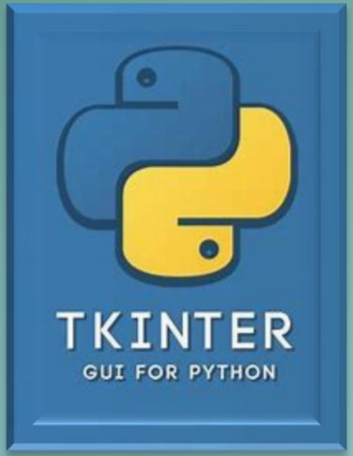
OBJECT ORIENTED PROGRAMMING

PROJECT BASED

2nd semester (Lect3)

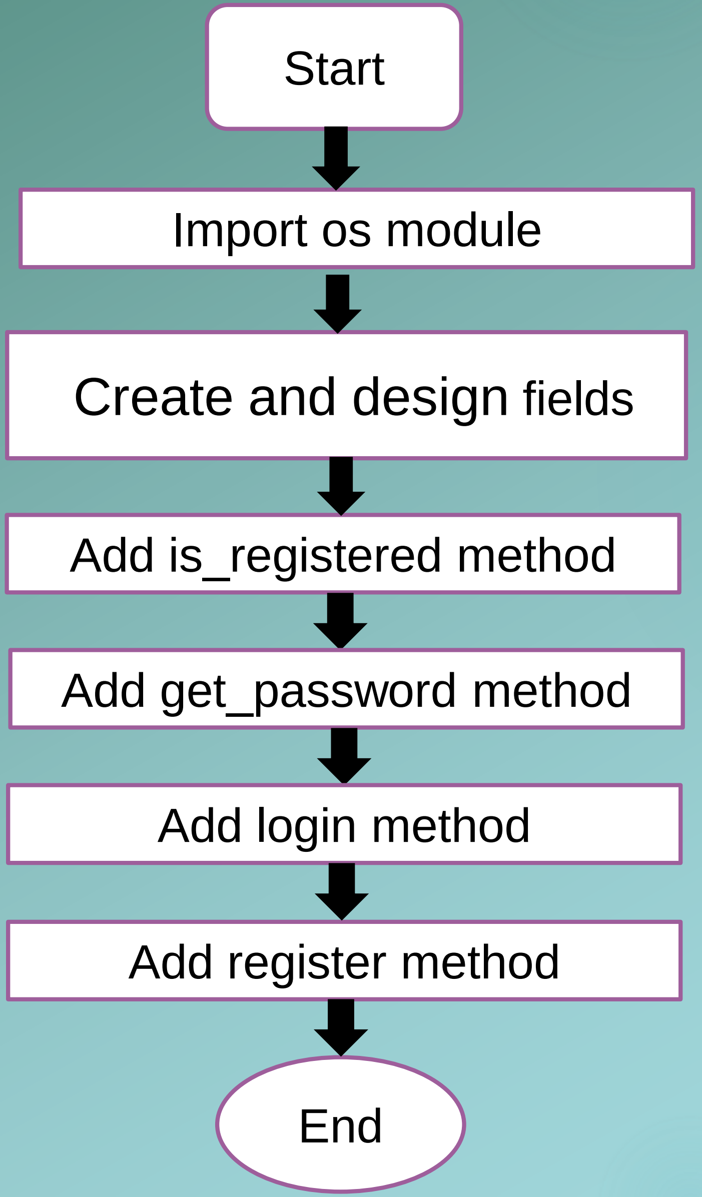
Creating Log in and Registration GUI window



Creating Log in and Registration GUI window

- **Q:** How can we create a GUI for player login and registration in the game? The window should include fields for entering a username and password, along with buttons for logging in and registering

Create GIU window



Creating Log in and Registration GUI window

1-Importing Modules

```
from tkinter import messagebox
import os
```

These lines import

- ❑ The `os` module for **file-related operations**, and
- ❑ The `messagebox` module from `tkinter` for **displaying message boxes**.
- ❑ We use `os` to check file existence and perform file operations,
- ❑ while `messagebox` allows us to provide interaction with user via message boxes.

Creating Log in and Registration GUI window

2- Create and Design Fields

```
self.label_username = tk.Label(self.window, text="Username:")
self.label_username.pack()  # Add username label to the window

self.label_password = tk.Label(self.window, text="Password:")
self.label_password.pack()  # Add password label to the window

# Create entry fields for username and password
self.entry_username = tk.Entry(self.window)
self.entry_username.pack()  # Add username entry field to the window

self.entry_password = tk.Entry(self.window, show="*")  # Show asterisks for password input
self.entry_password.pack()  # Add password entry field to the window

# Create buttons for login and register
self.button_login = tk.Button(self.window, text="Login", command=self.login)
self.button_login.pack()  # Add login button to the window

self.button_register = tk.Button(self.window, text="Register", command=self.register)
self.button_register.pack()  # Add register button to the window
```

Creating Log in and Registration GUI window

First, we need to integrate entry fields into the existing code. To accomplish this, we'll write the following lines:

```
# Create entry fields for username and password
        self.entry_username = tk.Entry(self.window)
        self.entry_username.pack()  # Add username entry field to the window

        self.entry_password = tk.Entry(self.window, show="*")  # Show asterisks for password input
        self.entry_password.pack()  # Add password entry field to the window
```

These lines introduce two entry fields:

- ❑ One for the username, and,
- ❑ Another for the password. For added security, the characters in the password field are hidden, denoted by the show="*" parameter.
- ❑ The show="*" parameter ensures that the characters entered in the password field are displayed as asterisks, increasing security.

Creating Log in and Registration GUI window

```
# Create buttons for login and register
self.button_login = tk.Button(self.window, text="Login", command=self.login)
self.button_login.pack()  # Add login button to the window

self.button_register = tk.Button(self.window, text="Register", command=self.register)
self.button_register.pack()  # Add register button to the window
```

- ❑ Create buttons for login and register, specifies that the button should be placed inside the **self.window** window, displays the text "Login“ and ‘Register” on the buttons, and associates the **self.login** and **self.register** methods with the button's action. This means that when the buttons are clicked, the **self.login** and **self.register** methods will be executed.

Creating Log in and Registration GUI window

3- Add is-registered method

```
def is_registered(self, username):
    # Check if username exists in the saved file
    with open("user_info.txt", "r") as file:
        for line in file:
            if f"Username: {username}" in line:
                return True
    return False
```

- ❑ This method checks if a username exists in the saved file "user_info.txt".
- ❑ We need this method to verify if a username is already registered before attempting to log in with it.
- ❑ It reads the file line by line and searches for the specified username.

Creating Log in and Registration GUI window

4- Add get_password method

```
def get_password(self, username):
    # Retrieve the password associated with the username
    with open("user_info.txt", "r") as file:
        for line in file:
            if line.startswith(f"Username: {username}"):
                return line.split("Password: ")[1].strip()
    return None
```

- This method retrieves the password associated with a given username from the saved file "user_info.txt".
- We need this method to fetch the correct password associated with a username during the login process.
- It parses the file content to extract the password corresponding to the username.

Creating Log in and Registration GUI window

- **Note :**

```
return line.split("Password: ")[1].strip()
```

- **line**: This variable represents a line of text from a file. It likely contains user information in the format "Username: [username], Password: [password]".
- **split("Password: "):** This method splits the line into two parts based on the string "Password: ". After splitting, we have a list containing two elements. The first element is everything before "Password: ", and the second element is everything after "Password: ".
- **[1]:** This index accesses the second element of the split result, which corresponds to the password part of the line.
- **strip():** This method removes any leading or trailing whitespace characters from the password string. It ensures that there are no extra spaces or newline characters in the password.

Creating Log in and Registration GUI window

5- Add login method

```
def login(self):
    # Retrieve username and password input from entry fields
    username = self.entry_username.get()
    password = self.entry_password.get()

    # Check if user_info.txt exists before attempting to read from it
    if os.path.exists("user_info.txt"):
        # Check if username exists in the saved file
        if self.is_registered(username):
            # Retrieve the password associated with the username
            correct_password = self.get_password(username)
            if correct_password == password:
                messagebox.showinfo("Login", "Login successful")  # Show login success message box
            else:
                messagebox.showerror("Error", "Incorrect password")  # Show incorrect password error message box
        else:
            messagebox.showerror("Error", "Username is not registered")  # Show username not registered error message
box
    else:
        messagebox.showerror("Error", "File 'user_info.txt' does not exist")  # Show file not found error message box
```

Creating Log in and Registration GUI window

- ❑ The login method now includes logic to check the validity of the entered username and password.
- ❑ We use **os.path.exists()** to verify if the file "user_info.txt" exists. If the file exists, we proceed to validate the entered credentials by comparing them with the stored data.
- ❑ We use message boxes to provide informative feedback to the user about the login process.

Creating Log in and Registration GUI window

6- Add Register method

```
def register(self):
    # Retrieve username and password input from entry fields
    username = self.entry_username.get()
    password = self.entry_password.get()

    # Implement registration functionality (store username and password to a file)
    # Placeholder code:
    with open("user_info.txt", "a") as file:
        file.write(f"Username: {username}, Password: {password}\n")
    messagebox.showinfo("Registration", "Registration successful")  # Show registration success msg box
```

- ❑ The register method is responsible for storing newly registered usernames and passwords in the file "user_info.txt".
- ❑ We use file I/O operations to append the new user information to the file.
- ❑ After successful registration, we inform the user via a message box.