Python Interview Exercises

Purpose

The purpose of these exercises is to establish a common, non-biased baseline for comparisons of developer skills and abillity using the development language Python. Further, it is expected that:

* Each developer will write code to satisfy the requirement in the exercise,
* The interviewer will evaluate the code and run it against a standard Python environment at the time (see procedures below)
* The code will be discussed during the call and used as an evaluation criteria for hire

Procedures

* These exercises will tend to 'build' on each other - so feel free to leverage the same code between the exercises
* The developer will complete all exercises by themselves - and will represent that the code is their own / authored by the person being interviewed
  * Do not copy other's code - it should be entirely 'yours'
  * If during the interview process we find out that someone else has authored the code, the candidate will be disqualified from the position
* Each python exercise should be completed as a single, stand-alone file. Importing from other files and libraries is fine, provided that you provide references in the code documenting the dependencies you're using
  * Exercise 1 should be named `exercise1.py`, Exercise 2 should be named `exercise2.py`, etc
  * Test cases (if applicable) should be named using the same nomeclaiture for each exercise; `test_exercise1.py` corresponds to test cases for exercise 1, `test_exercise2.py` for exercise 2, etc
  * If common libraries are being written (for exercises 3-4), it's fine to keep them generic and add them into the list of files
* When completed, zip up all exercises, common files, and test cases into a single ZIP file and return it to HPE ahead of any interviews.

Exercises

1. **Exercise 1:** create a simple API listener / server process that can respond to unauthenticated `/ping` calls. Be prepared to discuss any framework decisions you make during the interview
2. **Exercise 2:** Come up with a method that can be used for authentication using pre-shared secrets that can be added to the header of the request. Create an `/authorize` endpoint for the API that will authorize the header against a pre-shared key that can be loaded as a secret when the API starts
3. **Exercise 3:** Assume that you will be reading from a database (use `sqlite` for this exercise). Create an endpoint called `/save` that allows POSTING of a key / value pair into a database table.
   * Create another endpoint that allows the user to `/get` the data from the database
   * Create a final endpoint that removes the data from the database `/delete`
4. **Exercise 4:** Employ a caching mechanism for your application to speed up operations on `/save` and `/get`. Be prepared to discuss / defend your choice and approach

5. **Exercise 5:** Write test cases (unit and end-to-end) that verify both the operation and business logic you may have written. Assume the tests will use `pytest` and be prepared to discuss testing topics in general.