TDD FileLoader v1.0

In this solution, the production class actually loads a file from the disk.

Unit Test

```
import unittest
from app.file_loader import FileLoader

class LiveFileLoaderTest(unittest.TestCase):
    """
    The initial design is described in this test.

    The weakness should be obvious — the file to be loaded and its location.
    I use a shared network drive to run this code from different machines, normally
    developing from a PC. When I ran the code on the laptop from a cafe it
    immediately failed because the C: on the laptop was completely different
    to the PC, so the original file C:/tmp/KeyboardHandler.txt did not exist.

    THIS IS A GREAT EXAMPLE OF WHY THE UNIT TEST AND CUT SHOULD NOT BE STRONGLY
    LINKED TO ANY IO — NETWORK, DB, AND FILE SYSTEM.
    """

    def test_load_all_of_file_using_inbuilt_files_type(self):
        # Arrange
        file_to_load = "sample.txt"
        expected_bytes_read = 12
        cut = FileLoader(file_to_load)

        # Act
        bytes_read = cut.load_file(file_to_load)

        # Assert
        self.assertEqual(expected_bytes_read, bytes_read)

if __name__ == "__main__":
    unittest.main()
```

The CUT FileLoader

```
class FileLoader:
    def __init__(self, file_to_load):
        self.file_to_load = file_to_load
        self.lines = []

    def load_file(self, fname):
        try:
            with open(fname, encoding='utf-8') as f:
                self.lines = f.readlines()
        except IOError:
            self.lines = []

        return self._calculate_file_size()

    def _calculate_file_size(self):
        total_length = sum(len(line) for line in self.lines)
        return total_length
```