A3/CIS*4150

Mocking

courselink.uoguelph.ca

A. Hamilton-Wright

Overview

In this assignment we will write a test suite using mocking.

Note that this is an individual assignment. All of the submitted work should be your own, and based on the starting code and course examples. Any work you have included from elsewhere (*e.g.*, the examples) must be noted and cited.

Skills

specification (6/6)	██████
testing tools (3/6)	███░░░
mocking frameworks (6/6)	██████
unit testing (6/6)	██████

(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Main Focus).]

Image description

Pretending to be a pirate. Photo source shutterstock.com Shutterstock license.

CIS 4150/F21: Assignment Three

This assignment will see us write real test cases using a mocking framework, as well as explain the test cases we have created.

Python mocking frameworks

There are several Python mocking frameworks, however we will use the mocking tools available within the `unittest` framework.

The class documentation for these tools is available here:

- unittest – the main `unitest` framework
- unittest.mock – the mocking tools within the `unittest` framework

These pages are a good reference, but difficult to learn from. This tutorial gives an overview of how to set up mocks in this framework:

- Kay, N. (2013): An Introduction to Mocking in Python, `toptal.com`

Key points

- In general, we can create a mock using either the class `unittest.Mock` or `unittest.MagicMock` to replace either an object instance OR a method on an instance
- One way to do this is to manually create an instance of a mock, and simply plug it into the object (Python allows runtime edits to objects)
- Another (more common) way to do this is to use `unitest.mock.patch` to do this temporarily, so that the "patch" is added and removed for a single block of our testing code
  - to replace the whole object, we will use "`unittest.mock.patch`"
  - to replace an instance on an object, we will instead use the syntax of "`unittest.mock.patch.object`"
    - (yes this is very confusing)
  - we can use "patch" in a decorator pattern application using `@patch` OR as a context manager. We typically see it applied as a decorator

Example

The pattern we see is:

1) replace some functionality with a mock
2) make a call through to the mock
3) check that the mock saw what was expected
4) remove the mock and replace everything as it was (patch makes this easier)

The following files contain examples of this:

- `example_manual_mocking.py` – a fully manual example
- `example_patch_mocking.py` – an example using a `patch.object` mock

Objective

You have been given an updated copy of the "GPA" calculation library written in python. We want to create test cases to determine whether it is indeed doing the right thing, now that it has been updated.

We will write a test suite using a mixture of the unittests that we used previously, as well as mocking when appropriate. Part of the assignment is to determine when we wish to use which strategy.

A3/CIS*4150

Mocking

courselink.uoguelph.ca

A. Hamilton-Wright

Overview

In this assignment we will write a test suite using mocking.

Note that this is an individual assignment. All of the submitted work should be your own, and based on the starting code and course examples. Any work you have included from elsewhere (*e.g.*, the examples) must be noted and cited.

Skills

specification (6/6)

testing tools (3/6)

mocking frameworks (6/6)

unit testing (6/6)

(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Main Focus).]

Image description

Pretending to be a pirate. Photo source shutterstock.com Shutterstock license.

We wish to write tests to verify whether the following functionality is correct:

- do the lettergrades function correctly over their entire range?
  - for each of A+ through F work on their full range
- do the additions of grades use the lettergrade conversion functions properly?
  - do they call the lettergrade conversion functions the appropriate number of times and with the correct parameters?
- do the term objects get created and cleaned up properly?
  - are the right grades placed in the terms?
  - does term removal operate correctly?

Assignment Task

We wish to design, and then implement, a set of tests to verify whether the code works as intended.

Consider which of the above tasks need mocks, and which don't.

Design

Design a list of test cases to verify whether the above stipulations are met. For each, indicate:

- what the test case will verify
- whether mocks are used, and if so, how
- what supplied arguments will be provied, and to which functions
- what expected values will be observed
- (based on the results below) whether the test passes

This will go into your README file. It should be a human-readable format that someone familiar with testing but not with Python can easily understand.

Coding

Write up your test cases, using the `unittest` framework. Ensure that each test case in your code matches with a test case in your document, and has the same values as described.

Note that ONLY writing the code will not suffice for the document information.

Ensure that your code works using `python3` on `linux.socs.uoguelph.ca`.

Handing In Your Assignment

Collect all of the files you wish to submit into one archive (`.zip` or `.tar`) and upload to the Courselink dropbox.