Exercise 1: Ethereum Wallet Management

1. Install the web3.py Python library for interacting with Ethereum

```
!pip install web3
```

```
  Downloading eth_keyfile-0.8.1-py3-none-any.whl.metadata (8.5 kB)
Collecting eth-keys>=0.4.0 (from eth-account>=0.13.1->web3)
  Downloading eth_keys-0.6.1-py3-none-any.whl.metadata (13 kB)
Collecting eth-rlp>=2.1.0 (from eth-account>=0.13.1->web3)
  Downloading eth_rlp-2.2.0-py3-none-any.whl.metadata (3.3 kB)
Collecting rlp>=1.0.0 (from eth-account>=0.13.1->web3)
  Downloading rlp-4.1.0-py3-none-any.whl.metadata (3.2 kB)
Collecting ckzg>=2.0.0 (from eth-account>=0.13.1->web3)
  Downloading ckzg-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (865 bytes)
Collecting pycryptodome<4,>=3.6.6 (from eth-hash[pycryptodome]>=0.5.1->web3)
  Downloading pycryptodome-3.22.0-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.4 kB)
Collecting cytoolz>=0.10.1 (from eth-utils>=5.0.0->web3)
  Downloading cytoolz-1.0.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.6 kB)
Downloading rlp-4.1.0-py3-none-any.whl (19 kB)
Installing collected packages: ckzg, bitarray, websockets, types-requests, pyunormalize, pycryptodome, parsimonious, hexbytes, eth-typing, eth-hash, cytoolz, eth-utils, rlp,
  Attempting uninstall: websockets
    Found existing installation: websockets 14.2
    Uninstalling websockets-14.2:
      Successfully uninstalled websockets-14.2
Successfully installed bitarray-3.2.0 ckzg-2.1.0 cytoolz-1.0.1 eth-abi-5.2.0 eth-account-0.13.5 eth-hash-0.7.1 eth-keyfile-0.8.1 eth-keys-0.6.1 eth-rlp-2.2.0 eth-typing-5.2.
```

2. Create a new Ethereum wallet using web3.py.

```python
from web3 import Web3
from eth_account import Account

# Tạo ví Ethereum mới
account = Account.create()
private_key = account.key.hex()
print(f"Địa chỉ ví: {account.address}")
```

```
Địa chỉ ví: 0xa0e0B4fa1740E8F059196e046478e1be5700F95B
```

3. Retrieve the wallet's address and balance using web3.py.

```python
infura_url = "https://mainnet.infura.io/v3/11ca5d6e091c455b8ba2607cdb007c23"
web3 = Web3(Web3.HTTPProvider(infura_url))

# Địa chỉ ví cần kiểm tra số dư
wallet_address = account.address
balance = web3.eth.get_balance(wallet_address)
print(f"Số dư: {web3.from_wei(balance, 'ether')} ETH")
```

```
Số dư: 0 ETH
```

4. Send Ether from one wallet to another using web3.py.

```python
sender_address = "0xa0e0B4fa1740E8F059196e046478e1be5700F95B"
receiver_address = "0xBF408BcAF0701D69874197E3288d83dC320D5967"

amount = web3.to_wei(0.01, 'ether')

balance = web3.eth.get_balance(sender_address)
print(f"Số dư hiện tại: {web3.from_wei(balance, 'ether')} ETH")

amount = web3.to_wei(0.01, 'ether')
nonce = web3.eth.get_transaction_count(sender_address)

# Kiểm tra nếu số dư không đủ
amount = web3.to_wei(0.01, 'ether')
if balance < amount:
    print("Lỗi: Số dư không đủ để thực hiện giao dịch!")
else:
    try:
        # Lấy nonce của ví gửi
        nonce = web3.eth.get_transaction_count(sender_address)

        # Tạo giao dịch
        tx = {
            'nonce': nonce,
            'to': receiver_address,
            'value': amount,
            'gas': 21000,
            'gasPrice': web3.to_wei('50', 'gwei'),
            'chainId': 5
        }

        # Gửi giao dịch
        signed_tx = web3.eth.account.sign_transaction(tx, private_key)
        tx_hash = web3.eth.send_raw_transaction(signed_tx.rawTransaction)
        print(f"Giao dịch đã gửi, hash: {tx_hash.hex()}")

    except Exception as e:
        print(f"Lỗi giao dịch: {e}")
```

5. Check the transaction status and balance updates after the transaction.

```
tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)

if tx_receipt.status == 1:
    print("✅ Giao dịch thành công!")
else:
    print("❌ Giao dịch thất bại!")

# Kiểm tra số dư sau giao dịch
balance_after = web3.eth.get_balance(sender_address)
print(f"Số dư sau giao dịch: {web3.from_wei(balance_after, 'ether')} ETH")
```

6. Explore additional features of web3.py, such as querying transaction details and interacting with smart contracts.

```
contract_address = "0x1234567890abcdef1234567890abcdef12345678"
abi = [{"constant": True, "inputs": [], "name": "getValue", "outputs": [{"name": "", "type": "uint256"}], "payable": False, "stateMutability": "view", "type": "function"}]

try:
    balance = web3.eth.get_balance(sender_address)
    print(f"Số dư hiện tại: {web3.from_wei(balance, 'ether')} ETH")

    tx = {'nonce': web3.eth.get_transaction_count(sender_address), 'to': receiver_address, 'value': web3.to_wei(0.01, 'ether'), 'gas': 21000, 'gasPrice': web3.to_wei('50', 'gwei')
    signed_tx = web3.eth.account.sign_transaction(tx, private_key)
    tx_hash = web3.eth.send_raw_transaction(signed_tx.rawTransaction)
    print(f"Giao dịch đã gửi, hash: {tx_hash.hex()}")
    tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
    print("✅ Giao dịch thành công!" if tx_receipt.status == 1 else "❌ Giao dịch thất bại!")

    contract = web3.eth.contract(address=contract_address, abi=abi)
    print(f"Giá trị từ contract: {contract.functions.getValue().call()}")

    tx = contract.functions.setValue(42).build_transaction({'from': sender_address, 'nonce': web3.eth.get_transaction_count(sender_address), 'gas': 100000, 'gasPrice': web3.to_wei
    signed_tx = web3.eth.account.sign_transaction(tx, private_key)
    tx_hash = web3.eth.send_raw_transaction(signed_tx.rawTransaction)
    tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
    print("✅ Cập nhật dữ liệu thành công!" if tx_receipt.status == 1 else "❌ Thất bại!")

except Exception as e:
    print(f"❌ Lỗi: {e}")
```

Exercise 2: Smart Contract Deployment

1. Write a Solidity smart contract defining basic functionality (e.g., a token contract, a simple voting contract).

```
// SimpleStorage.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleStorage {
    uint256 private storedValue;

    event ValueUpdated(uint256 newValue);

    function setValue(uint256 _value) public {
        storedValue = _value;
        emit ValueUpdated(_value);
    }

    function getValue() public view returns (uint256) {
        return storedValue;
    }
}
```

2. Compile the smart contract using solc, the Solidity compiler

```
!pip install py-solc-x
```

```
Collecting py-solc-x
  Downloading py_solc_x-2.0.3-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: requests<3,>=2.19.0 in /usr/local/lib/python3.11/dist-packages (from py-solc-x) (2.32.3)
Collecting packaging<24,>=23.1 (from py-solc-x)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.19.0->py-solc-x) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.19.0->py-solc-x) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.19.0->py-solc-x) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.19.0->py-solc-x) (2025.1.31)
Downloading py_solc_x-2.0.3-py3-none-any.whl (18 kB)
Downloading packaging-23.2-py3-none-any.whl (53 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.0/53.0 kB 2.0 MB/s eta 0:00:00
Installing collected packages: packaging, py-solc-x
  Attempting uninstall: packaging
    Found existing installation: packaging 24.2
    Uninstalling packaging-24.2:
      Successfully uninstalled packaging-24.2
Successfully installed packaging-23.2 py-solc-x-2.0.3
```

```python
from solcx import compile_source, install_solc, set_solc_version
install_solc("0.8.0")
set_solc_version("0.8.0")

contract_source = """
pragma solidity ^0.8.0;
contract SimpleStorage {
    uint256 private storedValue;
    event ValueUpdated(uint256 newValue);
    function setValue(uint256 _value) public {
        storedValue = _value;
        emit ValueUpdated(_value);
    }
    function getValue() public view returns (uint256) {
        return storedValue;
    }
}
"""

compiled_sol = compile_source(contract_source)
contract_interface = compiled_sol["<stdin>:SimpleStorage"]

bytecode = contract_interface["bin"]
abi = contract_interface["abi"]

print("Bytecode:", bytecode)
print("ABI:", abi)
```

```
Bytecode: 608060405234801561001057600080fd5b5061017e806100206000396000f3fe608060405234801561001057600080fd5b506004361061003657600035 60e01c806320965255146100 3b57806355241077146
ABI: [{'anonymous': False, 'inputs': [{'indexed': False, 'internalType': 'uint256', 'name': 'newValue', 'type': 'uint256'}], 'name': 'ValueUpdated', 'type': 'event'}, {'inputs
```

3. Initialize a connection to an Ethereum node using web3.py

```
infura_url = "https://mainnet.infura.io/v3/11ca5d6e091c455b8ba2607cdb007c23"
web3 = Web3(Web3.HTTPProvider(infura_url))

if web3.is_connected():
    print("Kết nối Ethereum node thành công!")
    print(f"Block hiện tại: {web3.eth.block_number}")
else:
    print("Kết nối thất bại!")
```

```
Kết nối Ethereum node thành công!
Block hiện tại: 22109932
```

4. Deploy the compiled smart contract to the Ethereum blockchain using web3.py

```
wallet_address = account.address
nonce = web3.eth.get_transaction_count(wallet_address)

SimpleStorage = web3.eth.contract(abi=abi, bytecode=bytecode)
tx = SimpleStorage.constructor().build_transaction({'from': wallet_address, 'nonce': nonce, 'gas': 2000000, 'gasPrice': web3.to_wei('50', 'gwei')})
signed_tx = web3.eth.account.sign_transaction(tx, private_key)
tx_hash = web3.eth.send_raw_transaction(signed_tx.rawTransaction)
tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
print(f"Hợp đồng được triển khai tại: {tx_receipt.contractAddress}")
```

5. Verify the successful deployment by retrieving the contract's address and bytecode.

```
# Lấy địa chỉ hợp đồng đã triển khai
contract_address = tx_receipt.contractAddress
print(f"Hợp đồng được triển khai tại: {contract_address}")

# Lấy bytecode của hợp đồng từ blockchain
deployed_bytecode = web3.eth.get_code(contract_address)
print(f"Bytecode của hợp đồng đã triển khai: {deployed_bytecode.hex()}")
```

6. Interact with the deployed smart contract using Python to perform various transactions and calls.

```
contract = web3.eth.contract(address=contract_address, abi=abi)

# Gửi giao dịch để cập nhật giá trị trong hợp đồng (gọi hàm setValue)
tx = contract.functions.setValue(100).build_transaction({
    'from': wallet_address,
    'nonce': web3.eth.get_transaction_count(wallet_address),
    'gas': 2000000,
    'gasPrice': web3.to_wei('50', 'gwei')
})
signed_tx = web3.eth.account.sign_transaction(tx, private_key)
tx_hash = web3.eth.send_raw_transaction(signed_tx.rawTransaction)
tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
print(f"Giao dịch cập nhật giá trị thành công: {tx_receipt.transactionHash.hex()}")
```

```
stored_value = contract.functions.getValue().call()
print(f"Giá trị trong hợp đồng: {stored_value}")
```