Prompt Engineering: Github Copilot

Making your code Intelligent

Evolution of Coding

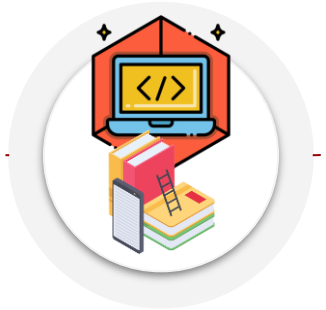
From manual coding to AI-assisted Development

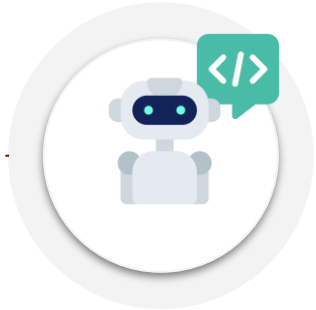
1970s

Handwritten code and punch cards

Integrated Development Environments (IDEs) and code libraries

1990s





2020s

AI is revolutionizing the way we write code

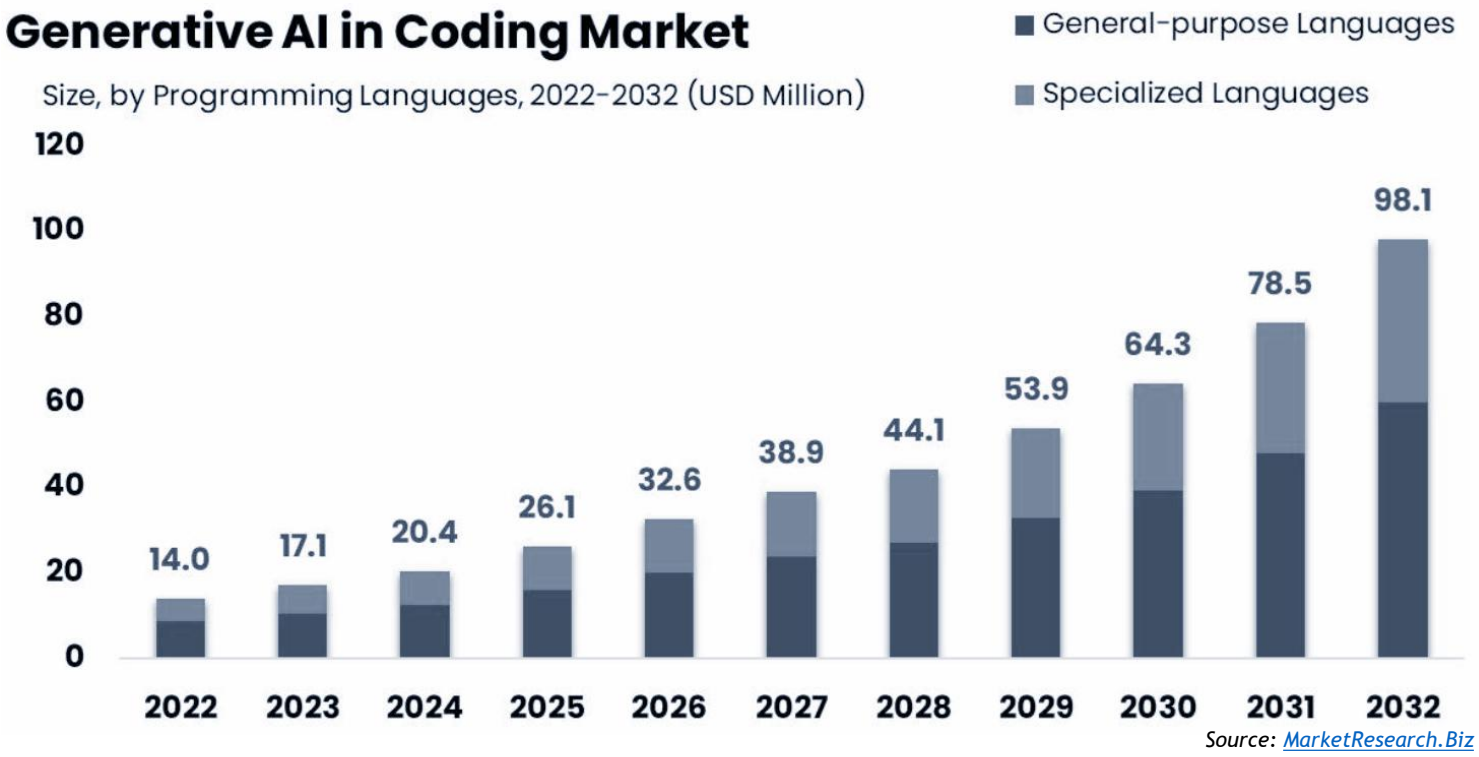
Challenges Developers Face Today

The Coding Pain Points

- *Tedious repetition of boilerplate code*
- *Keeping up with multiple programming languages*
- *Difficulty in debugging & understanding existing code*
- *Long development cycles and tight deadlines*

Need for Automation & Vision of AI in Coding

Can AI make developers more efficient?



Source: 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

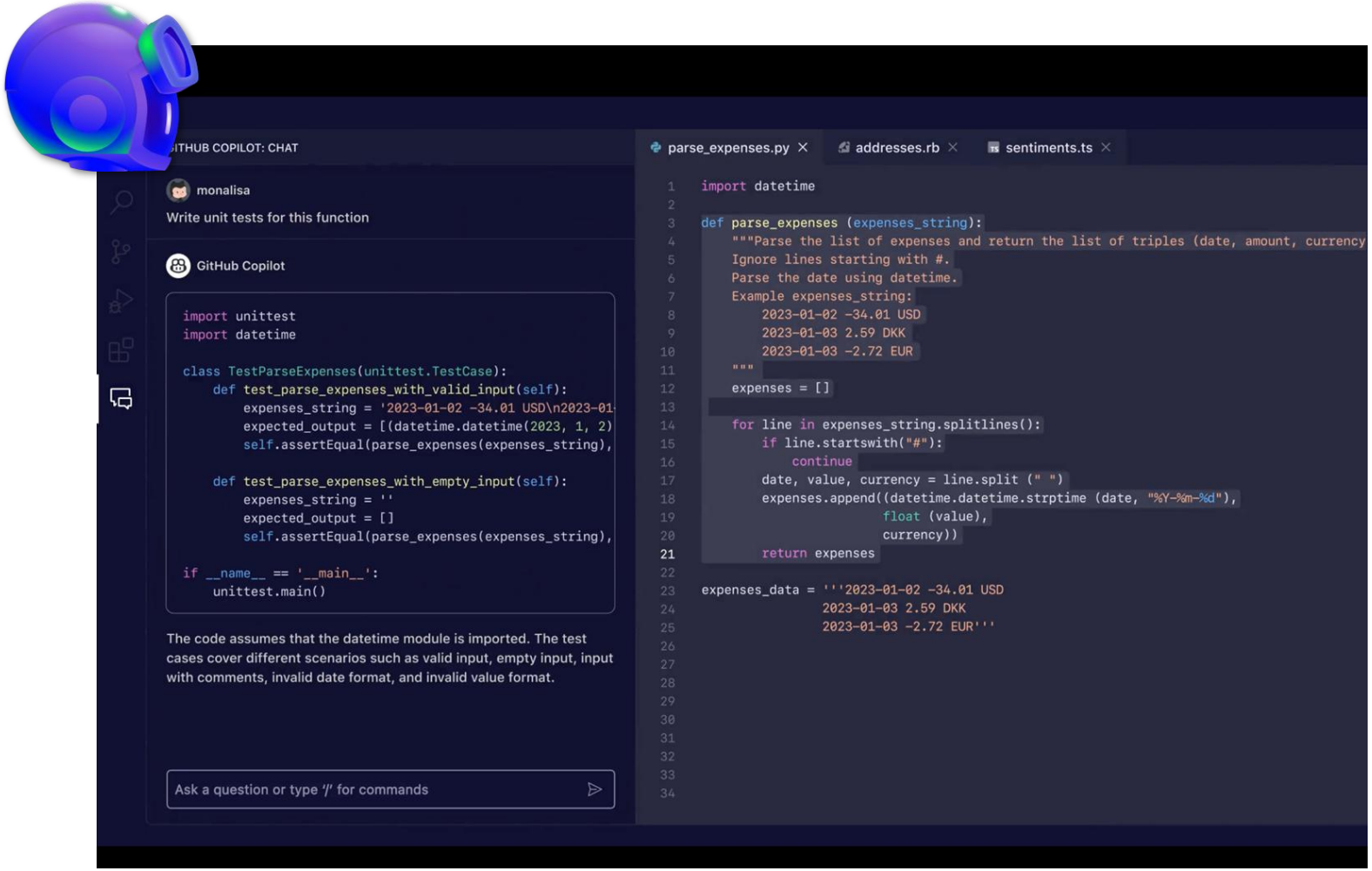
Introducing Github Copilot

Meet Your AI Coding Partner: GitHub Copilot

- GitHub Copilot is the world's first at-scale AI developer tool that can help you write code faster with less work
- GitHub Copilot draws context from comments and code to suggest individual lines and whole functions instantly
- GitHub Copilot helps developers code faster, focus on solving bigger problems, stay in the flow longer, and feel more fulfilled with their work

Introducing Github Copilot

Meet Your AI Coding Partner: GitHub Copilot



Introducing Github Copilot

Meet Your AI Coding Partner: GitHub Copilot

50,000+

Businesses have adopted GitHub Copilot

1 in 3

Fortune 500 companies are using Github copilot

55%

of developers preferred Github Copilot over others

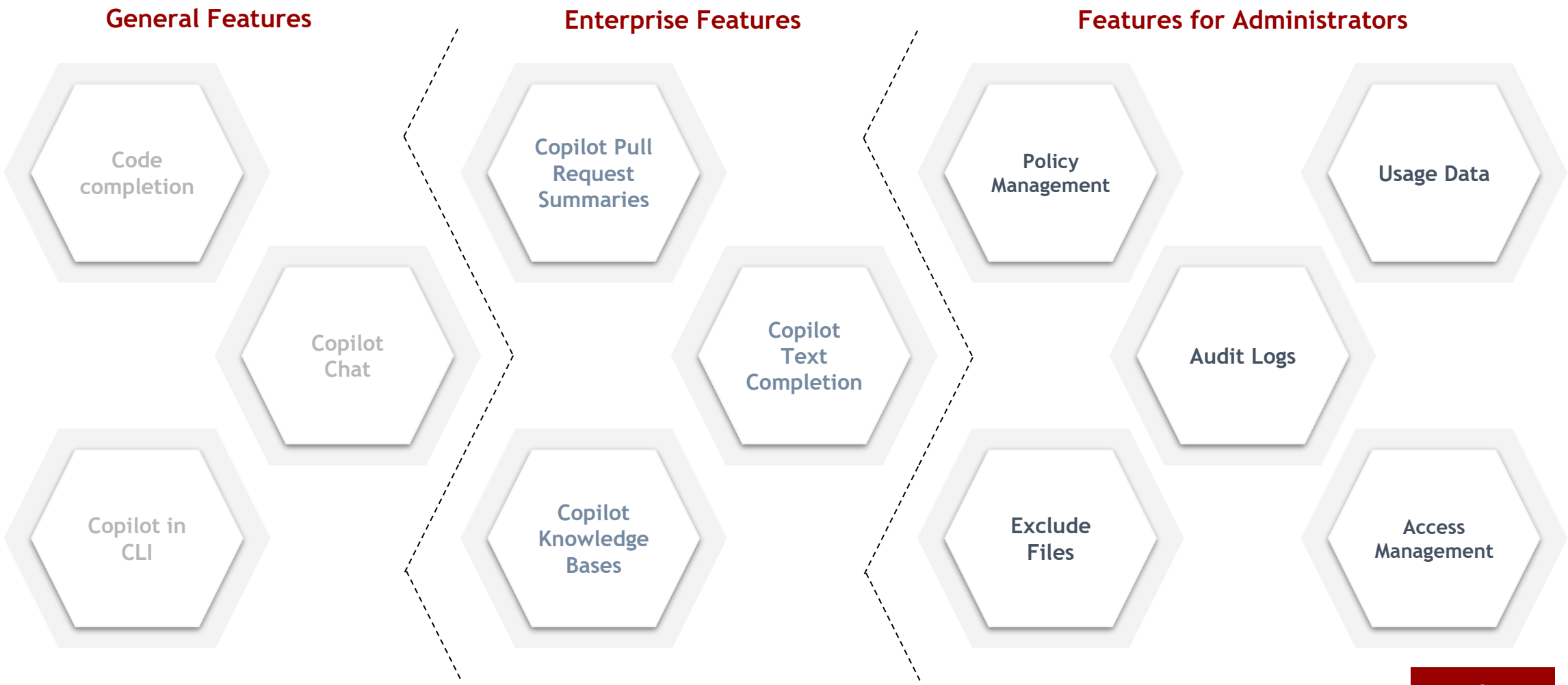
Features of Github Copilot

The AI coding assistant changing coding paradigm

1. *Get code suggestions as you type in your IDE*
2. *Chat with Copilot to ask for help with your code*
3. *Ask Copilot for help using the command line*
4. *Generate a description of the changes in a pull request*
5. *Create & manage collections of documentation, called knowledge bases, to use as a context for chatting with Copilot*

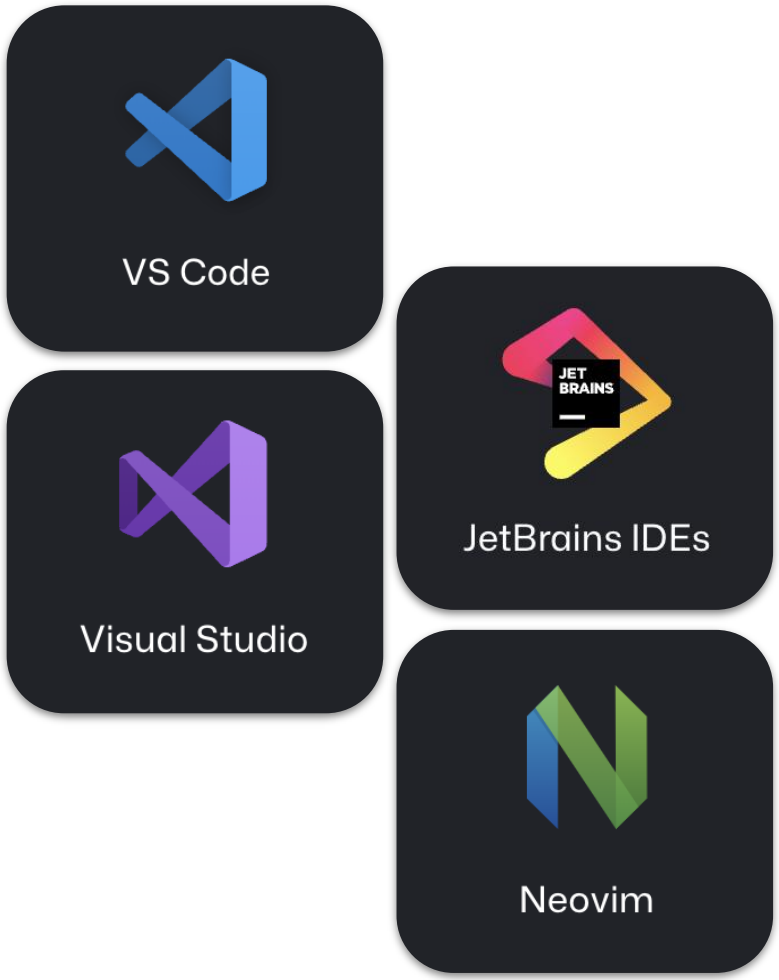
Features of Github Copilot

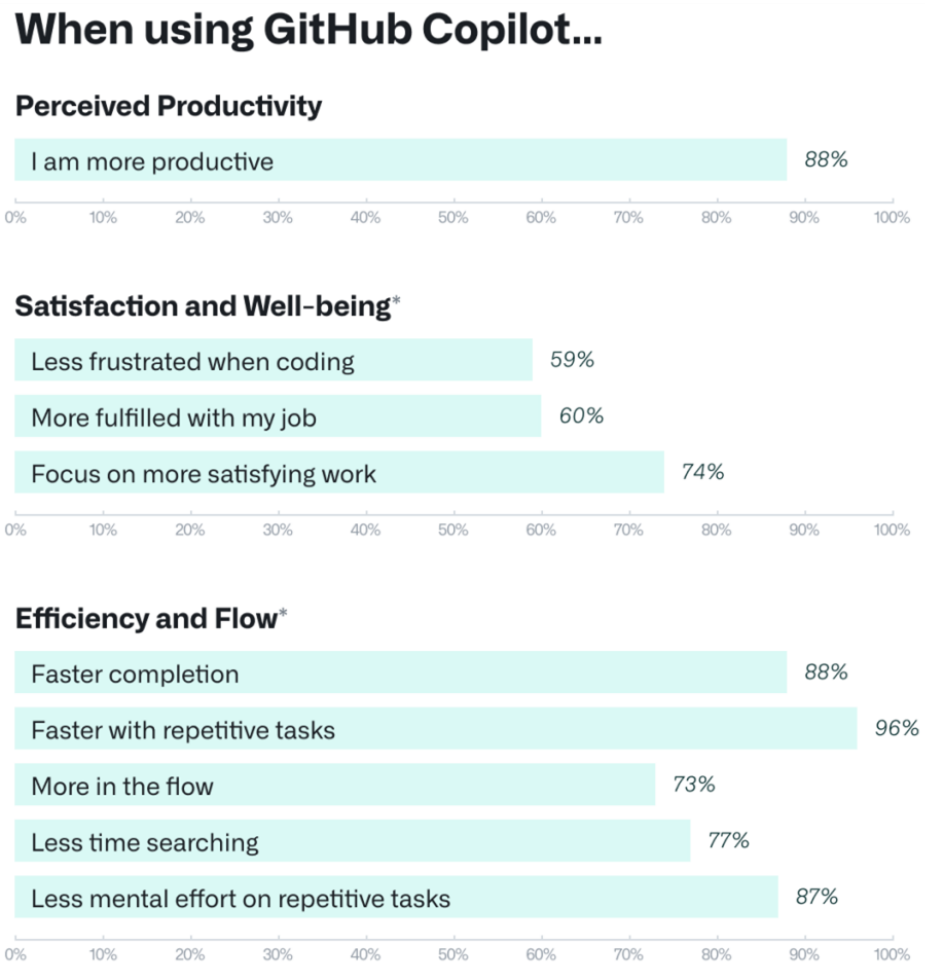
The AI coding assistant changing coding paradigm



Keep flying with your favourite editor

Ask for assistance right in your IDE

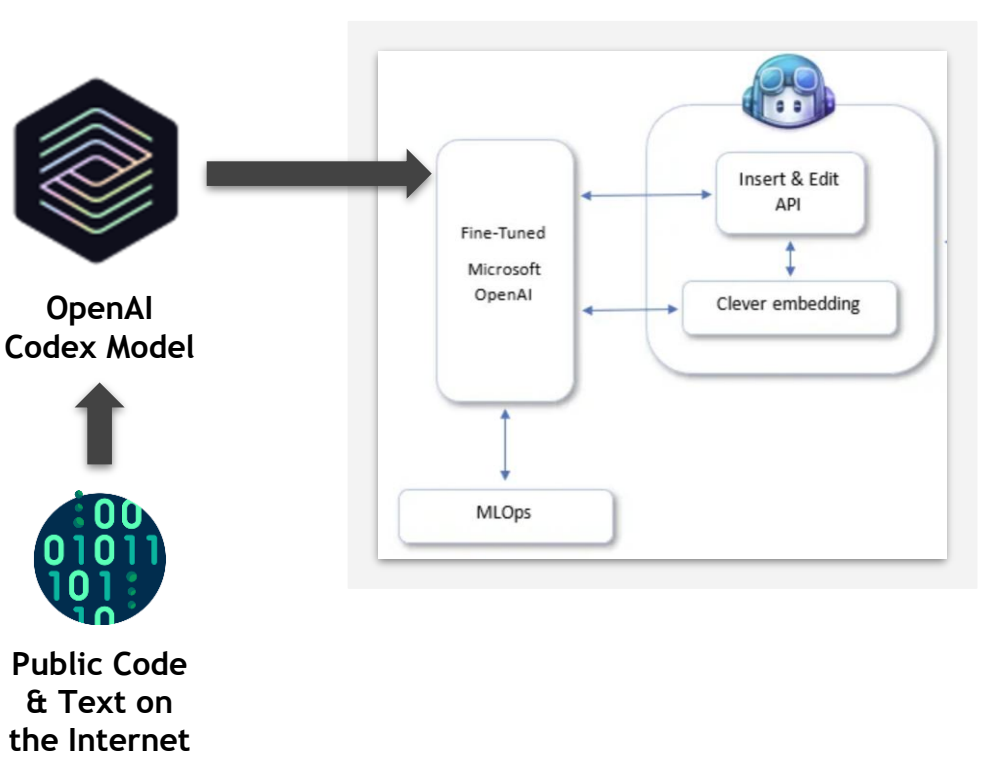


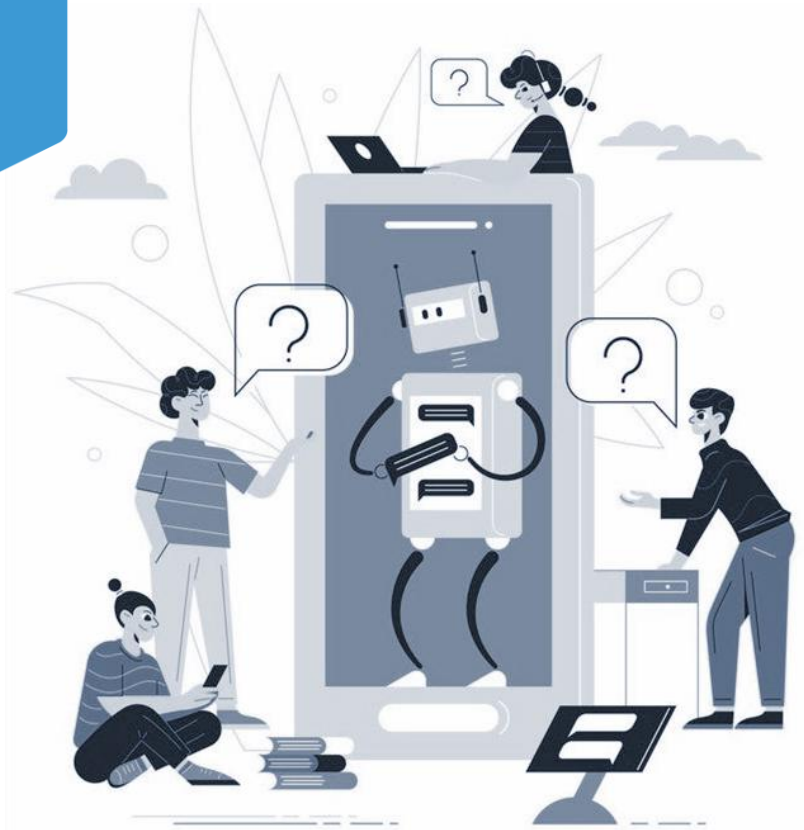


Source:
https://visualstudiomagazine.com/articles/2022/09/13/~/media/ECG/visualstudiomagazine/Images/2022/09/Copilot1.ashx

How does Github Copilot works?

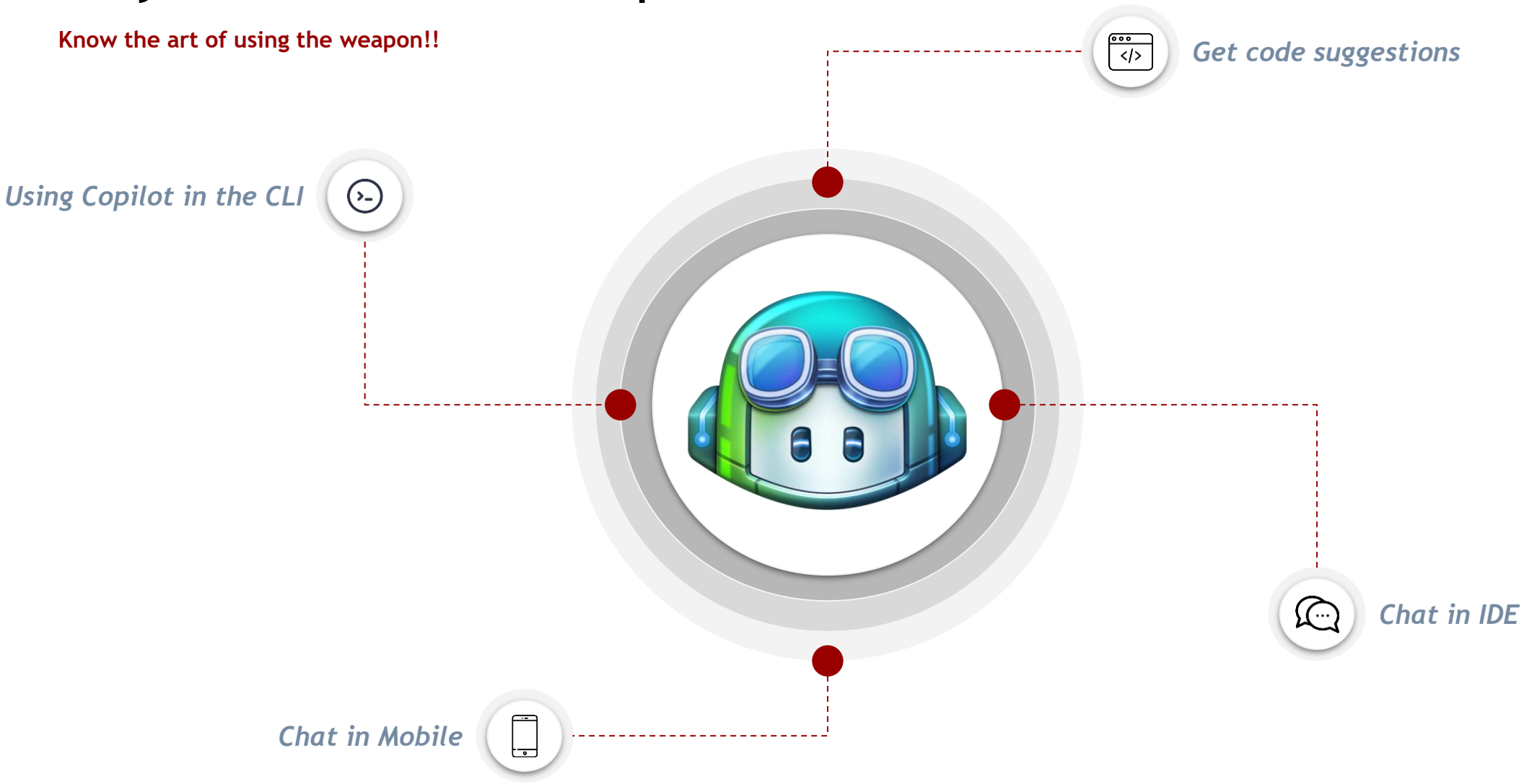
Know how magic is happening!!





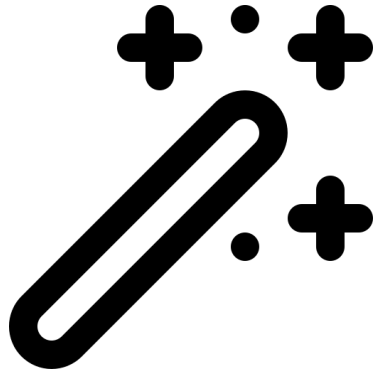
Ways to use Github Copilot

Know the art of using the weapon!!



Common Use Cases of “Get Code Suggestion”

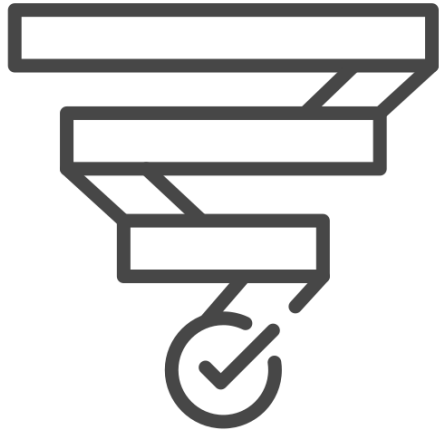
Building a Full CRUD API using Flask and Github Copilot



- *Initialize Flask Project*
- *Use Github copilot to generate boilerplate code*
- *Create data models for resources*
- *Implement CRUD operation for API*
- *Test API Endpoints*
- *Deploy and Document API*

Common Use Cases of “Get Code Suggestion”

Making Data Processing Pipelines with Pandas and Github Copilot



Initialize Data Processing Pipeline

Load CSV

Use Github Copilot to Clean, Transform & Process data

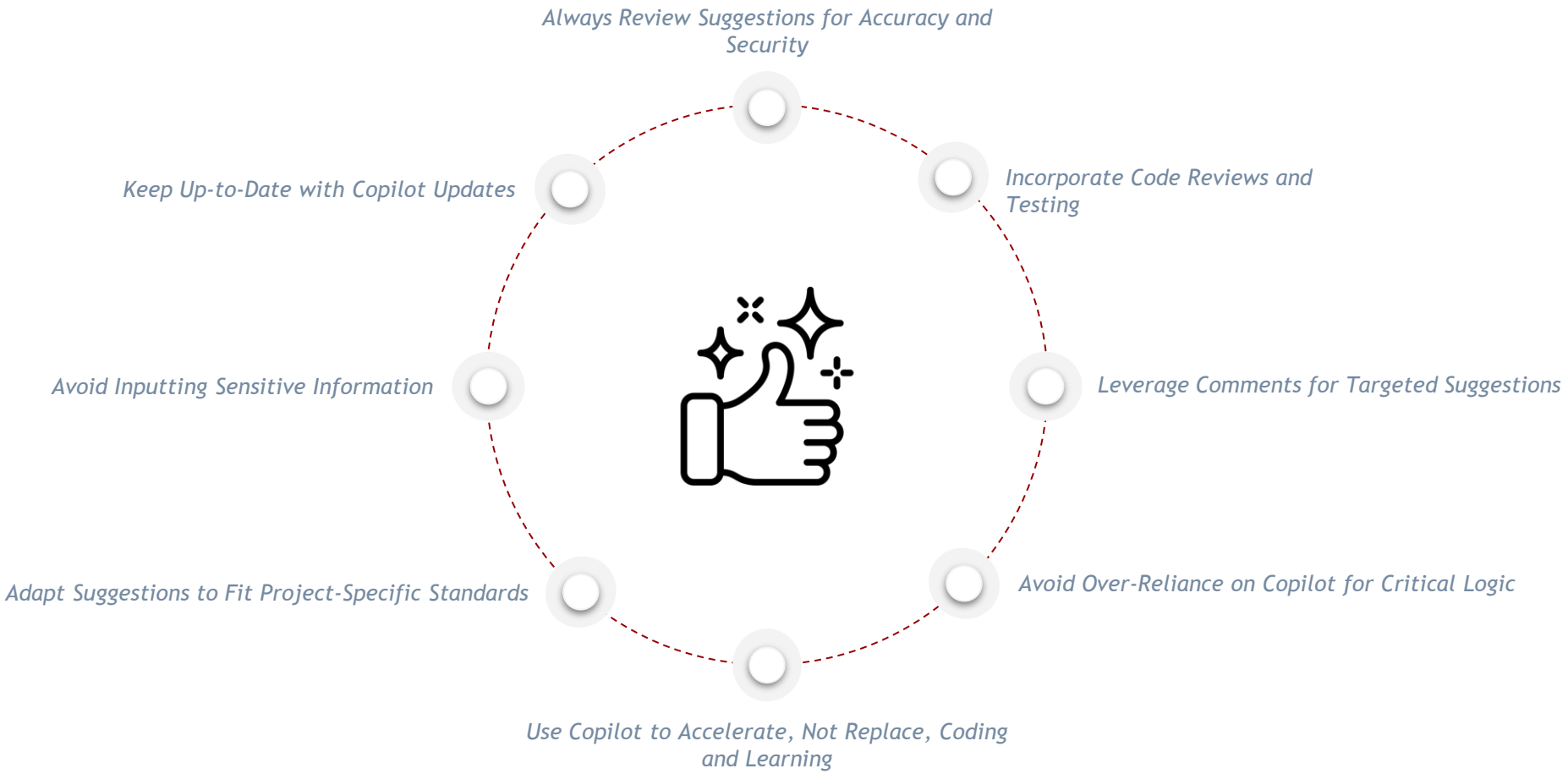
Generate Summary Statistics

Visualize the Data

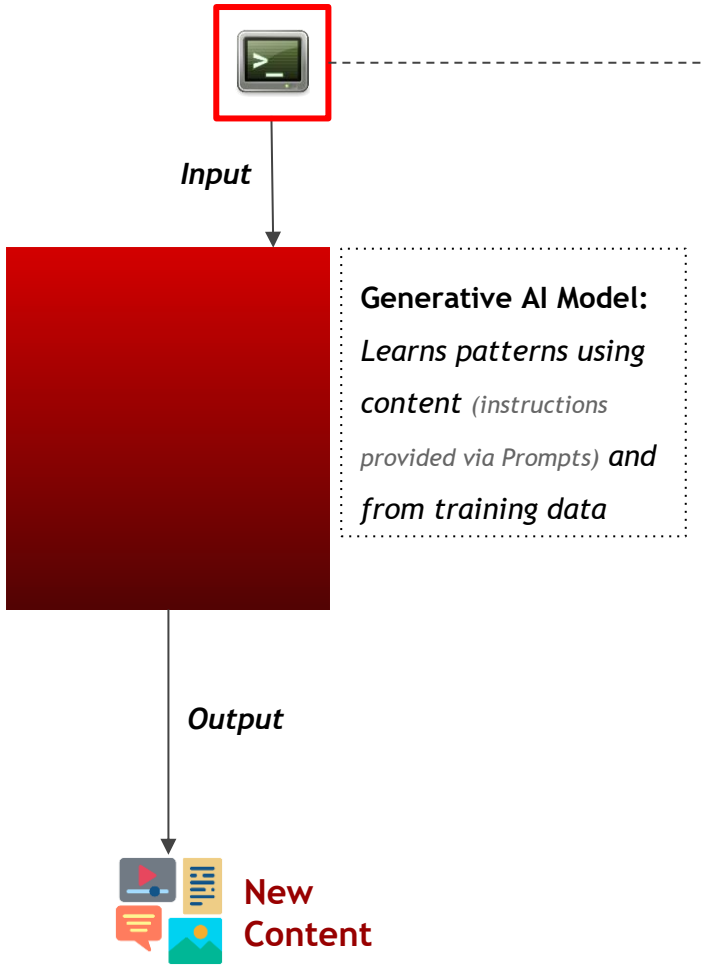
Document the Pipeline

Best Practices for “Get Code Suggestion”

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

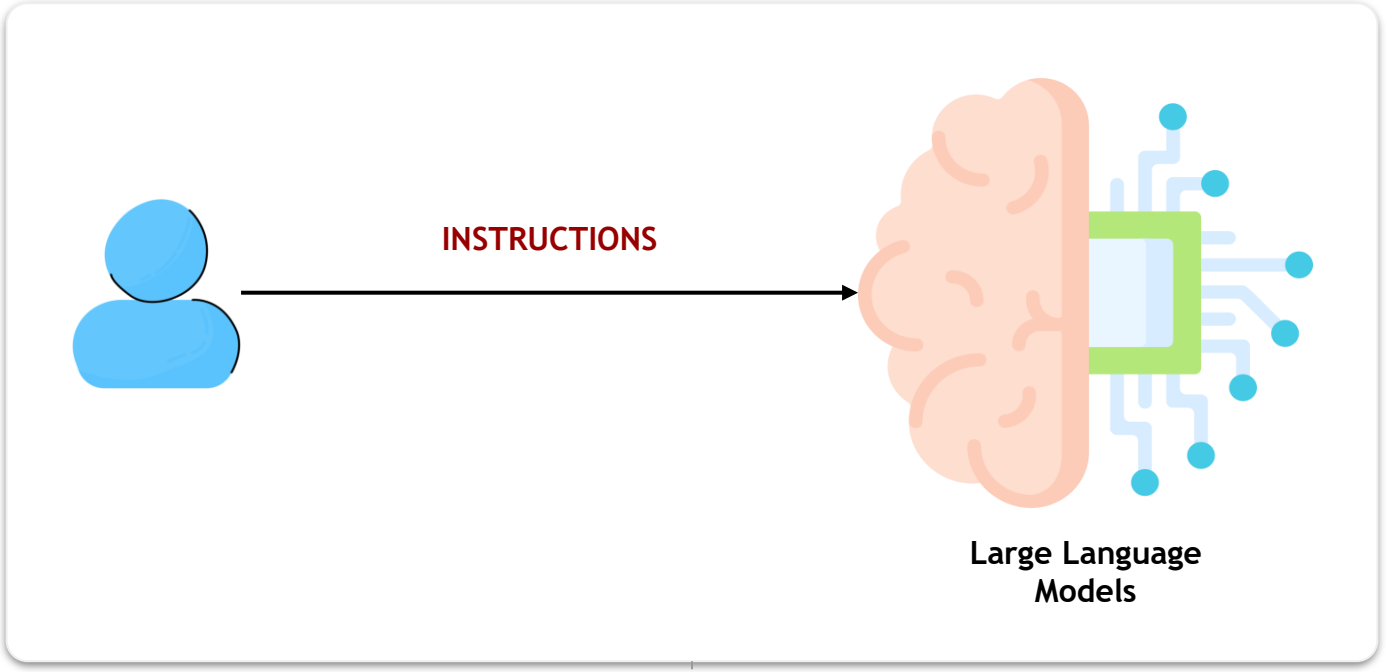


What is a Prompt?



- *A prompt refers to a specific instruction, query, or input given to an AI model to generate a response or perform a task*
- *It serves as cue, guidance, or a starting point for the AI system to produce an output based on its learned knowledge and language comprehension*
- *The form and tone of a prompt may vary depending on the AI application and the desired outcome*
- *Well-defined, context-driven, and unambiguous prompts bring AI output closer to the intended results*

What is Prompt Engineering?



Prompt Engineering is an art of asking the right question to get the best response from an LLM

What is Prompt Engineering?

Developer

As per me,

PROMPT

Code blocks, individual lines of code, or natural language comments a developer writes to generate a specific suggestion from GitHub Copilot

PROMPT ENGINEERING

Providing instructions or comments in the IDE to generate specific coding suggestions

CONTEXT

Details that are provided by a developer to specify the desired output from a generative AI coding tool

What is Prompt Engineering?

ML Researcher

As per me,

PROMPT

Compilation of IDE code & relevant context (IDE comments, code in open files, etc.) that is generated by algorithms & sent to the model of a Gen AI coding tool

PROMPT ENGINEERING

Creating algorithms that will generate prompts (compilations of IDE code and context) for a large language model

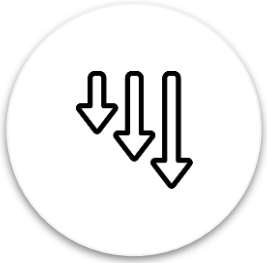
CONTEXT

Details (like data from your open files & code you've written before) that algorithms send to a large language model (LLM) as additional information about the code

Why is Prompt Engineering Important?

Improve the Accuracy

Improve the usefulness of AI generated content



Reduce the risk of generating harmful and biased content

Key elements of a Prompt

Let’s look at aspects that make up a good prompt:

Task

Clearly specify the desired action or response from the model

Context

Provide relevant background information to help the model understand the setting or scenario

Exemplar

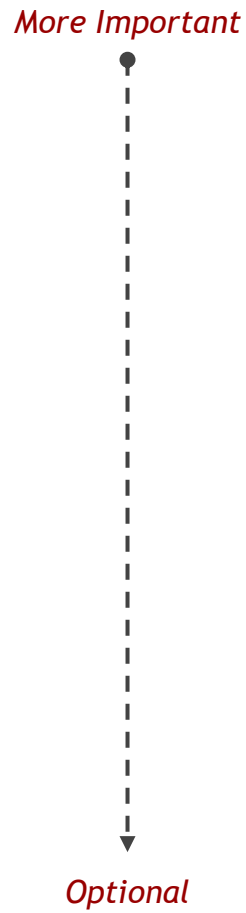
Offer a sample of the desired output to guide the model

Persona

Define the character or role the model should respond to

Format

Specify the structure or style of the response



Key elements of a Prompt

Let's look at aspects that make up a good prompt:

Task

Always start the Task sentence with an Action Verb, e.g.

- ***GENERATE***
- ***GIVE***
- ***WRITE***
- ***ANALYZE***
- ***and many more …***

Key elements of a Prompt

Let's look at aspects that make up a good prompt:

Context

- *What is the user's background?*
- *What does success look like?*
- *What environment are they in?*

Create a Python script for an e-commerce website to automate customer order processing. Keep in mind that the user is a mid-level software engineer familiar with web development and APIs but new to handling large-scale transaction data. Success means the script is efficient, easy to maintain, and includes error handling for failed API calls. The user is working in a fast-paced startup environment with limited resources and a tight deadline.

Key elements of a Prompt

Let's look at aspects that make up a good prompt:

Exemplar

Exemplars are not necessary for every prompt but including them as relevant examples greatly improves the quality of your output

Develop a REST API using Flask that handles user authentication for a web application. The user is a backend developer new to Flask but familiar with Django. Success involves creating an API with endpoints for registration, login, and logout. Provide an example of an API response for a successful login, including the structure of the JSON data and the status codes. Additionally, include sample error responses to demonstrate proper error handling.
Example Pseudocode:
First, import Flask and the necessary modules. Initialize a Flask application. Create a simple data dictionary to store user credentials:
from flask import Flask, request, jsonify
app = Flask(**name**)
users = {"test_user": "password123"}
Define a '/login' endpoint using the POST method. Retrieve the username and password from the incoming JSON request. Check if the username exists and matches the password. If successful, return a JSON response with a success status, a message, and an example token with a 200 status code.
If the credentials do not match, return an error response with a status of "error" and an "Invalid credentials" message, using a 401 status code.
Finally, run the Flask application in debug mode.
Expected Successful Response:
{ "status": "success",
"message": "Login successful!",
"token": "example_token_123"
}
Expected Error Response:
{ "status": "error",
"message": "Invalid credentials."
}

Key elements of a Prompt

Let’s look at aspects that make up a good prompt:

Persona

Who do you want the AI to be? Think of someone you wish you had instant access to with the task you’re facing

IF YOU’RE A SENIOR BACKEND ENGINEER

Imagine you are a senior backend engineer with over 10 years of experience in building secure and scalable web applications.

You've worked extensively with Flask, Django, and REST API design. I'm a mid-level developer building a REST API for user authentication in Flask. Guide me as a mentor would, helping me design the login endpoint to handle user credentials securely, implement proper error handling, and return appropriate JSON responses. Break down complex concepts into simple steps, and provide best practices to ensure the API is robust and secure.

Key elements of a Prompt

Let's look at aspects that make up a good prompt:

Format

Set the structure of the output from GPT-Models

Create a detailed guide for building a user authentication REST API using Flask. Structure the response into the following sections:

1. ***Introduction**: Briefly explain the purpose of user authentication and why Flask is a good choice.*
2. ***Requirements**: List all the libraries and tools needed to build the API.*
3. ***Step-by-Step Instructions**:*
    - *Include a clear breakdown of each step with explanations.*
    - *Provide code snippets for each part, starting from setting up the Flask app, creating endpoints for registration, login, and logout.*
4. ***Code Walkthrough**: Explain the key parts of the code, focusing on how authentication is handled.*
5. ***Best Practices**: Highlight important considerations such as security measures, error handling, and scalability.*
6. ***Sample API Responses**: Show example JSON responses for successful login, registration, and error scenarios.*
7. ***Conclusion**: Summarize the key takeaways and next steps.*

Ensure each section is clearly labeled and presented in a concise manner. Use simple language and provide comments within code snippets for better understanding

Key elements of a Prompt

Let's look at aspects that make up a good prompt:

You are a senior software architect with over 15 years of experience in developing secure and scalable web applications. I am a mid-level developer, working in a fast-paced startup environment, and I need to create a REST API for user authentication using Flask. I have experience with Django but am new to Flask and handling large-scale authentication systems.

Your task is to guide me through building a simple but secure authentication API. Please provide clear, step-by-step instructions. The goal is to ensure the API has endpoints for registration, login, and logout, implements token-based authentication, and includes proper error handling.

Structure the response as follows:

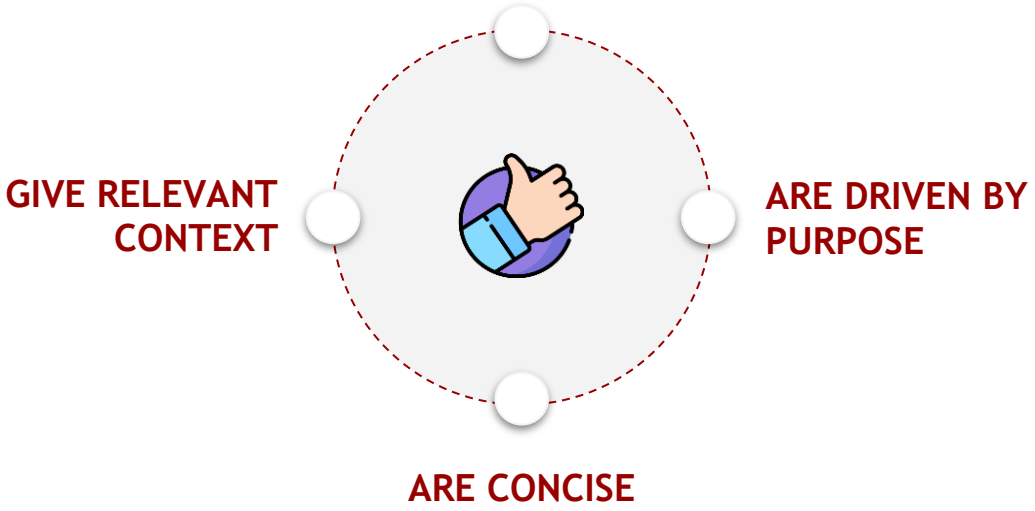
- ***Introduction**: A brief overview of user authentication and why Flask is suitable for this task.*
- ***Requirements**: List of libraries and tools needed.*
- ***Step-by-Step Implementation**: Break down the process with code snippets and explanations. Include comments within the code for clarity.*
- ***Best Practices**: Advice on security measures, error handling, and scalability.*
- ***Sample Code Response**: Provide an example response for a successful login and an error case.*

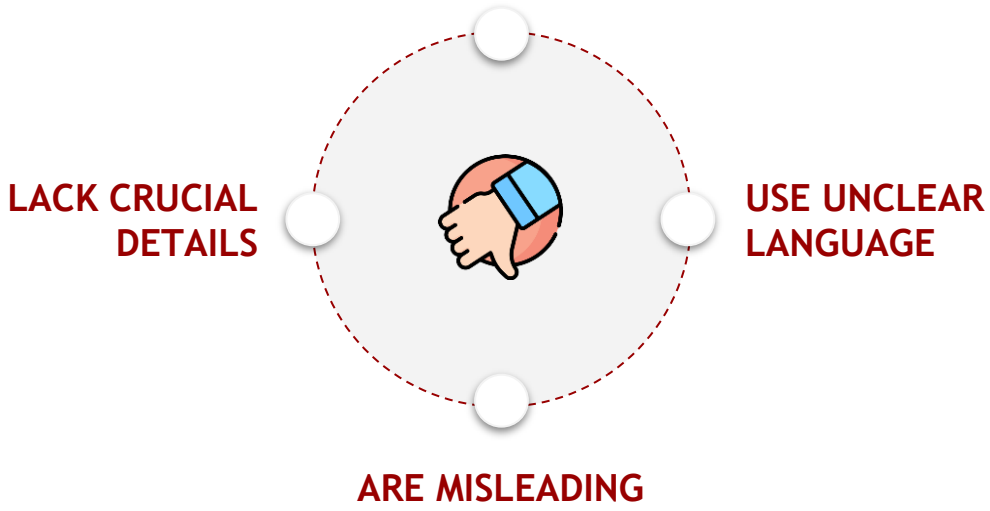
For the 'Step-by-Step Implementation' section, include a sample code snippet. For example, when implementing the login endpoint, show how to handle user input, validate credentials, and return a JSON response with a success message and token

PERSONA CONTEXT TASK FORMAT EXEMPLAR

Good Prompts vs. Bad Prompts

The quality of the prompt significantly influences the output generated by the model. Here are the traits of Good & Bad Prompts:





Tips for designing Prompts

Write Clear Instructions

Help me build an authentication system

Guide me through building a user authentication system in Flask. Include steps for creating endpoints for user registration, login, and logout. Use token-based authentication and include code examples for each step. Make sure to explain how to handle errors and security best practices

Change your prompt as per following to make it clearer:

- *Include details in your query to get more relevant answers*
- *Ask the model to adopt a persona*
- *Use delimiters to clearly indicate distinct parts of the input*
- *Specify the steps required to complete a task*
- *Provide examples*

Tips for designing Prompts

Ask to provide reference text when required

Explain data encryption

Explain data encryption as it relates to API security. Provide reference examples from standard security guidelines like OWASP if possible. Include practical examples for implementing encryption in a Python application

Change your prompt like this to make it clearer:

- *Instruct the model to answer using a reference text/website*
- *Instruct the model to answer with citations from a reference text*


THANK YOU