GitHub Copilot Training

Course content
puun.ch/to/pictet-copilot

Jeremy GOBET & Joachim DESROCHES 2024

Trainers

▶ **Jeremy Gobet**

Lecturer @HEPIA HES-SO
Web App Developer

jeremy.gobet@socraft.ch

Joachim Desroches ◀

IT Architect @Kleis Technology
Software Craftsmanship Trainer

joachim.desroches@socraft.ch

Agenda

▷ Theory

▷ Hands-on

01

Introduction

02

GitHub Copilot

03

Discovering features

04

Refactoring

05

Limits & Issues

06

Conclusion

01

Introduction

Training objectives, previous experience and expectations from participants.

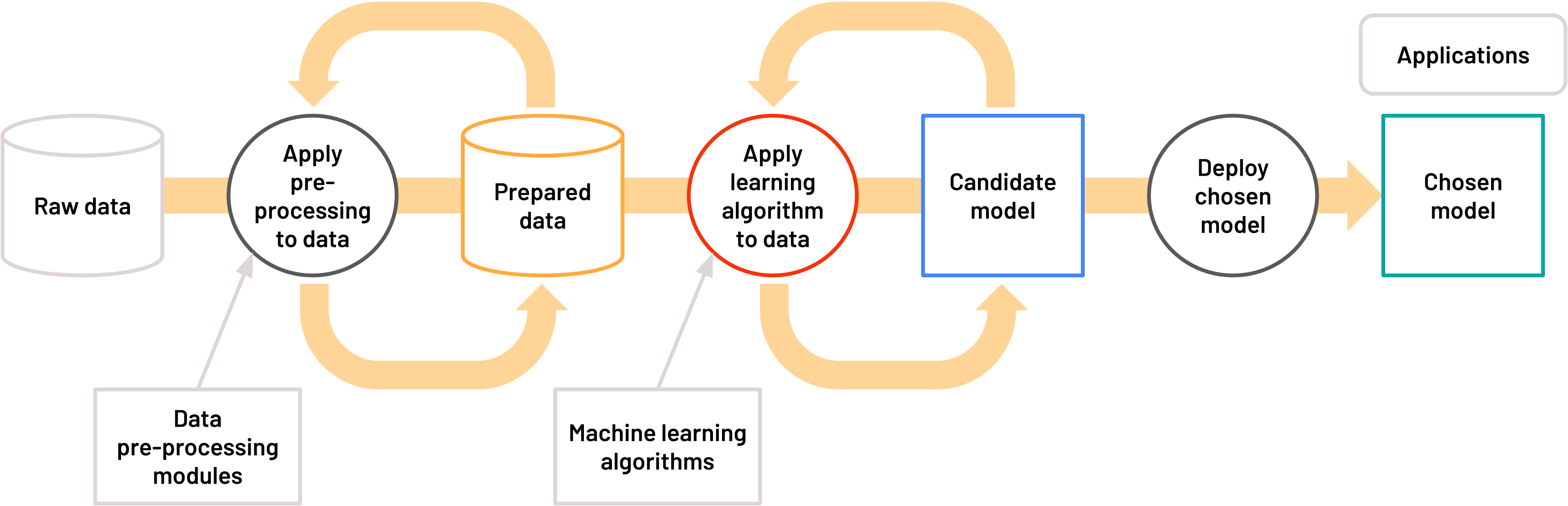
Training objectives

30% of theory and 70% of practice

- To provide insights about what GitHub Copilot does and does not through best practice usage considerations.
- To experiment with GitHub Copilot in order to forge an opinion of when and why to use it.
- User from the early beginning. We will share with you our recommendations.

What are your expectations ?

Machine Learning process


Raw data
Apply pre-processing to data
Prepared data
Apply learning algorithm to data
Candidate model
Deploy chosen model
Chosen model
Applications
Data pre-processing modules
Machine learning algorithms

Jeremy Gobet & Joachim Desroches. Github Copilot. 2024

02

GitHub Copilot

Presentation of GitHub Copilot: capabilities, forces and weakness.

GitHub Copilot provides contextualized assistance throughout the software development lifecycle

GitHub
Copilot

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

GitHub Copilot integrations

IDE Integration

- **VS Code**
- Visual Studio
- JetBrains IDEs
- Xcode
- Neovim

GitHub Integration

- Code review
- GitHub actions
- Issues
- Third-party extensions
- ...

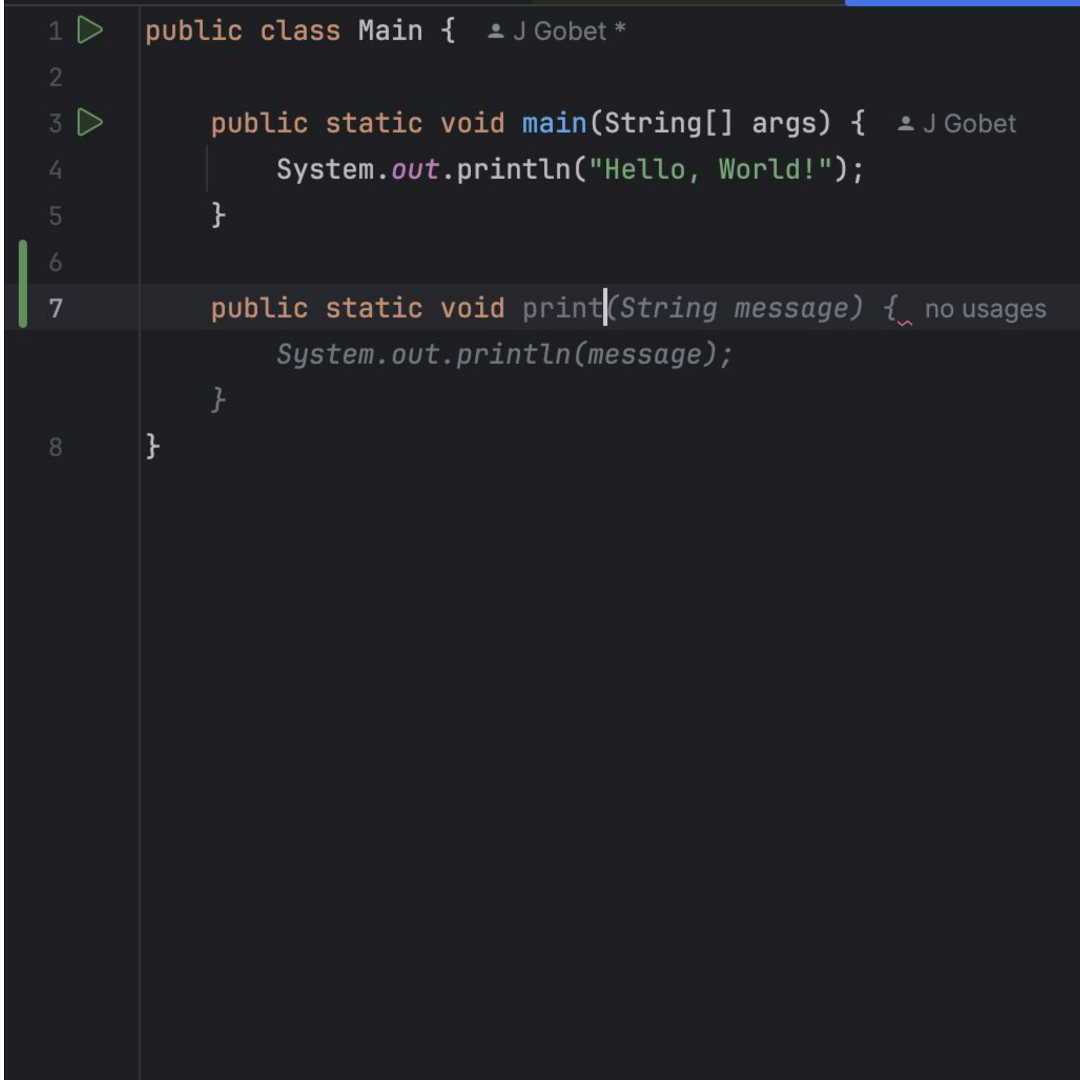
Other integrations

- Android application
- IOS application
- CLI
- ...

https://github.com/features/copilot

What is GitHub Copilot ?

- **Code Completion**
- Chat assistance
- Context-aware



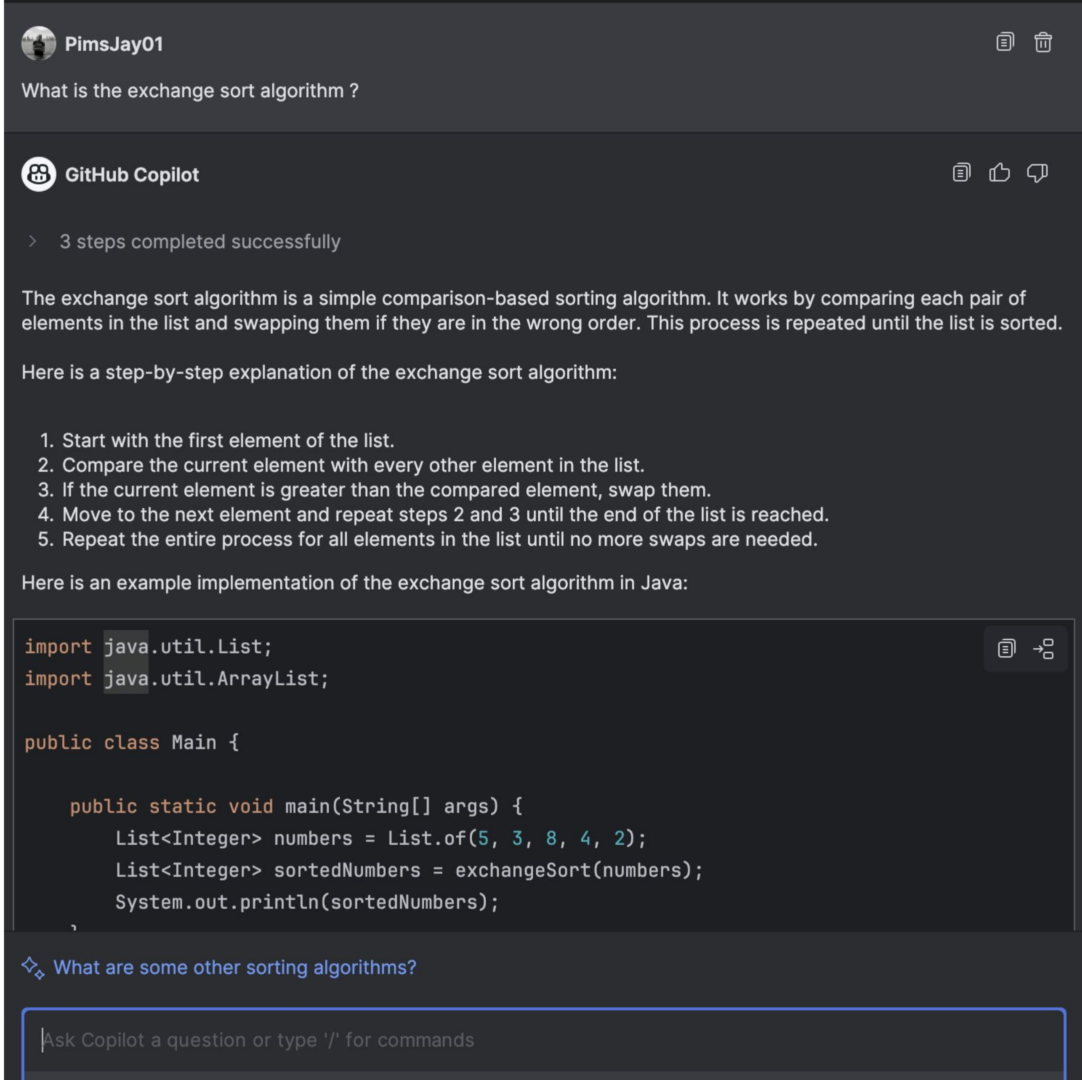
```java
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }

    public static void print(String message) {
        System.out.println(message);
    }
}
```

What is GitHub Copilot ?

Code Completion

Chat assistance

Context-aware



PimsJay01

What is the exchange sort algorithm ?

GitHub Copilot

The exchange sort algorithm is a simple comparison-based sorting algorithm. It works by comparing each pair of elements in the list and swapping them if they are in the wrong order. This process is repeated until the list is sorted.

Here is a step-by-step explanation of the exchange sort algorithm:

1. Start with the first element of the list.
2. Compare the current element with every other element in the list.
3. If the current element is greater than the compared element, swap them.
4. Move to the next element and repeat steps 2 and 3 until the end of the list is reached.
5. Repeat the entire process for all elements in the list until no more swaps are needed.

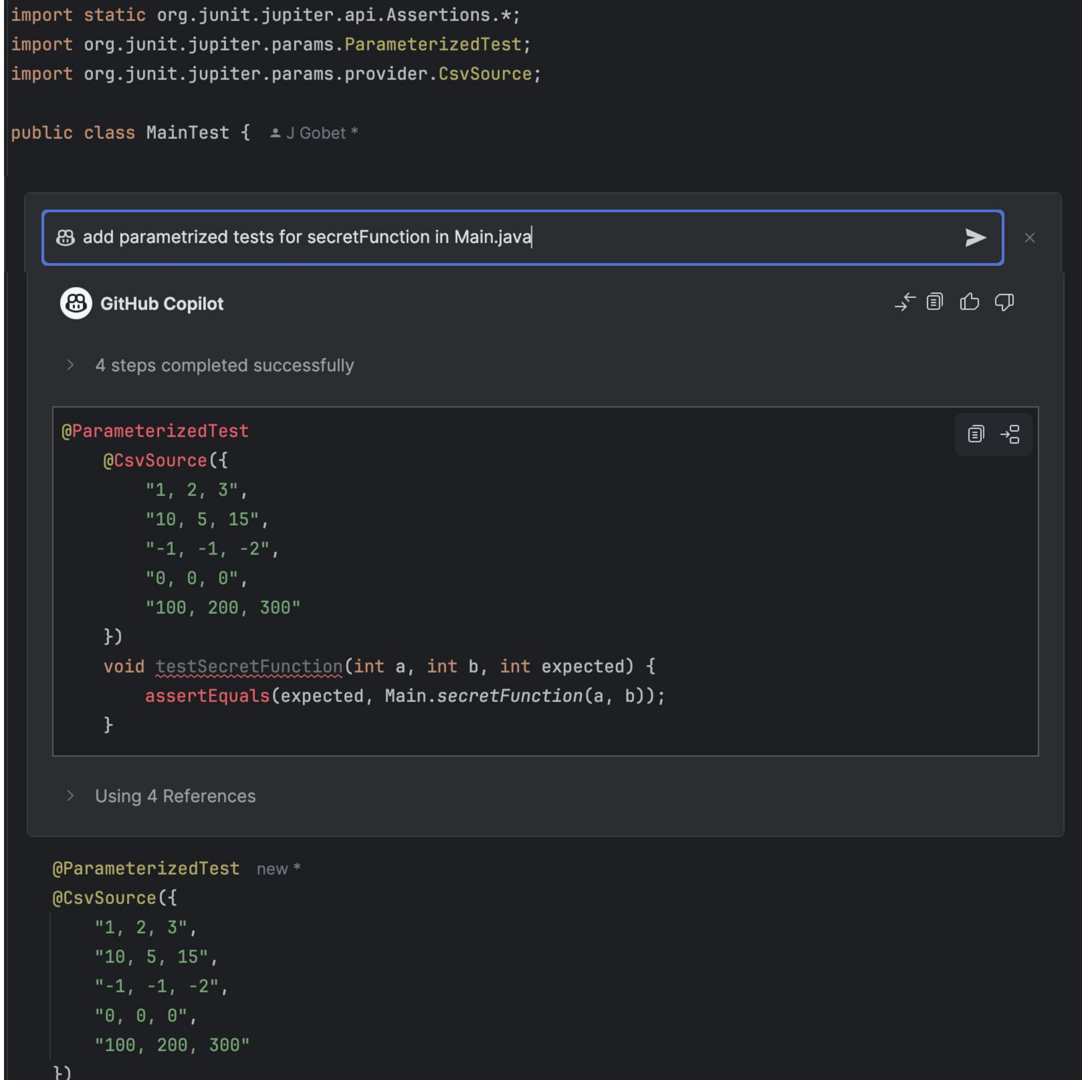
Here is an example implementation of the exchange sort algorithm in Java:

```java
import java.util.List;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        List<Integer> numbers = List.of(5, 3, 8, 4, 2);
        List<Integer> sortedNumbers = exchangeSort(numbers);
        System.out.println(sortedNumbers);
```

What are some other sorting algorithms?
Ask Copilot a question or type '/' for commands

What is GitHub Copilot ?

- Code Completion
- Chat assistance
- **Context-aware**



```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;

public class MainTest {  J Gobet *
```

add parametrized tests for secretFunction in Main.java

GitHub Copilot

4 steps completed successfully

```java
@ParameterizedTest
    @CsvSource({
        "1, 2, 3",
        "10, 5, 15",
        "-1, -1, -2",
        "0, 0, 0",
        "100, 200, 300"
    })
    void testSecretFunction(int a, int b, int expected) {
        assertEquals(expected, Main.secretFunction(a, b));
    }
```

Using 4 References

```java
@ParameterizedTest  new *
@CsvSource({
    "1, 2, 3",
    "10, 5, 15",
    "-1, -1, -2",
    "0, 0, 0",
    "100, 200, 300"
})
```

What is GitHub Copilot ?

Copilot is an AI-powered code assistant tool developed by GitHub in collaboration with OpenAI

Code Completion

AI-driven suggestions from a comment, a function signature, the beginning of a line, the contextual menu, etc.

Chat assistance

Developers can interact with GitHub Copilot using natural language using the chat or directly in the code.

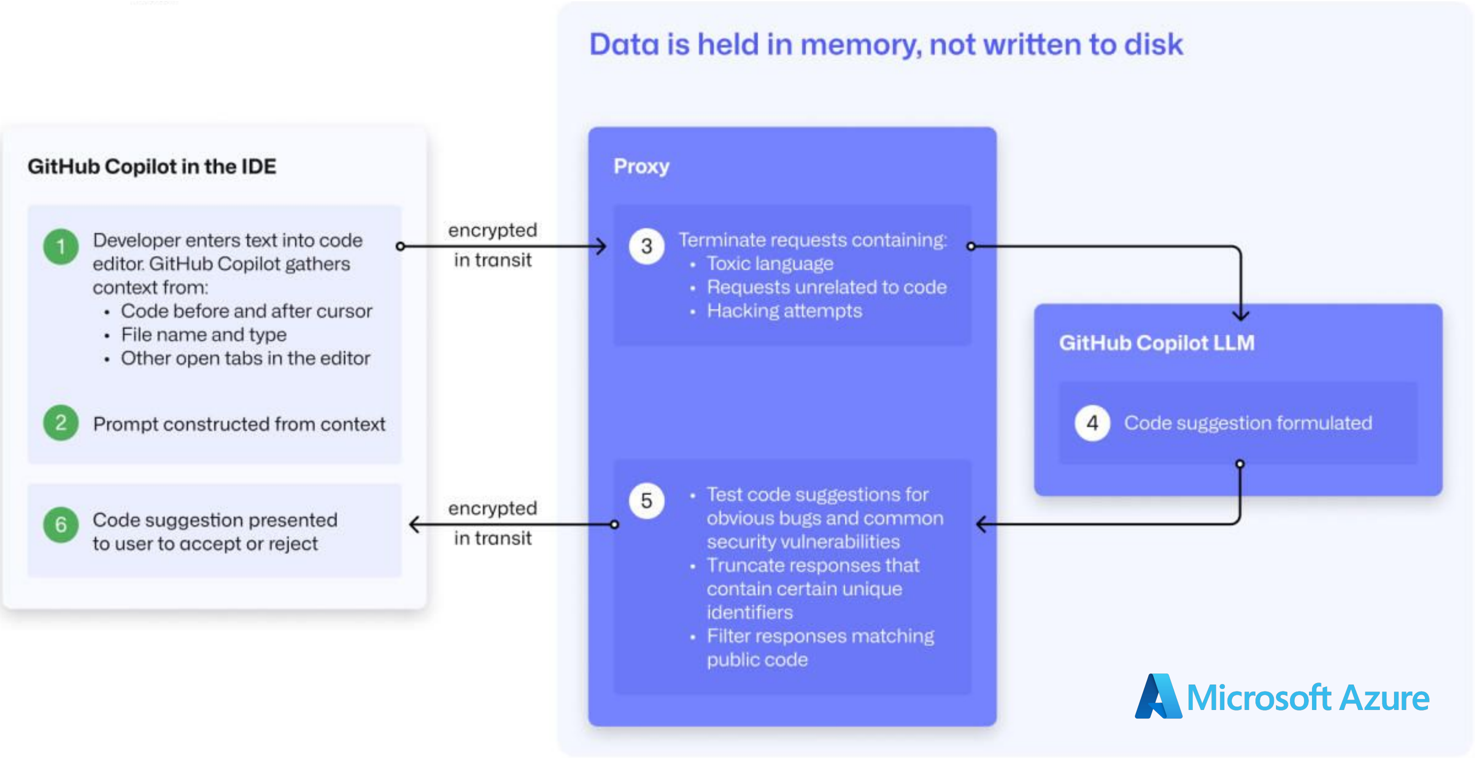
Context-aware

GitHub Copilot analyzes the context of the code being written and generates suggestions accordingly.

GitHub Copilot keeps your data secure on trusted Microsoft Azure infrastructure

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

Lifecycle of a code suggestion



Data is held in memory, not written to disk

GitHub Copilot in the IDE

① Developer enters text into code editor. GitHub Copilot gathers context from:
- Code before and after cursor
- File name and type
- Other open tabs in the editor

② Prompt constructed from context

encrypted in transit

Proxy

③ Terminate requests containing:
- Toxic language
- Requests unrelated to code
- Hacking attempts

GitHub Copilot LLM

④ Code suggestion formulated

⑤
- Test code suggestions for obvious bugs and common security vulnerabilities
- Truncate responses that contain certain unique identifiers
- Filter responses matching public code

encrypted in transit

⑥ Code suggestion presented to user to accept or reject

Microsoft Azure

https://resources.github.com/learn/pathways/copilot/essentials/how-github-copilot-handles-data

How Copilot addresses plagiarism

Models learn to avoid recitation

Different techniques help models generate new code rather than output code from the training data.

Duplication detection filter

Checks suggestions against public repositories for matches of 65 or more lexemes.

Contractual protection

GitHub and Microsoft provide indemnity. But only if duplication detection filter is enabled.

https://resources.github.com/learn/pathways/copilot/essentials/establishing-trust-in-using-github-copilot/

Our findings

No plan for an on-premises version of Copilot

But GitHub declares about Copilot that they do not

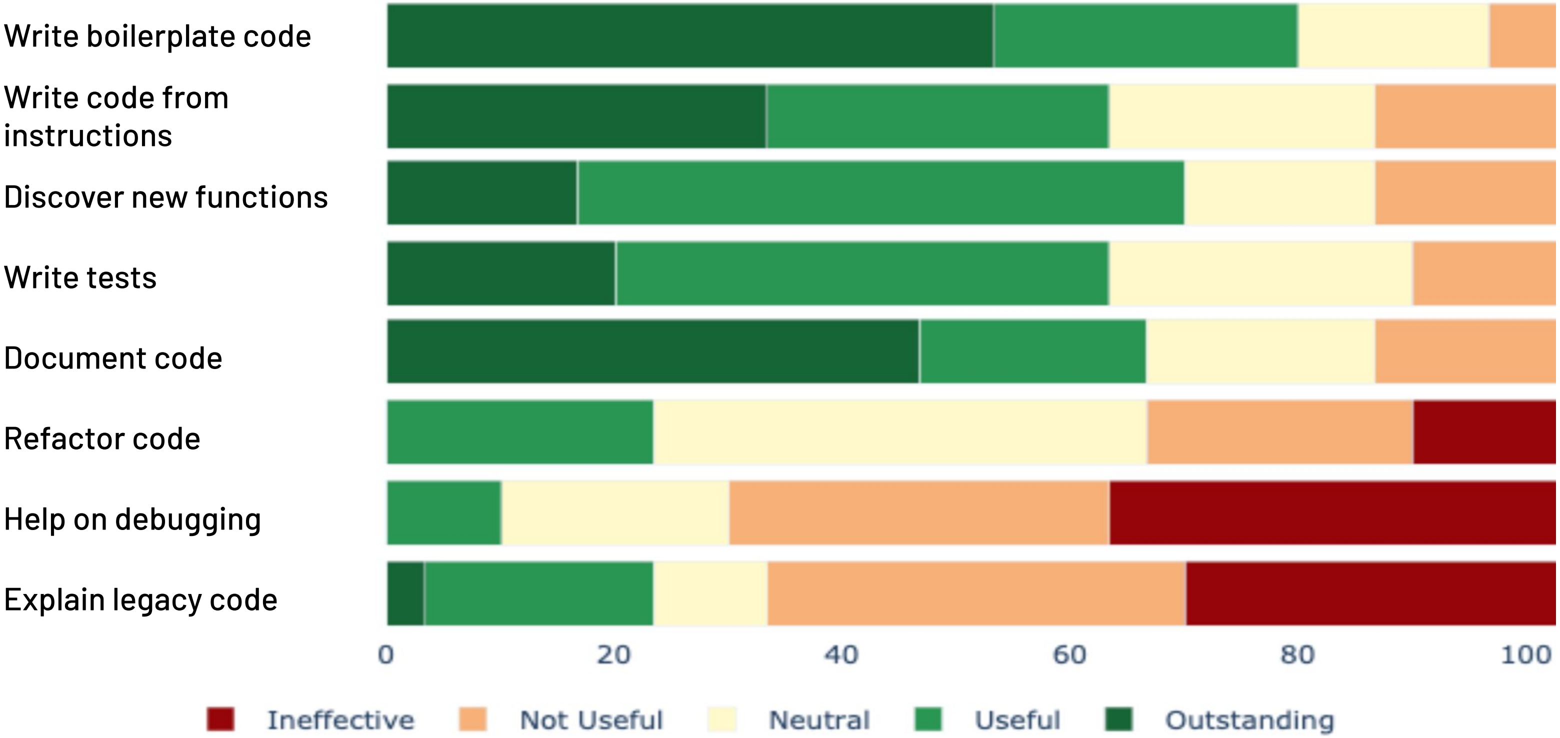
- ❌ retain user content (e.g. code from user's editor)
- ❌ use code in context to improve the model
- ❌ retain suggestions produced by Copilot
- ❌ send information to OpenAI
- ❌ own the code generated by Copilot

GitHub Copilot handles the tasks developers dislike

GitHub Copilot

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

Usefulness on different tasks



Prometeia. Alessandro Benetti & Michele Filannino. GitHub Copilot: a systematic study. 29-30-.05.2024

Integration with documentation

"**GitHub Copilot's integration with our documentation**
allows our engineers to ask specific questions of our documentation,
instead of searching by keyword and scanning the results for what they need.
It's a more natural way of interacting with technical content
that's saving our developers time and effort."

Jun Li

Engineering Manager at Lyft,
american company offering
mobility as a service

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

Our findings

Unfortunately Copilot is not going to meetings for us yet

But GitHub Copilot could be good for

- Repetitive or boilerplate code
- Keeping focus (avoiding context-switch)
- Writing tests (at least the “given” part)
- Explaining error messages and suggesting fixes
- Onboarding with new code libraries
- Generating mockup data

Try it yourself to find how it can help you

GitHub Copilot boosts speed and efficiency

GitHub Copilot

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

Better productivity ?

Slalom experimented with 4 teams, working on identical projects. Half used GitHub Copilot. The developers had no prior experience with the codebase or Copilot and completed their tasks within one sprint.

Observations

- **20-50% faster code production**
- Code quality varied by language
- **Led to more unnecessary code duplication in some languages**

Spend Less Time:

- Writing boilerplate and repetitive code
- Searching google and Stack Overflow
  → **Decrease context switching**

Spend More Time:

- Writing pseudo-code and code comments of the problems team are solving
- Reviewing and tweaking AI generated code
- Thinking about how to structure code to solve business problems

Slalom, business and technology consulting company. Test of productivity with GitHub Copilot. 2024

Our findings

Copilot decreases context-switching

But all studies we found on GitHub Copilot try to make a link between productivity and lines of code written.

Unfortunately none uses DORA metrics such as **Lead Time for Changes** or **Deployment Frequency** which seem much more appropriate to measure team performance.

We can not confirm increase in productivity.

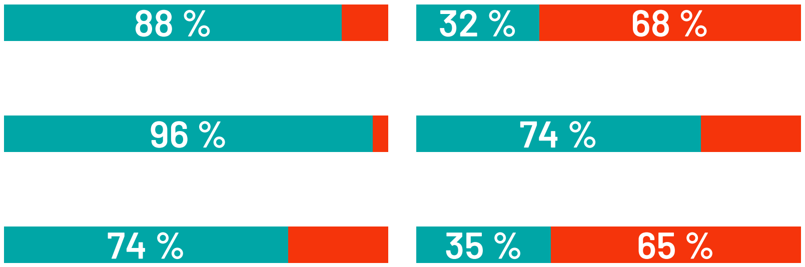
GitHub Copilot can
~~help attract~~
~~and retain talent~~
improve coding satisfaction

GitHub
Copilot

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

Developer satisfaction

GitHub Copilot study performance metrics



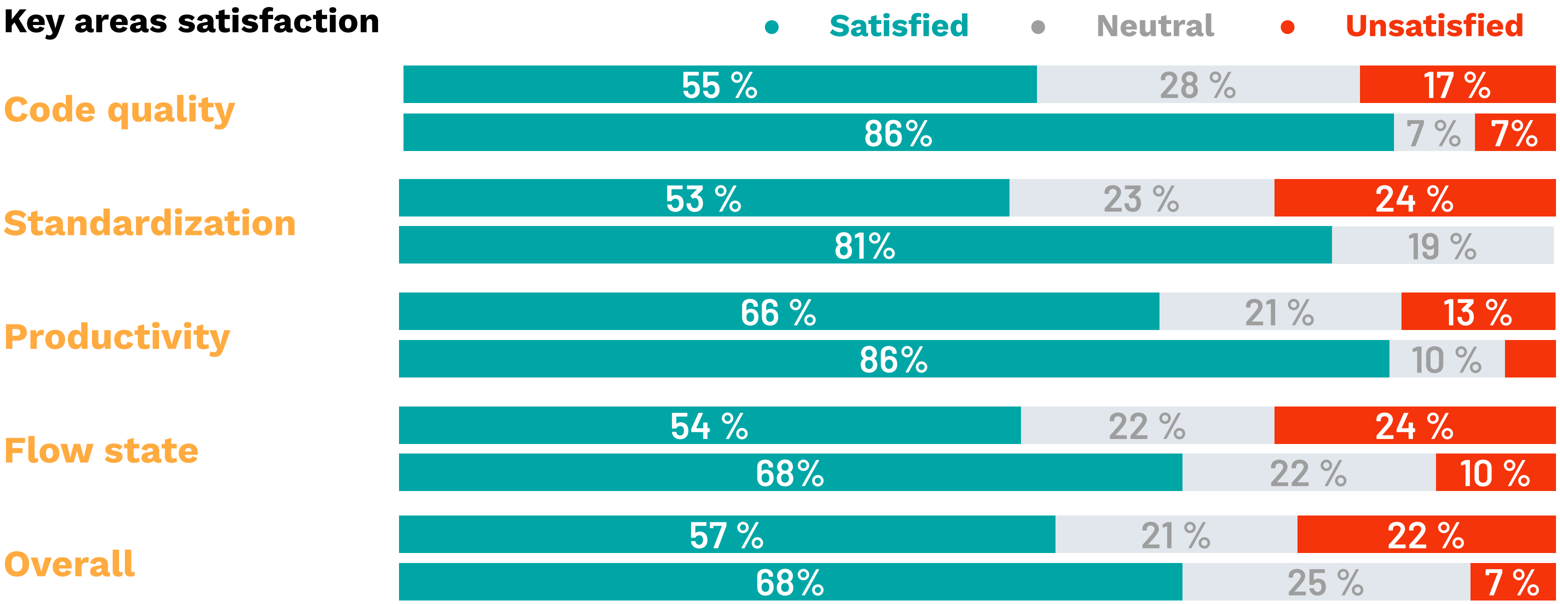
Yes or no questions	**GitHub** study results	**Prometeia** study results
Feel more productive	88 %	32 % 68 %
Feel faster with repetitive tasks	96 %	74 %
Could focus on more satisfying tasks	74 %	35 % 65 %

“The differences may be attributed to the nature of the experiments conducted by GitHub. [...] Our study population consisted of individuals with impending deadlines, which could influence their perceptions and experiences with the tool. Our results still reflect a highly positive sentiment“

Prometeia. Alessandro Benetti & Michele Filannino. GitHub Copilot: a systematic study. 29-30-.05.2024

Developer satisfaction

Satisfaction with key areas of job without and with Copilot



Slalom. Test of productivity with GitHub Copilot. 2024

Our findings

Like every tool, learning to use it **takes some time** and personal investment.

The return on time invested is very good for some activities.

- **Might not attract your next genius dev, but can help with developer satisfaction**

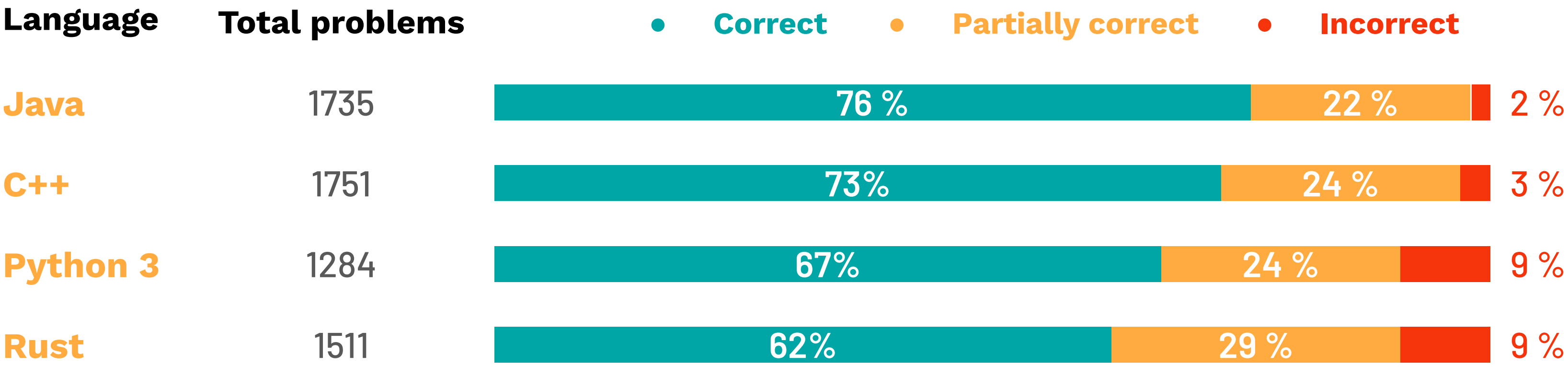
GitHub Copilot can improve code quality

GitHub
Copilot

https://resources.github.com/learn/pathways/copilot/essentials/essentials-of-github-copilot

Support for programming languages

Submission results for each language on LeetCode problems



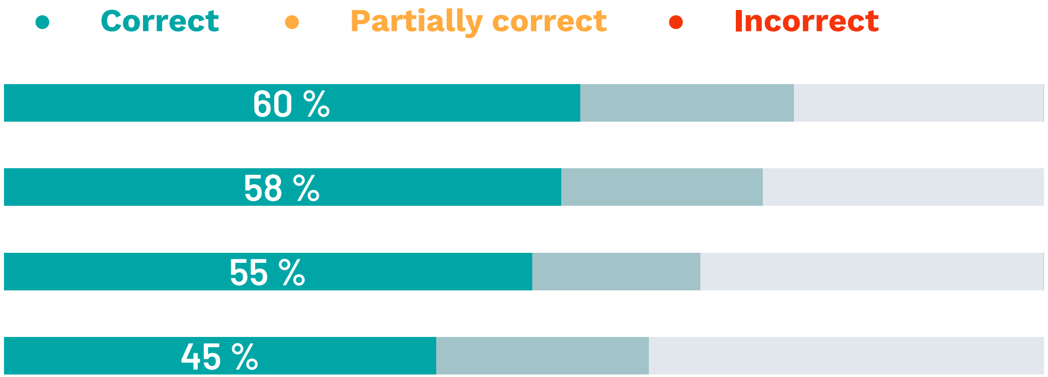
“For each problem, we asked Copilot to generate as many suggestions as possible and tested all of them [...] Here we consider Copilot’s solution for the problem to be Correct if Copilot generated at least one suggestion that passes all the tests“

Ilja Siros, Dave Singelée & Bart Preneel. GitHub Copilot: the perfect Code compLeeter?. 17.06.2024

Support for programming languages

Top Copilot proposal is not always the best one.

Language	Total problems
Java	1735
C++	1751
Python 3	1284
Rust	1511



“We concluded that the code generated by Copilot is more efficient, both time- and memory-wise, than the code written by the average human (user on LeetCode platform).“

Ilja Siros, Dave Singelée & Bart Preneel. GitHub Copilot: the perfect Code compLeeter?. 17.06.2024

More time available for quality ?

”Many participants reported reallocating the time gained to enhance the **quality** of their products further by focusing on rigorous **testing**, refining **documentation**, or dedicating effort to areas of the project that could benefit from manual oversight”

Authors of the study

People from Prometeia, provider of advisory services, tech solutions and research insights.

Prometeia. Alessandro Benetti & Michele Filannino. GitHub Copilot: a systematic study. 29-30-.05.2024

Our findings

More time for quality <u>yes but</u>

- Will not change your company culture (craftsmanship vs assembly line)
- May be a kiss cool effect
- Tends to produce code similar to the existing codebase: good or bad

03

Discovering features

Quick exploration of all the ways
to trigger Copilot

Discovering features

Open the provided **Java** project
and generate suggestion from Copilot with:

1. a comment
2. a function signature
3. the in-line prompt in code
4. multiple suggestions
5. chat
6. contextual menu

github.com/
Copilot-Training-by-socraft/
discovering-features-java

Discovering features

Open the provided **Python** project
and generate suggestion from Copilot with:

1. a comment
2. a function signature
3. the in-line prompt in code
4. multiple suggestions
5. chat
6. contextual menu

github.com/
Copilot-Training-by-socraft/
discovering-features-python

Discovering features (1/2)

Open the provided **Java** project and generate suggestion from Copilot with:

1. **a comment**:

```
// method to compute a bubble sort
```

2. **a function signature**:

```
public static int sum(int[] arr)
```

3. **the in-line prompt in code**:

   Select the previous method, open in-line prompt , and ask:
   "refactor to use the stream API"

Discovering features (1/2)

Open the provided **Python** project and generate suggestion from Copilot with:

1. **a comment**:

   ```
   # method to compute a bubble sort
   ```

2. **a function signature**:

   ```
   def calcul_sum_for_even_values(number_list):
   ```

3. **the in-line prompt in code**:

   Select the previous method, open in-line prompt , and ask:
   "refactor to use list comprehensions"

Discovering features (2/2)

Still on the provided **Java** project, use Copilot's:

4. **multiple suggestions**:

   Open the suggestions pane, and then prompt for:
   `// memoized fibonacci function`

5. **chat**:

   Open the chat, and ask:
   "write parameterized tests with five examples for a generic sorting function"

6. **contextual menu**:

   Select the bubble sort and in the contextual menu, click
   "**Simplify This**" and then "**Generate Docs**" or "**/docs**" directly in prompt

Discovering features (2/2)

Still on the provided **Python** project, use Copilot's:

4. **multiple suggestions**:

   Open the suggestions pane, and then prompt for:
   `# memoized fibonacci function`

5. **chat**:

   Open the chat, and ask:
   "write parameterized tests with five examples for a generic sorting function"

6. **contextual menu**:

   Select the bubble sort and in the contextual menu, click
   "Simplify This" and then **"Generate Docs"** or "**/docs**" directly in prompt

04

Refactoring

Experimentation of how to refactor code
using GitHub Copilot.

The 9 rules of object calisthenics

Guidelines to keep your code maintainable, readable, reusable and testable

- one level of indentation per method.
- don't use the ELSE keyword.
- wrap all primitives and Strings in classes.
- first class collections.
- one dot per line.
- don't abbreviate.
- keep all classes less than 50 lines.
- no classes with more than two instance variables.
- no getters or setters → Tell. Don't ask.

github.com/
Copilot-Training-by-socraft/
tictactoe-java

The 9 rules of object calisthenics

Guidelines to keep your code maintainable, readable, reusable and testable

- one level of indentation per method.
- don't use the ELSE keyword.
- wrap all primitives and Strings in classes.
- first class collections.
- one dot per line.
- don't abbreviate.
- keep all classes less than 50 lines.
- no classes with more than two instance variables.
- no getters or setters → Tell. Don’t ask.

github.com/Copilot-Training-by-socraft/tictactoe-python

Refactoring with GitHub Copilot

Guidelines to keep your code maintainable, readable, reusable and testable

- ☐ one level of indentation per method.
- ☐ don't use the ELSE keyword.
- ☐ wrap all primitives and Strings in classes.
- ☐ first class collections.
- ☐ one dot per line.
- ☐ don't abbreviate.
- ☐ keep all classes less than 50 lines.
- ☐ no classes with more than two instance variables.
- ☐ no getters or setters → Tell. Don’t ask.

GitHub Copilot features to use at least once

Generate suggestions

- ☐ from function signature
- ☐ from in-line or chat prompt
- ☐ from multiple suggestion pane

Use command from contextual menu

- ☐ Explain This
- ☐ Simplify This
- ☐ Generate Tests
- ☐ Generate Docs

06

Limits & Issues

GitHub Copilot limitations and main issues.

GitHub Copilot does not think for you

GitHub Copilot & LLMs do not understand or interpret meaning but rather predict words that are likely to follow a given sequence. They operate by recognizing and generating text based on patterns they have learned from vast amounts of data during training.

GitHub Copilot focuses on the syntax of the code rather than the underlying semantics or meaning.

A limited world

The codebase knowledge of Copilot is limited to the training date. **This might cause the suggestion of deprecated methods** for libraries that change significantly over time.

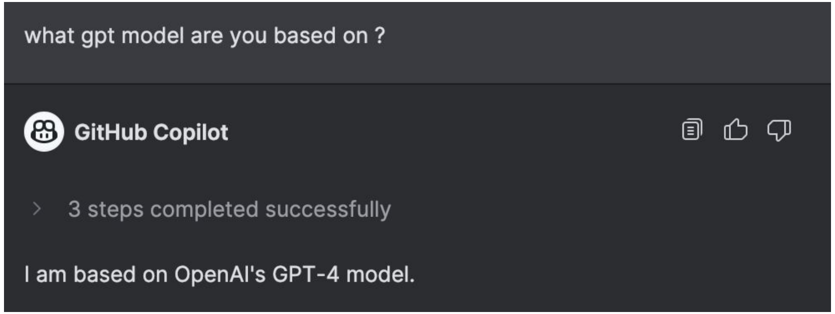
Java SE 21 – 19th of September 2023

→ **Last Java's version: SE 25**

Python 3.11.10 – 6th of September 2023

→ **Last Python's version: Python 3.13.0**

GitHub Copilot's knowledge base seems to be **updated every 6 months**.

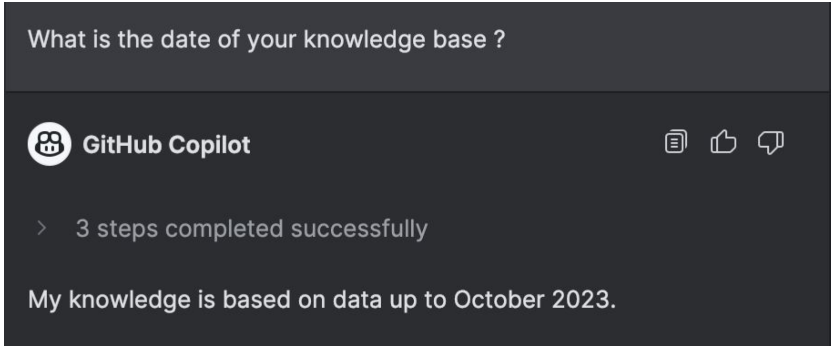


what gpt model are you based on ?

GitHub Copilot

› 3 steps completed successfully

I am based on OpenAI's GPT-4 model.



What is the date of your knowledge base ?

GitHub Copilot

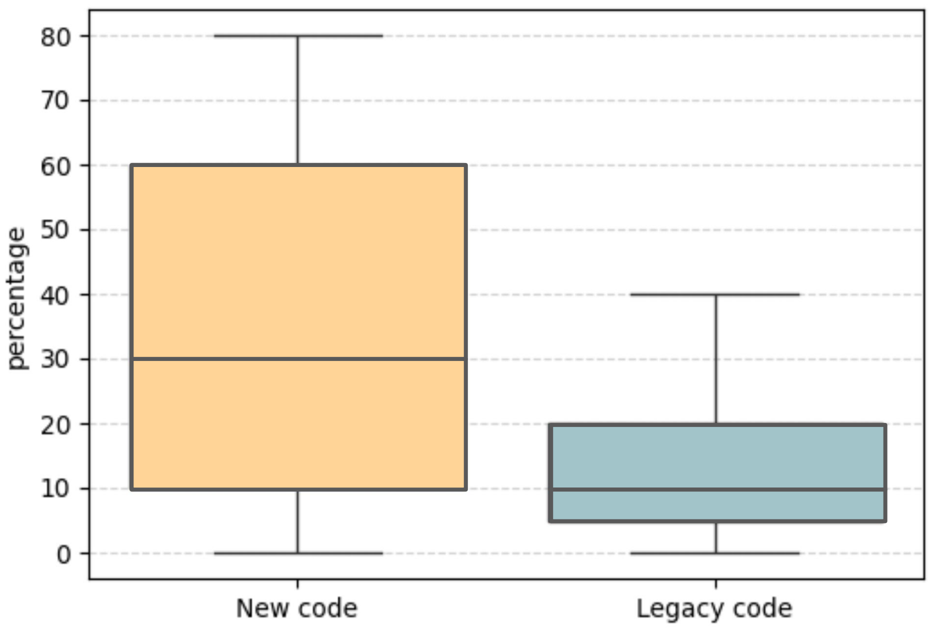
› 3 steps completed successfully

My knowledge is based on data up to October 2023.

Main drawbacks

- **Performance with large or legacy codebases presents challenges**
- Conflicts between Copilot and IDE support
- Suggests repetitive code.

Percentage of code lines written by Copilot



Prometeia. Alessandro Benetti & Michele Filannino. GitHub Copilot: a systematic study. 29-30-.05.2024

Vulnerabilities generated by Copilot

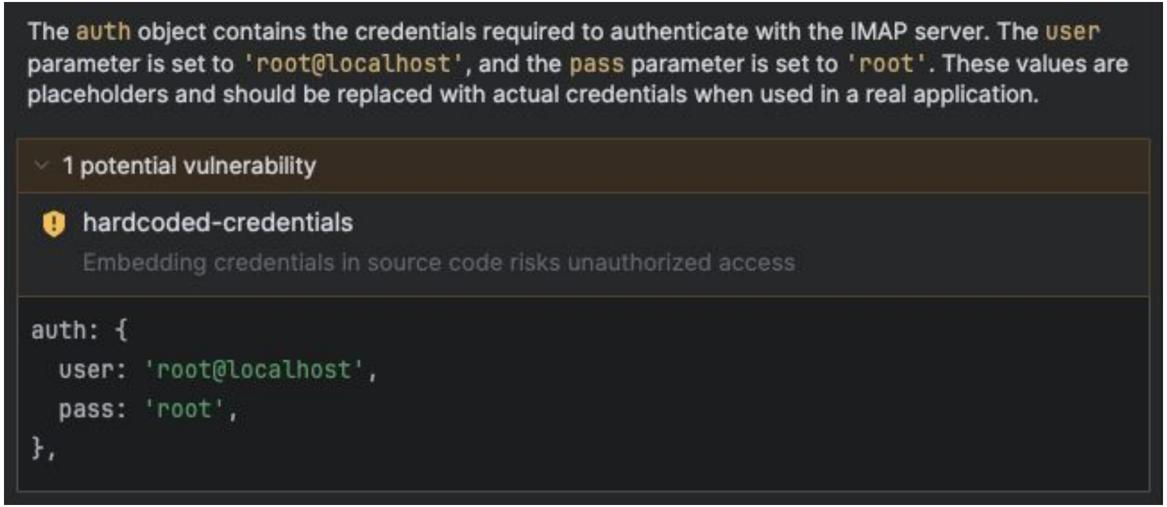
“across all axes and languages, Copilot generated vulnerabilities for **39 % of their suggestions**.”

Authors of the study

People from ECE & CSE New York University and ESE University of Calgary

Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Contributions. 16.12.2021

Detection of potential vulnerabilities



The `auth` object contains the credentials required to authenticate with the IMAP server. The `user` parameter is set to `'root@localhost'`, and the `pass` parameter is set to `'root'`. These values are placeholders and should be replaced with actual credentials when used in a real application.

1 potential vulnerability

hardcoded-credentials

Embedding credentials in source code risks unauthorized access

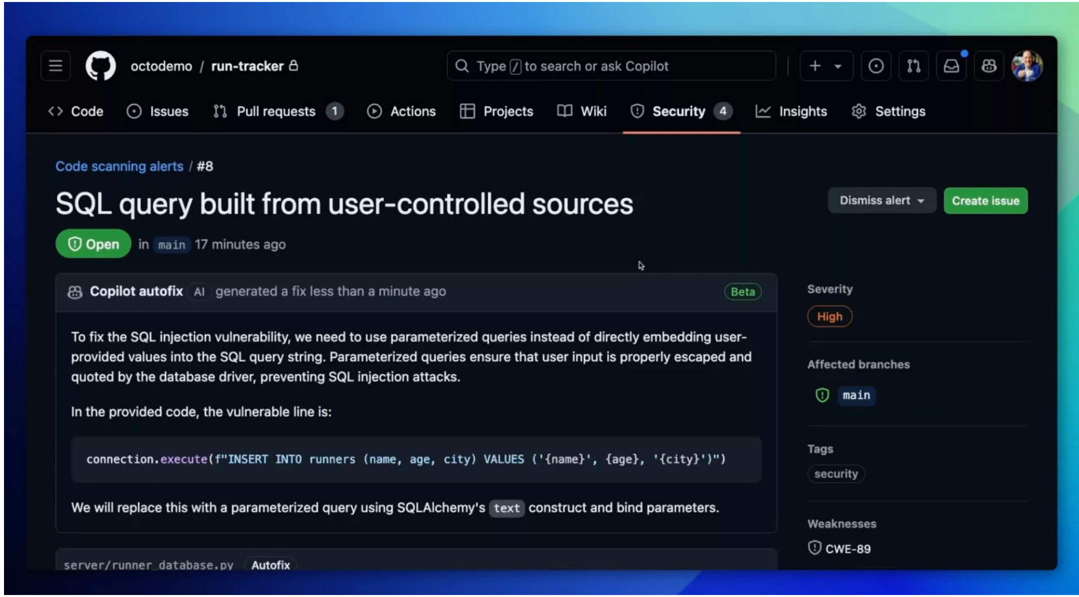
```
auth: {
  user: 'root@localhost',
  pass: 'root',
},
```

Copilot Autofix

GitHub Advanced Security (GHAS) is the native Static Application Security Testing (SAST) solution for GitHub Enterprise.

It searches for potential security vulnerabilities, coding errors and secrets in your code.

It performs AI-powered remediation with **Copilot Autofix** directly in the GitHub platform.



SQL query built from user-controlled sources

Open in `main` 17 minutes ago

Copilot autofix AI generated a fix less than a minute ago

To fix the SQL injection vulnerability, we need to use parameterized queries instead of directly embedding user-provided values into the SQL query string. Parameterized queries ensure that user input is properly escaped and quoted by the database driver, preventing SQL injection attacks.

In the provided code, the vulnerable line is:

```
connection.execute(f"INSERT INTO runners (name, age, city) VALUES ('{name}', {age}, '{city}')")
```

We will replace this with a parameterized query using SQLAlchemy's `text` construct and bind parameters.

Severity: High

Affected branches: `main`

Tags: security

Weaknesses: CWE-89

https://docs.github.com/en/get-started/learning-about-github/about-github-advanced-security

Our findings

Copilot is trained on public code of varying quality

- **You remain accountable for the final code.**
  Don't blindly trust suggestions made by Copilot

Copilot will imitate your bad patterns and potentially duplicate the vulnerabilities in your code base.

- **Review code regularly**
- **Include security analysis in continuous integration**
- **Train staff, especially juniors**

06

Conclusion

Recommendations to use Copilot.

Daily routine with GitHub Copilot

You are accountable

You should understand code before running it and use Copilot to enhance, not replace, your problem-solving skills.

Prompt effectively

Ask good question prompts as minor changes can lead to different outcomes, ensuring the input structure is clear to Copilot.

Iterative use

Employ Copilot iteratively by creating small code snippets and gradually building on them for comprehensive solutions.

Copilot is not an architect

Define design by yourself. Use Copilot only to write more quickly the code of your predetermined solution.

Rubber duck on steroïds

Try to engage Copilot in discussions to refine your ideas and benefit from creative solutions and suggestions.

Keep related files open in editor

Keep only relevant files open to help Copilot in understanding context due to its limited context window capability.

What next ?

Latest announces from GitHub Universe 2024 Conference

Multi-models

New available models for Copilot:
- Claude 3.5 Sonnet
- o1-preview & o1-mini
- Gemini 1.5 Pro (available soon)

GitHub Spark

Allows a non-developers to create micro apps powered by natural language.

Local config. file

Add local configuration file in your project to give general instructions to Copilot.

Experience

- Multi-file editing in VS Code
- Many languages support
- Copilot Autotfix
- Copilot Extensions

https://github.blog/news-insights/product-news/universe-2024-previews-releases

Training conclusion

Who here doesn't use Google or Stack Overflow ?

- Will not replace human
  → Just a new tool in our development toolbox
- Provides enough value for its price
  → Copilot deserves to be experimented
- Everybody uses it differently and
  Everybody has an opinion
  → Try it by yourself
- Many different integrations getting better
  → Not yet on the full scope of software lifecycle

A revolution may be underway

What use should you make of Copilot in your context ?

See you at Q/A & REX session
on January 2025

jeremy.gobet@socraft.ch

joachim.desroches@socraft.ch

R.O.T.I

Return on time invested

From 1 → To 5

Annexes

Alternatives

Local IA model for autocompletion available in following IDE

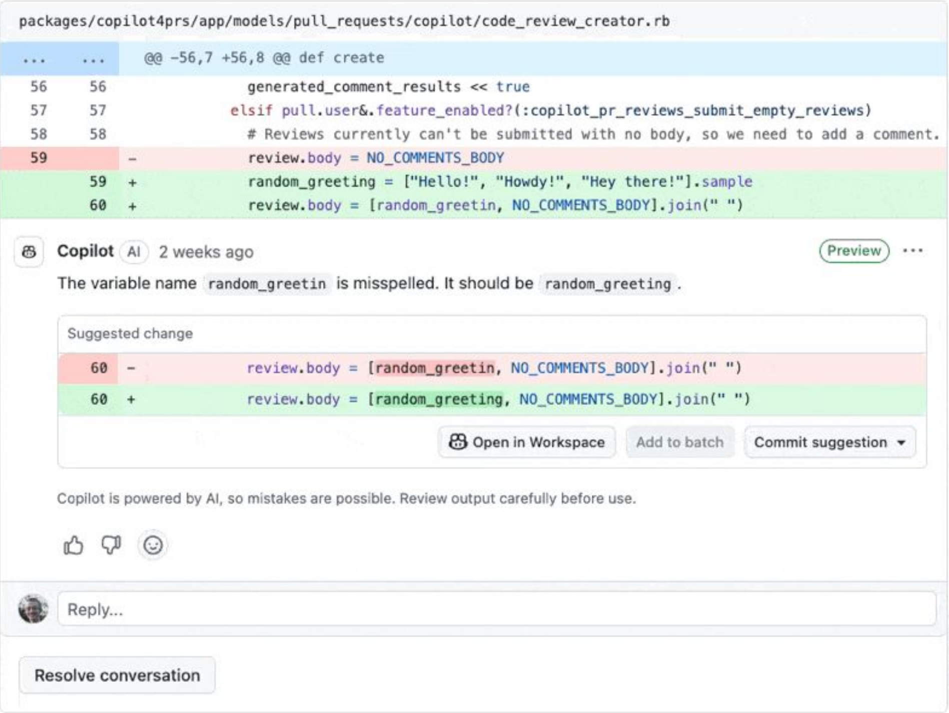
- **JetBrains In-IDE AI Assistant** and AI Service
- **Cursor IA IDE**

Other IA Assistants available on JetBrains Marketplace

- Blackbox
- Tabnine
- Sourcery

GitHub Copilot in code reviews

- **Request review by Copilot**
  or configure automatic reviews
  for every pull-request
- **Copilot attach its comments**
  to specific lines of your code,
  including one-click fixes
- **Copilot Workspace**
  allows to refine and validate
  Copilot’s suggestions in the context
  of the pull request
- **Copilot describes pull request**
  with more details and highlight
  the most important areas for review



packages/copilot4prs/app/models/pull_requests/copilot/code_review_creator.rb

```
...   ...      @@ -56,7 +56,8 @@ def create
 56    56                  generated_comment_results << true
 57    57                elsif pull.user&.feature_enabled?(:copilot_pr_reviews_submit_empty_reviews)
 58    58                  # Reviews currently can't be submitted with no body, so we need to add a comment.
 59          -               review.body = NO_COMMENTS_BODY
       59    +               random_greeting = ["Hello!", "Howdy!", "Hey there!"].sample
       60    +               review.body = [random_greetin, NO_COMMENTS_BODY].join(" ")
```

Copilot AI 2 weeks ago

The variable name `random_greetin` is misspelled. It should be `random_greeting`.

Suggested change

```
 60  -               review.body = [random_greetin, NO_COMMENTS_BODY].join(" ")
 60  +               review.body = [random_greeting, NO_COMMENTS_BODY].join(" ")
```

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

https://github.blog/changelog/2024-10-29-github-copilot-code-review-in-github-com-public-preview, 29.10.2024