PROMPTING - GETTING STARTED

Limitations

TRAINING DATA IMPACTS SUGGESTIONS

The quality of code suggestions depends on training data, popular languages like JavaScript yield better recommendations than less common ones

GITHUB COPILOT != COMPILER

Copilot does not guarantee it will deliver compilable code, especially with partial results

THE AI CAN'T READ YOUR MIND

Copilot does not anticipate what you are trying to do. You need to be as declarative as possible in your requests and provide as much context as possible

Good Code Begets Good Code

Use good names

GitHub Copilot understands natural language

Spell out variable names

Single letter variables and abbreviations are ambiguous

Keep functions functional

Follow strong principles when creating named blocks

Be consistent

Generated code follows contextual patterns

Patterns

Use Cases

Code Suggestion / Translation
Unit Test Development
Code Refactoring
RegEx Pattern Matching
Documentation

Anti Patterns

Non Coding Tasks
Production Data Generation
Natural Language Generation

PROMPT ENGINEERING

Cornerstones of Prompting

INTENT

Give Directions

The specific goal or purpose you have in mind when creating a prompt

CONTEXT

Provide Examples

Information to help GitHub Copilot understand the task better

CLARITY

Easy to Understand

Being clear, transparent, and easily understood,

SPECIFICITY

Precision in Detail

Providing precise and detailed information, leaving little room for ambiguity or interpretation.

Best Practices

Provide references
Improve relevance of the response by providing an example and context

Write clear instructions
Refine your prompt, provide context, write clearly, and give Copilot ample input for better results

Split up big tasks
Breaking down complex tasks minimizes errors and utilizes previous outcomes for efficiency

Give Copilot time to think
Requesting Copilot's thought process will enhance Copilot accuracy, but it may prolong wait times

Neighboring Tabs

Keep relevant files open, things related to your current code

GitHub Copilot analyzes all files open in a developer's IDE, not just the actively edited one, scanning through the data to find code matches and incorporating them into its suggestions.

Let's Think Step by Step

Ensure Clarity
Take time to spell everything out

Itemize Tasks
Break down each step

Organized approach
Avoid Overwhelm

Token Limits

Break Down Tasks
Divide requests into smaller tasks, to maintain token limits

Be Concise
Be clear and concise in your prompts, cutting out unnecessary words

Iterative Development
Use one prompt's output for the next, building increasingly complex solutions

PROMPTING - TIPS & TECHNIQUES

Innovative Strategies

COPILOT IS MULTILINGUAL

Copilot understands many languages, code as well as natural

You can communicate with Copilot in a variety of languages. Write your comments in your native tongue, and Copilot will not only complete your comment but also generate the expected code.

Copilot can assist in translating code content, enabling you to internationalize your application effortlessly.

OFFERING ALTERNATIVES

Copilot can provide multiple suggestions to choose from

Use Copilots Suggestions Tab to browse over a selection of recommendations.
If you don't see any suggestion you like, change the comment and/or open Tabs to provide new context and let Copilot try again.

COPILOT EXPLAINS CODE

Copilot can help to document your code

Simply add an empty comment next to the code in question or at the beginning of a function, to let Copilot provide you with an explanation.

CREATE UNIT TESTS

Generate general unit tests

Copilot understands the complexities of your code and suggests tests that cover critical paths and corner cases.

At the beginning of your test file, create a description to tell Copilot to suggest Unit-Tests.