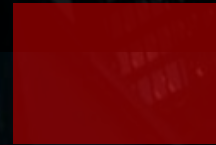




Introduction to Github Copilot

Making your code Intelligent



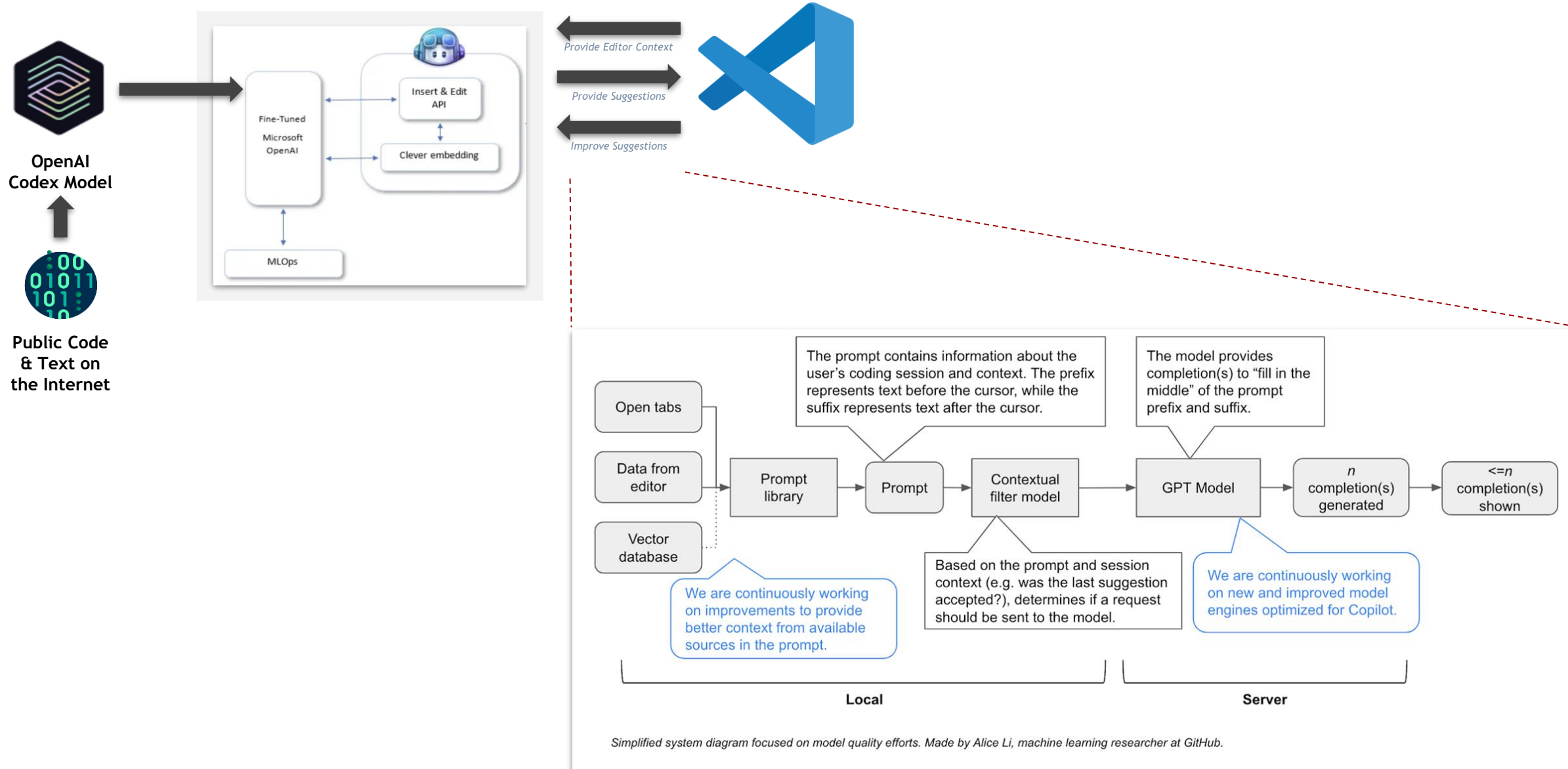


AGENDA

- Copilot in Daily Workflows
- Chat-Powered Development & AI-Driven Code Assistance
- Feature Development with Limited Context
- AI-Assisted Test-Driven Development (TDD) & Code Quality Enforcement
- Intelligent Code Refactoring & Performance Optimization
- Seamless DevOps Integration & CI/CD Automation
- AI-Generated Documentation & Code Summarization
- Contributing to External Projects with GitHub Copilot

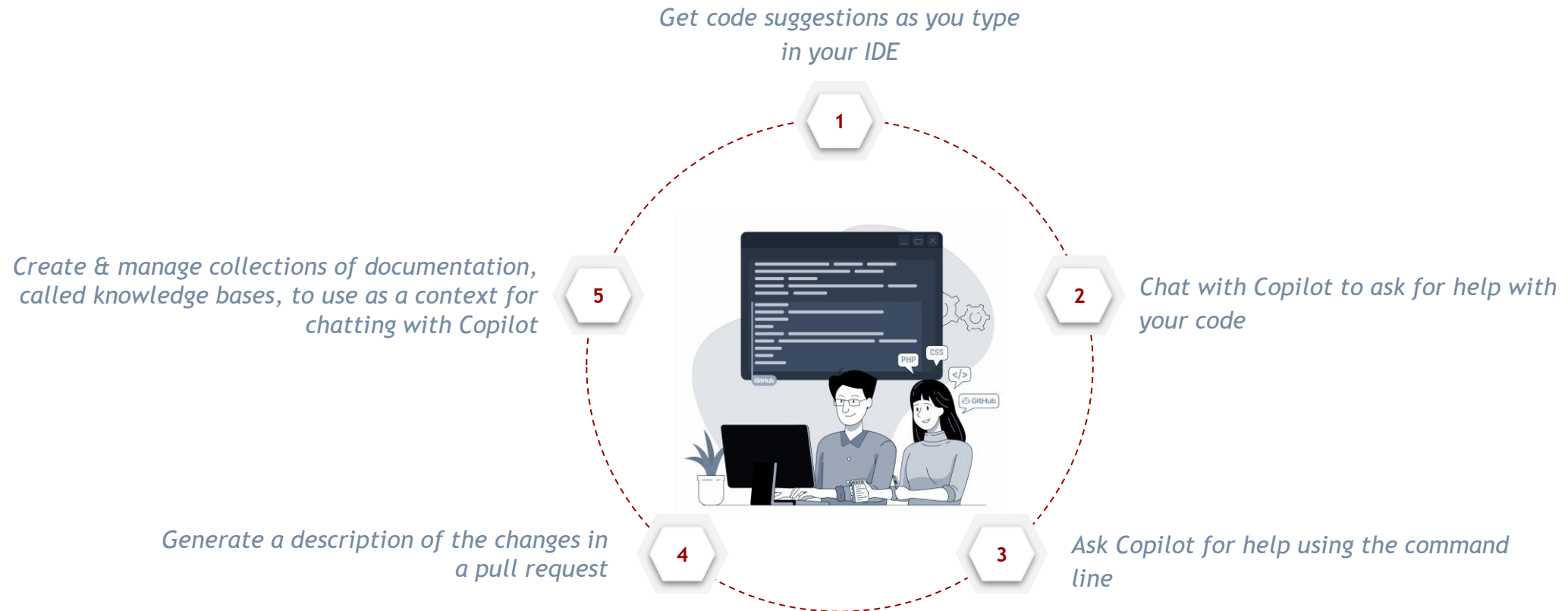


How Github Copilot understand your code?



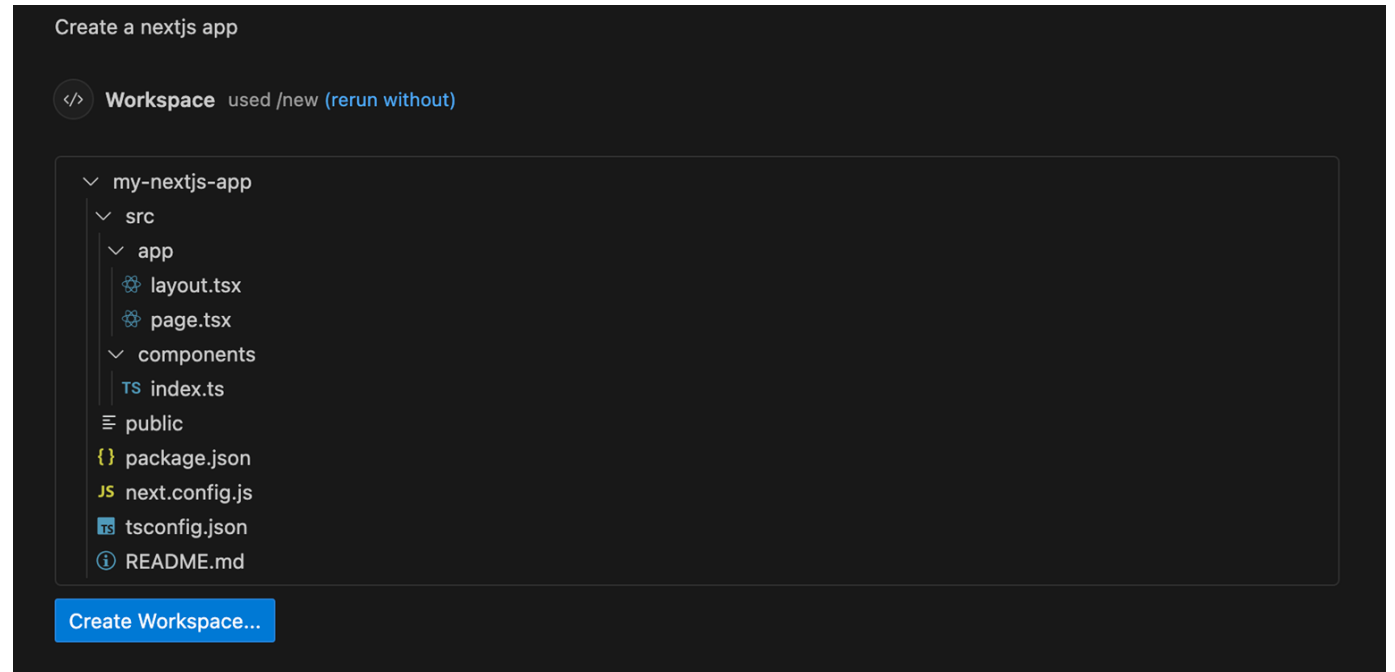
Features of Github Copilot

The AI coding assistant changing coding paradigm



Copilot in Daily Workflows

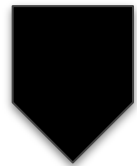
- *Rapid Project Initialization*
- *Boilerplate Code Generation*
- *Prototyping Minimal Viable Features*
- *Best Practices & Efficiency*



Strengths & Weaknesses

Understand your coding partner in depth!

Some of the things Copilot does best are



Writing tests and repetitive code



Debugging and correcting syntax



Explaining and commenting code



Generating regular expressions

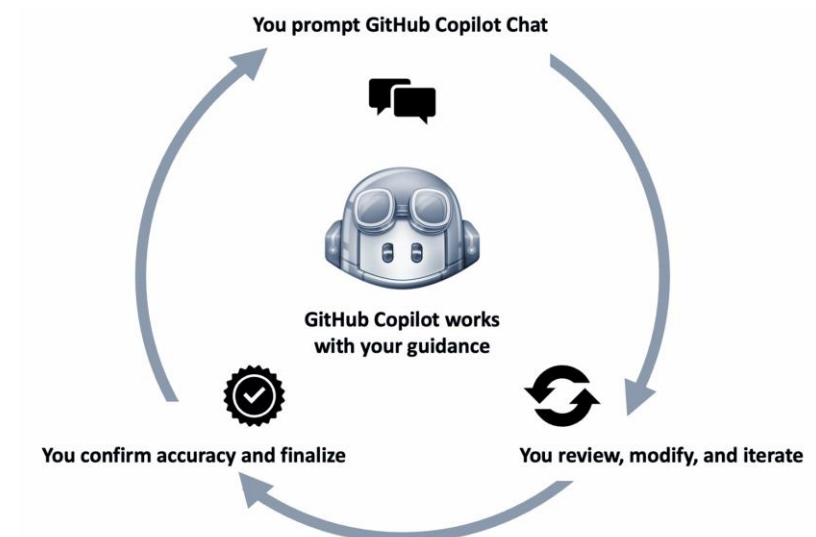
Copilot is not designed to



Respond to prompts unrelated to coding and technology



Replace your expertise & skills. Remember that you are in charge, & Copilot is a powerful tool at your service





Hands-On Exercise 1:

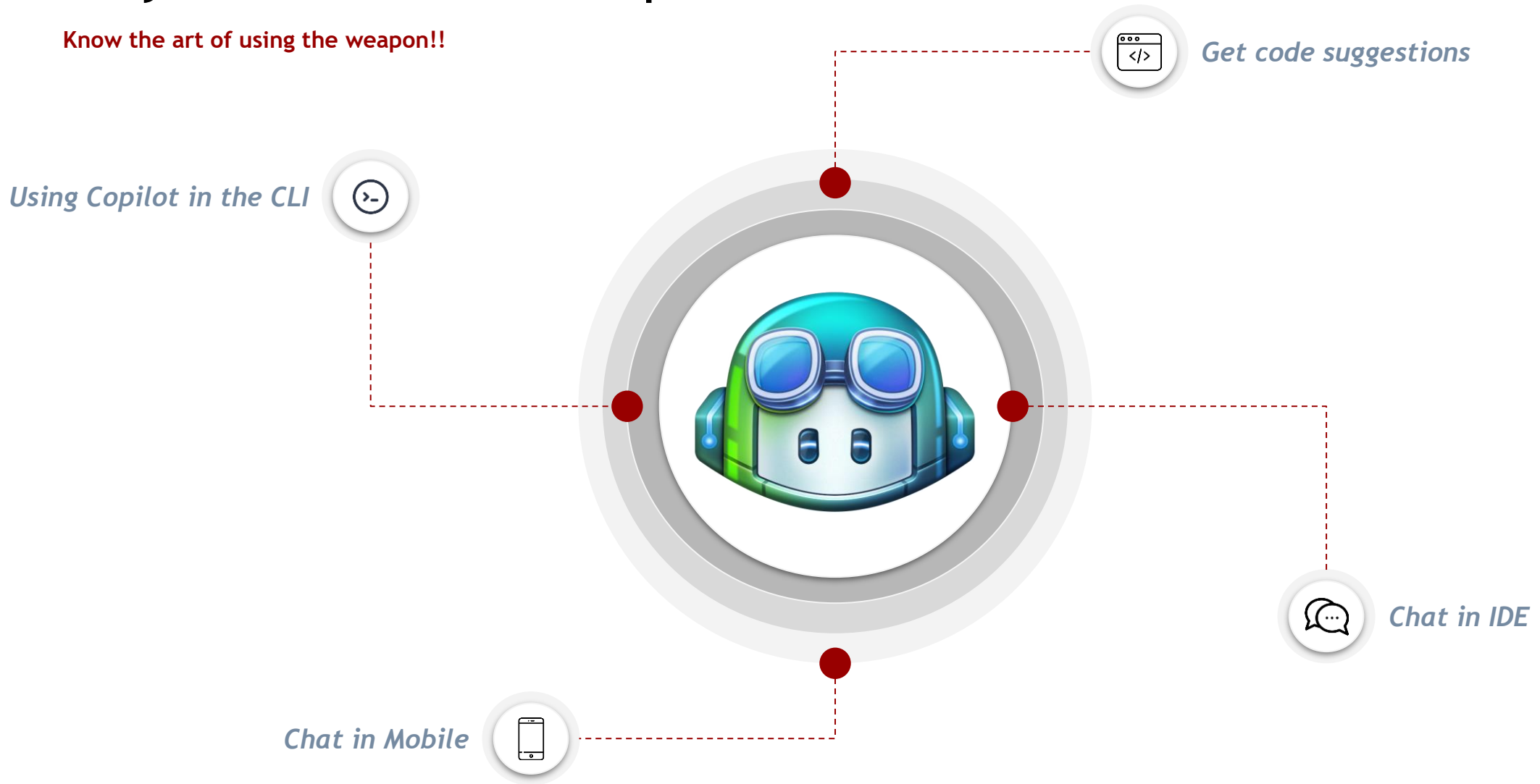


Github Copilot

Chat-Powered Development & AI-Driven Code Assistance

Ways to use Github Copilot

Know the art of using the weapon!!



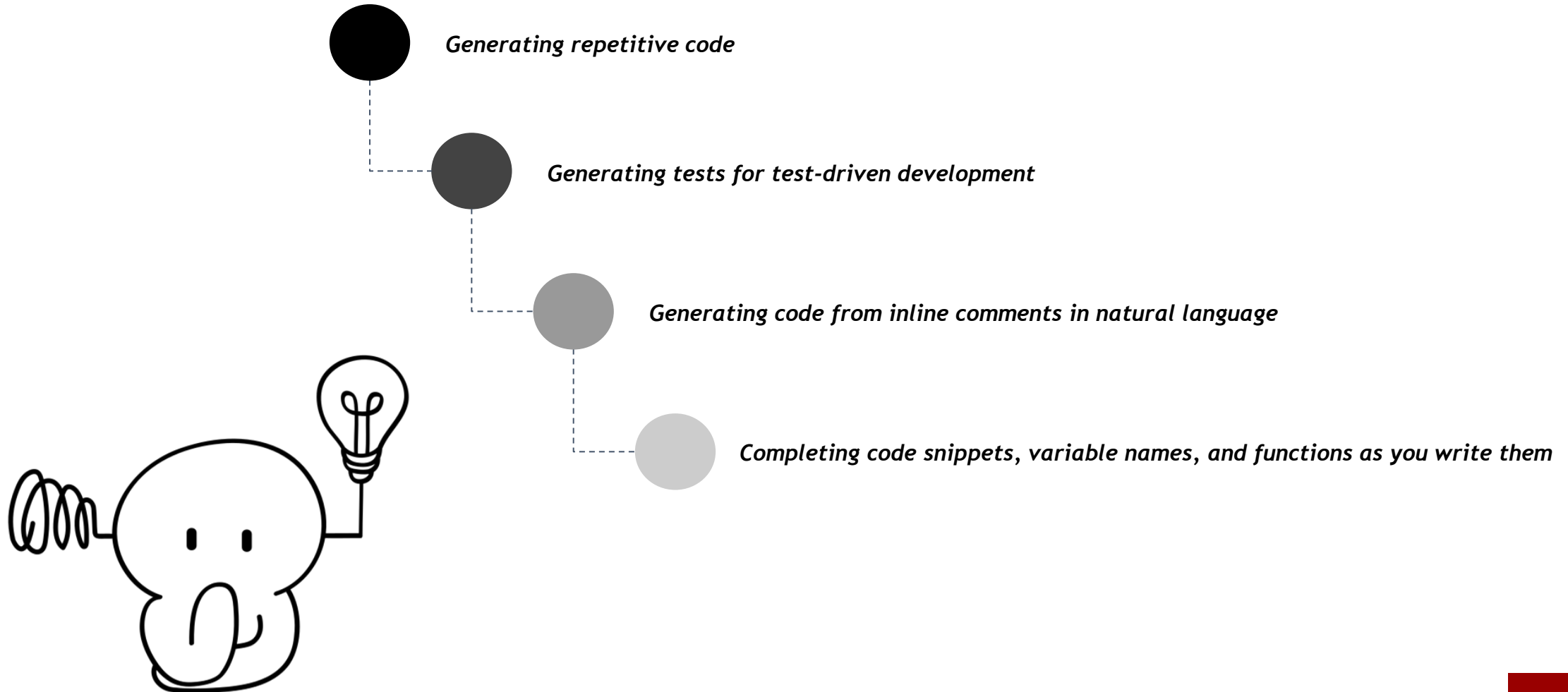
Chat-Powered Development & AI-Driven Code Assistance

Use GitHub Copilot to get code suggestions in your editor

- **Real-time Suggestions:** GitHub Copilot provides code suggestions as you type in your IDE
- **Context-Aware:** The suggestions are based on the context of the code you're writing
- **AI-Powered:** The tool uses advanced AI models trained on vast amounts of code to predict and suggest relevant snippets
- **Code Completion:** It offers both simple code completions (e.g., finishing a line of code) & complex suggestions (e.g., entire functions or code blocks)
- **Multi-Language Support:** GitHub Copilot supports a wide range of programming languages and frameworks
- **Improves Efficiency:** It helps developers write code faster by reducing manual coding effort
- **Learning from Patterns:** The tool learns from common coding patterns and your personal style to refine suggestions

Chat-Powered Development & AI-Driven Code Assistance

When does it work best?



Get Code Suggestion

Use GitHub Copilot to get code suggestions in your editor

>_

```
// write a function to  
// find all images without alternate text  
// and give them a red border
```

|..



GitHub Copilot will automatically suggest the rest of the function

Press **TAB** to accept the suggestion



>_

```
// write a function to  
// find all images without alternate text  
// and give them a red border
```

```
function findMyImage(begin, end) { ...
```

Get Code Suggestion

Show alternative suggestions for your code

Press **TAB** to accept the suggestion



```
>_  
  
import pandas as pd  
data = ...
```

```
>_  
  
import pandas as pd  
data = pd.read_csv(...)
```



Use the following keyboard shortcuts
for viewing suggestions

OS	See next suggestion	See previous suggestion
macOS	Option (⌘) +]	Option (⌘) + [
Windows or Linux	Alt+]	Alt+[

Get Code Suggestion

Didn't like the initial suggestions? You can even view multiple Suggestions as well

```
>_  
  
import pandas as pd  
|..
```



Press **CONTROL + ENTER** to see multiple suggestions in new tab

GitHub Copilot Suggestions

8 Suggestions

Suggestion 1

```
import numpy
```

Accept suggestion 1

Suggestion 2

```
import numpy as np  
import matplotlib
```

Accept suggestion 2

Suggestion 3

```
import numpy as np  
  
df = pd.DataFrame(np.random.randn(6, 4), columns=  
list('AB
```

Accept suggestion 3

Get Code Suggestion

Show alternative suggestions for your code

>_

```
import pandas as pd
```

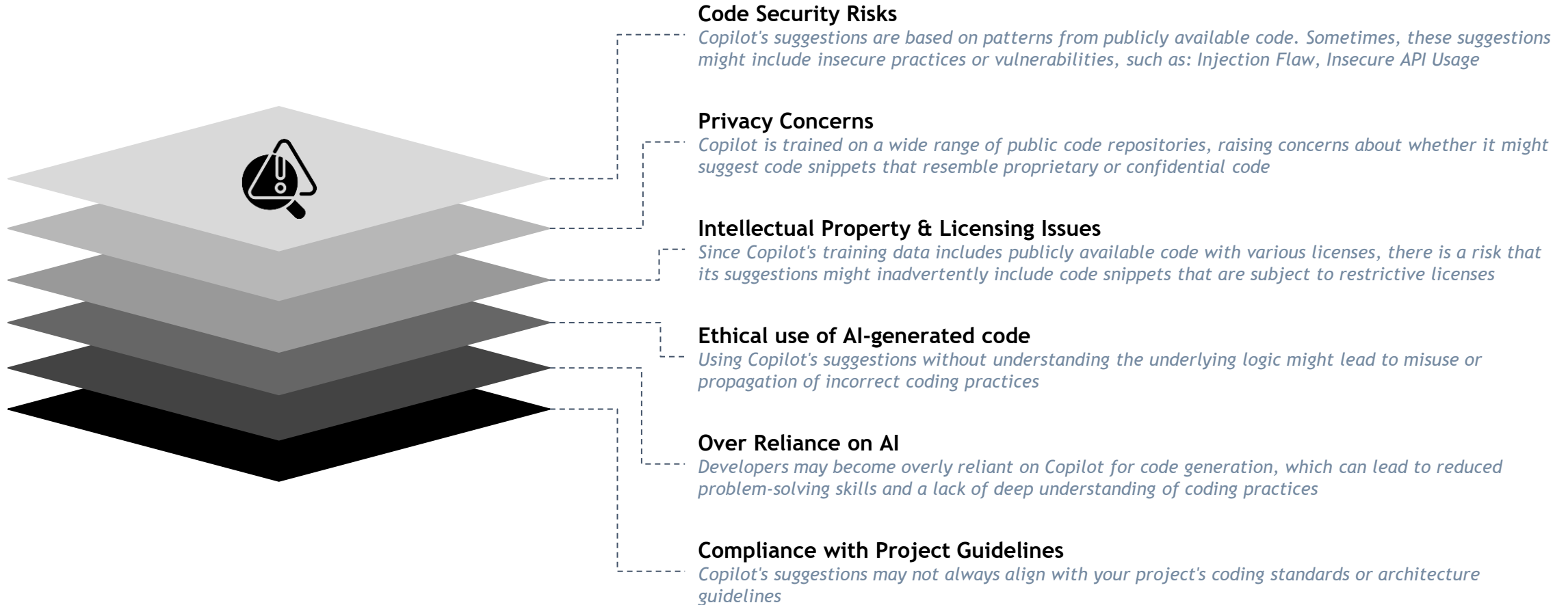
```
data = pd.read_csv(data.csv)
```



Press **CONTROL** + **→** to accept
the words out the complete suggestion

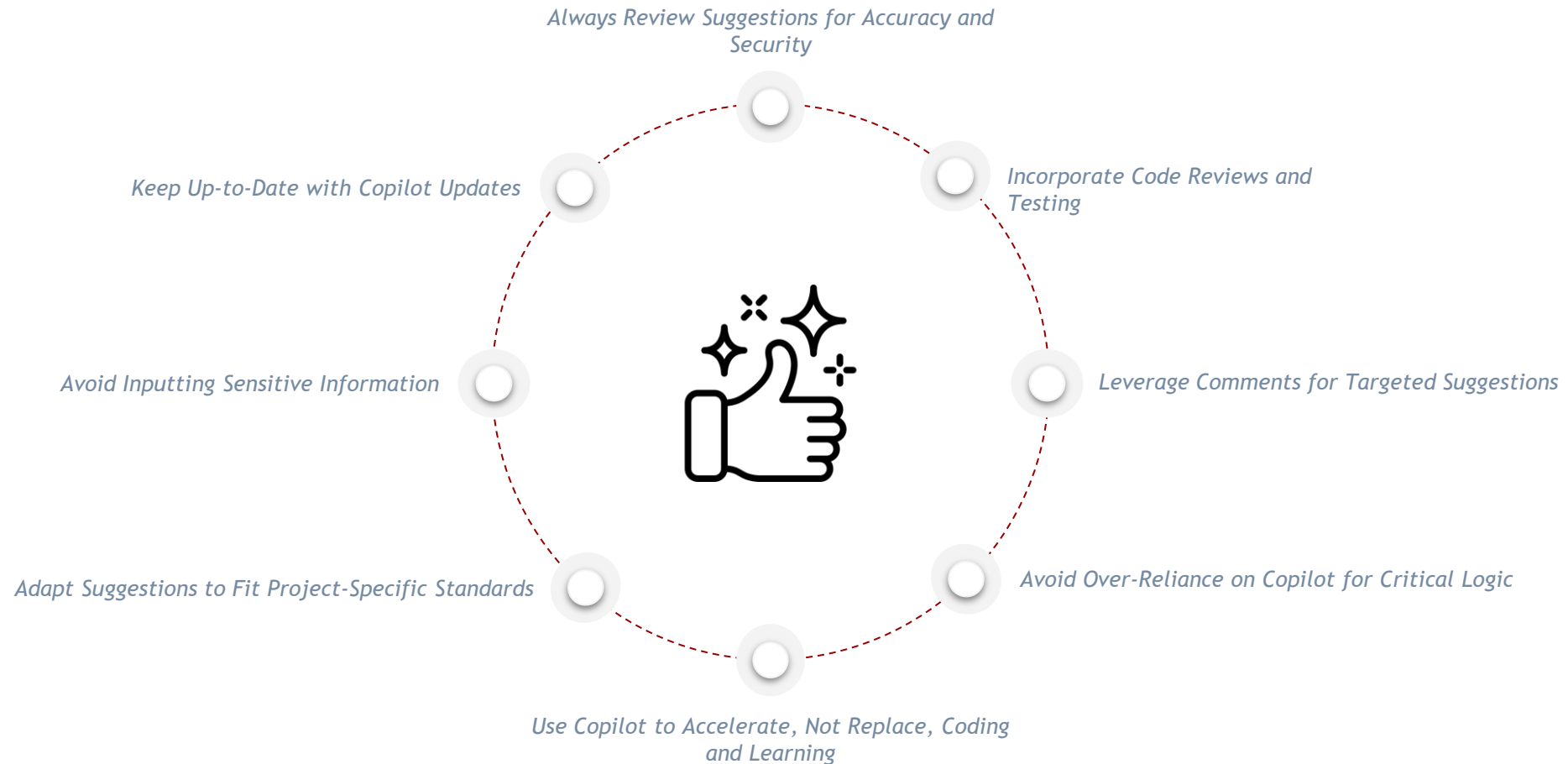
Get Code Suggestion

Security & Ethical Consideration



Best Practices for “AI-Driven Code Assistance”

Optimizing GitHub Copilot Usage: Best Practices for Effective and Secure Code Generation



Otto Group - Elevate Developer Engagement

PROBLEM STATEMENT

Otto Group's **decentralized IT teams** led to security inconsistencies and a lack of visibility across subsidiaries, making it difficult to standardize software security policies

SOLUTION

GitHub Copilot, combined with GitHub Advanced Security, enabled **automated security code reviews**, **secret scanning to prevent credential leaks**, and the **enforcement of standardized security policies** across development teams

<https://github.com/customer-stories/otto-group>

otto group

Sample Prompts

Implementing AI-driven Security Checks

Analyze this Python script for potential security vulnerabilities (e.g., input validation, data handling, authentication) by referencing OWASP best practices. Provide recommended fixes, code snippets for each identified issue, and brief explanations on mitigating risks

Automating Secret Scanning in CI/CD Pipelines

Generate a GitHub Actions workflow that scans commits for exposed API keys using a tool like GitLeaks or truffleHog. Configure the workflow to trigger on commit events, support customizable API key patterns, log detailed notifications for detected secrets, and include inline documentation along with fallback steps for handling false positives

Standardizing Security Policies with AI-powered Best Practices

Generate a JSON policy for role-based access control (RBAC) in GitHub that defines distinct roles (e.g., admin, contributor, viewer) with explicit, least-privilege permissions. Include inline comments to explain each permission group and ensure the policy aligns with GitHub's security best practices

Accenture - Double Down on Github's Platform

PROBLEM STATEMENT

Accenture's vast technical footprint often means there isn't enough time in the day to tackle technical debt

SOLUTION

With GitHub Copilot, Accenture's developers produce **better quality code** faster, leaving time for other endeavors

95%

of developers enjoy coding more when using copilot

67%

of developers use Github Copilot everyday

96%

success rate with Github Copilot amongst users

<https://github.com/customer-stories/accenture>


accenture

Sample Prompts

Enhancing developer experience

Review this Python function and suggest improvements to enhance its readability and maintainability. Provide a refactored version that adheres to PEP8 guidelines, uses clear variable names, includes comprehensive docstrings and inline comments, and simplifies complex logic where possible

Boosting Code Quality

Refactor this JavaScript code to adhere to industry best practices by applying ES6+ syntax, enforcing consistent naming conventions, and modularizing the code. Include clear inline comments that explain changes and ensure the code is optimized for readability, maintainability, and performance



Hands-On Exercise 2



Github Copilot

AI-Assisted Test-Driven Development (TDD) & Code Quality Enforcement

AI-Assisted Test-Driven Development (TDD) & Code Quality Enforcement

Leverage GitHub Copilot for Test-Driven Development (TDD) to write tests before implementing features.

```
Suggest test cases and boundary conditions for the code. Give solution for each issue that may arise.
J LegacyETL.java:24-47

Workspace used /tests (rerun without)
> Used 1 reference

Let me plan the test cases step by step:

1. Test Structure Setup:
• Create LegacyETLTest class with necessary imports
• Setup mock server for HTTP requests
• Create temporary directory for output files

2. Test Cases for fetchSalesData:
• Success case with valid JSON response
• Empty JSON array response
• Invalid JSON response
• Network timeout
• HTTP error responses (404, 500, etc.)
• Connection issues
• Malformed URL

3. Test Cases for saveToFile:
• Success case with valid data
• Empty data list
• File permission issues
• Directory doesn't exist
• File already exists
• Special characters in path
```


- Automate code quality checks and enforce best practices using AI-driven suggestions.
- Generate unit, integration, and end-to-end tests based on function signatures.
- Use Copilot to suggest edge cases and improve test coverage.

Enforcing Code Quality with Copilot

- Automate PR reviews with AI-generated feedback.
- Use Copilot to identify security vulnerabilities and suggest fixes.
- Maintain coding standards consistently with AI-driven linting and refactoring.

```
gh pr create --title "Modernize Java Pipeline" --body "This PR modernizes the legacy Java-based ETL pipeline by upgrading outdated constructs to modern features such as try-with-resources and Java Streams. It replaces System.out.println with SLF4J logging, and implements robust error handling to improve code maintainability and performance." --base main
```

Review this pull request for best practices, security vulnerabilities, and code quality improvements. Check for adherence to coding standards, potential performance optimizations, and missing test coverage. Provide a summary of key issues and suggest actionable improvements.

 **GitHub** (rerun without)

I'll help review the pull request details you've shared. However, I notice that this seems to be a command to create a new PR rather than an existing PR to review. Let me check if there's an existing PR with these details.

```
<plan> 1. searchPullRequests to find PRs in datacouch-io/acme-corp with title "Modernize Java Pipeline" 2. If found, getPullRequest to get detailed information 3. If not found, inform user about the situation </plan>
```

Let me check the pull requests for this repository.

I see that you're using the GitHub CLI command to create a new pull request. Before reviewing any code changes, we'll need:

1. The actual code changes you want to submit in this PR
2. The current branch you're working from

Would you like me to help you:

1. Create this pull request with the title and description you've provided
2. Review specific code changes you have ready
3. Provide best practices for modernizing Java pipelines based on the description you've given

Please let me know which you'd prefer, and I'll be happy to assist further.

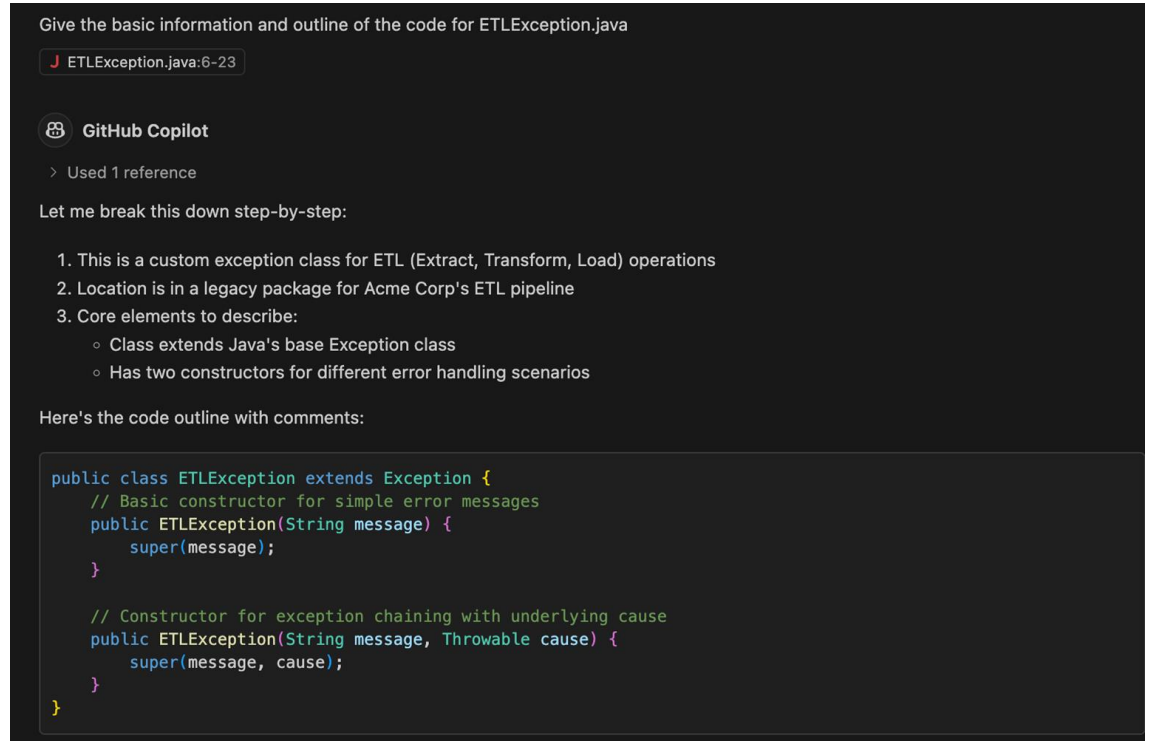


Hands-On Exercise 3

Feature Development with Limited Context


Quickly implement features in unfamiliar codebases using GitHub Copilot.

- Leverage AI to analyze existing code and generate context-aware suggestions.
- Use Copilot Chat to analyze repositories and identify key components.
- Prompt-first approach: refine messy code into structured solutions.



Give the basic information and outline of the code for ETLEException.java

ETLEException.java:6-23

 **GitHub Copilot**

> Used 1 reference

Let me break this down step-by-step:

1. This is a custom exception class for ETL (Extract, Transform, Load) operations
2. Location is in a legacy package for Acme Corp's ETL pipeline
3. Core elements to describe:
 - Class extends Java's base Exception class
 - Has two constructors for different error handling scenarios

Here's the code outline with comments:

```
public class ETLEException extends Exception {  
    // Basic constructor for simple error messages  
    public ETLEException(String message) {  
        super(message);  
    }  
  
    // Constructor for exception chaining with underlying cause  
    public ETLEException(String message, Throwable cause) {  
        super(message, cause);  
    }  
}
```

Feature Development with Limited Context

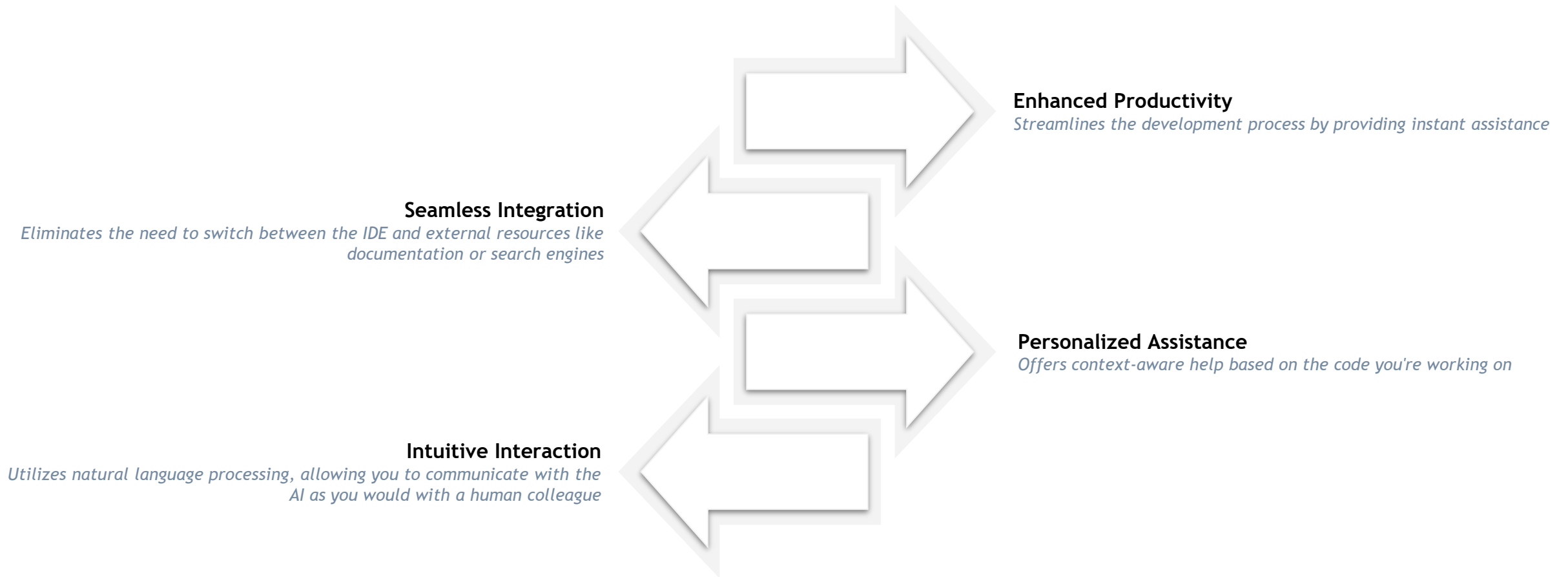
Use Copilot Chat in your editor to give code suggestions, explain code, generate unit tests, and suggest code fixes

- "Chat in IDE" refers to an interactive chat feature integrated within your Integrated Development Environment (IDE)
- Specifically, with tools like GitHub Copilot, it provides an AI-powered conversational assistant directly inside your coding environment
- This feature allows developers to engage in a dialogue with the AI to receive code suggestions, explanations, and assistance without leaving the IDE



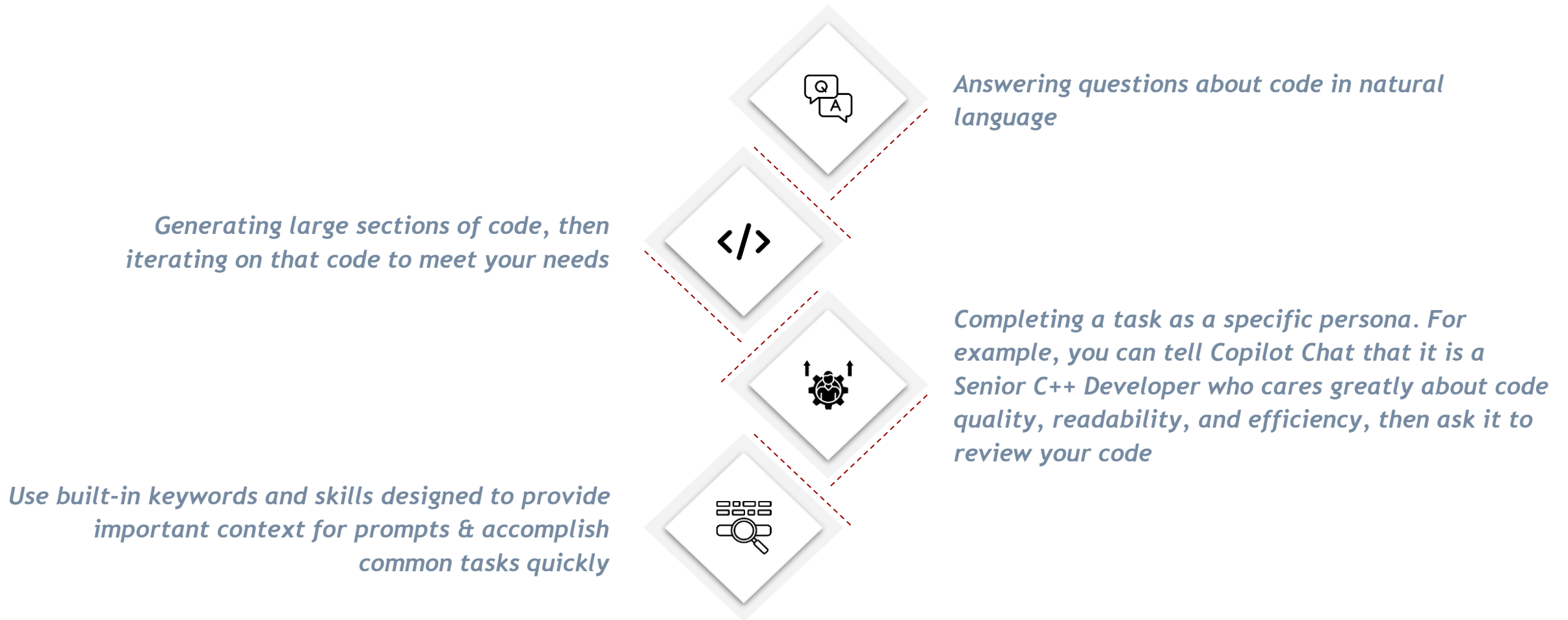
Use “Chat in IDE” for Feature Development

“Chat in IDE” refers to an interactive chat feature integrated within your Integrated Development Environment (IDE)



“Chat in IDE” is best suited for

Know the benefits



Using keywords in your prompt

Chat Participants

Use chat participants to scope your prompt to a specific domain. To use a chat participant, type @ in the chat prompt box, followed by a chat participant name

1

@workspace

Has context about the code in your workspace. Use @workspace when you want Copilot to consider the structure of your project, how different parts of your code interact, or design patterns in your project

2

@vscode

Has context about Visual Studio Code commands and features. Use it when you want help with Visual Studio Code

3

@terminal

Has context about the Visual Studio Code terminal shell and its contents. Use it when you want help creating or debugging terminal commands

Additional Ways to access Github Copilot Chat

Get suggestions for code at your fingertips



INLINE

QUICK
CHAT

SMART
ACTIONS

Additional Ways to access Github Copilot Chat

Get suggestions for code at your fingertips

INLINE

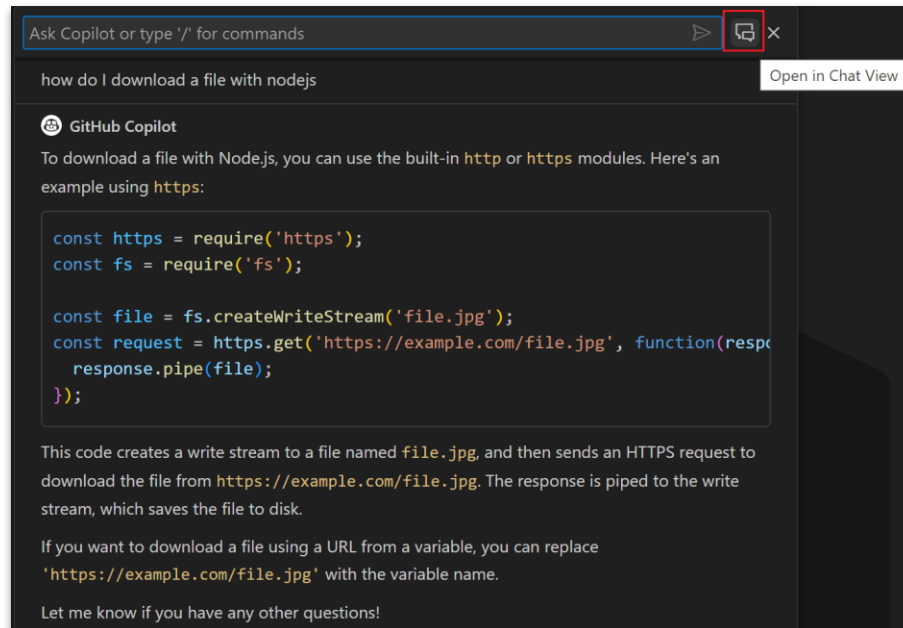
```
2
3 def sort(arr):
4
5     convert this to bubble sort
6
7     GitHub Copilot
8     Made changes.
9     Accept Discard Accept or discard 1 change
10
11     n = len(arr)
12     for i in range(n):
13         for j in range(0, n-i-1):
14             if arr[j] > arr[j+1]:
15                 arr[j], arr[j+1] = arr[j+1], arr[j]
16     return arr
```

```
what sorting algo is this
GitHub Copilot used /explain (rerun without)
> Used 1 reference
The code you provided is an implementation of the QuickSort algorithm. QuickSort is a popular sorting algorithm known for its efficiency and average-case time complexity of  $O(n \log n)$ . It works by selecting a
def sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot_index = random.randint(0, len(arr) - 1)
        pivot = arr[pivot_index]
        arr[0], arr[pivot_index] = arr[pivot_index], arr[0]
        less, greater = partition(arr[1:], pivot)
        return sort(less) + [pivot] + sort(greater)
def partition(arr, pivot):
    less = [x for x in arr if x <= pivot]
    greater = [x for x in arr if x > pivot]
    return less, greater
```

Additional Ways to access Github Copilot Chat

Get suggestions for code at your fingertips

QUICK
CHAT



Ask Copilot or type '/' for commands

how do I download a file with nodejs

GitHub Copilot

To download a file with Node.js, you can use the built-in `http` or `https` modules. Here's an example using `https`:

```
const https = require('https');
const fs = require('fs');

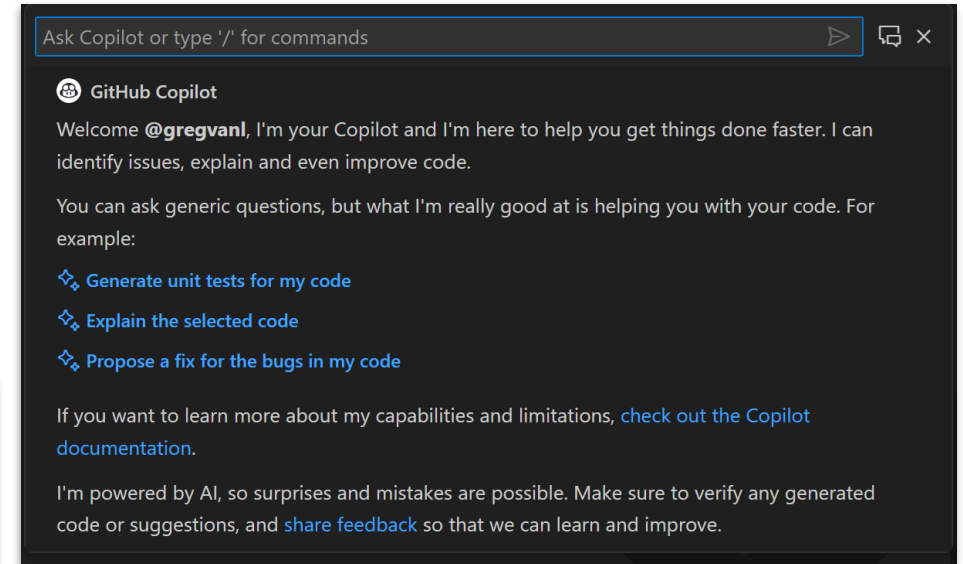
const file = fs.createWriteStream('file.jpg');
const request = https.get('https://example.com/file.jpg', function(response) {
  response.pipe(file);
});
```

This code creates a write stream to a file named `file.jpg`, and then sends an HTTPS request to download the file from `https://example.com/file.jpg`. The response is piped to the write stream, which saves the file to disk.

If you want to download a file using a URL from a variable, you can replace `'https://example.com/file.jpg'` with the variable name.

Let me know if you have any other questions!

Open in Chat View



Ask Copilot or type '/' for commands

GitHub Copilot

Welcome @gregvanl, I'm your Copilot and I'm here to help you get things done faster. I can identify issues, explain and even improve code.

You can ask generic questions, but what I'm really good at is helping you with your code. For example:

- Generate unit tests for my code
- Explain the selected code
- Propose a fix for the bugs in my code

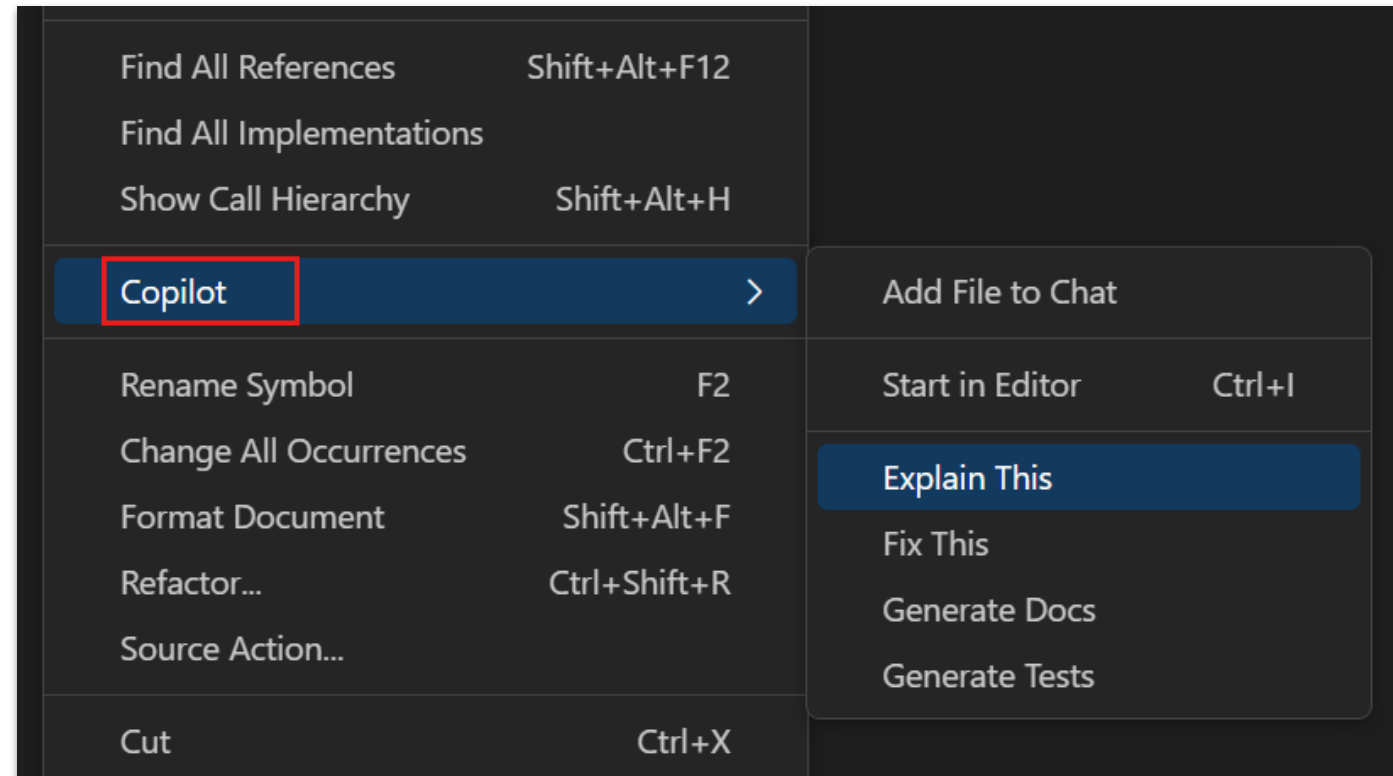
If you want to learn more about my capabilities and limitations, [check out the Copilot documentation](#).

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.

Additional Ways to access Github Copilot Chat

Get suggestions for code at your fingertips

SMART
ACTIONS






Hands-On Exercise 4

Intelligent Code Refactoring & Performance Optimization

- Use GitHub Copilot

Optimise the code and check for any possible bugs in the code.

tradingviewwidget.jsx:21-39

 **GitHub Copilot**

> Used 1 reference

Let me analyze the code step by step and suggest optimizations:

1. Analysis Steps:

- Review dependency usage
- Check useEffect cleanup
- Verify script loading
- Examine styling approach
- Validate memo usage
- Check accessibility

2. Issues Found:

- Missing cleanup in useEffect
- No loading state handling
- Hardcoded styles
- No error handling
- Missing prop-types validation

Here's the optimized code:

```
import { memo, useEffect, useRef, useState } from 'react';
import PropTypes from 'prop-types';

const TradingViewWidget = ({ height = '100%', width = '100%' }) => {
  const container = useRef(null);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    // Optimized code logic
  }, []);
}
```

What other optimizations can we apply?

Intelligent Code Refactoring & Performance Optimization

- Key Benefits of AI-Assisted Refactoring:
 - **Code Simplification:** Identify redundant logic, remove unnecessary complexity, and enhance readability.
 - **Performance Boost:** Optimize loops, recursive functions, and database queries for better execution speed.
 - **Error Reduction:** Identify memory leaks, inefficient data structures, and bottlenecks before they impact performance.
 - **Scalability Enhancements:** Ensure the codebase is future-proofed for larger workloads

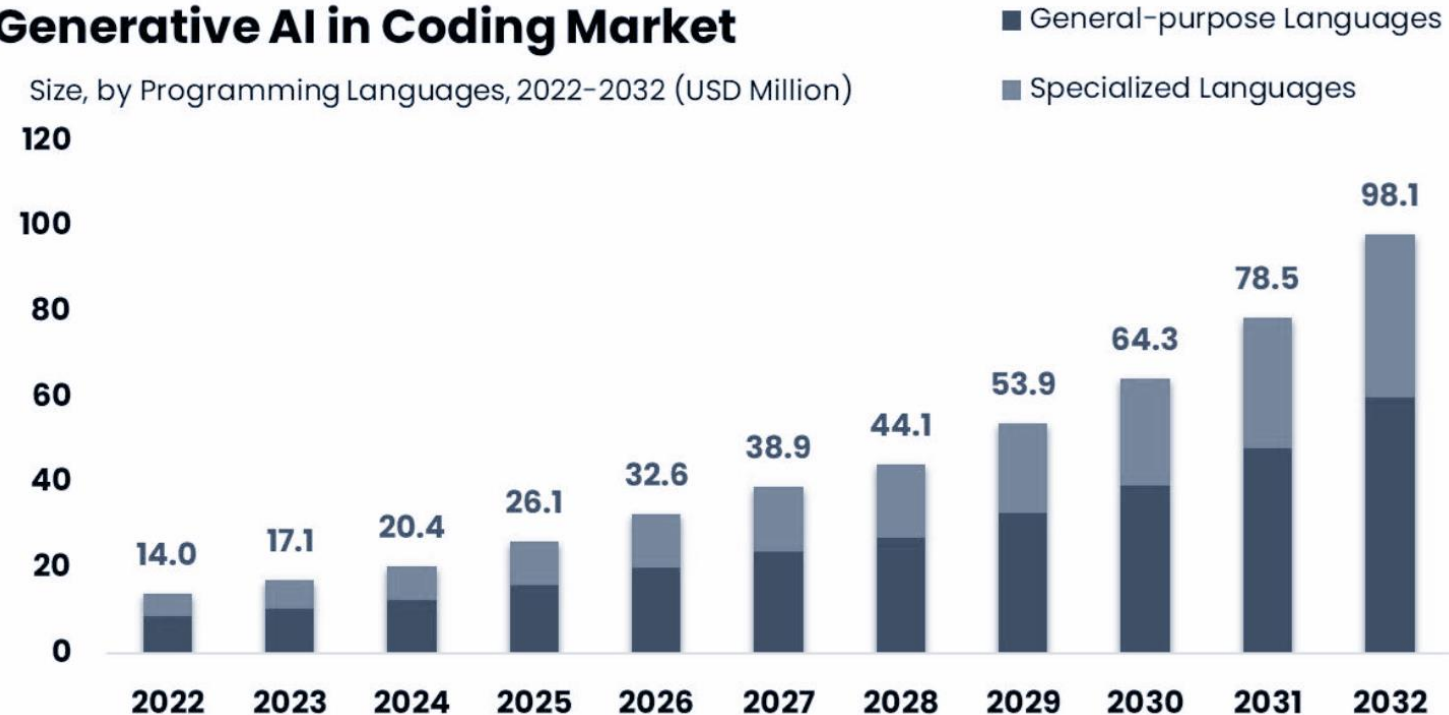


Intelligent Code Refactoring & Performance Optimization

Need for Automation & Vision of AI in Coding

Generative AI in Coding Market

Size, by Programming Languages, 2022-2032 (USD Million)



Source: [MarketResearch.Biz](https://www.marketresearch.biz)

GenAI in coding Market will
grow at the CAGR of

22.1%

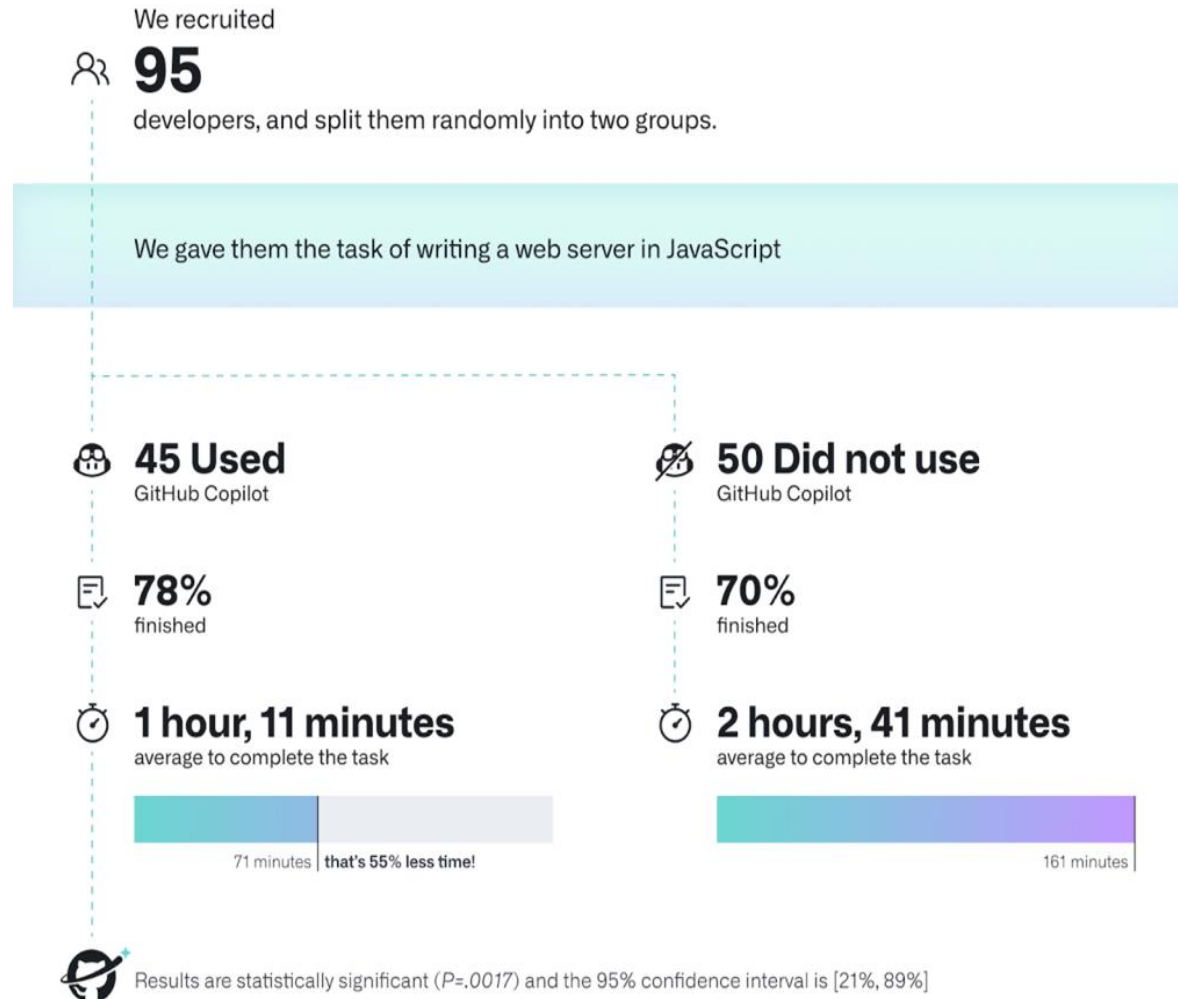
As per BCG,

30-50%

productivity boost can be achieved in software development by integrating automation tools, leading to faster coding, testing, and deployment with fewer manual errors and higher quality results

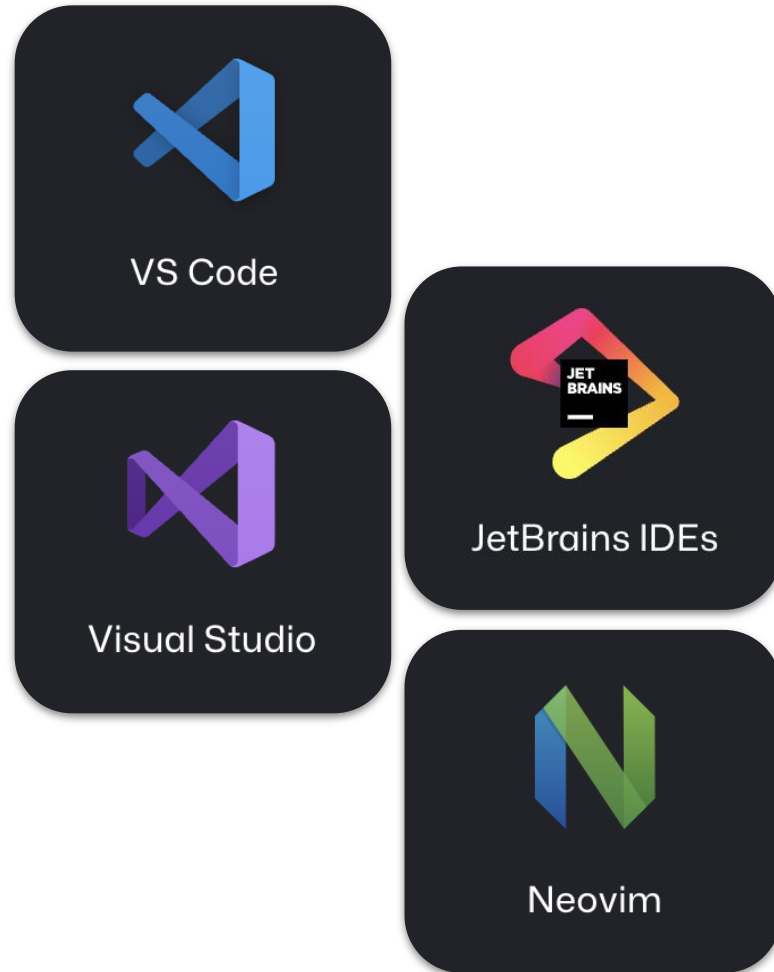
Performance Optimization

This AI coding assistant is changing the coding paradigm



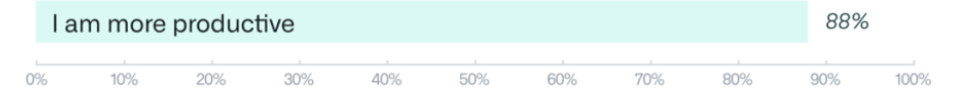
Keep flying with your favourite editor

Ask for assistance right in your IDE

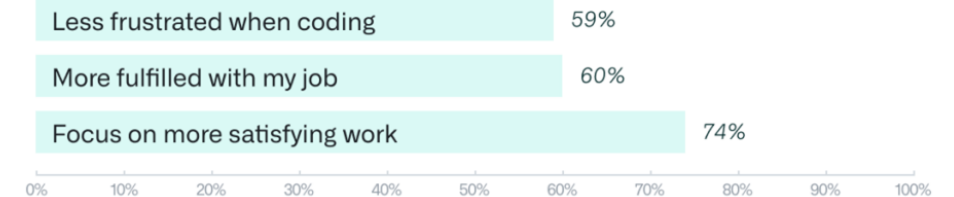


When using GitHub Copilot...

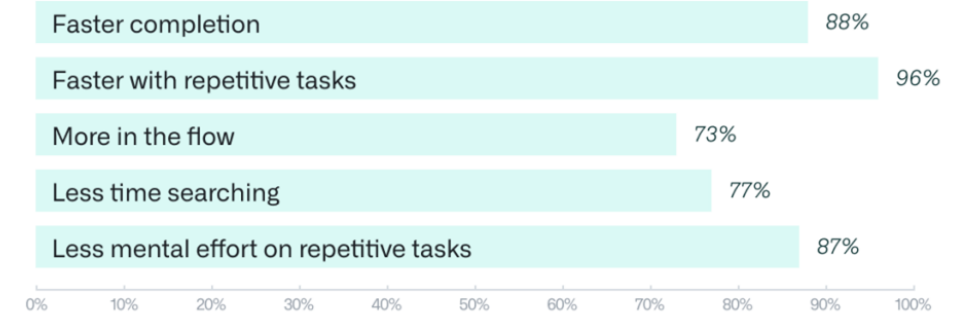
Perceived Productivity



Satisfaction and Well-being*



Efficiency and Flow*



Source:
https://visualstudiomagazine.com/articles/2022/09/13/~/_media/ECG/visualstudiomagazine/Images/2022/09/Copilot1.ashx



Hands-On Exercise 5

Indra - Github Copilot helps to focus on making world safer

PROBLEM STATEMENT

As competition for top talent heated up during the pandemic, Indra searched for ways to attract developers and give them more opportunities to focus on higher value work

SOLUTION

The company adopted GitHub Copilot to **handle boilerplate and accelerate development**, increasing job satisfaction and efficiency among existing talent and making it more attractive for new hires

30%

less time on boilerplate code

20%

productivity boost when developing new features

20%

more time to solving new complex problems

<https://github.com/customer-stories/indra>

indra

Sample Prompts

Automating Repetitive Tasks

Generate boilerplate code for a REST API in Node.js using Express and ES6 syntax. Include endpoints for CRUD operations, robust error handling, middleware for logging and security, environment variable management, and clear inline documentation

Improving Developers Productivity

Generate a Python script that automates data processing tasks by reading a CSV file, cleaning & transforming the data (handling missing values, type conversions, and filtering records), & writing the output to a new CSV file. Use modular functions, include error handling, logging, & command-line argument parsing with argparse. Provide comprehensive docstrings and inline comments for clarity and maintainability

Carlsberg - Unifies Development & accelerate innovation

PROBLEM STATEMENT

Carlsberg's development processes were impeded by a lack of unified tooling and forced context-switching, alongside potential security threats due to undetected secrets within their source code

SOLUTION

*Carlsberg unified its development process and bolstered its security using **GitHub Advanced Security**, while the integration of Copilot enabled efficient coding practices and minimized the need for context-switching*

600

secrets detached in source code with Github Secret Scanning

30000+

security vulnerabilities removed since activating Github Advanced Security

6

tools reduced to a single platform for software development with Github Enterprise



<https://github.com/customer-stories/carlsberg-group>

Sample Prompts

Unified Development on Github Enterprise

Generate a standardized CI/CD pipeline for microservices using GitHub Actions. Include stages for building, testing, and deploying to a Kubernetes cluster, parameterized environment variables, robust error handling, and notifications on failures. Provide inline documentation and adhere to industry best practices for maintainability and scalability

Accelerating Innovation

Suggest an AI model architecture for demand forecasting using deep learning. Include a data preprocessing pipeline to handle time series data (trends, seasonality, and external factors), and design a sequential neural network—such as an LSTM or Transformer—for capturing temporal dependencies. Provide a high-level diagram, component descriptions, and recommendations for hyperparameter tuning and evaluation metrics

Intelligent Code Refactoring & Performance Optimization

1

Automating Documentation with AI

- Copilot generates docstrings, inline comments, and summaries automatically.
- Ensures consistent documentation across projects, improving code readability.

2

Faster Code Understanding & Onboarding

- AI analyzes repositories to provide concise overviews of key components.
- Automates README generation and highlights dependencies & architecture.

3

Enhancing Collaboration & Knowledge Sharing

- Standardizes documentation, making code easier to maintain and share.
- Reduces reliance on manual updates, improving developer efficiency.



Intelligent Code Refactoring & Performance Optimization

Automatic Inline Code Comments


```
<p className={`font-semibold ${stock.percentChange >= 0 ? 'text-green-600' : 'text-red-600'}`}>  
  Change: {stock.percentChange || 'Loading...'}%  
  // If the percentChange is greater than 0, the text will be green, otherwise it will be red.  
</p>  
</div>
```

Intelligent Code Refactoring & Performance Optimization

Faster Code Understand

Create a short summary of this project.

Codebase





 **Workspace** (rerun without)

> Used 29 references

- ✓ Searching codebase for "project summary"...
- ✓ Collecting workspace information
- ✓ Filtering to most relevant information

Based on the code, this appears to be a stock market analysis and prediction web application with:

Frontend (React):

- A main dashboard showing a TradingView widget chart
- Components for displaying stock data:
 -  **CandlestickChart** for visualizing stock price movements
 -  **StockCard** showing basic stock info
 -  **StockForm** for user input
 -  **StockRecommendation** for displaying trading signals

Backend (Python):

- Flask server in app.py that:
 - Fetches stock data using yfinance
 - Implements machine learning prediction model
 - Provides API endpoints for stock data and predictions
- Uses a trained Keras model for predictions
- Stores static assets like datasets and plots

Phillips - Build & Deploy applications faster with innersourcing on Github

PROBLEM STATEMENT

Philips, a global health technology leader, struggled with **siloed toolchains**, preventing **code reuse** and collaboration across its business units. The lack of a **unified development environment** slowed down innovation.

SOLUTION

Philips integrated GitHub Copilot with GitHub Enterprise to foster an **inner source development culture**, **accelerate onboarding** with Copilot-generated best practices, and **automate infrastructure as code (IaC) provisioning** through AI-driven prompts

80%

reduction in infrastructure costs from self-hosted runners

85%

repos shareable with entire software organization

70%

developers reported innersource ecosystem improved developer experience

<https://github.com/customer-stories/philips>

PHILIPS

Sample Prompts

Enhancing Code Reuse with AI-generated Templates

Generate a reusable Python microservice template for healthcare data processing using the FastAPI framework. Ensure compliance with HIPAA guidelines by securely handling patient data, include Docker configurations for containerization, and provide robust docstrings, error handling, and basic unit tests for critical functions

Automating DevOps Infrastructure Setup

Create a Terraform script that provisions an AWS EC2 instance (Amazon Linux 2) with minimal inbound/outbound rules, key-based SSH access, and a least-privileged IAM role. Include inline commentary, parameterized variables, and maintain a clear resource structure

5

Create a PyTorch deep learning model for medical image classification, leveraging transfer learning and data augmentation. Include clear docstrings, basic unit tests, and ensure secure data handling

Travelport - Transforming Complexity to Consistency

PROBLEM STATEMENT

Travelport, a global travel technology company, had **1,600+ applications** and **6,000+ repositories** with multiple CI/CD tools, causing inefficiencies in managing software pipelines

SOLUTION

Travelport integrated GitHub Copilot **streamlined development** by helping developers **quickly understand APIs**, simplifying CI/CD pipeline configurations, and **reducing boilerplate code** for automated migrations

4000

repos migrated to new EMU instance in just a few days with GitHub Enterprise Importer

<https://github.com/customer-stories/travelport>

1

developer manages CI/CD pipeline of 1,000+ repos with GitHub Actions



Sample Prompts

Accelerating API Integration with AI-generated Code

Create a Node.js API client to fetch flight booking data from Travelport's API, using environment variables for authentication and robust error handling. Include JSDoc comments, asynchronous patterns (async/await), and a brief usage example for easy implementation

Simplifying Microservices Deployment with CI/CD Pipelines

Create a Kubernetes deployment YAML for a microservices architecture with multiple services, specifying resource requests/limits, readiness/liveness probes, and environment variables. Use clear labels/annotations, version each service, and include a Horizontal Pod Autoscaler for scaling

CloudZero - Five minutes away from Production

PROBLEM STATEMENT

As a small team committed to rapid production and customer delight, CloudZero sought a comprehensive platform to stay close to production with agile development practices

SOLUTION

*Through GitHub's unified platform, CloudZero centralized **development tooling, accelerated innovation, and established a streamlined developer environment** to promote success at scale*

4x

faster at coding when
using Copilot

300%

faster at bug fixes when
using copilot

99%

of deployments just take
five minutes to production

<https://github.com/customer-stories/cloudzero>



CLOUDZERO

Sample Prompts

Accelerate coding speed

Generate a boilerplate code to build a real-time data processing module. Ensure the code follows agile development practices by using modular functions, comprehensive inline documentation, unit tests, and robust error handling

Expedite Bug Fixes

Analyze the following faulty microservice component code. The component is expected to process API requests and return JSON responses with comprehensive logging and standardized error handling. Identify any bugs—such as issues with asynchronous operations, input validation, or missing error captures—and suggest code fixes. Ensure your recommendations include enhancements to logging practices (with appropriate log levels and detailed error messages) and robust error handling mechanisms

Accelerating deployment with Github Actions

Generate a GitHub Actions workflow configuration for automated deployments in a fast-paced production environment. The workflow should trigger on code commits, manage environment variables securely, include error handling and logging, and feature inline documentation to ensure that deployments reach production within five minutes



Hands-On Exercise 6

A low-angle, upward-looking photograph of several modern skyscrapers with glass facades. The image is overlaid with a semi-transparent dark grey horizontal band across the middle. Three solid red rectangular blocks are positioned: one at the top center, one at the bottom left, and one at the bottom right. The text 'THANK YOU' is centered within the dark band in a white, bold, sans-serif font.

THANK YOU