Practical Computing: Exercise 3

Information about unit testing and profiling modules in python:

Unit Testing:

Unit Testing is the first level of software testing where the smallest testable parts of a software are tested. This is used to validate that each unit of the software performs as designed. The unit test framework is python's x Unit style framework. Unit testing is a technique in which particular module is tested to check by developer himself whether there are any errors. The primary focus of unit testing is testing an individual unit of system to analyse, detect, and fix the errors.

The developers are expected to write automated test scripts, which ensures that each and every section or a unit meets its design and behaves as expected. Though writing manual tests for your code is definitely a tedious and time-consuming task, Python's built-in unit testing framework has made life a lot easier. The unit test framework in Python is called unit test, which comes packaged with Python.

Unit testing makes your code future proof since you anticipate the cases where your code could potentially fail or produce a bug. Though you cannot predict all of the cases, you still address most of them.

Profiling:

Python provides many excellent modules to measure the statistics of a program. This makes us know where the program is spending too much time and what to do in order to optimize it. It is better to optimize the code in order to increase the efficiency of a program. So, perform some standard tests to ensure optimization and we can improve the program in order to increase the efficiency.

Using Timers:

Timers are easy to implement and they can be used anywhere at a program to measure the execution time. By using timers, we can get the exact time and we can improve the program where it takes too long. Time module provides the methods in order to profile a program.

cProfile and profile provide deterministic profiling of Python programs. A profile is a set of statistics that describes how often and for how long various parts of the program executed. These statistics can be formatted into reports via the pstats module.

- The Python standard library provides two different implementations of the same profiling interface:

  cProfile is recommended for most users; it's a C extension with reasonable overhead that makes it suitable for profiling long-running programs. Based on lsprof, contributed by Brett Rosen and Ted Czotter.

- cProfile, a pure Python module whose interface is imitated by cProfile, but which adds significant overhead to profiled programs. If you're trying to extend the profiler in some way, the task might be easier with this module.