**🧠 Prompt Engineering Template for Codebase Porting**

You are a senior developer which has been tasked with porting a old text-based video adventure game into a more modern language. You need to port the COMPLETE game with all logic, objects, data, items, and so on.  
  
The source code will be in Python 3.13 and should be divided into files and modules in a meaningful way as to promote reuse and simplicity to port other similar games in the future. Game elements should be 100% data-driven with no data elements hard coded into the engine code to help promote portability to other game porting projects.

The project plan is below. Steps 1 through 6 have ALREADY been completed and can be referenced to complete steps 7 through 10 which are not started yet.

**📌 Objective**

Port an existing codebase from **ZIL which is based on MDL** to **Python 3.13** while preserving functionality, structure, and maintainability. Each steps’ output should be placed in its own directory with the same name as the following steps.

The original source code is written in **ZIL which is based on MDL** which is intended to be compiled into zcode and then run within a zcode interpreter. The ported code base should simplify this with the complete game written in python and running as a **Python 3.13** executable.

The original game’s source code can be found here: <https://github.com/caliraftdude/deadline?tab=readme-ov-file>

References for the game, its design, how it was played, etc.. can be found in with the following resources:

* [Wikipedia](https://en.wikipedia.org/wiki/Deadline_(video_game))
* [The Digital Antiquarian](https://www.filfre.net/2012/07/deadline/)
* [The Interactive Fiction Database](https://ifdb.tads.org/viewgame?id=p976o7x5ies9ltdh)
* [The Infocom Gallery](http://infocom.elsewhere.org/gallery/deadline_grey/)
* [IFWiki](http://www.ifwiki.org/index.php/Deadline)

References for how the zMachine interpreter works (to help eliminate it and understand the zil code better) can be found here: <https://github.com/caliraftdude/zmachine>. This repository also includes references and explanations of ZIL called ZilCourse.pdf, ZilLearning.pdf and zil.pdf. An explanation on how a zil code written adventure works is found here: https://medium.com/swlh/zork-the-great-inner-workings-b68012952bdc . Lastly an online reference for the MDL language is here: <https://mdl-language.readthedocs.io/en/latest/>

Steps 1 through 6 area already completed and are located the following repository within the “design docs” directory. You should use these documents to prevent doing any repeat work: <https://github.com/caliraftdude/py_adv_Deadline>

**1. 🔍 Project Overview**

**Prompt:**

"Summarize the purpose and functionality of the current codebase written in **ZIL which is based on MDL**. Include key modules, dependencies, and architectural patterns."

**Expected Output:**

* High-level system architecture
* Key features and workflows
* External integrations (APIs, databases, services)

**2. 📦 Inventory and Analysis**

**Prompt:**

"List all files, modules, and dependencies in the codebase. Identify language-specific constructs, third-party libraries, and platform-specific code."

**Expected Output:**

* File/module map
* Dependency graph
* Language-specific features (e.g., decorators in Python, generics in Java)

**3. 🧪 Testing Strategy**

**Prompt:**

"Extract and summarize the existing test coverage. Identify unit, integration, and end-to-end tests. Highlight gaps in coverage."

**Expected Output:**

* Test coverage report
* List of critical test cases
* Recommendations for additional tests

**4. 🔁 Mapping Constructs**

**Prompt:**

"Create a mapping between **ZIL which is based on MDL** constructs and their equivalents in **Python 3.13**. Include syntax, idioms, and design patterns."

**Expected Output:**

* Syntax translation table
* Idiomatic usage comparison
* Design pattern adaptation (e.g., MVC in different languages)

**5. 🛠️ Tooling and Automation**

**Prompt:**

"Recommend tools or frameworks to assist in porting from **ZIL which is based on MDL** to **Python 3.13**. Include transpilers, linters, formatters, and static analysis tools."

**Expected Output:**

* Toolchain suggestions
* Setup instructions
* Automation scripts (if applicable)

**6. 🧩 Porting Plan**

**Prompt:**

"Generate a phased migration plan for porting the codebase. Include milestones, risk assessment, and rollback strategies."

**Expected Output:**

* Migration roadmap
* Risk matrix
* Rollback and fallback procedures

**7. 🧬 Code Translation**

**Prompt:**

"Translate the repository <https://github.com/caliraftdude/deadline?tab=readme-ov-file> written in **ZIL which is based on MDL** to Python 3.13, preserving logic and performance. Include comments explaining the translation choices. Be sure to include ALL game logic, locations, items. Ensure the game engine is data driven, keep all game elements in a data format so that its easy to port other similar games in this manner."

**Expected Output:**

* Translated code
* Inline comments
* Performance considerations

**8. 🧹 Refactoring and Optimization**

**Prompt:**

"Suggest refactoring opportunities in the translated code to make it idiomatic and efficient in Python 3.13."

**Expected Output:**

* Refactored code
* Language-specific best practices
* Performance improvements

**9. ✅ Validation and Testing**

**Prompt:**

"Generate test cases to validate the translated code against the original functionality. Include edge cases and performance benchmarks."

**Expected Output:**

* Test suite
* Benchmark results
* Validation checklist

**10. 📚 Documentation**

**Prompt:**

"Generate updated documentation for the ported codebase in [Target Language]. Include setup instructions, API references, and developer notes."

**Expected Output:**

* README files
* API docs
* Developer onboarding guide