A Python programming Prompt and surprising insights of LLMs - ChatGPT 4

# Version Information

Satya Komatineni, 3/7/24

Contents

[Version Information 1](#_Toc160705485)

[Abstract 1](#_Toc160705486)

[English Spec for the program 1](#_Toc160705487)

[ChatGPT said the following: 2](#_Toc160705488)

[Then it wrote the code (Not surprising part) 2](#_Toc160705489)

[Key observations: The surprising part 3](#_Toc160705490)

[Then it advises on Typing and Type hints (Surprising as well) 4](#_Toc160705491)

# 

# Abstract

1. Illustrates “Creating English specs at the beginning of a python file”.
2. Ask ChatGPT to write the code based on the Spec.
3. There is a surprising insight it has demonstrated (Oh my! Moment)

# English Spec for the program

I wrote the following at the top of the python file:

"""

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Introduction

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Goal:

1. Create an interface to create objects

2. usually based on configuration files

3. A singleton

4. replaceable with future implementations

5. Can support many kinds of dependency injection

6. Closely works with IApplication and its interfaces

"""

"""

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Code spec

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Create an abstract clas with name IFactory

2. Abstractt Methods

public Object getObjectAbsolute(String identifier, Object args)

3. Regular method

1. public getObjectInvocationPrefix()

should return "request"

2. public Object getObject(String identifier, Object args)

call getObjectInvocationPrefix() and add its output to the identifier to call getObjectAbsolute()

3. public Object getObjectWithDefault(String identifier, Object args, Object defaultObject)

This method should call the getObject() and

if that throws exception

log the exception

return the default

if it doesn't throw the exception return the object from getObject()

“””

# ChatGPT said the following:

To implement the specifications you've provided, we'll translate the concept to Python, adapting to Python's syntax and conventions.

**Since Python does not use access modifiers like public (as in Java or C#), we'll omit those**.

Python also doesn't have a built-in Object type as in Java; instead, **we use Any from the typing module** to represent any object.

Note that Python's abstract base classes (ABCs) and method definitions slightly differ from Java/C# syntax.

# Then it wrote the code (Not surprising part)

1. from abc import ABC, abstractmethod

2. from typing import Any

3.

4. class IFactory(ABC):

5.

6. @abstractmethod

7. def getObjectAbsolute(self, identifier: str, args: Any) -> Any:

8. """

9. Abstract method to be implemented by subclasses.

10. """

11. pass

12.

13. def getObjectInvocationPrefix(self) -> str:

14. """

15. Regular method that returns the invocation prefix.

16. """

17. return "request"

18.

19. def getObject(self, identifier: str, args: Any) -> Any:

20. """

21. Calls getObjectAbsolute with a modified identifier.

22. """

23. modified\_identifier = self.getObjectInvocationPrefix() + identifier

24. return self.getObjectAbsolute(modified\_identifier, args)

25.

26. def getObjectWithDefault(self, identifier: str, args: Any, defaultObject: Any) -> Any:

27. """

28. Attempts to call getObject and returns defaultObject if an exception occurs.

29. """

30. try:

31. return self.getObject(identifier, args)

32. except Exception as e:

33. print(f"Exception caught: {e}") # Simple logging of the exception

34. return defaultObject

35.

# Key observations: The surprising part

It said the following after the code.

I have highlighted its surprising insights.

1. The getObjectAbsolute method is abstract and must be implemented by any subclass of IFactory. It's designed to retrieve an object based on the identifier and arguments.
2. The getObjectInvocationPrefix method is a regular method that returns a fixed string. ***This method could be overridden in subclasses if a different prefix is needed.***
3. The getObject method utilizes getObjectInvocationPrefix to modify the identifier before calling getObjectAbsolute. This showcases method interaction within the class.
4. The getObjectWithDefault method demonstrates exception handling in Python. It tries to get an object using getObject and returns defaultObject if any exception is caught during the process. This method logs the caught exception to the console (***a placeholder for more sophisticated logging mechanisms you might use in a real application***).

# Then it advises on Typing and Type hints (Surprising as well)

**Using Any** allows these methods to accept and return objects of any type, mirroring the flexibility of the Object type in languages like Java.

The exception handling in getObjectWithDefault is basic and for demonstration purposes. ***Depending on your needs, you might want to handle specific exceptions or log them differently.***