Project On
CSE-3632
Operating Systems Lab

Simple CLI and GUI Shell in Python

Submitted to —

Mohammad Zainal Abedin

Assistant Professor ,Dept of CSE

Submitted by—

Md. Faisal Hoque Rifat C221076

Istahadul Hoque C221059

Project Report: SimpleShell CLI and GUI in Pyhton

Abstract:

- This project implements a simple command-line interface (CLI) and graphical user interface (GUI) shell in Python. The shell supports basic commands: ls, pwd, cd, and exit. The purpose of this project is to provide a minimalistic shell environment that demonstrates the integration of core Python functionalities for file system navigation and GUI development.

Introduction:

- Shells are an essential part of operating systems, enabling users to interact with the file system and execute commands. In this project, a Python-based shell was implemented to provide basic functionality for navigating the file system. The implementation includes both a CLI for text-based command execution and a GUI for a more user-friendly interaction.

- The commands supported by the shell include:
  - **ls:** Lists the files and directories in the current working directory.
  - **pwd:** Displays the current working directory.
  - **cd:** Changes the current working directory to a specified path.
  - **exit:** Exits the shell.

Features:

- **Basic Shell Implementation**:

- The project implements a basic shell interface for interacting with the file system.
- Users can run common file system commands like ls, pwd, cd, and exit.

- **Command-Line Interface (CLI)**:
  - A text-based interface where users type commands directly.
  - Commands like ls, pwd, and cd execute in the context of the current working directory.
- **Graphical User Interface (GUI)**:
  - A simple GUI that provides a user-friendly way to interact with the shell.
  - Commands can be entered in a text field, and their output is displayed in a designated area.
  - Buttons may allow quick execution of predefined commands.
- **Cross-Platform**:
  - The project uses Python libraries, making it compatible with multiple operating systems (Windows, macOS, Linux).
- **Core Command Support**:
  - **ls**: Lists files and directories in the current working directory.
  - **pwd**: Prints the current working directory.
  - **cd**: Changes the current working directory.
  - **exit**: Terminates the shell or GUI session

Implementation:

1. **Command-Line Interface (CLI):**
   - The CLI was implemented using Python's built-in modules, such as os and sys. It provides the user with a text-based interface to execute commands. The following steps outline the functionality:
     - **Input Parsing**: The shell accepts user input, parses it, and executes the appropriate command.
     - **Command Execution:**
       - **ls: Uses os.listdir()** to list files and directories.
       - **pwd: Uses os.getcwd()** to display the current working directory.
       - **cd: Uses os.chdir()** to change directories, with error handling for invalid paths.
       - **exit:** Terminates the shell.

2. **Graphical User Interface (GUI):**
   - The GUI was implemented using the tkinter library to provide a basic graphical representation of the shell. The key features include:

- A text box for user input.
- A display area for command outputs.
- Buttons for executing commands and exiting the shell.
- The GUI processes the same commands as the CLI and displays the results in the output area.

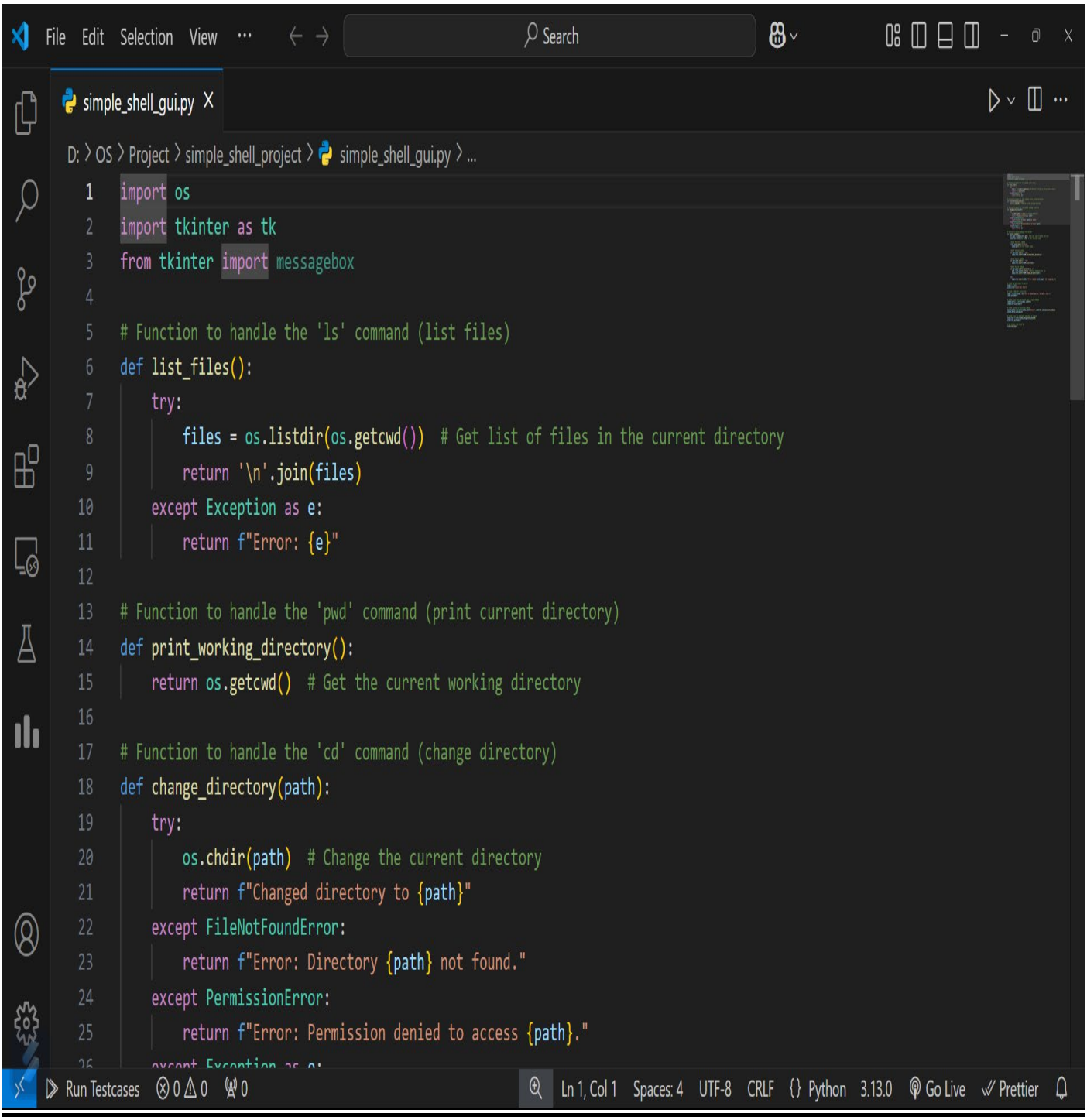
Code Structure:

1. Main CLI Implementation:

```python
import os

# Function to handle the 'ls' command (list files)
def list_files():
    try:
        files = os.listdir(os.getcwd())  # Get list of files in the current directory
        return '\n'.join(files)
    except Exception as e:
        return f"Error: {e}"

# Function to handle the 'pwd' command (print current directory)
def print_working_directory():
    return os.getcwd()  # Get the current working directory

# Function to handle the 'cd' command (change directory)
def change_directory(path):
    try:
        os.chdir(path)  # Change the current directory
        return f"Changed directory to {path}"
    except FileNotFoundError:
        return f"Error: Directory {path} not found."
    except PermissionError:
        return f"Error: Permission denied to access {path}."
    except Exception as e:
        return f"Error: {e}"
```

- **Link:** https://github.com/FaisalHoqueRifat/OS/blob/main/Project/simple_shell_project/simple_shell_cli.py

2. Main GUI Implementation:



- **Link:** https://github.com/FaisalHoqueRifat/OS/blob/main/Project/simple_shell_project/simple_shell_gui.py

Testing and Results:

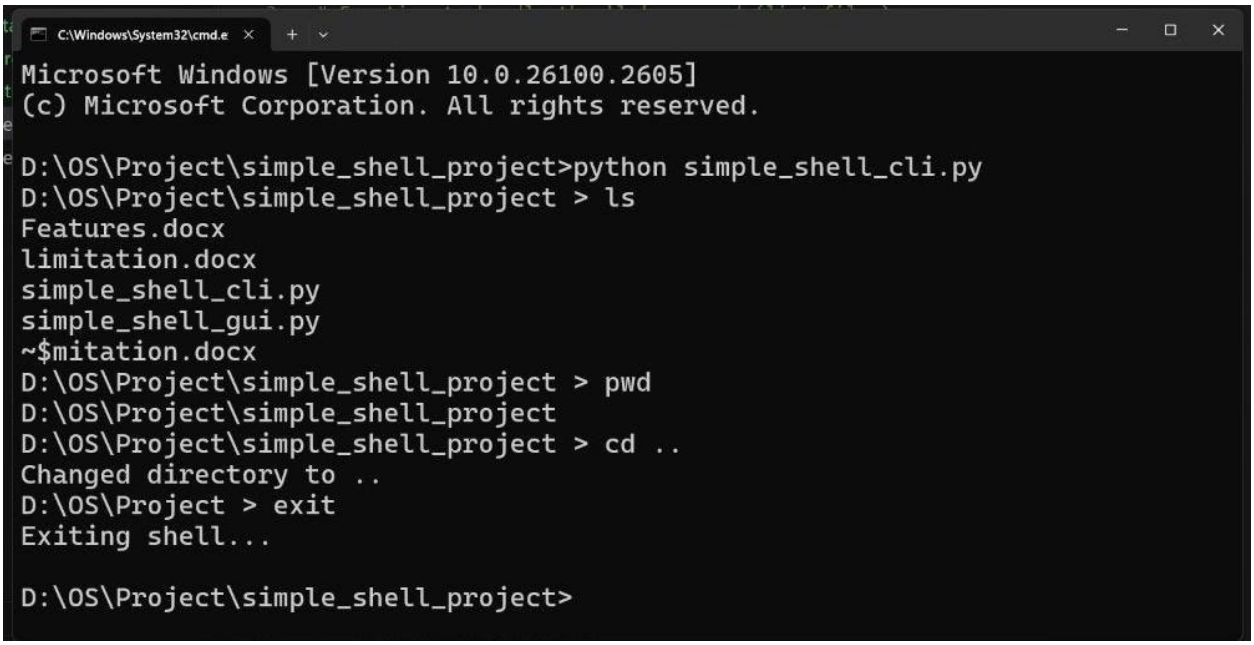


Fig 1.1 CLI Output

Fig 1.2 GUI Output

Fig 1.3 GUI Output

Fig 1.4 GUI Output

Conclusion:

This project successfully implemented a simple shell in Python with both CLI and GUI interfaces. The shell demonstrates basic file system navigation and command execution, serving as a foundation for more complex shell functionalities. Python's os and tkinter libraries proved to be effective tools for this purpose.

Future Work:

- Adding more commands, such as file manipulation (touch, rm, etc.).
- Improving the GUI for better usability and aesthetics.
- Implementing command history and auto-completion features.

References:

- Python os module documentation: *https://docs.python.org/3/library/os.html*
- Python tkinter module documentation: *https://docs.python.org/3/library/tkinter.html*