	Create a Python script that: • Connects to a blockchain network • Loads a wallet using a private key • Fetches wallet address • Checks wallet balance • Demonstrates transaction preparation (without real funds)	

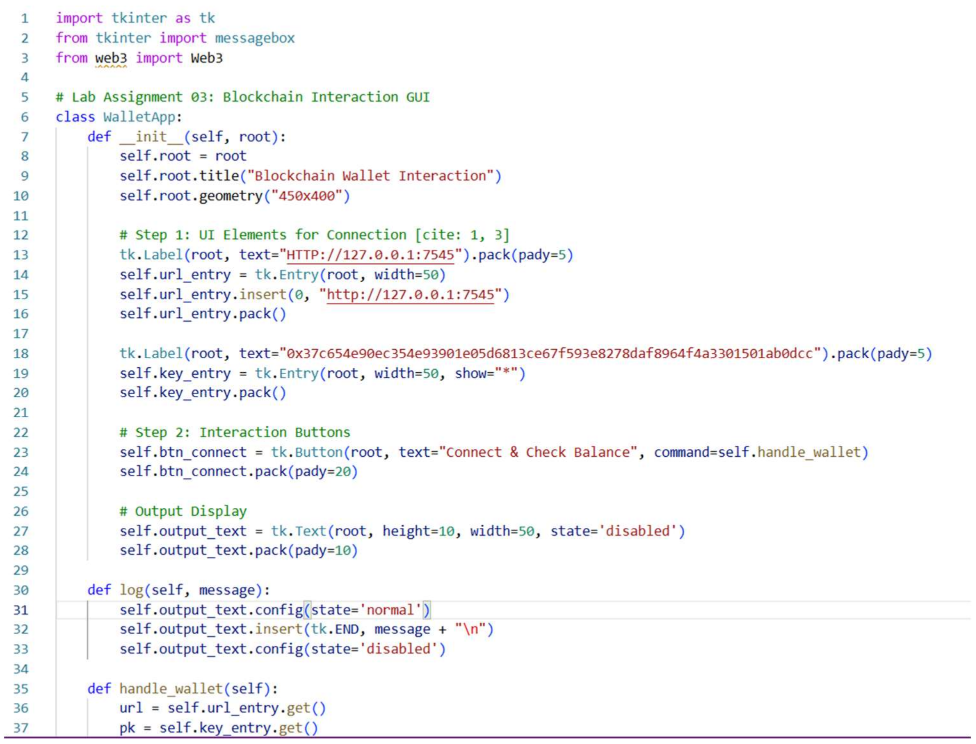
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**	
Program Name: B. Tech		**Assignment Type:** Lab	**Academic Year:** 2025-26
Course Coordinator Name		Dr. Jagjit Singh Dhatterwal	
Instructor(s) Name		Dr. Jagjit Singh Dhatterwal	
Course Code	23CS201PE401	**Course Title**	Blockchain Engineering
Year/Sem	III/II	**Regulation**	R25
Date and Day of Assignment	06-01-2026	**Time(s)**	9:00AM to 11:00AM
Duration	2 Hours	**Applicable to Batches**	(23CSBTB19, 23CSBTB20 23CSBTB21, 23CSBTB22 23CSBTB23, 23CSBTB24 23CSBTB25, 23CSBTB26)
Assignment Number: 03/12			

Q. No.	**Question**	**Expected Time to complete**
	Objective: To study blockchain interaction using Python and Web3 through wallet balance checking and transaction simulation.	
1	Requirements: • Install Python 3.x • Set up VS Code with Python extension • Install required Python libraries: • `pip install web3` • Use a test blockchain network (Ethereum Sepolia / Ganache local blockchain) • Basic understanding of blockchain wallets and private keys	

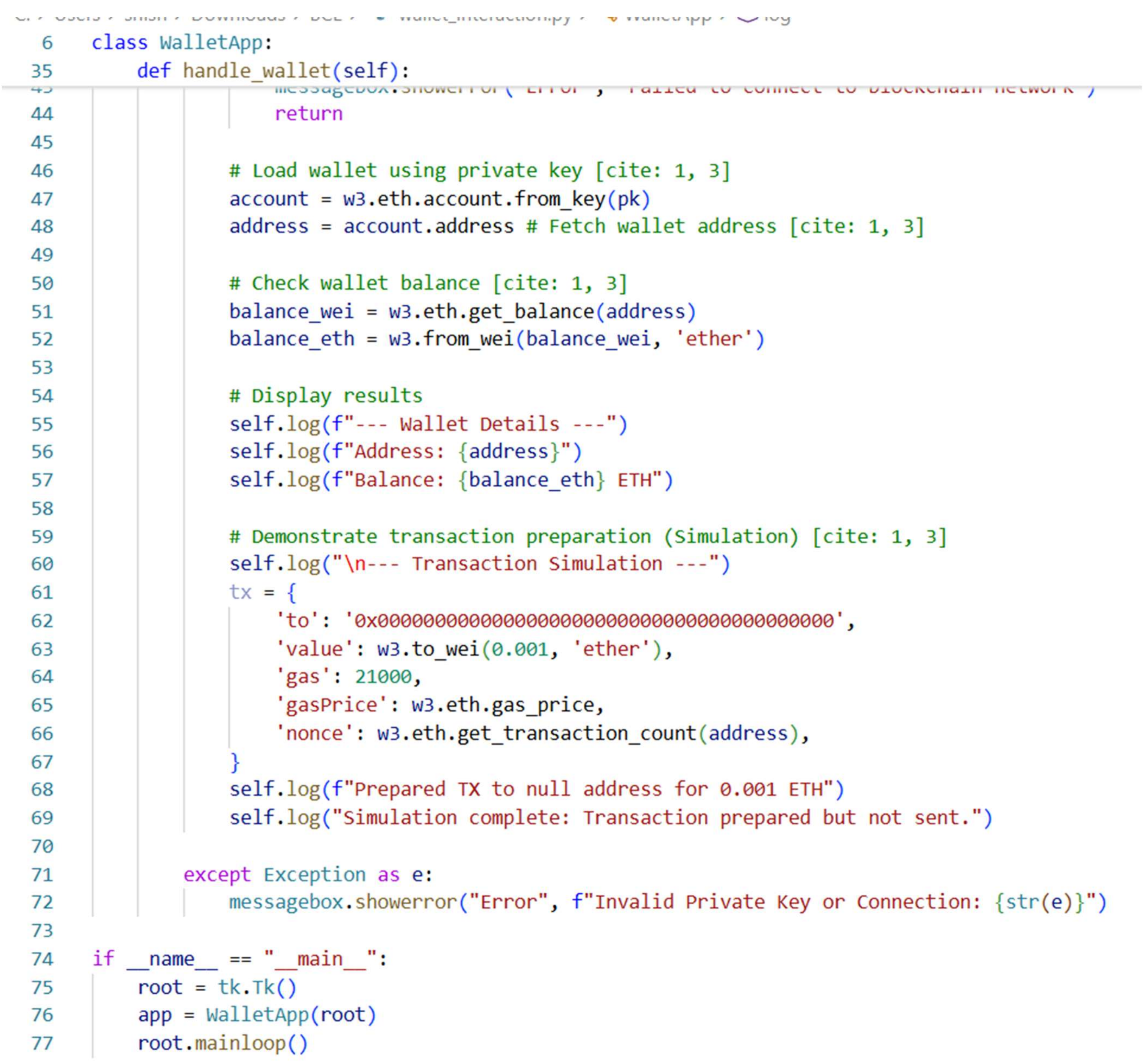
	Practical Description: Step 1: Environment Setup • Install Python and VS Code • Install Web3.py library	

	• Create a Python file named wallet_interaction.py	
	Step 2: Wallet and Blockchain Interaction Script Create a Python script that: • Connects to a blockchain network • Loads a wallet using a private key • Fetches wallet address • Checks wallet balance • Demonstrates transaction preparation (without real funds)	

Code Input:

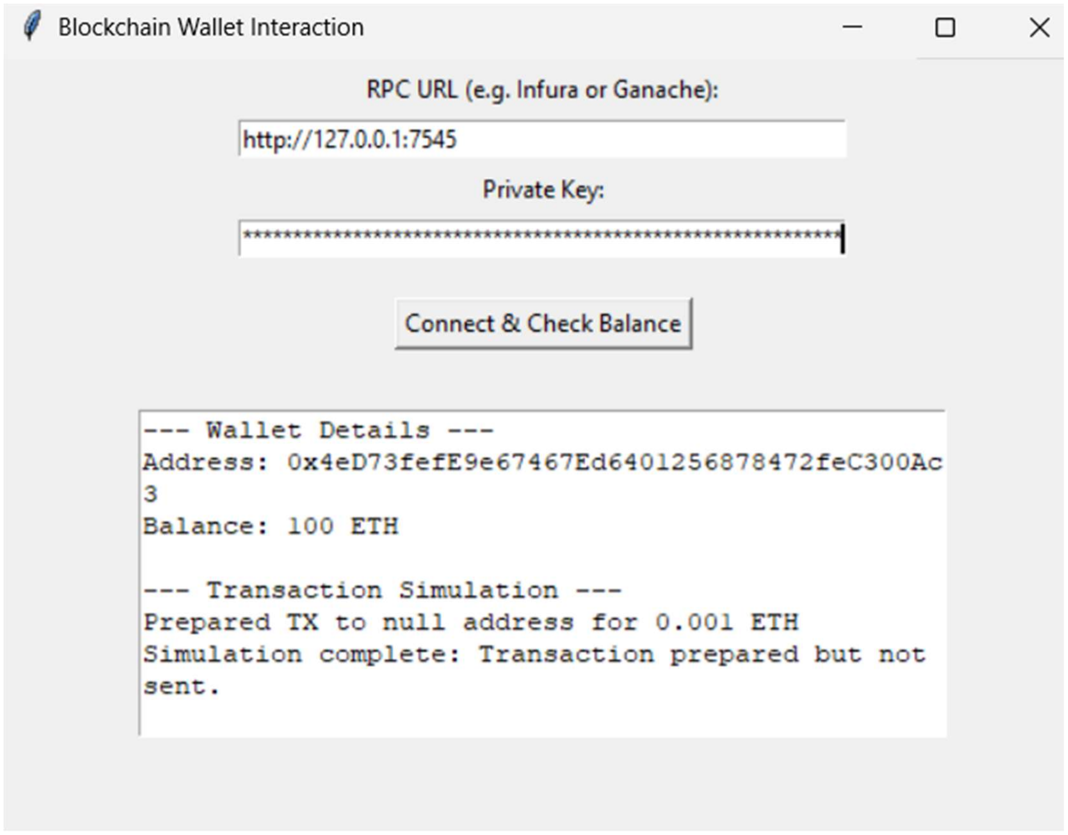


```python
import tkinter as tk
from tkinter import messagebox
from web3 import Web3

# Lab Assignment 03: Blockchain Interaction GUI
class WalletApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Blockchain Wallet Interaction")
        self.root.geometry("450x400")

        # Step 1: UI Elements for Connection [cite: 1, 3]
        tk.Label(root, text="HTTP://127.0.0.1:7545").pack(pady=5)
        self.url_entry = tk.Entry(root, width=50)
        self.url_entry.insert(0, "http://127.0.0.1:7545")
        self.url_entry.pack()

        tk.Label(root, text="0x37c654e90ec354e93901e05d6813ce67f593e8278daf8964f4a3301501ab0dcc").pack(pady=5)
        self.key_entry = tk.Entry(root, width=50, show="*")
        self.key_entry.pack()

        # Step 2: Interaction Buttons
        self.btn_connect = tk.Button(root, text="Connect & Check Balance", command=self.handle_wallet)
        self.btn_connect.pack(pady=20)

        # Output Display
        self.output_text = tk.Text(root, height=10, width=50, state='disabled')
        self.output_text.pack(pady=10)

    def log(self, message):
        self.output_text.config(state='normal')
        self.output_text.insert(tk.END, message + "\n")
        self.output_text.config(state='disabled')

    def handle_wallet(self):
        url = self.url_entry.get()
        pk = self.key_entry.get()
```



```python
class WalletApp:
    def handle_wallet(self):
                messagebox.showerror("Error", "Failed to connect to blockchain network")
                return

            # Load wallet using private key [cite: 1, 3]
            account = w3.eth.account.from_key(pk)
            address = account.address # Fetch wallet address [cite: 1, 3]

            # Check wallet balance [cite: 1, 3]
            balance_wei = w3.eth.get_balance(address)
            balance_eth = w3.from_wei(balance_wei, 'ether')

            # Display results
            self.log(f"--- Wallet Details ---")
            self.log(f"Address: {address}")
            self.log(f"Balance: {balance_eth} ETH")

            # Demonstrate transaction preparation (Simulation) [cite: 1, 3]
            self.log("\n--- Transaction Simulation ---")
            tx = {
                'to': '0x0000000000000000000000000000000000000000',
                'value': w3.to_wei(0.001, 'ether'),
                'gas': 21000,
                'gasPrice': w3.eth.gas_price,
                'nonce': w3.eth.get_transaction_count(address),
            }
            self.log(f"Prepared TX to null address for 0.001 ETH")
            self.log("Simulation complete: Transaction prepared but not sent.")

        except Exception as e:
            messagebox.showerror("Error", f"Invalid Private Key or Connection: {str(e)}")

if __name__ == "__main__":
    root = tk.Tk()
    app = WalletApp(root)
    root.mainloop()
```

CODE OUTPUT:



```
--- Wallet Details ---
Address: 0x4eD73fefE9e67467Ed6401256878472feC300Ac
3
Balance: 100 ETH

--- Transaction Simulation ---
Prepared TX to null address for 0.001 ETH
Simulation complete: Transaction prepared but not
sent.
```