The code creates a basic GUI wallet application using PyQt5, which interacts with the Sepolia test network through Web3.py. At the beginning, the program imports necessary modules including sys, Web3 for blockchain interaction, and PyQt5 widgets to build the user interface.

Inside the SepoliaWalletApp class, the constructor initializes the user interface and sets up a connection to the Sepolia network using an HTTP provider URL. You'll notice the placeholder 'YOUR_INFURA_PROJECT_ID', which needs to be replaced with an actual Infura project ID to connect successfully to the network.

The initUI method sets up the layout of the interface. It displays labels to show the wallet address and balance. It includes a line edit for entering a private key, which is essential for accessing the wallet. There's a connect button that triggers the wallet connection process. Additionally, there are fields to enter a recipient address and an amount of ETH to send, along with a send button to initiate the transaction.

The connect_wallet method reads the private key entered by the user. It checks if the key is provided and then uses Web3 to create an account object from the private key. After connecting, it fetches the wallet address and the current balance of ETH from the Sepolia test network. This data is displayed in the GUI labels for the user to see.

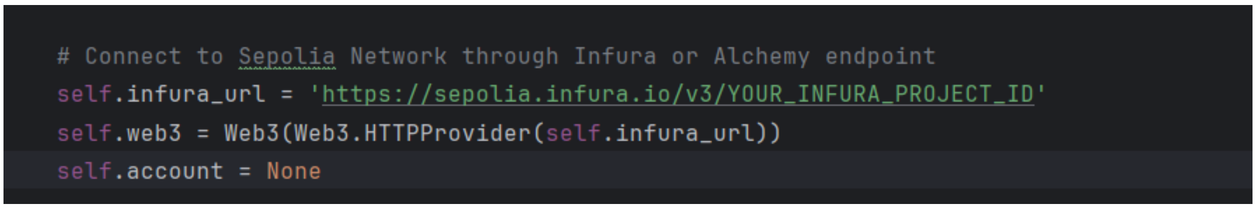
The send_transaction method executes after the user clicks the send button. It verifies that a wallet has been connected, and that both recipient and amount fields are filled. It prepares a transaction dictionary, specifying the nonce, recipient address, amount to send (converted from ETH to Wei), gas limit, and gas price. It then signs the transaction with the private key and sends the raw transaction to the Sepolia network. If successful, it shows a message with the transaction hash so the user can track it on a blockchain explorer.

Finally, at the end of the script, it initializes the PyQt application and runs the main loop to display the wallet GUI.

Connecting to the Ethereum Network (Senpolia)

```
self.infura_url = 'https://sepolia.infura.io/v3/YOUR_INFURA_PROJECT_ID'

self.web3 = Web3(Web3.HTTPProvider(self.infura_url))
```



```
# Connect to Sepolia Network through Infura or Alchemy endpoint
self.infura_url = 'https://sepolia.infura.io/v3/YOUR_INFURA_PROJECT_ID'
self.web3 = Web3(Web3.HTTPProvider(self.infura_url))
self.account = None
```

The program connects to the Sepolia network via Inf ua.

YOUR_INFRA_PROJECT_ID needs to be replaced with your API key from Infra.

Web3.HTTP Provider(self.injury_url) creates a connection to the blockchain.

Connecting a wallet

self.account = self.web3.eth.account.from_key(private_key)

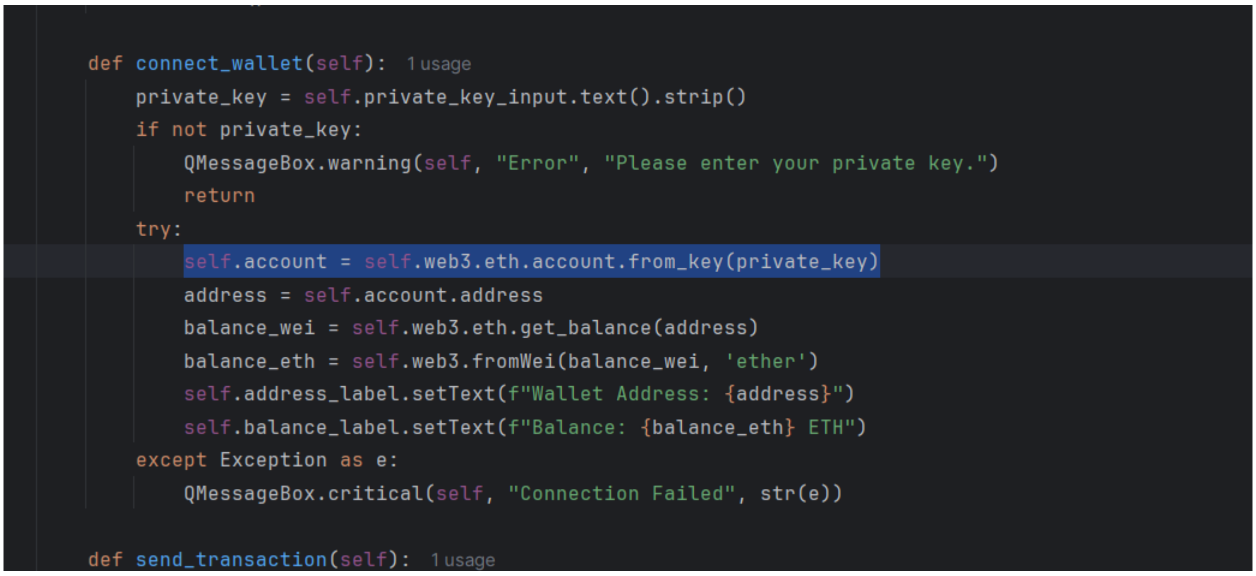
address = self.account.address

balance_wei = self.web3.eth.get_balance(address)

balance_eth = self.web3.fromWei(balance_wei, 'ether')

self.address_label.setText(f"Wallet Address: {address}")

self.balance_label.setText(f"Balance: {balance_eth} ETH")



```
def connect_wallet(self):  1 usage
    private_key = self.private_key_input.text().strip()
    if not private_key:
        QMessageBox.warning(self, "Error", "Please enter your private key.")
        return
    try:
        self.account = self.web3.eth.account.from_key(private_key)
        address = self.account.address
        balance_wei = self.web3.eth.get_balance(address)
        balance_eth = self.web3.fromWei(balance_wei, 'ether')
        self.address_label.setText(f"Wallet Address: {address}")
        self.balance_label.setText(f"Balance: {balance_eth} ETH")
    except Exception as e:
        QMessageBox.critical(self, "Connection Failed", str(e))

def send_transaction(self):  1 usage
```

1.We read the user's private key and create a wallet.

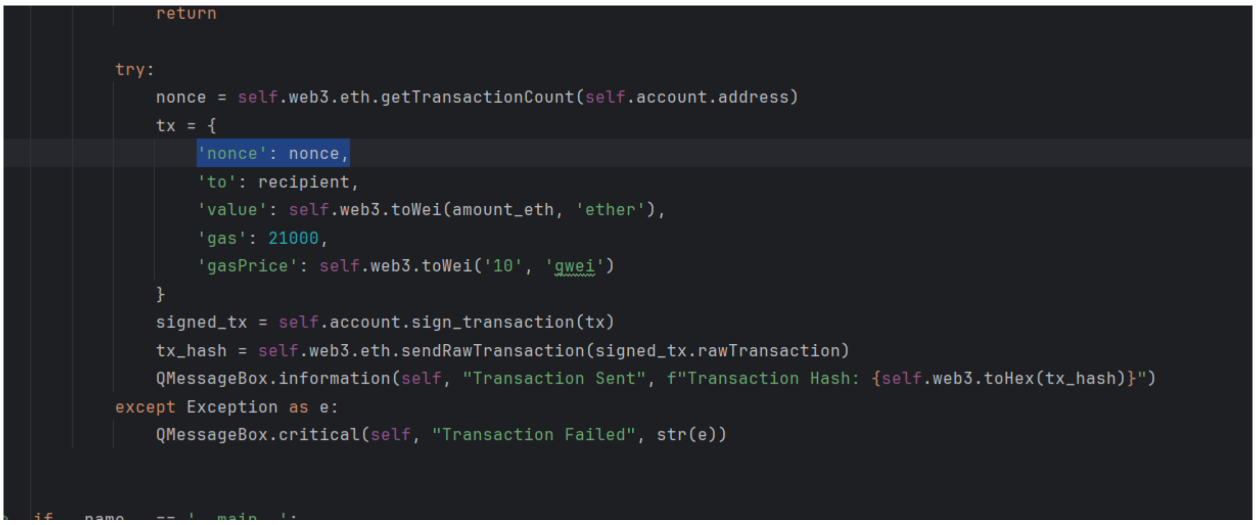
2.We get the balance in wei (the smallest unit of ETH).

3.Converting the balance from wei to ether.

4.We display the wallet address and balance in the interface.

Creating and sending a transaction

```
tx = {
  'nonce': nonce,
  'to': recipient,
  'value': self.web3.toWei(amount_eth, 'ether'),
  'gas': 21000,
  'gasPrice': self.web3.toWei('10', 'gwei')
}
signed_tx = self.account.sign_transaction(tx)
tx_hash = self.web3.eth.sendRawTransaction(signed_tx.rawTransaction)
```



```
            return

        try:
            nonce = self.web3.eth.getTransactionCount(self.account.address)
            tx = {
                'nonce': nonce,
                'to': recipient,
                'value': self.web3.toWei(amount_eth, 'ether'),
                'gas': 21000,
                'gasPrice': self.web3.toWei('10', 'gwei')
            }
            signed_tx = self.account.sign_transaction(tx)
            tx_hash = self.web3.eth.sendRawTransaction(signed_tx.rawTransaction)
            QMessageBox.information(self, "Transaction Sent", f"Transaction Hash: {self.web3.toHex(tx_hash)}")
        except Exception as e:
            QMessageBox.critical(self, "Transaction Failed", str(e))


if __name__ == '__main__':
```

Generating a nonce is the transaction number for the wallet.

We are forming a transaction with the fields:

to — recipient's address.

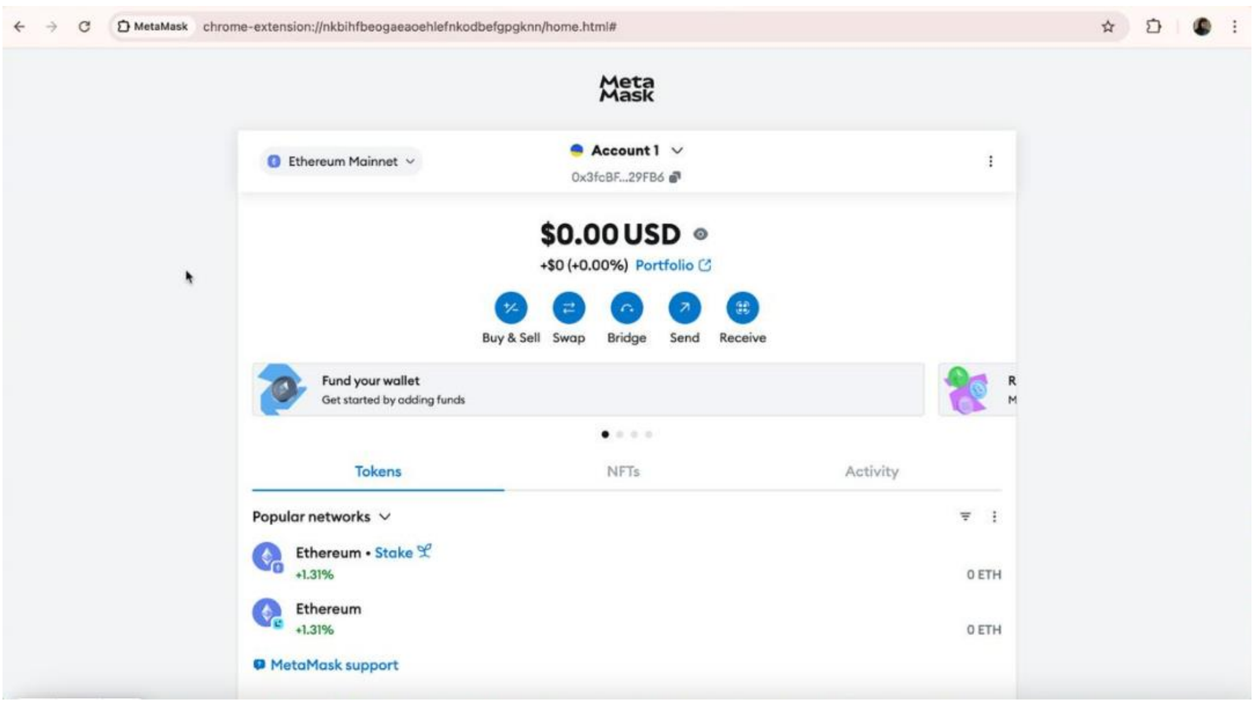
value — the amount of ETH (transferred from ether to wei).

gas is a fixed value of 21000 (the standard for regular ETH transfers).

gasPrice — the price for gas (10 Gwei).

We sign the transaction with a private key.

We send it to the network.



chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#

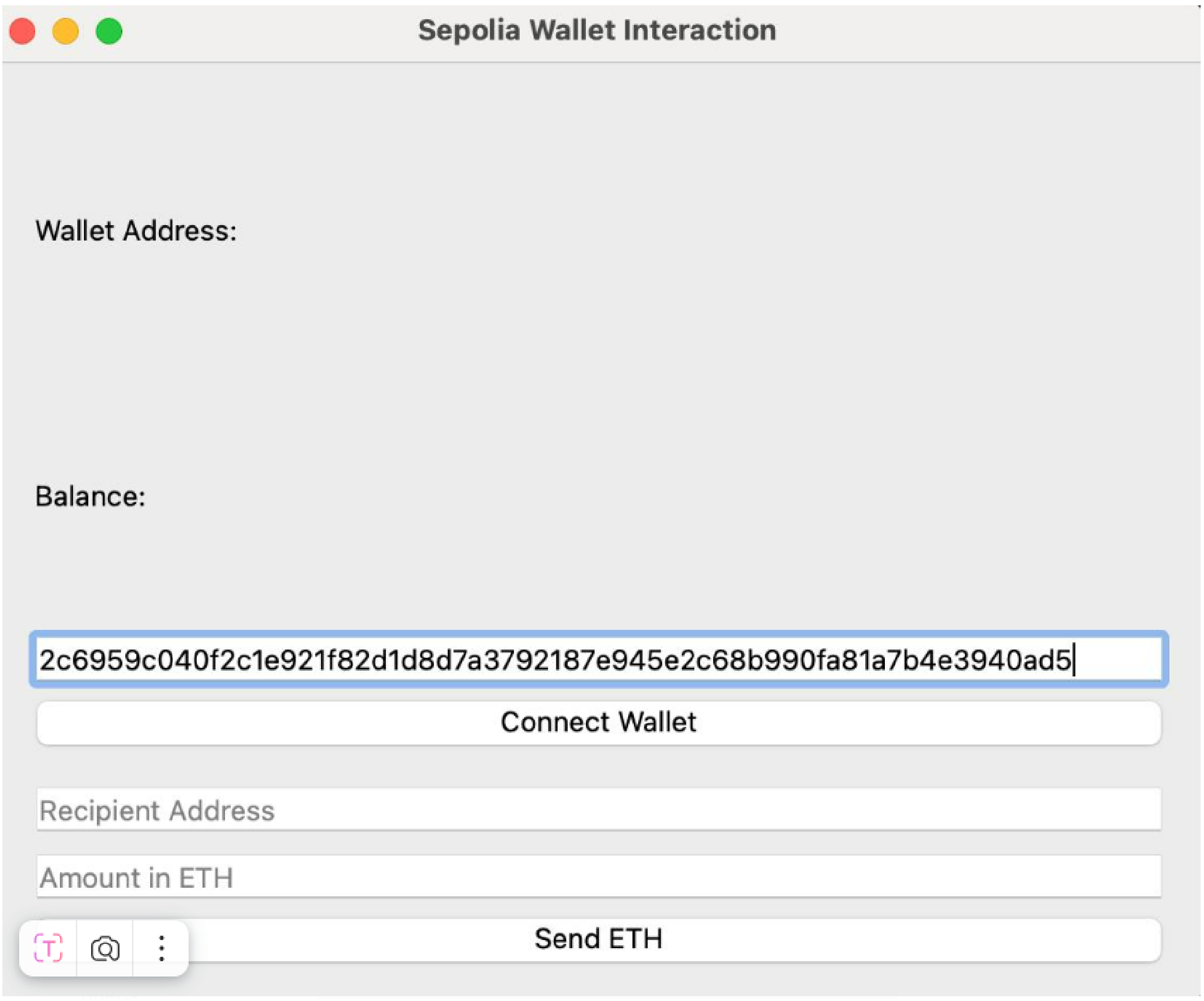
Account 1
0x3fcBF...29FB6

$0.00 USD
+$0 (+0.00%) Portfolio

Fund your wallet
Get started by adding funds

Ethereum • Stake
+1.31%
0 ETH

Ethereum
+1.31%
0 ETH



Sepolia Wallet Interaction

Wallet Address:

Balance:

2c6959c040f2c1e921f82d1d8d7a3792187e945e2c68b990fa81a7b4e3940ad5

Recipient Address

Amount in ETH