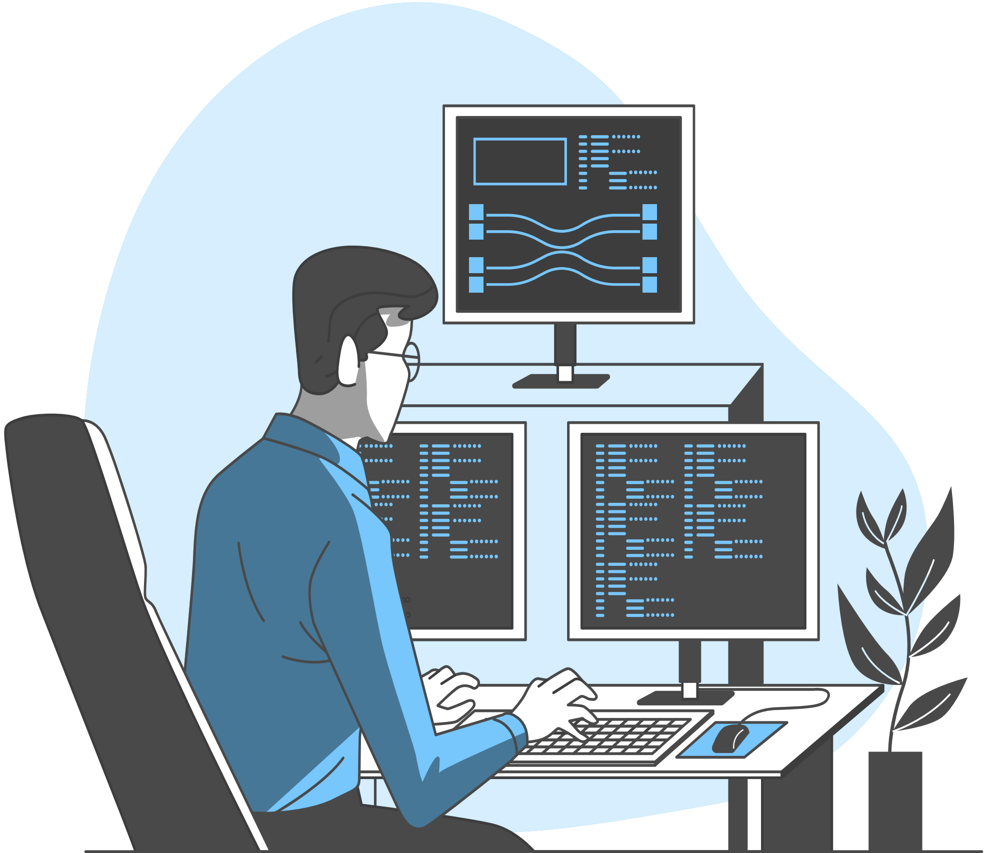


PyLight XL

Group 13:

Mitchell Hicks 100707709
Austin Page 100703400
Taha Hashmat 100689792
Yakhneshan Siva 100725236

Table of Contents

01

Introduction

What our software does and why we choose it

02

Features

Implementation of added functionality that makes the software better

03

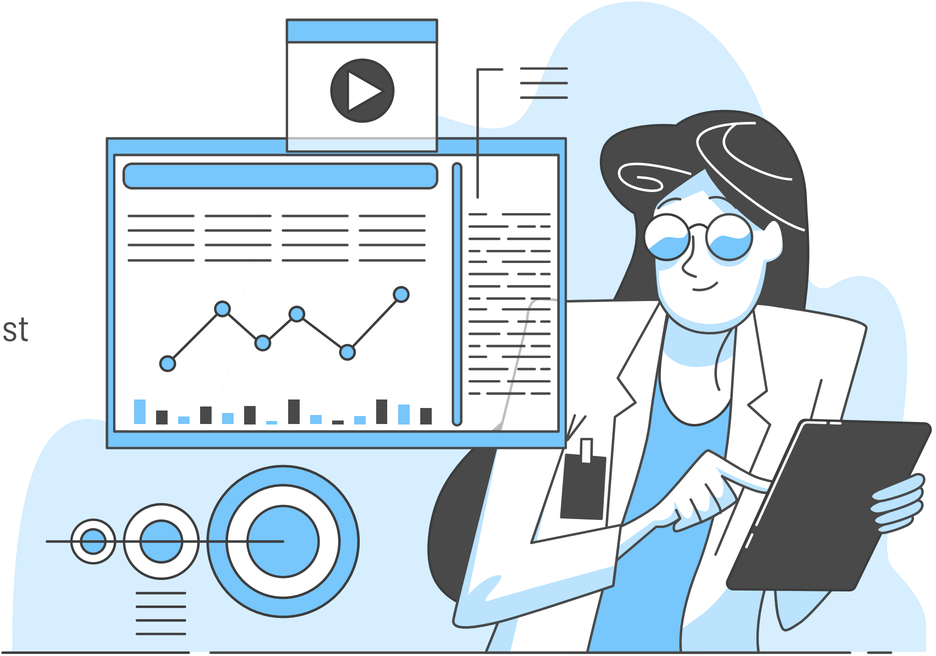
Testing

The test data we generate and test cases we implement for our new features

04

Improvements

What measures we took to make the overall software better



01

Introduction

WHAT IS PYLIGHTXL

Pylightxl is a lightweight microsoft excel reader/writer. The Pylightxl library allows developers to access data represented in an excel file and make computations on it.

WHY WE CHOSE PYLIGHTXL

- Pylightxl is not very popular which allows us to make valuable upgrades to the code
- Pylightxl is used for data analytics and getting data from excel file, as a group we are all interested in data analysis
- Pylightxl is not as complex as a library like Pandas, this will help us focus on a specific improvement without overwhelming us

02

Features

Description of Features

01. New File Types

Implementation:

→ Writes a .txt file from pylightxl database.

→ For db that have more than one sheet, multiple files with the sheet name tagged on the end

Improvements:

→ Gives the user an option to edit a wider range of file types, instead of just .csv files

02. User Manual

Implementation:

→ Takes in key words from the user and prints out the description

→ If no keyword entered, a list of all keywords is shown

Improvements:

→ Primarily implemented to assist the user if they are confused with how to work the software

03. Statistics

Implementation:

→ Returns the number or rows and columns present in a sheet

Improvements:

→ Gives the user a clearer idea of the data they are working with

03

Testing

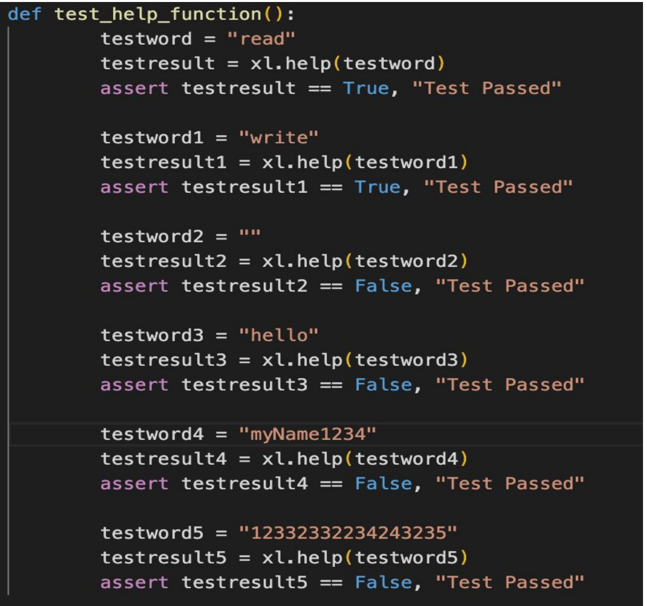
Tools Used

1. Test Cases - Pytest Unit Testing
2. Test Data - parameterized

Test Cases

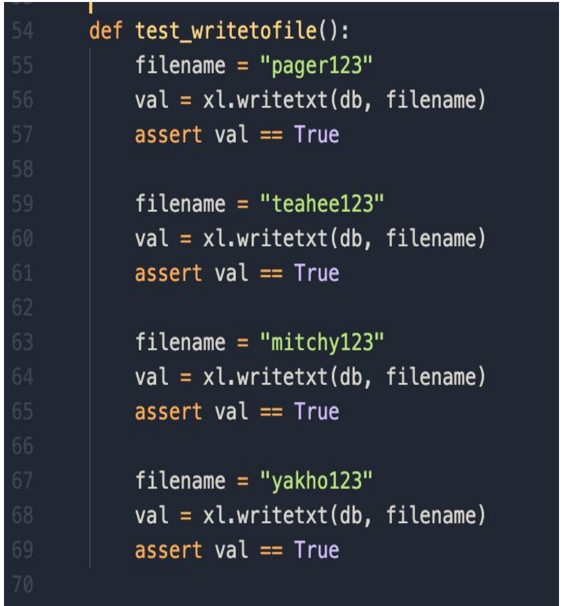
pytest

User Manual Test



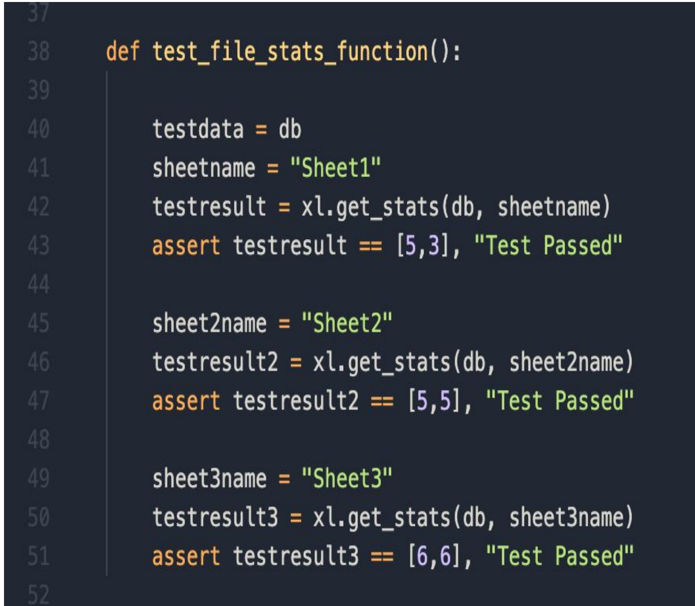
```python
def test_help_function():
        testword = "read"
        testresult = xl.help(testword)
        assert testresult == True, "Test Passed"

        testword1 = "write"
        testresult1 = xl.help(testword1)
        assert testresult1 == True, "Test Passed"

        testword2 = ""
        testresult2 = xl.help(testword2)
        assert testresult2 == False, "Test Passed"

        testword3 = "hello"
        testresult3 = xl.help(testword3)
        assert testresult3 == False, "Test Passed"

        testword4 = "myName1234"
        testresult4 = xl.help(testword4)
        assert testresult4 == False, "Test Passed"

        testword5 = "12332332234243235"
        testresult5 = xl.help(testword5)
        assert testresult5 == False, "Test Passed"
```

Write to File Test



```python
def test_writetofile():
    filename = "pager123"
    val = xl.writetxt(db, filename)
    assert val == True

    filename = "teahee123"
    val = xl.writetxt(db, filename)
    assert val == True

    filename = "mitchy123"
    val = xl.writetxt(db, filename)
    assert val == True

    filename = "yakho123"
    val = xl.writetxt(db, filename)
    assert val == True
```

Statistics Test



```python
def test_file_stats_function():

    testdata = db
    sheetname = "Sheet1"
    testresult = xl.get_stats(db, sheetname)
    assert testresult == [5,3], "Test Passed"

    sheet2name = "Sheet2"
    testresult2 = xl.get_stats(db, sheet2name)
    assert testresult2 == [5,5], "Test Passed"

    sheet3name = "Sheet3"
    testresult3 = xl.get_stats(db, sheet3name)
    assert testresult3 == [6,6], "Test Passed"
```

Test Data Generation

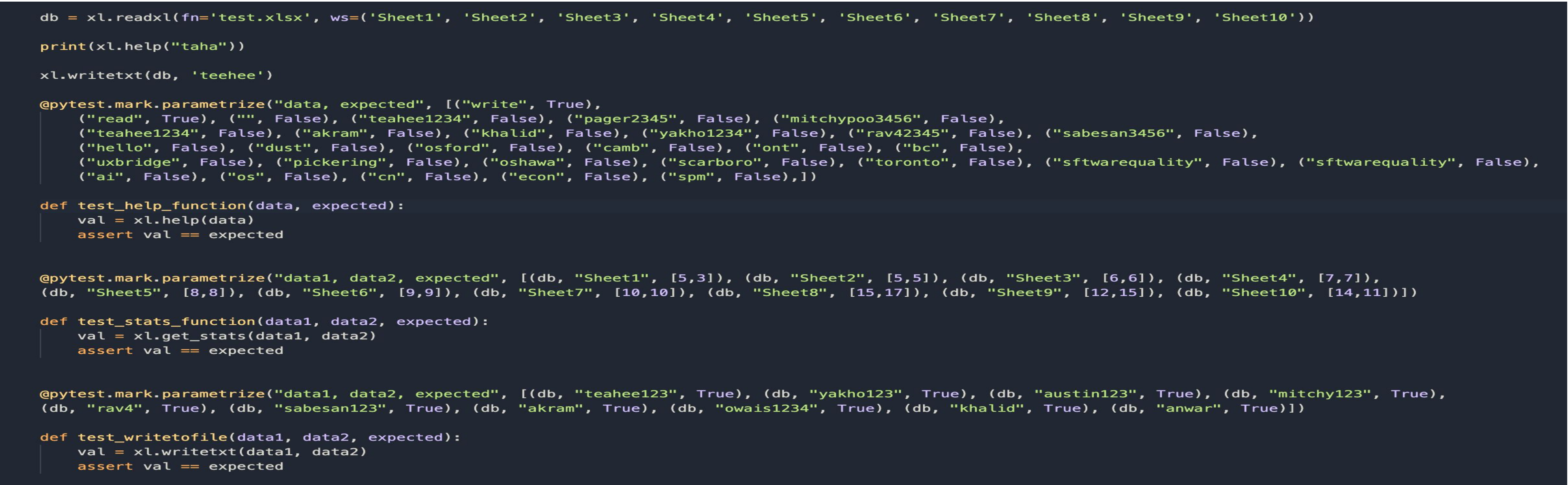
parameterized

What is Parameterized?

- A pytest library that allows you to generate large data input with their expected outputs.

- Parameterized streamlines the process of data generation by allowing the user to test their test cases with large data inputs to ensure efficient code

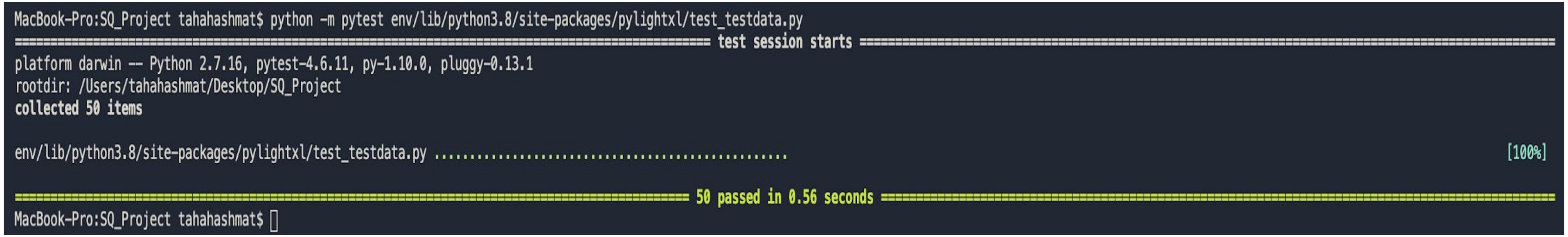
- Over 50 test cases for our testing functions which provides us a with a thorough test coverage



```
db = xl.readxl(fn='test.xlsx', ws=('Sheet1', 'Sheet2', 'Sheet3', 'Sheet4', 'Sheet5', 'Sheet6', 'Sheet7', 'Sheet8', 'Sheet9', 'Sheet10'))

print(xl.help("taha"))

xl.writetxt(db, 'teehee')

@pytest.mark.parametrize("data, expected", [("write", True),
    ("read", True), ("", False), ("teahee1234", False), ("pager2345", False), ("mitchypoo3456", False),
    ("teahee1234", False), ("akram", False), ("khalid", False), ("yakho1234", False), ("rav42345", False), ("sabesan3456", False),
    ("hello", False), ("dust", False), ("osford", False), ("camb", False), ("ont", False), ("bc", False),
    ("uxbridge", False), ("pickering", False), ("oshawa", False), ("scarboro", False), ("toronto", False), ("sftwarequality", False), ("sftwarequality", False),
    ("ai", False), ("os", False), ("cn", False), ("econ", False), ("spm", False),])

def test_help_function(data, expected):
    val = xl.help(data)
    assert val == expected


@pytest.mark.parametrize("data1, data2, expected", [(db, "Sheet1", [5,3]), (db, "Sheet2", [5,5]), (db, "Sheet3", [6,6]), (db, "Sheet4", [7,7]),
(db, "Sheet5", [8,8]), (db, "Sheet6", [9,9]), (db, "Sheet7", [10,10]), (db, "Sheet8", [15,17]), (db, "Sheet9", [12,15]), (db, "Sheet10", [14,11])])

def test_stats_function(data1, data2, expected):
    val = xl.get_stats(data1, data2)
    assert val == expected


@pytest.mark.parametrize("data1, data2, expected", [(db, "teahee123", True), (db, "yakho123", True), (db, "austin123", True), (db, "mitchy123", True),
(db, "rav4", True), (db, "sabesan123", True), (db, "akram", True), (db, "owais1234", True), (db, "khalid", True), (db, "anwar", True)])

def test_writetofile(data1, data2, expected):
    val = xl.writetxt(data1, data2)
    assert val == expected
```

Test Cases Output

- Out of the 50 written test cases, we had a 100% success rate with our testing, with all 50 test cases returning the desired value



```
MacBook-Pro:SQ_Project tahahashmat$ python -m pytest env/lib/python3.8/site-packages/pylightxl/test_testdata.py
=================================================================================================== test session starts ===================================================================================================
platform darwin -- Python 2.7.16, pytest-4.6.11, py-1.10.0, pluggy-0.13.1
rootdir: /Users/tahahashmat/Desktop/SQ_Project
collected 50 items

env/lib/python3.8/site-packages/pylightxl/test_testdata.py ..................................................                                                                                                  [100%]

================================================================================================ 50 passed in 0.56 seconds ================================================================================================
MacBook-Pro:SQ_Project tahahashmat$ 
```

04

Improvements

Tools Used

Dynamic Analysis - Monitors

Timestamps – Monitor

To look at the improvements are group made to the added functions, we compared the original execution and improved execution times of each feature

	User Manual Feature	New File Type Feature	getStats Feature
Original Time (in seconds)	0.00205	0.00502	0.0314
Improved Time (in seconds)	0.00199	0.00100	0.00102
Difference (in seconds)	0.00185	0.0492	0.0212

Thanks!