateDaysBetweenDates(begin, end) {' with a cursor at the end of the opening brace. The rest of the function body is shown in a lighter gray font, indicating it is a suggestion. The suggested code includes variable declarations for 'beginDate' and 'endDate', a calculation for 'days' using 'Math.round', and a 'return days;' statement. The function is closed with a closing brace '}' on line 2.

```
JS test.js 1 ●
JS test.js > calculateDaysBetweenDates
1  function calculateDaysBetweenDates(begin, end) {
    var beginDate = new Date(begin);
    var endDate = new Date(end);
    var days = Math.round((endDate - beginDate) / (1000 * 60 * 60 * 24));
    return days;
  }
2
```

4. Press **Tab** to accept the suggestion.

3. Choosing alternate suggestions

For any given input, GitHub Copilot can provide multiple suggestions. As the developer you are always in charge; you can select which suggestion to use, or reject them all.

1. Clear the file (or start a new one), and type the following again:

```
function calculateDaysBetweenDates(begin, end) {
```

2. GitHub Copilot will again show you a suggested completion.

3. Instead of pressing `Tab` :

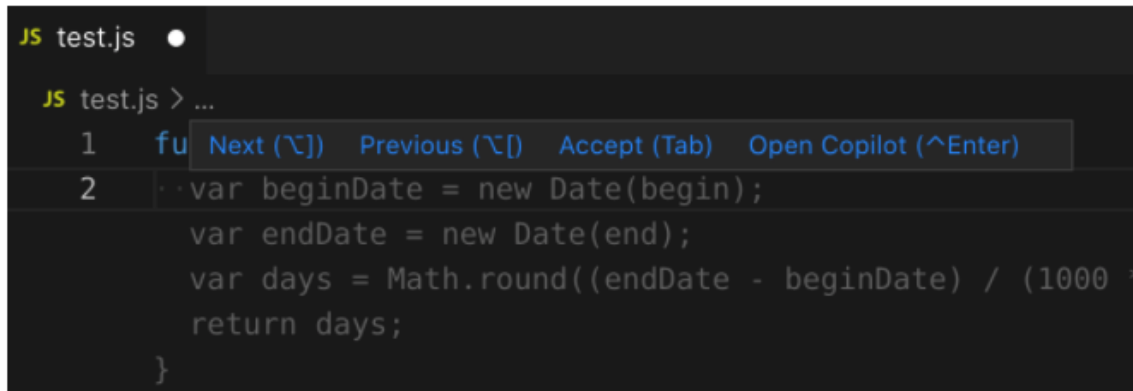
- On macOS, press `Option +]` (or `Option + [`).
- On Windows or Linux, press `Alt +]` (or `Alt + [`).

GitHub Copilot will cycle through other alternative suggestions.

4. When you see a suggestion you like, press `Tab` to accept it.

5. If you don't like any of the suggestions, press `Esc` .

You can also hover over a suggestion to see the GitHub Copilot command palette for choosing suggestions.



The screenshot shows a code editor with a file named 'test.js'. The code being edited is a JavaScript function 'calculateDaysBetweenDates'. A GitHub Copilot suggestion is visible, showing the function's completion. A command palette is open over the suggestion, displaying four actions: 'Next (\\])', 'Previous (\\[)', 'Accept (Tab)', and 'Open Copilot (^Enter)'.

```
JS test.js •
JS test.js > ...
1  fu
2  var beginDate = new Date(begin);
    var endDate = new Date(end);
    var days = Math.round((endDate - beginDate) / (1000 *
    return days;
}
```

Users > luisbeltran > Projects > copilotjs > C# test.cs

```
30 public double calculateReLu(double x)
31 {
32     if (x > 0)
33     {
34         return x;
35     }
36     else
37     {
38         return 0;
39     }
40 }
41
42 public double calculateSigmoid(double x)
43 {
44     return 1 / (1 + Math.Exp(-x));
45 }
46
47 public double calculateTanh(double x)
48 {
49     return Math.Tanh(x);
50 }
51
52 }
```

Ctrl+Enter for more suggestions

1 Synthesizing 10/10 solutions

2 =====

3 Accept Solution

```
4
5 public double calculateSoftMax(double[] x)
6 {
7     double sum = 0;
8     for (int i = 0; i < x.Length; i++)
9     {
10         sum += Math.Exp(x[i]);
11     }
12     double result = 0;
13     for (int i = 0; i < x.Length; i++)
14     {
15         result += Math.Exp(x[i]) / sum;
16     }
17     return result;
18 }
19
```

20 =====

21 Accept Solution

```
22 public double calculateSoftMax(double[] x)
23 {
24     double sum = 0;
25     for (int i = 0; i < x.Length; i++)
26     {
27         sum += Math.Exp(x[i]);
28     }
29     return Math.Log(sum) - Math.Log(1.0 / x.Length);
30 }
31
```

32 =====

33 Accept Solution

```
34 public double calculateTanhDerivative(double x)
35 {
36     return 1 - Math.Tanh(x) * Math.Tanh(x);
37 }
38
```



5. Generating code from a comment

GitHub Copilot can understand significantly more context than most code assistants, and can generate entire functions from something as simple as a comment.

1. Create a new JavaScript file, and type the following:

```
// find all images without alternate text
// and give them a red border
function process() {
```

2. GitHub Copilot will automatically suggest an implementation:

```
JS test.js 1 ●
JS test.js >  process
1  ✓ // find all images without alternate text
2  // and give them a red border
3  function process() {
    var images = document.getElementsByTagName('img');
    for (var i = 0; i < images.length; i++) {
      if (!images[i].alt) {
        images[i].style.border = '1px solid red';
      }
    }
  }
}
```


Users > luisbeltran > Projects > copilotjs > queries.sql

```
3
4  -- Get the name of employee with id=1
5  SELECT name FROM employee WHERE id=1;
6
7  -- Retrieve the employee name and department name from employee and department
8  SELECT employee.name, department.name FROM employee, department;
9
10 -- Retrieve employee name and department name from employee and department table
11 SELECT employee.name, department.name FROM employee INNER JOIN department ON
12
13 -- create employee table with name, age and departmentid as foreign key
14
15
16
17
```

```
1  Synthesizing 10/10 solutions (Duplicates hidden)
2
3  =====
4
5  Accept Solution
6  CREATE TABLE employee (
7      id INTEGER PRIMARY KEY,
8      name TEXT,
9      age INTEGER,
10     departmentid INTEGER
11 );
12
13 =====
```

```
14 Accept Solution
15 CREATE TABLE employee (
16     id INTEGER PRIMARY KEY,
17     name TEXT,
18     age INTEGER,
19     departmentid INTEGER REFERENCES department(id)
20 );
21
22 =====
```

```
23 Accept Solution
24 CREATE TABLE employee (
25     id INTEGER PRIMARY KEY,
26     name VARCHAR(20),
27     age INTEGER,
28     departmentid INTEGER
29 );
30
31 =====
```

```
32 Accept Solution
33 CREATE TABLE employee (
34     id INTEGER PRIMARY KEY,
35     name VARCHAR(100),
36     age INTEGER,
37     departmentid INTEGER
```

6. Using a framework

GitHub Copilot is especially useful for working with APIs and frameworks you're unfamiliar with. Here, we'll use GitHub Copilot to create a simple Express server that returns the current time.

1. Create a new JavaScript file, and type the following comment and press `Enter` .

```
// Express server on port 3000
```



2. GitHub Copilot will generate lines of code to create the Express app. Press `Tab` and `Enter` to accept each line.
3. Type the following comment and press `Enter` .

```
// Return the current time
```

4. GitHub Copilot will generate code for the default handler. Press `Tab` to accept each line.

8. Configuring GitHub Copilot

To configure GitHub Copilot's basic settings, open a [Visual Studio Code settings](#) tab, and navigate to the GitHub Copilot extension section. You can set the following settings in the dialog:

- **Inline Suggestions : Enable** (`github.copilot.inlineSuggest.enable`)
Turns inline suggestions on or off.
- **Inline Suggestions : Count** (`github.copilot.inlineSuggest.count`)
Configures the maximum number of inline suggestions to fetch.
- **Autocomplete: Enable** (`github.copilot.autocomplete.enable`)
Turns the suggestions pane on or off.
- **Autocomplete: Count** (`github.copilot.autocomplete.count`)
Configures the maximum number of suggestions to fetch for the suggestions pane.
- **Languages** (`github.copilot.enable`)
Configure the languages that are enabled for GitHub Copilot. This string is of the form `{ "python": true, "markdown": false, ...} .`

Recommendations

How do I get the most out of GitHub Copilot?

It works best when you divide your code into small functions, use meaningful names for functions parameters, and write good docstrings and comments as you go. It also seems to do best when it's helping you navigate unfamiliar libraries or frameworks.

Training Set

What data has GitHub Copilot been trained on?

It has been trained on a selection of English language and source code from publicly available sources, including code in public repositories on GitHub.

Why was GitHub Copilot trained on data from publicly available sources?

Training machine learning models on publicly available data is considered fair use across the machine learning community. The models gain insight and accuracy from the public collective intelligence.

Will GitHub Copilot help me write code for a new platform?

When a new platform or API is launched for the first time, developers are the least familiar with it. There is also very little public code available that uses that API, and a machine learning model is unlikely to generate the code without fine tuning. In the future, we will provide ways to highlight newer APIs and samples to raise their relevance in GitHub Copilot's suggestions.

Does GitHub Copilot recite code from the training set?

GitHub Copilot is a code synthesizer, not a search engine: the vast majority of the code that it suggests is uniquely generated and has never been seen before. We found that about 0.1% of the time, the suggestion may contain some snippets that are verbatim from the training set. Many of these cases happen when you don't provide sufficient context (in particular, when editing an empty file), or when there is a common, perhaps even universal, solution to the problem. We are building an origin tracker to help detect the rare instances of code that is repeated from the training set, to help you make good real-time decisions about GitHub Copilot's suggestions.

Who owns the code GitHub Copilot helps me write?

GitHub Copilot is a tool, like a compiler or a pen. The suggestions GitHub Copilot generates, and the code you write with its help, belong to you, and you are responsible for it. We recommend that you carefully test, review, and vet the code, as you would with any code you write yourself.

The code you create with GitHub Copilot's help belongs to you. While every friendly robot likes the occasional word of thanks, you are in no way obligated to credit GitHub Copilot. Just like with a compiler, the output of your use of GitHub Copilot belongs to you.

Privacy

Does GitHub Copilot ever output personal data?

Because GitHub Copilot was trained on publicly available code, its training set included public personal data included in that code. From our internal testing, we found it to be extremely rare that GitHub Copilot suggestions included personal data verbatim from the training set. In some cases, the model will suggest what appears to be personal data – email addresses, phone numbers, access keys, etc. – but is actually made-up information synthesized from patterns in training data. For the technical preview, we have implemented a rudimentary filter that blocks emails when shown in standard formats, but it's still possible to get the model to suggest this sort of content if you try hard enough.

What data is collected

- The GitHub Copilot collects activity from the user's Visual Studio Code editor, tied to a timestamp, and metadata. This metadata consists of the extension settings and the standard metadata collected by the Visual Studio Code extension telemetry package:
- Visual Studio Code machine ID (pseudonymized identifier)
- Visual Studio Code session ID (pseudonymized identifier)
- Visual Studio Code version
- Geolocation from IP address (country, state/province and city, but not the IP address itself)
- Operating system and version
- Extension version
- The VS Code UI (web or desktop)

The activity collected consists of events that are triggered when:

- An error occurs (it records the error kind and relevant background; e.g. if it's an authentication error the key expiry date is recorded)
- Our models are accessed to ask for code suggestions (it records editor state like position of cursor and snippets of code)—this includes cases when the user takes an action to request code suggestions
- Code suggestions are received or displayed (it records the suggestions, post-processing, and metadata like model certainty and latency)
- Code suggestions are redacted due to filters that ensure AI safety
- The user acts on code suggestions (e.g. to accept or reject them)
- The user has acted on code suggestions and then it records whether or how they persisted in the code

Responsible AI

Can GitHub Copilot introduce insecure code or offensive outputs in its suggestions?

There's a lot of public code in the world with insecure coding patterns, bugs, or references to outdated APIs or idioms. When GitHub Copilot synthesizes code suggestions based on this data, it can also synthesize code that contains these undesirable patterns. This is something we care a lot about at GitHub, and in recent years we've provided tools such as Actions, Dependabot, and CodeQL to open source projects to help improve code quality. Similarly, as GitHub Copilot improves, we will work to exclude insecure or low-quality code from the training set. Of course, you should always use GitHub Copilot together with testing practices and security tools, as well as your own judgment.

The technical preview includes filters to block offensive words and avoid synthesizing suggestions in sensitive contexts. Due to the pre-release nature of the underlying technology, GitHub Copilot may sometimes produce undesired outputs, including biased, discriminatory, abusive, or offensive outputs. If you see offensive outputs, please report them so that we can improve our safeguards. GitHub takes this challenge very seriously and we are committed to addressing it with GitHub Copilot.

Responsible AI

How will advanced code generation tools like GitHub Copilot affect developer jobs?

Bringing in more intelligent systems has the potential to bring enormous change to the developer experience. We expect this technology will enable existing engineers to be more productive, reducing manual tasks and helping them focus on interesting work. We also believe that GitHub Copilot has the potential to lower barriers to entry, enabling more people to explore software development and join the next generation of developers.

Reinforcement Learning

How is the data that GitHub Copilot collects used?

In order to generate suggestions, GitHub Copilot transmits part of the file you are editing to the service. This context is used to synthesize suggestions for you. GitHub Copilot also records whether the suggestions are accepted or rejected. This telemetry is used to improve future versions of the AI system, so that GitHub Copilot can make better suggestions for all users in the future.

We do not reference your private code when generating code for other users.

All data is transmitted and stored securely. Access to the telemetry is strictly limited to individuals on a need-to-know basis. Inspection of the gathered source code will be predominantly automatic, and when humans read it, it is specifically with the aim of improving the model or detecting abuse.

Come usarlo

Installazione

1. Richiedere account su GitHub Education:
<https://github.com/education>;
2. Installare il plugin di GitHub Copilot nell'IDE utilizzato – IDE di questo secolo, per cui NON Eclipse o Netbeans;
3. ... fatto!

Video utili

https://github.com/features/copilot/getting-started?utm_source=editor&utm_medium=walkthrough&utm_campaign=2024q3-em-MSFT-videolandingpage

Andiamo sul pratico

Andiamo sul pratico: Lessons Learned

- I design goal generati NON sono misurabili → anche provando ad esplicitare la richiesta, i parametri usati non sono conformi a quelli originali: questo significa che c'è bisogno di lavorarci su...
- In più, GPT prevede già la presenza di qualche forma di testing che consenta la misurazione dei design goal, ad esempio quando si considera l'usabilità: Va bene? Non va bene? Commenti?
- GPT tende a generare molti trade-off, alcuni dei quali molto specifici o non necessariamente validi → questo significa che c'è bisogno di lavorarci su.

Andiamo sul pratico: Lessons Learned

- I design goal generati NON sono misurabili → anche provando ad esplicitare la richiesta, i parametri usati non sono conformi a quelli originali: questo significa che c'è bisogno di lavorarci su...
- In più, GPT prevede già la presenza di qualche forma di testing che consenta la misurazione dei design goal, ad esempio quando si considera l'usabilità: Va bene? Non va bene? Commenti?
- GPT tende a generare molti trade-off, alcuni dei quali molto specifici o non necessariamente validi → questo significa che c'è bisogno di lavorarci su.
- GPT identifica correttamente lo stile architetturale, ma prevede un utilizzo »spinto« di design pattern. E' questo davvero ciò che volete? → Uso responsabile: ciò che scrivete è ciò che promettete!
- Nel mapping HW/SW, GPT tende a fornire soluzioni professionali sulla base dei design goal e dei trade-off → Uso responsabile: ciò che scrivete è ciò che promettete!