The Developer’s Guide to Prompt Engineering

_ Mastering the Art of Communication with AI

Based on '200 Prompts Every Programmer Should Know' | BIG School - Master in AI Development

{ Garbage In, Garbage Out }

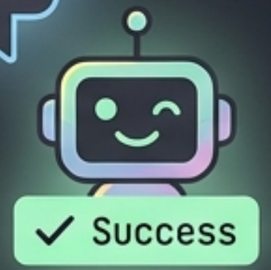
The Reality

> Project / src / main.py

User: Fix this code.

The Goal

User: Act as a Senior Python Dev.
Debug this loop for memory leaks...




```
1 def analyze_data(data):
2     # Optimized for memory efficiency
3     for item in data:
4         process(item)  # Efficiently handle resources
5     return "Memory leaks resolved."
```

Status: Resolved

⚠ Warning

A prompt is not a search query; it is a function call.
Undefined parameters return unpredictable values.

> Project / src / main.py

{ JetBrains Mono: The Anatomy of a Perfect Prompt }

tags badges

Role/Persona // Who is the AI?

Objective // What is the task?

Context // What's the backstory?

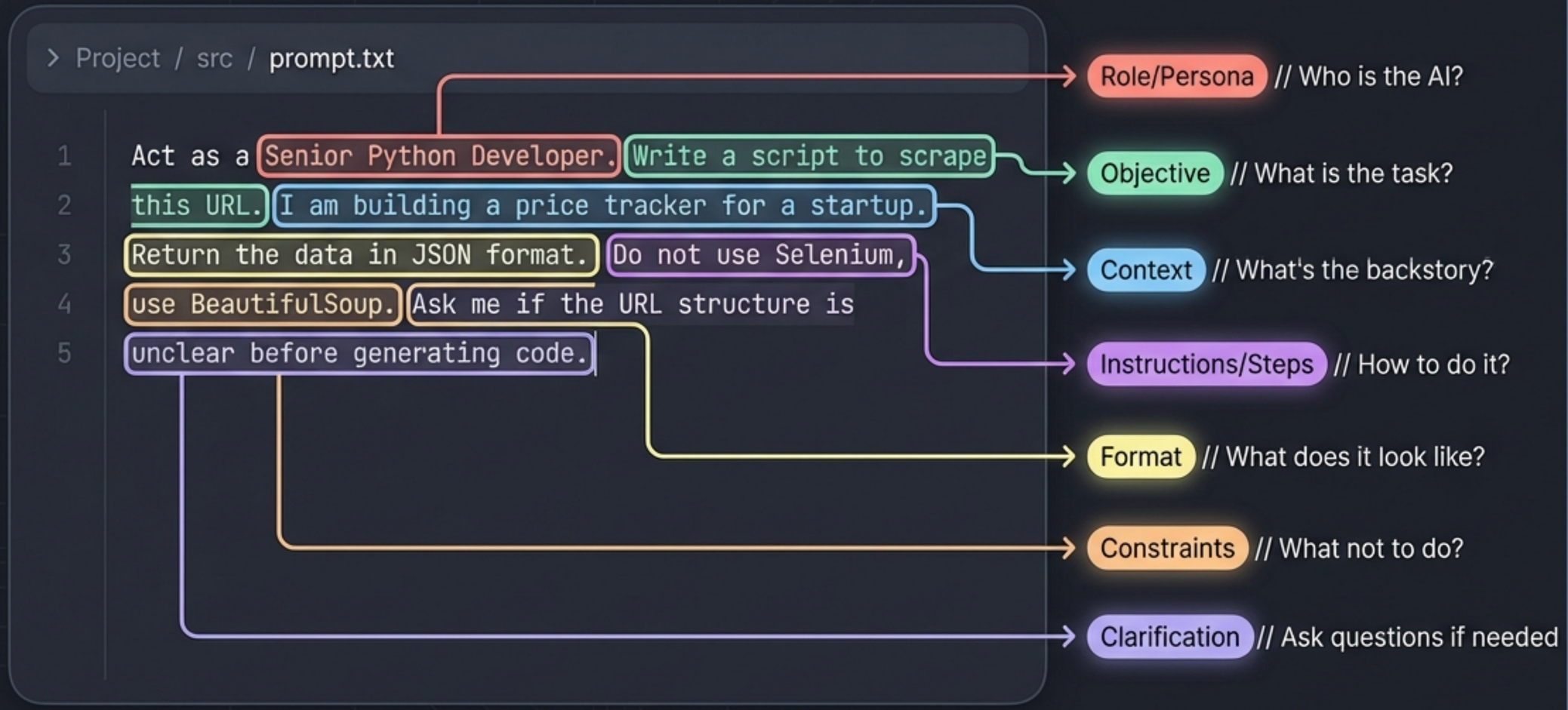
Instructions/Steps // How to do it?

Format // What does it look like?

Constraints // What not to do?

Clarification // Ask questions if needed

{ The Anatomy in Action }



Specificity eliminates ambiguity. Treat your prompt like a ticket for a junior developer.

Project / src / prompt.txt

{ Phase 1: Learning & Fundamentals }

Goal: Accelerating the learning curve and filling knowledge gaps.

The Simplifier

Analogy

Explain [Recursion] as if I were a 10-year-old. Use a real-world analogy.

The Syllabus Maker

Plan

```
[.
├── [.ai.]
```

Create a 1-week study plan for [React] including daily goals and resources.

```
│   └── ...
└── directory
```

The Socratic Tutor

Quiz

> Simulate a study session. Ask me 5 multiple-choice questions about [SQL] and explain the answers.

{ Phase 2: Planning & Architecture }

Goal: Structuring thought and making architectural decisions.

The Tech Stack Advisor

Recommend a frontend/backend stack for an [E-commerce app] and explain why.
Create a comparison table. _

[Stack]

The Pattern Matcher

Suggest 2 design patterns for [User Auth] and list pros/cons.

[Patterns]

The Data Architect

Design a JSON data structure for a [User Profile] including nested fields and types.

[JSON]

>_ { Phase 3: Build & Implement }

Goal: Translating requirements into functional, clean code.

[Snippet]

The Specialist

Write a function in [Python] to [Calculate Factorial]. Include error handling for edge cases.

[Refactor]

The Translator

Translate this [Java] code into [Python]. Maintain the logic but use Pythonic conventions.

[Scaffold]

The Boilerplate Gen

Generate a REST API endpoint in [Express.js] for [User Login] with standard status codes.

{ Phase 4: The Detective (Debugging) }

Goal: Identifying errors and understanding root causes.

The Error Analyst [Bug]

Here is an error message: **[Error 404]**. Explain the root cause and suggest 3 solutions ordered by probability.

The Code Reviewer [Review]

Debug this code block. Identify potential infinite loops or memory leaks.

The Fixer [Fix]

My program hangs when [loading large files]. Analyze this code and propose a fix.

{ Phase 5: Testing & QA }

Goal: Ensuring robustness and covering edge cases.

[Test]

The Unit Tester

Write unit tests for this function covering edge cases (nulls, zeros, special characters).

[Mock]

The Data Generator

Generate dummy JSON data for 5 users to test this import script. Include one invalid entry.

[QA]

The Edge Case Finder

What input values would break this function? Act as a QA engineer.

Phase 6: Refactoring & Optimization

Goal: Improving performance, readability, and security.

The Complexity Crusher

[Performance]

Analyze the Big-O complexity of this function and suggest an optimized version.

The Clean Coder

[Refactor]

Refactor this code to follow [Google Style Guide] and improve variable naming.

The Security Auditor

[Security]

Review this SQL query for injection vulnerabilities and suggest a parameterized version.

>_ Phase 6 / Refactoring & Optimization_

{ Phase 7: Documentation & Maintenance }

Goal: Future-proofing code and facilitating teamwork.

[Docs]

The Docstring Writer

Generate JSDoc comments for this function explaining parameters and return values.

[README]

The README Architect

Create a README.md for this project including installation steps and usage examples.

[Translate]

The Translator (Human)

Explain this code to a non-technical stakeholder using simple terms.

> Phase 7 / Documentation & Maintenance_

Advanced Prompt Flows

Chain of Thought

Inter: Explicitly ask the AI to "Think step-by-step" to activate reasoning.

Iterative Refinement

Inter: Feed errors back into the chat. It is a conversation, not a transaction.

Role Flipping

Inter: Ask the AI to act as a user trying to break the app.

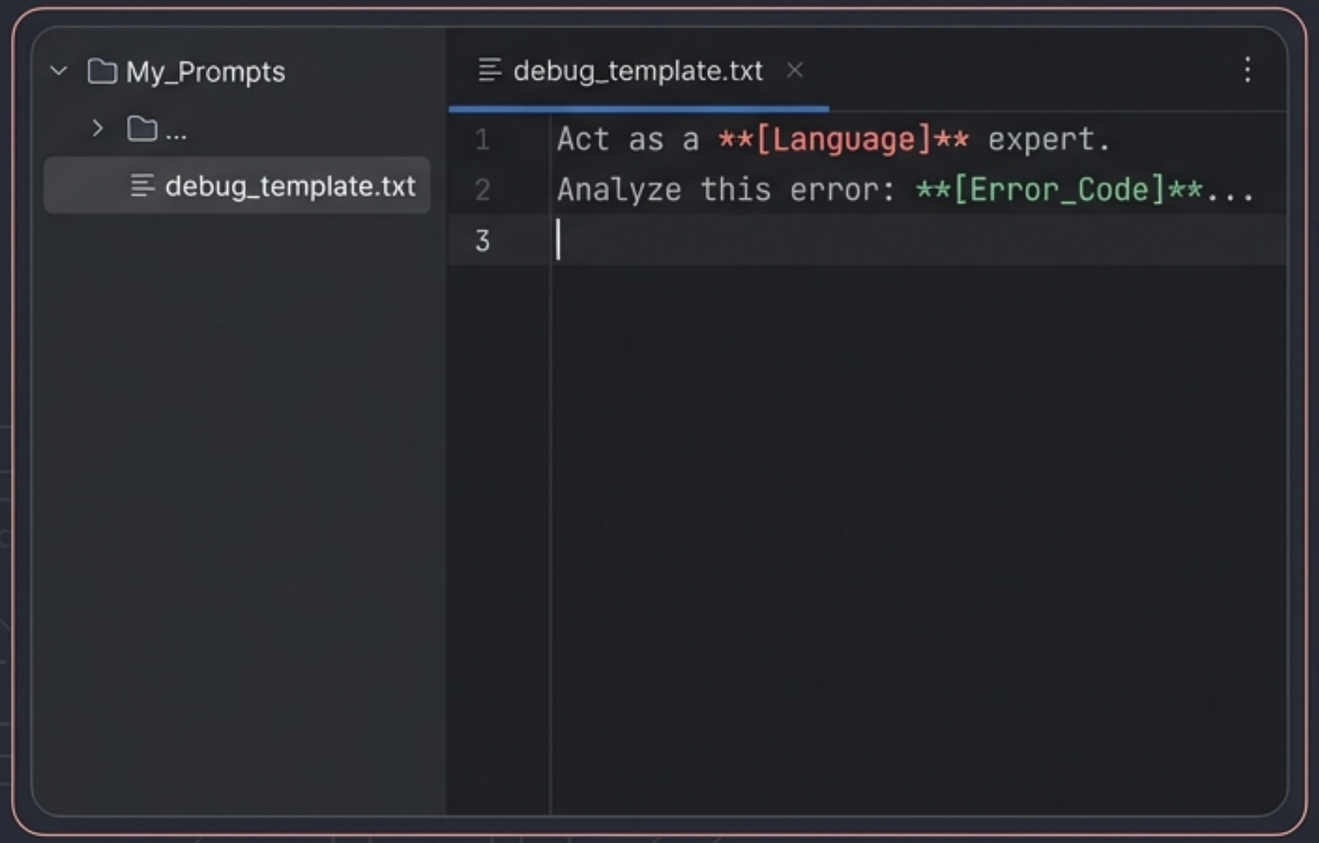
> Phase: Advanced Prompt Flows_

> { 10 Golden Rules for Effective Prompts }_

1. Define the objective first.
2. Specificity > Ambiguity.
3. Divide and Conquer (break down tasks).
4. Iterate (it's a conversation).
5. Use positive instructions (Do X, not "Don't do Y").
6. Ask for "Step-by-Step" thinking.
7. Trust but Verify (Review the code).
8. Build a prompt library.
9. Know the model's limitations.
10. Experiment constantly._

> Golden Rules for Prompts_

> Build Your Personal Library_



“Don’t reinvent the wheel. When a prompt gives you a perfect result, save it as a template. Replace specific details with [Placeholders].”

Source: Tip #8 from BIG School Master Class

> Phase: Building a Prompt Library_

> Conclusion_

{ The Prompt is the New Syntax. }

Programming is no longer just about writing code; it's about articulating logic and intent. Master the prompt, and you master the engine.

Content adapted from BIG School - Máster Desarrollo con IA.