MAPÚA UNIVERSITY
SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 4:
Design Patterns and Unit Testing

CPE106L (Software Design Laboratory)

Member 1 Aljehro R. Abante
Member 2 Cerdan Karl T. Alarilla
Member 3 Daryl Jake A. Fernandez

Group No.: **3**
Section: **E01**

PreLab

Readings, Insights, and Reflection

The selection of tools for creating Python apps is diverse and flexible. Software architecture can be clearly represented visually with the help of UMLet, an efficient Windows tool that makes creating UML diagrams easier. Furthermore, using a Linux environment inside a Hadoop virtual machine (VM) gives developers flexibility and is a substitute for the popular Anaconda terminal, which is renowned for its efficient package and environment management tools. With its extensive feature set and extensibility, Visual Studio Code further improves the development experience by providing a thorough integrated development environment for Python projects that include coding, debugging, and testing.

A comprehensive approach to learning Python for desktop application development and testing is provided via the assigned texts. Laura Cassell and Alan Gauld's "Python Projects" are very educational because it includes useful projects that encourage learning by doing. On page 221 of the most recent edition, the Model-View-Controller (MVC) pattern is emphasized, highlighting a basic design paradigm that is crucial for organizing applications. By providing comprehensive instructions for creating console and graphical user interface (GUI) versions of a Tic Tac Toe game, the previous edition strengthens this emphasis and highlights the significance of real-world application and user interface design.

The stated materials offer a thorough method for learning Python for developing and testing desktop applications. Because it contains practical projects that promote learning by doing, Laura Cassell and Alan Gauld's "Python Projects" are incredibly instructive. The Model-View-Controller (MVC) pattern is highlighted on page 221 of the latest edition, showcasing a fundamental design paradigm that is essential for structuring applications. The previous edition reinforces this focus and emphasizes the importance of real-world application and user interface design by offering thorough instructions on developing console and graphical user interface (GUI) versions of a Tic Tac Toe game.

METIS #1:
Python Projects
Laura Cassell, Alan Gauld
Edition 1

ISBN: 9781118909195 (new)
9781118908891 (old)
Wiley Professional Development (P&T)

****** Chapter 4: Building Desktop Applications *******
- MVC Pattern - page 221 (new ISBN)

**** OLD Edition***
**** Tic Tac Toe (Console App), pp. 162 to 173***
**** Tic Tac Toe (GUI App), pp. 186 to 193***

==========
=

METIS #2:
Testing Python
Sale, D. (2014). Testing Python. Wiley Professional, Reference & Trade (Wiley K&L). https://bookshelf.vitalsource.com/books/9781118901243

==========
Professional Python (NOT AVAILABLE IN METIS)
Luke Sneeringer
Edition 1
ISBN: 9781119070832
Wiley Professional Development (P&T)
- Unit Testing, (Chap 11)

Answers to Questions

None

InLab

Procedure:

tictactoe_ui.py 1

E: > FILES > 2024 New Docus > CPE106L-4 > Lab_Exercises > Lab4 > tictactoe_ui.py > ...

```python
import tkinter as tk
import tkinter.messagebox as mb
import oxo_logic  # Ensure this module is available

top = tk.Tk()
top.title("Tic-Tac-Toe")

def buildMenu(parent):
    menus = (
        ("File", (("New", evNew),
                  ("Resume", evResume),
                  ("Save", evSave),
                  ("Exit", evExit))),
        ("Help", (("Help", evHelp),
                  ("About", evAbout)))
    )

    menubar = tk.Menu(parent)
    for menu in menus:
        m = tk.Menu(parent)
        for item in menu[1]:
            m.add_command(label=item[0], command=item[1])
        menubar.add_cascade(label=menu[0], menu=m)

    return menubar

def evNew():
    global status
    status['text'] = "Playing game"
    game2cells(oxo_logic.newGame())  # Begin a new game

def evResume():
    global status
```

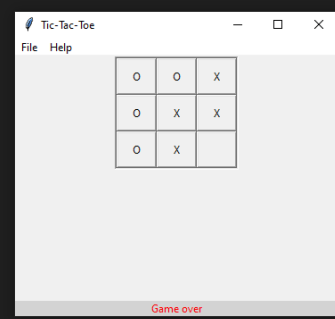


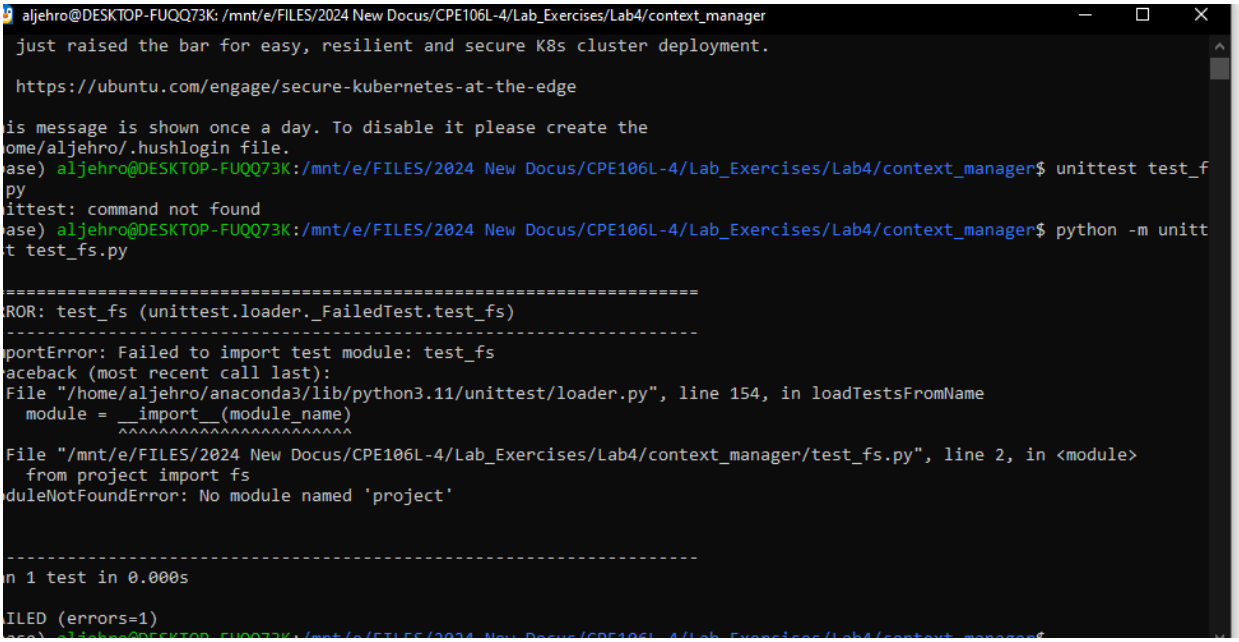
Figure 1. GUI and code of Tic-Tac-Toe

tictactoe_ui.py 1 | test_fs.py 1

> FILES > 2024 New Docus > CPE106L-4 > Lab_Exercises > Lab4 > context_manager > test_fs.py > TestExamples > test_print_contents_of_cwd_success

```python
from unittest import mock, TestCase
from project import fs


class TestExamples(TestCase):
    def test_print_contents_of_cwd_success(self):
        with mock.patch('project.fs.print_contents_of_cwd',return_value=b'test\nfoo\nbar\ntestx\n'):
            actual_result = fs.print_contents_of_cwd()
        expected_directory = b'tests'
        self.assertIn(expected_directory,actual_result)
```

Figure 2. unittest for context_manager

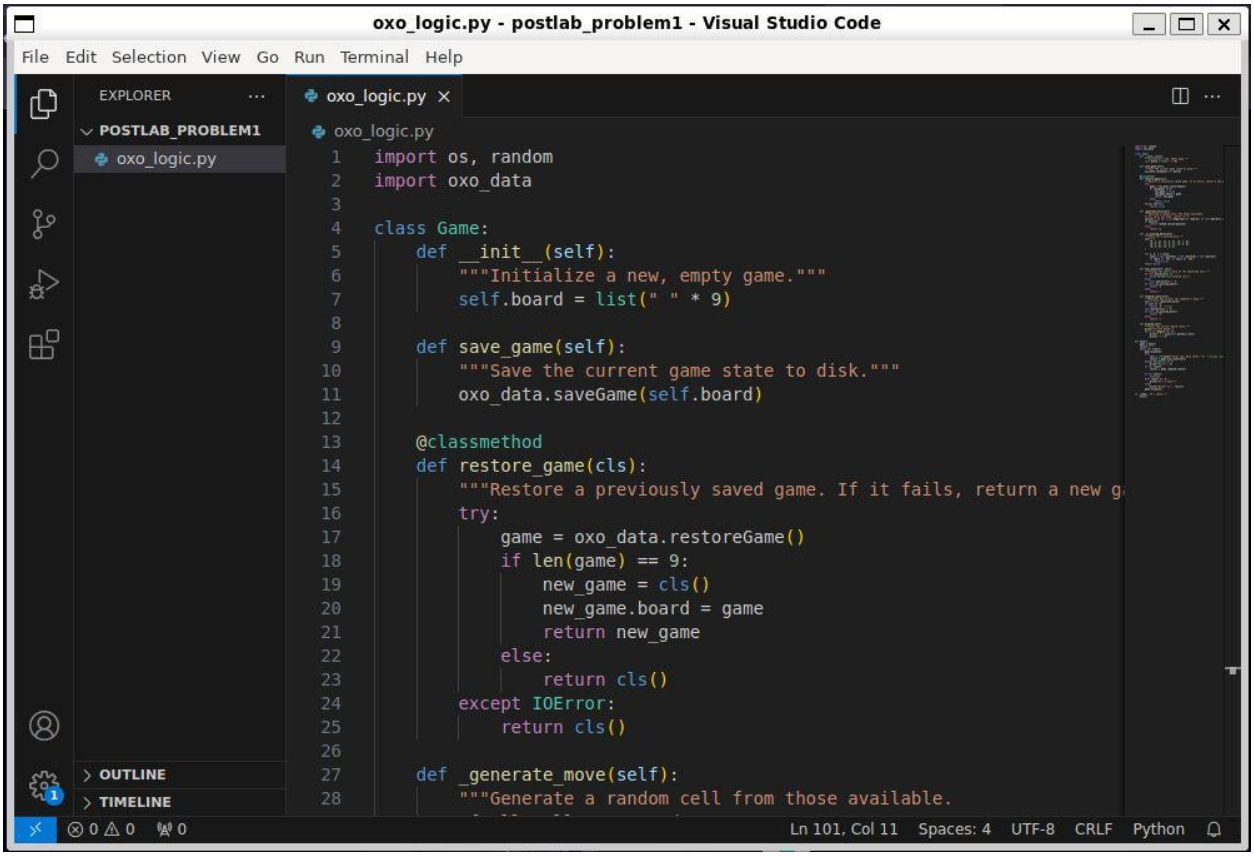


```
aljehro@DESKTOP-FUQQ73K: /mnt/e/FILES/2024 New Docus/CPE106L-4/Lab_Exercises/Lab4/context_manager
 just raised the bar for easy, resilient and secure K8s cluster deployment.

 https://ubuntu.com/engage/secure-kubernetes-at-the-edge

is message is shown once a day. To disable it please create the
ome/aljehro/.hushlogin file.
ase) aljehro@DESKTOP-FUQQ73K:/mnt/e/FILES/2024 New Docus/CPE106L-4/Lab_Exercises/Lab4/context_manager$ unittest test_f
py
ittest: command not found
ase) aljehro@DESKTOP-FUQQ73K:/mnt/e/FILES/2024 New Docus/CPE106L-4/Lab_Exercises/Lab4/context_manager$ python -m unitt
t test_fs.py

======================================================================
ROR: test_fs (unittest.loader._FailedTest.test_fs)
----------------------------------------------------------------------
portError: Failed to import test module: test_fs
aceback (most recent call last):
File "/home/aljehro/anaconda3/lib/python3.11/unittest/loader.py", line 154, in loadTestsFromName
  module = __import__(module_name)
           ^^^^^^^^^^^^^^^^^^^^^^^
File "/mnt/e/FILES/2024 New Docus/CPE106L-4/Lab_Exercises/Lab4/context_manager/test_fs.py", line 2, in <module>
  from project import fs
duleNotFoundError: No module named 'project'


----------------------------------------------------------------------
n 1 test in 0.000s

ILED (errors=1)
```

Figure 3. wsl for context_manager

PostLab

1. Convert the oxo-logic.py module to reflect OOP design by creating a Game class.




```python
import os, random
import oxo_data

class Game:
    def __init__(self):
        """Initialize a new, empty game."""
        self.board = list(" " * 9)

    def save_game(self):
        """Save the current game state to disk."""
        oxo_data.saveGame(self.board)

    @classmethod
    def restore_game(cls):
        """Restore a previously saved game. If it fails, return a new g
        try:
            game = oxo_data.restoreGame()
            if len(game) == 9:
                new_game = cls()
                new_game.board = game
                return new_game
            else:
                return cls()
        except IOError:
            return cls()

    def _generate_move(self):
        """Generate a random cell from those available.
```

Figure 1. code of oxo_logic.py OOP design with

Game class

2. Explore the Tkinter.filedialog module to get the name of a text file

3. Create a unit test program for testing the Tic Tac Toe Console App

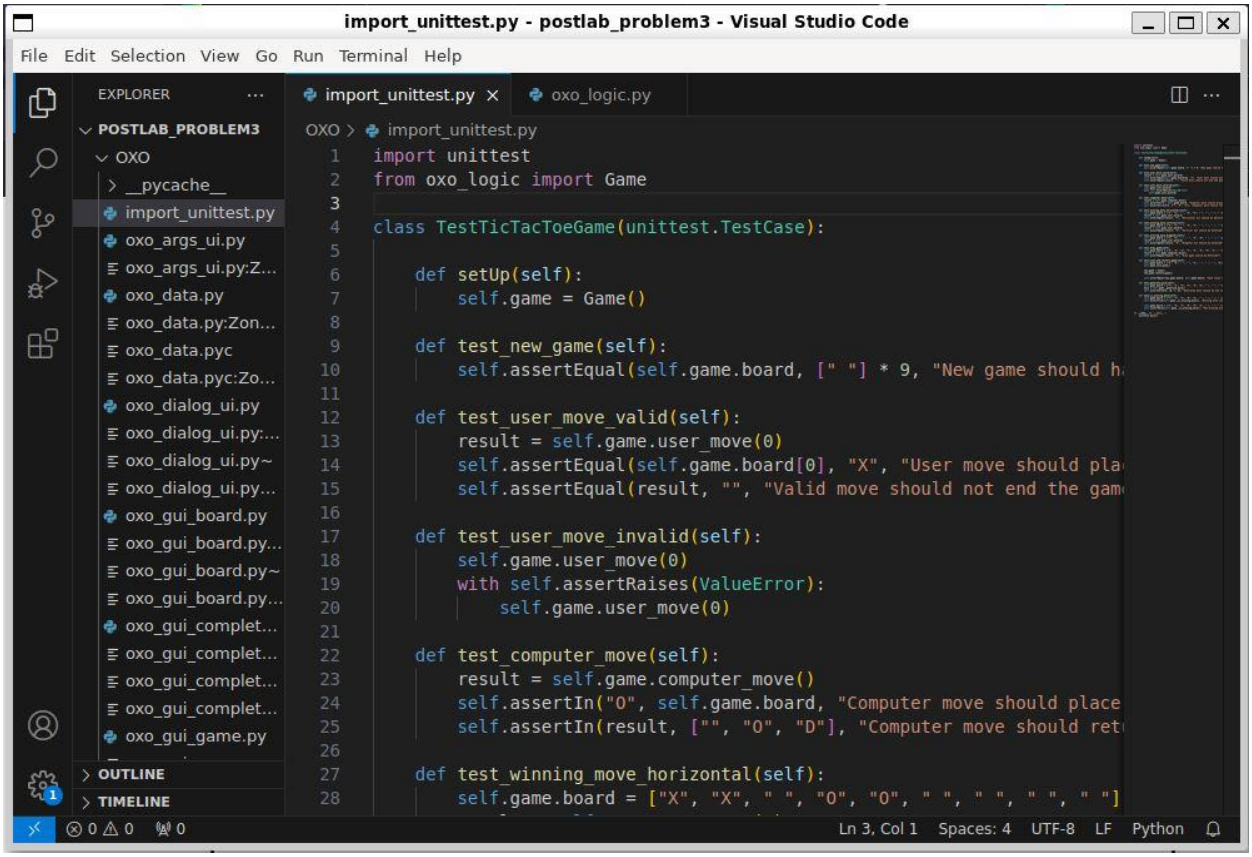


Figure 3. code of unit test program

```
(base) cerdan@DESKTOP-ICVPFLB:~/LocalRepo/cpe106l-4/Lab/Lab4/postlab/postlab_problem3/OXO$ python -m unittest import_un
ttest.py
E
======================================================================
ERROR: import_unittest (unittest.loader._FailedTest.import_unittest)
----------------------------------------------------------------------
ImportError: Failed to import test module: import_unittest
Traceback (most recent call last):
  File "/home/cerdan/anaconda3/lib/python3.12/unittest/loader.py", line 137, in loadTestsFromName
    module = __import__(module_name)
             ^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/cerdan/LocalRepo/cpe106l-4/Lab/Lab4/postlab/postlab_problem3/OXO/import_unittest.py", line 2, in <module>
    from oxo_logic import Game
ImportError: cannot import name 'Game' from 'oxo_logic' (/home/cerdan/LocalRepo/cpe106l-4/Lab/Lab4/postlab/postlab_prob
em3/OXO/oxo_logic.py)


----------------------------------------------------------------------
Ran 1 test in 0.000s
```

Figure 4. output of unit test program