Introduction to Prompt Engineering with GitHub Copilot

Mastering AI-Assisted Development for Maximum Efficiency

What is Prompt Engineering?

The Definition

Prompt engineering is the art and science of crafting inputs (prompts) to guide Generative AI models to produce the most accurate, relevant, and high-quality outputs.

It's not just asking. It's **instructing**.

Why it Matters

- **Efficiency:** Get the right code on the first try.
- **Accuracy:** Reduce hallucinations and bugs.
- **Creativity:** Unlock complex solutions you might not initially see.
- **Control:** Guide the style, structure, and logic of the output.

Core Foundations

Clarity

Ambiguity is the enemy. Be precise about what you want. Avoid vague terms like "fix this" without context.

Context

Copilot needs to know the "where" and "why". Provide relevant files, libraries, and coding standards.

Iterate

The first prompt is rarely perfect. Treat it like a conversation. Refine your request based on the AI's response.

The 4S Principles

Single

Focus on a single task or responsibility at a time.

Don't ask for a database migration and a frontend UI in one prompt.

Specific

Provide detailed instructions.

Mention variable names, specific libraries (e.g., pandas), and expected formats.

Short

Keep prompts concise. Long, rambling prompts confuse the context window. Be direct and to the point.

Surround

Open relevant files in your IDE tabs. Copilot uses these "surrounding" tabs as immediate context for generation.

The GCES Framework

A structured approach to prompt construction that ensures all necessary components are present for high-quality generation.

```
// Good Prompt Example

"Create a REST API endpoint (Goal) using
Express.js (Context). Example: GET /users
returns user list (Example). Include error
handling and JSON response (Structure)."
```

🏁 **Goal:**	What do you want to achieve?
Context:	Tech stack, constraints, libraries.
💡 **Example:**	Input/Output samples.
🔨 **Structure:**	Format, naming conventions.

Prompting Strategies

0

Zero-Shot

Providing no examples, just the task.

```
"Write a Python function to
calculate fibonacci."
```

1

One-Shot

Providing a single example to guide the model.

```
"Convert this date: '2023-01-
01' → 'Jan 1, 2023'. Now
convert '2023-12-25' ... "
```

Few-Shot

Multiple examples for complex pattern recognition.

```
"Input: A, Output: Z. Input:
B, Output: Y. Input: C,
Output: ?"
```

Chain Prompting

Break complex tasks into sequential, manageable steps to maintain context and accuracy.

1. **Define**

   Define the interface and data structures first.

2. **Logic**

   Implement the core business logic functions.

3. **Test**

   Generate unit tests for the specific logic.

4. **Refactor**

   Optimize for performance and readability.

Role Prompting

Assign a specific persona to Copilot to shift its perspective and output style.

Security Reviewer

"Act as a security expert. Review this code for vulnerabilities like SQL injection and XSS."

Solution Architect

"Act as a system architect. Design a scalable microservice structure for this user flow."

Testing Specialist

"Act as a QA engineer. Write comprehensive unit tests covering edge cases for this function."

Process Flow

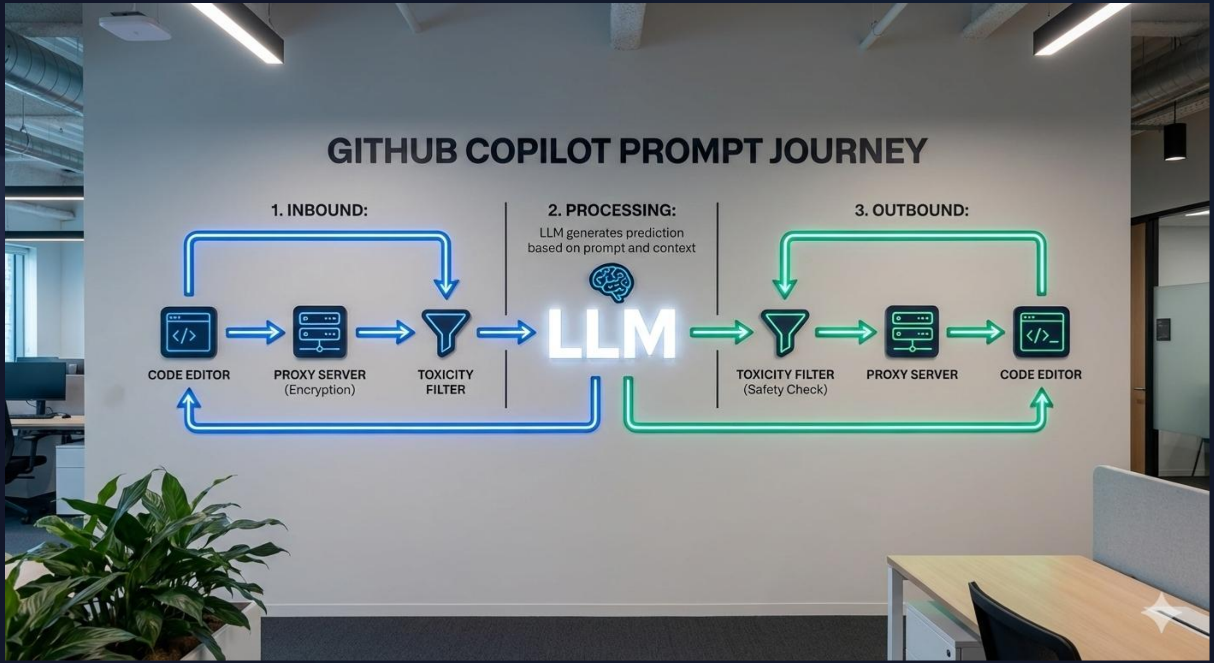
The Journey of a Prompt

1. Inbound: Code Editor → Proxy Server (Encryption) → Toxicity Filter → LLM.

2. Processing: LLM generates prediction based on prompt & context.

3. Outbound: LLM → Toxicity Filter (Safety Check) → Proxy Server → Code Editor.

> Safety mechanisms ensure that both the input context and the output suggestion are safe and relevant.


GITHUB COPILOT PROMPT JOURNEY
1. INBOUND:
2. PROCESSING:
LLM generates prediction based on prompt and context
3. OUTBOUND:
CODE EDITOR
PROXY SERVER (Encryption)
TOXICITY FILTER
LLM
TOXICITY FILTER (Safety Check)
PROXY SERVER
CODE EDITOR

Security & Privacy

Data Handling

- **Ephemeral:** Code snippets are used only to generate the suggestion and are discarded immediately after.
- **Encryption:** All data is transmitted over secure HTTPS channels.
- **No Retraining:** Your private code is NOT used to train the base models (unless you opt-in for telemetry).

User Control

- **Public Code Filter:** Option to block suggestions that match public code (preventing license issues).
- **Enterprise Policies:** Admins can enforce usage policies across the organization.
- **Compliance:** SOC 2, GDPR, and ISO 27001 compliant.

Supported Prompt Types

Direct Questions

"How do I center a div in CSS?"

Code Requests

"Write a function to validate email addresses."

Open-Ended

"Explain how this authentication middleware works."

Contextual

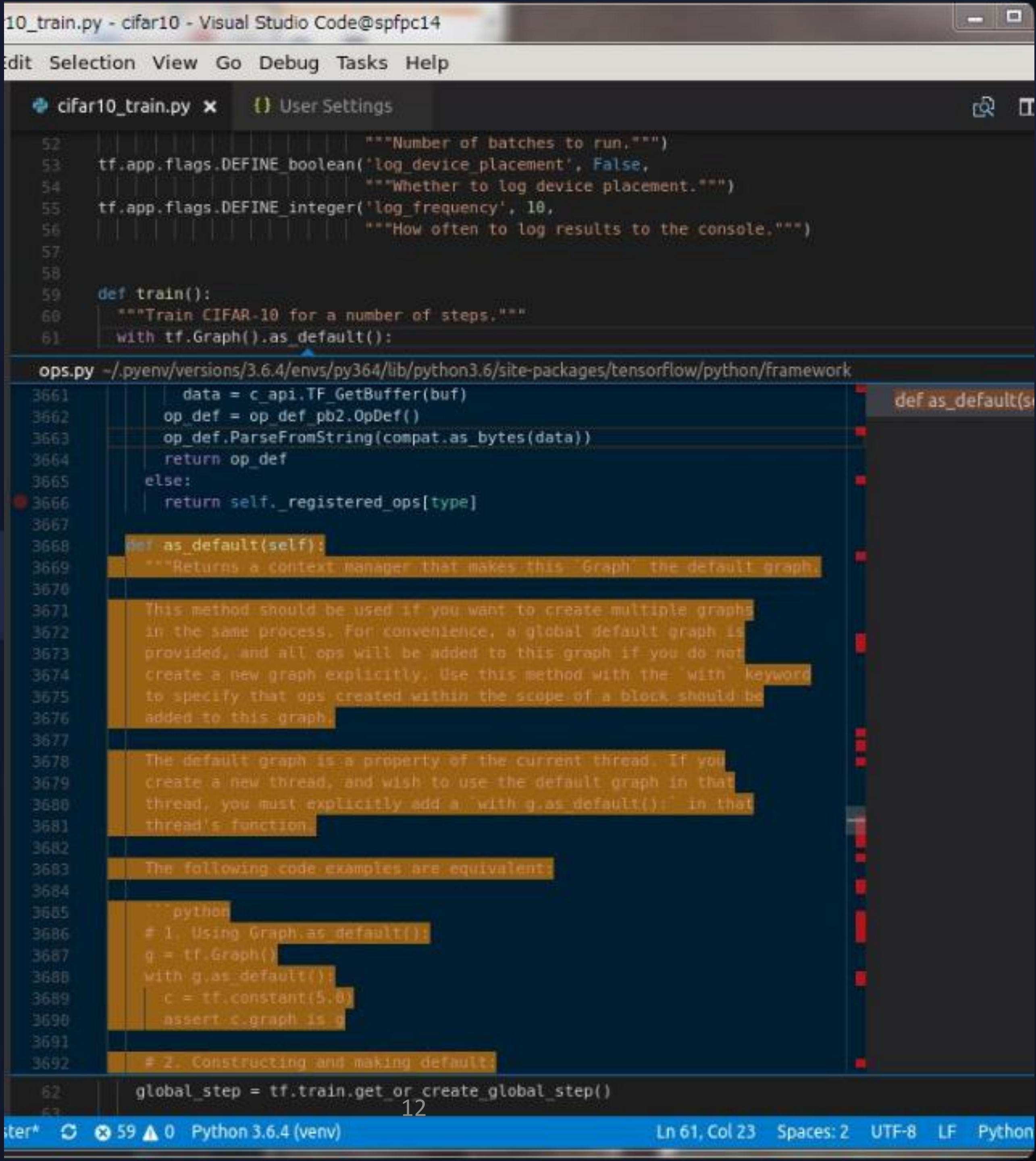
"Fix the bug in the selected code block."

Context Windows

Managing the "View"

The context window is the limit of text/code the LLM can process at once.

- **Current State:** Copilot uses a sophisticated context management system (neighboring tabs, recently active files).
- **Best Practice:** Keep relevant files open. Close unrelated files to avoid "noise".
- **Token Limit:** Be mindful of large files; only the most relevant chunks are sent.


10_train.py - cifar10 - Visual Studio Code@spfpc14
Edit Selection View Go Debug Tasks Help
cifar10_train.py
User Settings

```
                              """Number of batches to run.""")
tf.app.flags.DEFINE_boolean('log_device_placement', False,
                              """Whether to log device placement.""")
tf.app.flags.DEFINE_integer('log_frequency', 10,
                              """How often to log results to the console.""")


def train():
  """Train CIFAR-10 for a number of steps."""
  with tf.Graph().as_default():
```

ops.py ~/.pyenv/versions/3.6.4/envs/py364/lib/python3.6/site-packages/tensorflow/python/framework

~~~
          data = c_api.TF_GetBuffer(buf)
        op_def = op_def_pb2.OpDef()
        op_def.ParseFromString(compat.as_bytes(data))
        return op_def
      else:
        return self._registered_ops[type]

  def as_default(self):
    """Returns a context manager that makes this `Graph` the default graph.

    This method should be used if you want to create multiple graphs
    in the same process. For convenience, a global default graph is
    provided, and all ops will be added to this graph if you do not
    create a new graph explicitly. Use this method with the `with` keyword
    to specify that ops created within the scope of a block should be
    added to this graph.

    The default graph is a property of the current thread. If you
    create a new thread, and wish to use the default graph in that
    thread, you must explicitly add a `with g.as_default():` in that
    thread's function.

    The following code examples are equivalent:

    ```python
    # 1. Using Graph.as_default():
    g = tf.Graph()
    with g.as_default():
      c = tf.constant(5.0)
      assert c.graph is g

    # 2. Constructing and making default:
~~~

```
    global_step = tf.train.get_or_create_global_step()
```

ster* ⊗ 59 ⚠ 0 Python 3.6.4 (venv) Ln 61, Col 23 Spaces: 2 UTF-8 LF Python

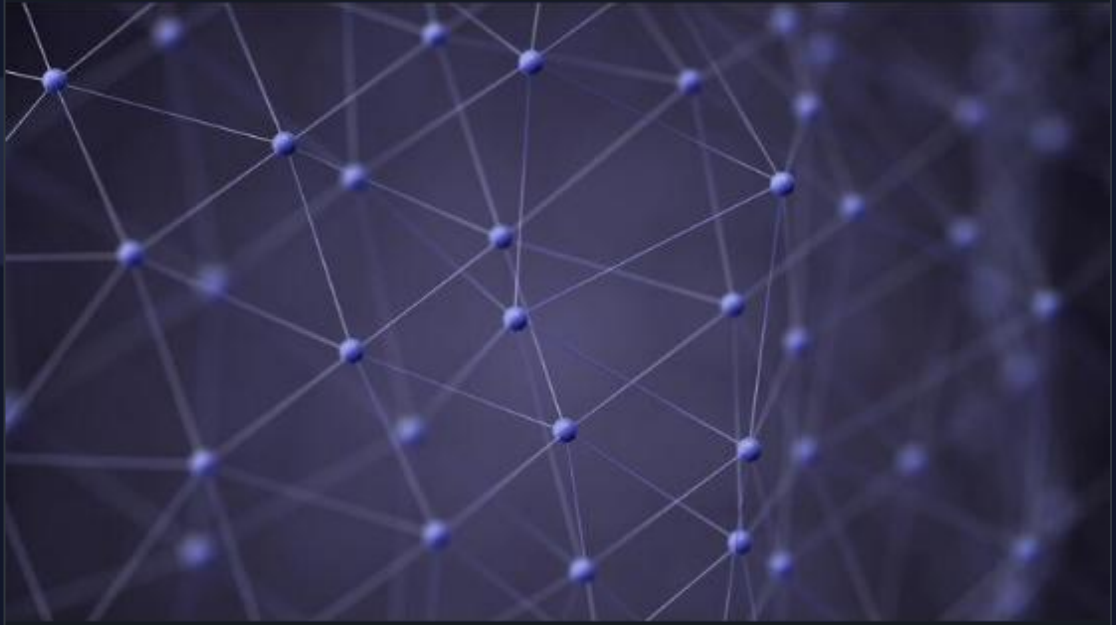
Introduction to Prompt Engineering with GitHub Copilot - Ritesh Singh

Under the Hood: LLMs

OpenAI Codex & GPT

GitHub Copilot is powered by highly specialized versions of OpenAI's models.

- **Codex:** A descendant of GPT-3, specifically fine-tuned on billions of lines of public code.
- **Optimization:** Optimized for low-latency generation to feel instant within the IDE.
- **Multilingual:** Fluent in dozens of programming languages, from Python to Go to TypeScript.



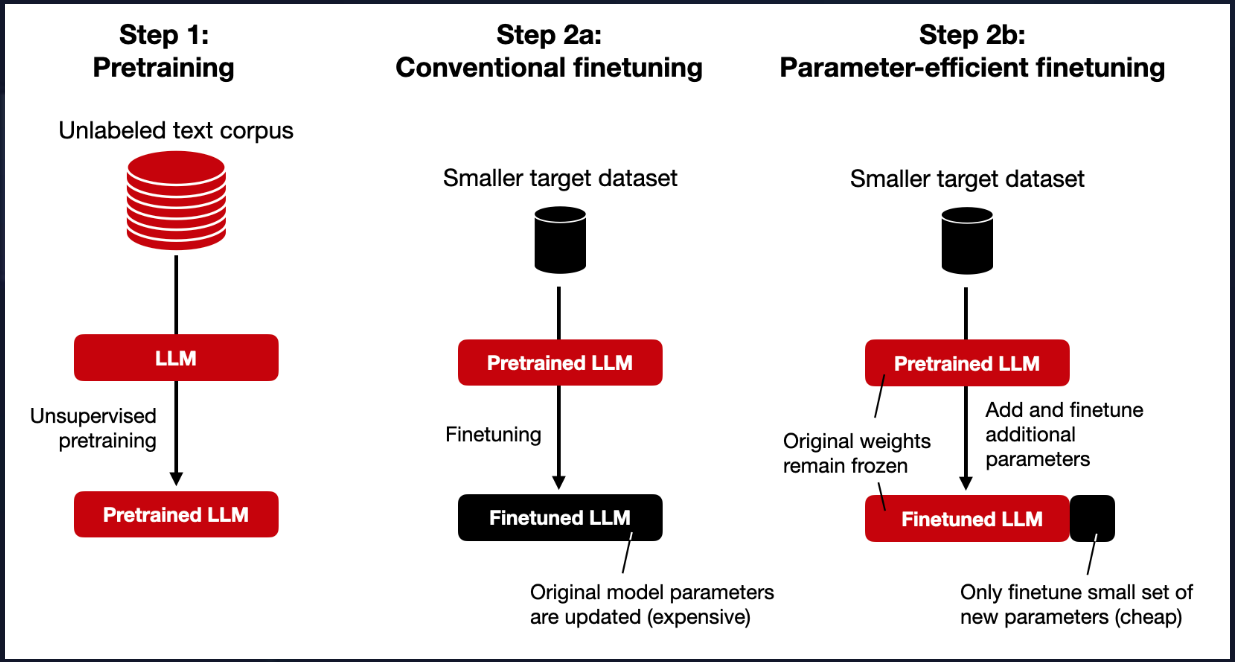
LoRA Fine-Tuning

Low-Rank Adaptation (LoRA)

A technique to fine-tune large models efficiently without retraining the entire network.

Role in Copilot:

Allows for specialized adaptation (e.g., to your organization's private coding style) by injecting small, trainable rank decomposition matrices into the model.



Key Takeaways

1
Context is King

Use the 4S Principles and GCES framework. Keep relevant files open and be specific.

2
Iterate

Don't stop at the first suggestion. Use chain prompting and role prompting to refine the output.

3
Secure

Trust in the enterprise-grade security filters and data handling privacy.

Q&A

Thank you for your time

Let's Learn & Grow Together

@riteshsingh84

@lalriteshsingh