



GitHub Expert Services



# Introduction to Prompt Engineering for GitHub Copilot

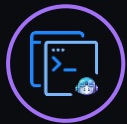
# Agenda



**What is Prompt Engineering?**



**Importance of Prompt Engineering**



**Key Strategies for Effective Prompts**



**Steps in Prompt Engineering**



**Examples of Effective Prompts**



**Common Challenges**



**Best Practices**



**Q&A**



# Prompt Engineering

# What is Prompt Engineering?



*Prompt engineering is the process of crafting clear instructions to guide AI systems, like GitHub Copilot, to generate context-appropriate code tailored to your project's specific needs.*

- Copilot

# Benefits of Well-Engineered Prompts



- Maximises utility of Copilot
- Minimises repeated iterations
- Determines code suggestion quality
- Guides Copilot to understand context & nuances
- Refined interactions reduces misunderstandings

# Principles of Prompt Engineering



- **Single:** Focus prompt on a single, well-defined task or question.
- **Specific:** Ensure that instructions are explicit and detailed
- **Short:** While being specific, keep prompts concise and to the point
- **Surround:** Provide context to help understanding the prompts

# Best Practices for Effective Prompts

- Provide enough clarity: Begin with a broad description and then list specific requirements.
- Give Examples: Use examples to help Copilot understand what you want.
- Break Complex Tasks into Simpler Tasks: Divide large tasks into smaller, manageable tasks.
- Provide enough context with details: The more contextual information provided, the more fitting the generated code suggestions are.
- Assert and iterate: Unlocking GitHub Copilot's full potential with the practice of iteration



# Techniques



Zero-Shot Prompting  
No Example



One-Shot Prompting  
One Example



Few-Shot Prompting  
Handful of Examples

# Techniques

## Zero-Shot Learning with Copilot



abelberhane

Write me unit tests for all of the operations in my calculator.js file.



GitHub Copilot

Sure! Here's an example of a test suite that covers all of the operations in a `calculator.js` file:

```
const { calculate } = require('./calculator');

describe('calculator', () => {
  describe('addition', () => {
    it('adds 1 + 2 to equal 3', () => {
      expect(calculate(1, 2, '+')).toBe(3);
    });

    it('adds -5 + 7 to equal 2', () => {
      expect(calculate(-5, 7, '+')).toBe(2);
    });

    it('adds 0.1 + 0.2 to equal 0.3', () => {
      expect(calculate(0.1, 0.2, '+')).toBeCloseTo(0.3);
    });
  });

  describe('subtraction', () => {
    it('subtracts 5 - 3 to equal 2', () => {
      expect(calculate(5, 3, '-')).toBe(2);
    });
  });
});
```



# Techniques

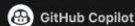
## One-shot Learning with Copilot



abelberhane

Write me unit tests for all of the operations in my calculator.js file.

Use the following example: `test('adds 1 + 2 to equal 3', () => { expect(calculate(1, 2, '+')).toBe(3); });`



GitHub Copilot

Here are some examples of unit tests for all of the operations in a `calculator.js` file:

```
const { calculate } = require('./calculator');

test('adds 1 + 2 to equal 3', () => {
  const result = calculate(1, 2, '+');
  expect(result).toBe(3);
});

test('subtracts 5 - 3 to equal 2', () => {
  const result = calculate(5, 3, '-');
  expect(result).toBe(2);
});

test('multiplies 2 * 4 to equal 8', () => {
  const result = calculate(2, 4, '*');
  expect(result).toBe(8);
});

test('divides 10 / 5 to equal 2', () => {
  const result = calculate(10, 5, '/');
  expect(result).toBe(2);
});

test('throws an error for an invalid operation', () => {
  expect(() => calculate(1, 2, '%')).toThrow('Invalid operation');
});

test('throws an error for division by zero', () => {
  expect(() => calculate(1, 0, '/')).toThrow('Division by zero');
});
```



# Techniques

## Few-shot Learning with Copilot


```
greeting.py X
greeting.py
1 # Example 1: Greeting message for morning
2 # Input: 9 AM
3 # Output: "Good morning!"
4
5 # Example 2: Greeting message for afternoon
6 # Input: 2 PM
7 # Output: "Good afternoon!"
8
9 # Example 3: Greeting message for evening
10 # Input: 7 PM
11 # Output: "Good evening!"
12
13 # Now, generate a python code that takes the current time as input using the datetime module
14 # and returns the appropriate greeting message
15
16 # Solution:
17 # Import datetime module
18 import datetime
19
20 # Get current time
21 current_time = datetime.datetime.now()
22
23 # Get current hour
24 current_hour = current_time.hour
25
26 # Check if it is morning (before 12 PM)
27 if current_hour < 12:
28     print("Good morning!")
29
30 # Check if it is afternoon (between 12 PM and 4 PM)
31 elif current_hour < 16:
32     print("Good afternoon!")
33
34 # Check if it is evening (after 4 PM)
35 elif current_hour < 21:
36     print("Good evening!")
37
38 # Else it is night time
39 else:
40     print("Good night!")
41
```

# Techniques

## Cue with Copilot

/explain #file:Program.cs The purpose of this file is

[Program.cs](#) [Program.cs:1-30](#)

 GitHub Copilot

> Used 1 reference

The purpose of the [Program.cs](#) file is to set up and run a .NET application that ingests data from GitHub using HTTP clients. Here's a detailed explanation:

### Using Directives

These bring in necessary namespaces for HTTP handling, dependency injection, and hosting:

```
using System.Net.Http.Headers;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.CopilotDashboard.DataIngestion.Services;
```

### Host Configuration

A [HostBuilder](#) is used to create and configure a host:

```
var host = new HostBuilder()
    .ConfigureFunctionsWebApplication()
    .ConfigureServices((ctx, services) =>
    {
        services.Configure<GithubMetricsApiOptions>(ctx.Configuration.GetSection<GithubMetricsApiOptions>());
        services.AddHttpClient<GithubCopilotMetricsClient>(ConfigureClient);
        services.AddHttpClient<GithubCopilotUsageClient>(ConfigureClient);
        services.AddHttpClient<GithubCopilotApiService>(ConfigureClient);
    })
    .Build();
```



# Techniques

## Supporting Content with Copilot

shin-akuma

create a powershell method that inserts a json file to cosmosdb collection.

> test.ps1:230-258

GitHub Copilot

> Used 1 reference

Here is a PowerShell method that inserts a JSON file into a CosmosDB collection:

```
function Insert-JsonToCosmosDb (
    param (
        [string]$cosmosDbEndpoint,
        [string]$cosmosDbKey,
        [string]$databaseName,
        [string]$collectionName,
        [string]$jsonFilePath
    )

    # Install the Az.CosmosDB
    if (-not (Get-Module -List
        Install-Module -Name Az.CosmosDB
        CurrentUser

    # Import the Az.CosmosDB
    Import-Module Az.CosmosDB

    # Read the JSON file
    $jsonContent = Get-Content $jsonFilePath -Raw

    # Convert JSON content to a PowerShell object
    $jsonObject = $jsonContent | ConvertFrom-Json
```

shin-akuma

@github create a powershell method that inserts a json file to cosmosdb collection. Use this repo as reference <https://github.com/PlagueHO/CosmosDB>

> get-changed-modules.ps1:24-47

GitHub

> Used 7 references

Repo not indexed for semantic code search

Do you wish to index the repo for semantic code search?

Based on the retrieved information, here's a PowerShell method that inserts a JSON file to a CosmosDB collection using the CosmosDB PowerShell module:

> get-changed-modules.ps1:24-47

GitHub

> Used 7 references

> get-changed-modules.ps1 scripts\azure-devops

<https://github.com/PlagueHO/CosmosDB/blob/9060ce4d4cd08b8e1a119349fb193e05b9acc263/RE...>  
<https://github.com/PlagueHO/CosmosDB/blob/9060ce4d4cd08b8e1a119349fb193e05b9acc263/C...>  
<https://github.com/PlagueHO/CosmosDB/blob/9060ce4d4cd08b8e1a119349fb193e05b9acc263/so...>  
<https://github.com/PlagueHO/CosmosDB/blob/9060ce4d4cd08b8e1a119349fb193e05b9acc263/so...>  
<https://github.com/PlagueHO/CosmosDB/blob/9060ce4d4cd08b8e1a119349fb193e05b9acc263/so...>

```
# GET THE COSMOSDB ACCOUNT CONTEXT
$cosmosDbContext = New-CosmosDbContext -Account $AccountName -ResourceGroup $ResourceGroup

# Read the JSON file
$jsonContent = Get-Content -Path $JsonFilePath -Raw | ConvertFrom-Json
```



# Techniques


**Provide enough  
context with  
details**

```
even_numbers.py •
even_numbers.py
1  # write a simple flask app that returns a list of even numbers from a list of numbers
2  # Create a function that takes a list of numbers and returns only the even values.
3  # create a sample list of numbers
4  # create a list of even numbers from the sample list
5  # return the list of even numbers
6
7
```




# Techniques

## Start with clear instructions with Copilot

 shin-akuma




@github #file:Program.cs #file:GithubMetricsApiOptions.cs


GithubMetricsApiOptions.cs Program.cs

 GitHub

> Used 2 references


The `Program.cs` file sets up dependency injection for the `GithubMetricsApiOptions.cs` file defined in the `Program.cs` file. Specifically, in `Program.cs`, the `GithubMetricsApiOptions` class is configured in the "GITHUB\_METRICS" section of the application's configuration.

 shin-akuma

@workspace /explain #file:Program.cs line 10

Program.cs Program.cs:10

 Workspace

> Used 1 reference

The `GithubMetricsApiOptions` class is used to configure settings for the GitHub Metrics API. It is being configured in the `ConfigureServices` method using the application's configuration.

Here's an example of what the `GithubMetricsApiOptions` class might look like:

```

namespace Microsoft.CopilotDashboard.DataIngestion.Services
{
    public class GithubMetricsApiOptions
    {
        public string BaseUrl { get; set; }
        public string ApiVersion { get; set; }
        public string Token { get; set; }
    }
}
    
```

In the `Program.cs` file, the `GithubMetricsApiOptions` class is configured like this:

```

services.Configure<GithubMetricsApiOptions>(ctx.Configuration.GetSection("GITHUB_METRICS"));
    
```

This line reads the configuration section named "GITHUB\_METRICS" from the application's configuration (e.g., `appsettings.json` or environment variables) and binds it to the `GithubMetricsApiOptions` class. This allows you to inject `IOptions<GithubMetricsApiOptions>` into your services to access these settings.

Example Configuration in `appsettings.json`:



# Techniques

## Break the task down with Copilot

Write a function to generate a 10 by 10 grid of letters.

Write a function to find all words in a grid of letters, given a list of valid words.

Write a function that uses the previous functions to generate a 10 by 10 grid of letters that contains at least 10 words.

Update the previous function to print the grid of letters and 10 random words from the grid.



# Role Prompting

Use copilot to assume a role during a task

Focus to improves results

Improves results by narrowing the AI's focus

Enhanced Results



abelberhane

I want you to act as a front end developer coach. I will provide some questions and the task that I have to complete and would like for you to walk me through the process before our sprint due date. This could involve offering advice on various topics, such as utilizing the react framework or handling API calls. My first request is to help setup my development environment.



GitHub Copilot

Sure, I can help you with that. What programming language and framework are you planning to use for your front-end development? Also, what operating system are you using?



# Techniques

## Getting Specific with Copilot

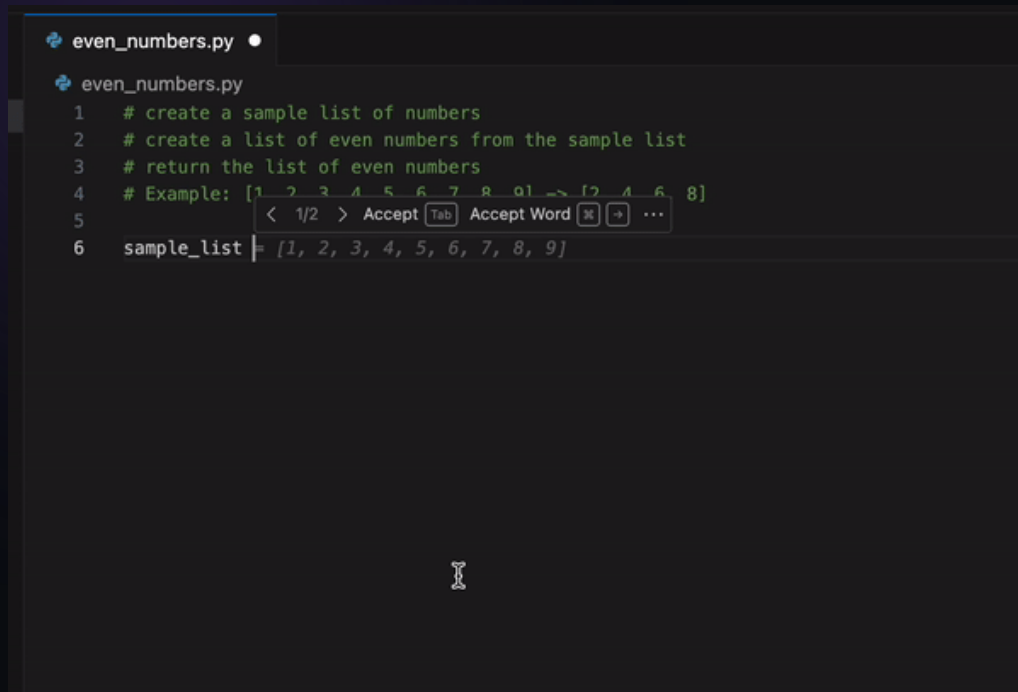
Write a function that tells me if a number is prime

The function should take an integer and return true if the integer is prime

The function should error if the input is not a positive integer

# Techniques

Provide examples  
for learning



```
even_numbers.py
even_numbers.py
1 # create a sample list of numbers
2 # create a list of even numbers from the sample list
3 # return the list of even numbers
4 # Example: [1, 2, 3, 4, 5, 6, 7, 8, 9] -> [2, 4, 6, 8]
5
6 sample_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

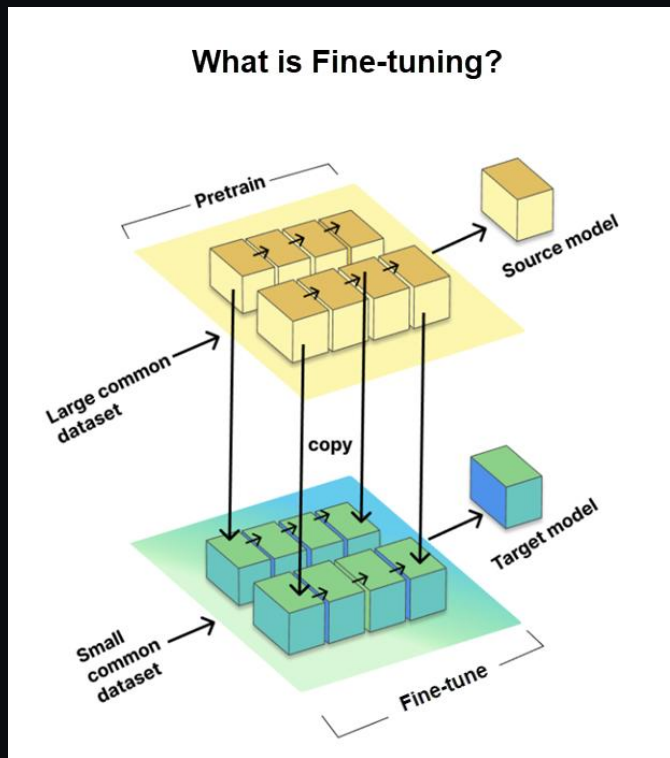
# What are LLMs?



*Large Language Models (LLMs) are artificial intelligence models designed and trained to understand, generate, and manipulate human language. These models are ingrained with the capability to handle a broad range of tasks involving text.*

- Copilot

# What are some core aspects?



- Volume of training data
- Contextual understanding
- Machine learning and AI integration
- Versatility through fine-tuning
- LoRA fine-tuning

# Model Selection

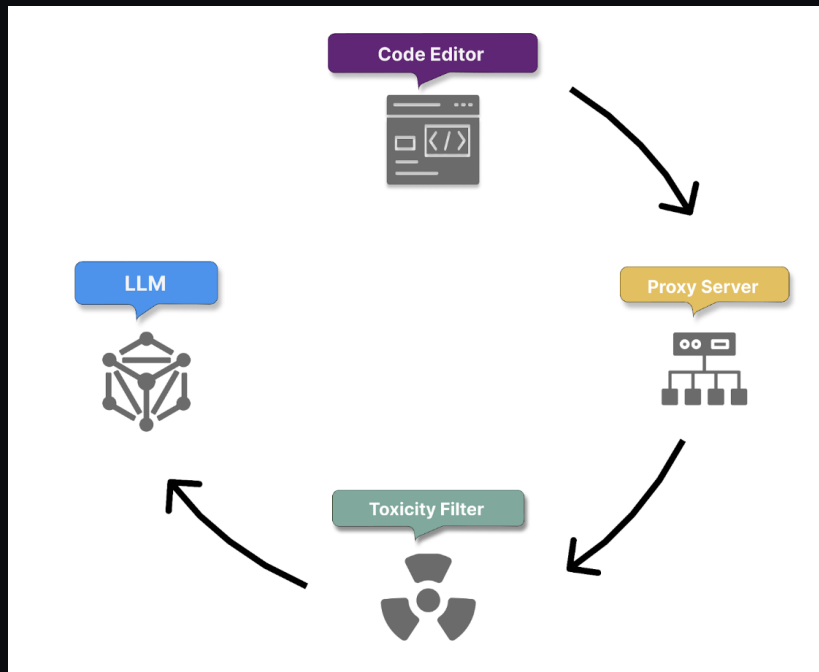
Auto	10% discount
GPT-4.1	0x
GPT-4o	0x
GPT-5 mini	0x
Grok Code Fast 1	0x
Raptor mini (Preview)	0x
Claude Haiku 4.5	0.33x
Claude Opus 4.5 (Preview)	3x
Claude Sonnet 4	1x
✓ Claude Sonnet 4.5	1x
Gemini 2.5 Pro	1x
Gemini 3 Pro (Preview)	1x
GPT-5	1x
GPT-5-Codex (Preview)	1x
GPT-5.1 (Preview)	1x
<a href="#">Manage Models...</a>	

- By default, Copilot Chat uses OpenAI's GPT-4.1 large language model.
- The following models are currently available in Copilot Chat:
  - Anthropic's Claude Haiku 4.5
  - Anthropic's Claude Sonnet 4 & 4.5
  - Google's Gemini 2.5 Pro
  - OpenAI's GPT 5
  - OpenAI's GPT-4o

# Model Selection

- For balance between cost and performance, try GPT-4.1 or 5 mini.
- For fast, low-cost support for simple repetitive tasks, try Claude Haiku 4.5.
- For deep reasoning or complex coding challenges, try GPT-5, Gemini 2.5 Pro or Claude Sonnet 4.5.
- For multimodal inputs (visuals) and real-time performance, try Claude Sonnet 4, Gemini 2.5 Pro or GPT-5 mini.

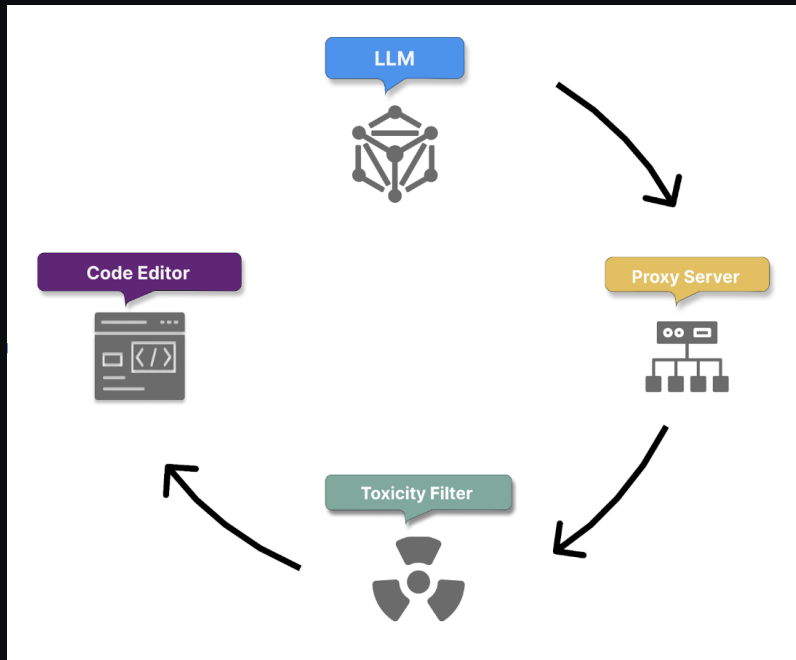
# Copilot User Prompt Process Flow



Inbound flow:

1. Secure prompt transmission and context gathering
2. Proxy filter
3. Toxicity filtering
4. Code generation with LLM

# Copilot User Prompt Process Flow



Outbound flow:

1. Post-processing and response validation
2. Suggestion delivery and feedback loop initiation
3. Repeat for subsequent prompts

# Prompt types supported by GitHub Copilot Chat

- Direct Questions: *Why is my React component not rendering?*
- Code-Related Requests: *Fix this error in my code*
- Open-Ended Queries: *What are the best practices when implementing ORM?*
- Contextual Prompts: *Here's a part of my code, can you suggest improvements?*



# Context Windows

- Context Window refers to the amount of surrounding code and text the model can process simultaneously to generate suggestions.
- Copilot Chat currently operates with a context window of 64k tokens\*
- Be mindful of context window limitations when crafting prompts.
- Breaking down complex problems into smaller, more focused queries or providing relevant code snippets.

\*Token is the basic unit of text that the model processes, acting as a building block for understanding and generating language.





Q & A

# Upcoming Sessions

- 1 GitHub Copilot for Developers
- 2 GitHub Copilot for DevOps
- 3 Q&A Session





**Thank you!**