Project report

Group 7

Mohamad Omar Eid Dalal

Moei21@student.bth.se

Ahmad Kamal Eddin

Ahka21@student.bth.se

Gubran Alshekh-ali

Gual21@student.bth.se

Introduction

The purpose of this project is to develop and execute a comprehensive test suite for the Python pickle module to test if the same input always creates the same (serialized) output. The pickle module is used for serializing and deserializing Python objects, allowing complex data structures to be saved to a file and later restored. Our goal is to ensure that various data types and structures are correctly serialized and deserialized by pickle, verifying their integrity through consistent hash values.

To tackle this project, we used several tools such as GitHub for multiple testing purposes and comprehensive workflow, we also used the python libraries “pickle” to serialize and un-serialize different objects, “hashlib” to hash the pickled objects for testing purposes and “threading” to test the pickle module functionality throw different threads.

Test suite

Test cases

To build a test suite that is as general as possible and would cover most of the pickle module, we thought of multiple test cases and designed them to cover a wide range of scenarios. These include:

1. Basic Types: Testing simple data types such as integers, floats, strings, lists, tuples, sets, and dictionaries.

2. Complex Structures: Testing nested and more complex data structures.

3. Recursive Structures: Testing objects that reference themselves.

4. Floating Point Accuracy: Ensuring that floating point arithmetic is correctly serialized and deserialized.

5. Customized Objects: Testing user-defined classes and objects.

6. Customized Objects Modified: Modifing the testing user-defined classes and objects, and then testing them after the modifications.

7. Circular References: Testing objects with circular references.

8. Binary Data: Testing serialization of binary data.

9. Shared Objects: Ensuring that shared objects are consistently serialized across multiple threads.

10. GitHub Actions: Testing the behaviour of the pickle module across different operating systems and different python versions.

Applied strategies

- **Hash Comparison**

  To test the different serialized objects, we used SHA-256 hashes to compare the serialized form of objects.

- **Custom Classes**

  To test serialization of user-defined objects, we defined custom classes in our test file.

- **Threading**

  To test thread safety, we run serialization in multiple threads.

- **GitHub Actions (workflows-testing)**

  To test the behaviour of the pickle module across different operating systems, such as Ubuntu Linux, and different Python (but not pickle) versions, such as python 3.7 and 3.8.

Completeness and Traceability Matrix

To show the competence of our tests, we made a traceability matrix that outlines the test cases, data types involved, and aspects being tested, ensuring comprehensive coverage of the pickle module functionalities.

Traceability Matrix

Test Case	Data Type	Cross-Operating System	Cross-version	Special Cases	Floating Point Accuracy
1. Basic Data Types	Integers, floats, strings, lists, sets, dicts	Yes	Yes	No	Yes
2. Complex Data Structures	Nested dicts, lists, tuples	Yes	Yes	No	Yes
3. Floating Point Accurecy	Floating-point numbers	Yes	Yes	Yes (Different precisions)	Yes
4. Recursive Structures	Recursive lists	Yes	Yes	Yes (Recursive structures)	No
5. Custom Object Serialization	Custom class with custom methods	Yes	Yes	Yes (Custom objects after modification)	No
6. Circular References	Objects with circular references	Yes	Yes	Yes (Circular references)	No
7. Binary Data	Byte arrays, streams	Yes	Yes	Yes (Binary objects)	No
8. Shared Objects Across Threads	Shared objects, multithreading	Yes	Yes	No	Yes
9. GitHub Actions	Testing across multiple OS and python versions	Yes	Yes	Yes	Yes

Summary and Findings

1. Basic Types

Description: Testing simple data types such as integers, floats, strings, lists, tuples, sets, and dictionaries.

Data Types: [123, 123.456, "hello", [1, 2, 3], (1, 2, 3), {1, 2, 3}, {'a': 1, 'b': 2}]

Results: All tests passed.

2. Complex Structures

Description: Testing nested and more complex data structures.

Data Types: [{'a': [1, 2, 3], 'b': {'x': 'y'}}, [[1, 2], [3, 4, 5], [6]], {"key1": (1, 2), "key2": {3, 4}}]

Results: All tests passed.

3. Recursive Structures

Description: Testing objects that reference themselves.

Data Types: Recursive lists.

Results: Test passed for recursive structure.

4. Floating Point Accuracy

Description: Ensuring that floating point arithmetic is correctly serialized and deserialized.

Data Types: [0.1 + 0.2, 0.001 + 0.002, 0.00100 + 0.002000]

Results: All tests passed, including tests for different precisions.

5. Customized Objects

Description: Testing user-defined classes and objects.

Data Types: Custom class Person with methods.

Results: Test passed for unmodified custom object.

6. Customized Objects Modified

Description: Modifying user-defined classes and objects, then testing them after modifications.

Data Types: Custom class Person with modified state.

Results: Test passed for modified custom object.

7. Circular References

Description: Testing objects with circular references.

Data Types: Custom classes Ahmad and Gubran with circular references.

Results: Test passed for circular references.

8. Binary Data

Description: Testing serialization of binary data.

Data Types: byte array([120, 3, 255, 0, 100])

Results: Test passed for binary data.

9. Shared Objects

Description: Ensuring that shared objects are consistently serialized across multiple threads.

Data Types: {"numbers": [1, 2, 3], "letters": ["a", "b", "c"]}

Results: Test passed for shared objects across threads.

10. GitHub Actions

Description: Testing the behaviour of the pickle module across different operating systems and different Python versions.

Environment: GitHub Actions CI environment on Ubuntu with Python versions 3.7, 3.8, 3.9.

Results: All tests passed across all specified environments.

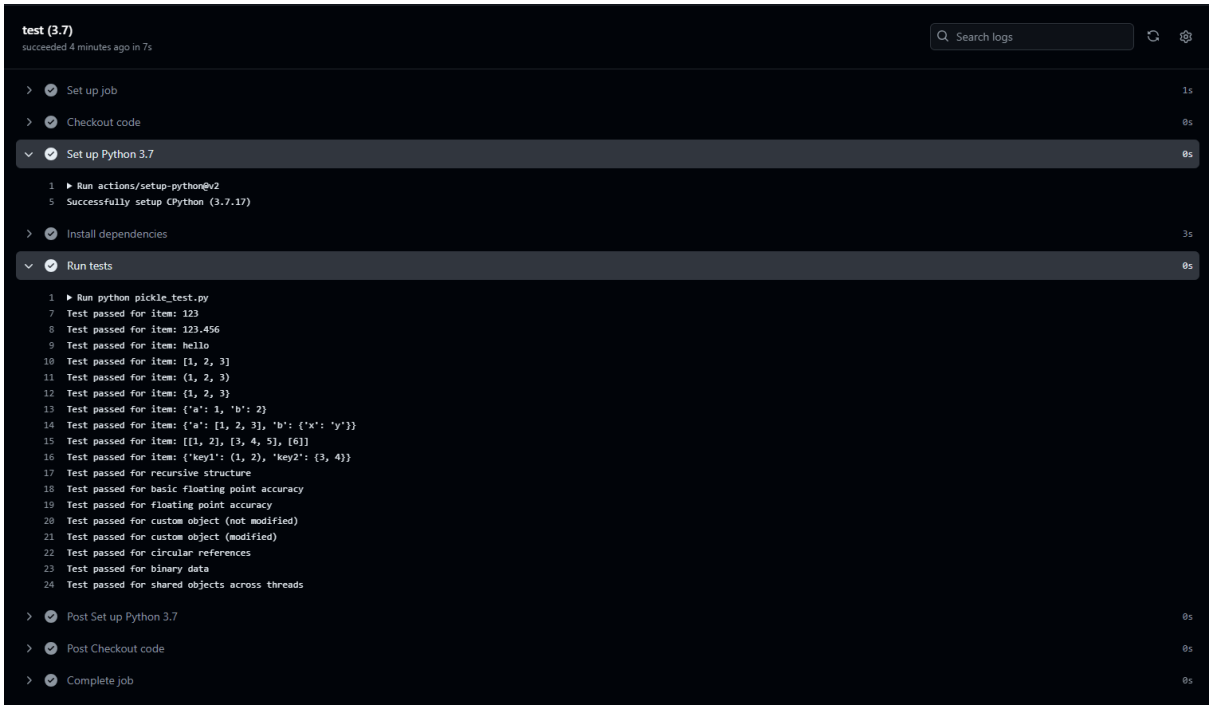

Figure 1: GitHub Actions CI environment - Example of a test summary

Limitations and Shortcomings

- **Code Coverage:** While our test suite is extensive, there may be edge cases not covered by our tests.

- **Performance Testing:** Our test suite does not measure the performance or efficiency of serialization and deserialization.

- **Environmental Differences:** Tests were conducted in a controlled CI environment (GitHub Actions on Ubuntu), which may not reflect all potential real-world environments.

Team Contributions

1. Omar: Developed initial test cases, created the create_pickle.py script for understanding and visualizing pickling, and implemented tests for basic types, floating point accuracy, custom objects and sharing across threads and GitHub actions environment. Also helped with writing the report

2. Ahmad: Contributed to the development of complex structure tests, Building the traceability matrix and writing the report.

3. Gubran: Worked on recursive structures, creating the GitHub actions test, Binary data test and circular references, as well as writing the report.

Repository Link

The code complies with PEP8 coding style guidelines and is available on GitHub. You can find the repository here:

GitHub Repository: https://github.com/Omardll001/projektReport_TestningAvMjukvara.git

Additional Points

Before starting the project, we created a file named "create_pickle.py" to understand how pickling works and to visualize it.

In this script, we created a simple Person object, modified its age, and serialized the object using pickle. After creating a pickled data file (testPerson.pickle), we imported the pickled data again into the Python file and deserialized it using pickle.load().