Summary of Analysis and Interaction Protocol – Python-Java Integration

Approaches

The problem statement is as follows:

- Students have been writing their solutions to the Game of Amazons project (which is Java-based application with both client and server sides written in Java using the server and client APIs for SmartFoxServer) for COSC 322 at UBC, and it would be more convenient if there was a way to allow students to write their solutions in Python and have the Python code communicate directly with the Java-based SmartFoxServer server API as there is no Python client API support for SmartFoxServer.

I researched a few approaches that could possibly work to solve this Python-Java integration problem, they are as follows:

1. Java Embedded Python (JEP): Capable of running Python in Java through the Java Native Interface (JNI), which could allow for students to write code in Python that will be able to communicate directly with the Java-based SmartFoxServer server API.
2. Jython: Could have executed python code statements directly from within Java code, but it does not have Python 3.x support, therefore, it cannot be considered as a serious option.
3. JPype: Can use Python-like syntax to write code that merges with Java code in a new syntax, which may require students to learn the syntax, which could potentially be a drawback.
4. Py4J: Another viable option for calling Java objects with Python code and can allow for users to write their code by pulling the Java objects that can communicate with the Java SmartFoxServer API.
5. PyJnius: Allows Python to access Java classes through the JNI, which could allow for students to write code in Python that will be able to communicate directly with the Java-based SmartFoxServer server API.
6. GraalVM: Allows clean compatibility and support between Python and Java and seems like a good option.
7. jpy – Java and Python communication allowed in both directions and does what JEP does and seems like a very good option as well.

From the final 7 approaches, more research and trial implementations will be needed to pick one approach, but I will do trial runs to see if some basic Python-Java communications work with GraalVM, jpy, and JPype, and PyJnius first before exploring the other approaches.

References

1. https://github.com/ninia/jep - JEP
2. https://www.jython.org/ - Jython
3. https://jpype.readthedocs.io/en/latest/index.html - JPype
4. https://www.py4j.org/ - Py4J
5. https://github.com/kivy/pyjnius - PyJnius
6. https://www.graalvm.org/ - GraalVM
7. https://github.com/jpy-consortium/jpy - jpy