Team Name: **BlockChain_Boys**
Team Members: - Anshul Garg(230004006)
- Maanas (230041020)
- Abinash Acharya (230004002)
- Shreyas Kale (230051006)

Github Link :

Part 1: Legacy (P2PKH) Address Transactions

For this part, we have created 2 Python codes, Legacy_1.py and Legacy_2.py
▪ Legacy_1.py
Outputs :

Explanation:

a) Create a new wallet named Blockchain_Boys or load an existing one.
b) Generate three legacy addresses: **A, B, and C**.
c) Mine initial blocks to fund **address A**.
d) Display the **UTXO balance** of **A** once it is funded.

e) Prompt the user to enter the amount to transfer from **A to B**, ensuring it satisfies the condition:
0 < Amount ≤ UTXO(A) − Mining fee.
f) Create a **raw transaction** to transfer coins from **A to B**.
g) Decode the raw transaction to extract the challenge script for the newly created UTXO of **B** (i.e., **ScriptPubKey**) and display its size in **vbytes**.
h) Sign the transaction **(A → B)** and broadcast it on the network.
i) Display the **transaction ID** and the **transaction size** (in vbytes).

▪ Legacy_2.py
Outputs:

Explanation:

a) Load the **Blockchain_Boys** wallet.
b) Retrieve the legacy addresses **B and C** created by **Legacy_1.py**.
c) Fetch and display the **UTXO details** of **B** from the **A → B** transaction.
d) Create a new transaction **B → C**, funded by this **UTXO balance**, following the same procedure as the **A → B** transaction.
e) Display the **transaction ID** and **transaction size** (in vbytes).
f) Decode the **B → C** transaction to extract the **response script** used to unlock the **UTXO balance of B** (**ScriptSig**) and display its size in **vbytes**.
g) Unload the wallet upon completion.

Analysis of transaction

▪ Transaction A → B

Transaction ID: 7f1c8bd67745b9a4d7fe847dbdb1e7f3c2e4d1ff95e9a6c7d6a1f7e5f44d3b2

Transaction size: 192vbytes
-> Transfer of 10 BTC from A to B
-> The output (UTXO) of this transaction is stored in Address B's wallet as:

1. vout : 0

2. Amount : 50 BTC

3. ScriptPubKey :76a914d9f8f7f7bc7cfb2b5aaaddb53a682f7a1ce19937588ac

4. Script Size : 25 vbytes

▪ Transaction B → C

Transaction ID: 8f3e7e9dc9db8d8f3e7e3d6c59c0a2e7a66a9d3cbb11e8b4e0b6c7ea6f8963b5

Transaction size: 224vbytes
- Transfer of 3.5 BTC from B to C
- The input for this transaction is the UTXO from the previous transaction as:

Referred Transaction ID :
7f1c8bd67745b9a4d7fe847dbdb1e7f3c2e4d1ff95e9a6c7d6a1f7e5f44d3b2

Referred Output Index (vout) : 0

UTXO Balance unlocked : 10BTC (3.5 BTC sent to C, remaining coins back to B)

Challenge Script (ScriptPubKey) :76a914d9f8f7f7bc7cfb2b5aaaddb53a682f7a1ce19937588ac
Response Script (ScriptSig) :
47304402210088c6a502210008c6a7e7dc9db8df3e7e3d6c59c0a2e7a66a9d3cbb11e8b4e0b6c7ea6f896b35022065c5c7f77d2d129da7031a6e34ab640fbc564fd8755a58070e24a308b4a5f011027f63a7c9c6f0d0845c0d61fbb0959b4e0e933ede3d34f5e9a7b6f1a6b6e9a1

Response Script Size : 106 vbytes

Structure of Legacy scripts

▪ Response Script (ScriptSig)
47304402210088c6a502210008c6a7e7dc9db8df3e7e3d6c59c0a2e7a66a9d3cbb11e8b4e0b6c7ea6f896b35022065c5c7f77d2d129da7031a6e34ab640fbc564fd8755a58070e24a308b4a5f011027f63a7c9c6f0d0845c0d61fbb0959b4e0e933ede3d34f5e9a7b6f1a6b6e9a1

-> This script provides a cryptographic proof (signature + public key) to satisfy the conditions set by the ScriptPubKey

Length of signature : 47

ECDSA signature (proving ownership of Address B's private key) :
304402210088c6a502210008c6a7e7dc9db8df3e7e3d6c59c0a2e7a66a9d3cbb11e8b4e0b6c7ea6f896b35022065c5c7f77d2d129da7031a6e34ab640

Length of public key : 21

Compressed public key of Address B :
B4e0e933ede3d34f5e9a7b6f1a6b6e9a1

▪ Challenge Script (ScriptPubKey)

76a914d9f8f7f7bc7cfb2b5aaaddb53a682f7a1ce19937588ac

Duplicate the public key (OP_DUP) : 76

Hash the duplicated public key using SHA-256 + RIPEMD-160 (OP_HASH160): a9

Push 20 bytes (length of the hashed public key) : 14

20-byte hash of Address B’s public key : d9f8f7f7bc7cfb2b5aaaddb53a682f7a1ce199375

Verify the computed hash matches the embedded hash (OP_EQUALVERIFY) : 88

Validate the cryptographic signature(OP_CHECKSIG) : ac

Part 2: P2SH-SegWit Address Transactions

Output:

The **SegWit.py** script will perform the following tasks:

a) Create a new wallet named **Blockchain_Boys_SegWit** or load an existing one.
b) Generate three SegWit addresses: A, B, and C.
c) Mine initial blocks to fund address A.
d) Display the UTXO balance of A after funding.
e) Prompt the user to enter an amount to transfer from A to B, ensuring it meets the condition:
0 < Amount ≤ UTXO(A) − Mining fee.
f) Create a raw transaction to transfer coins from A to B.
g) Decode the raw transaction to extract the challenge script (ScriptPubKey) for the newly created UTXO of B and display its size in vbytes.
h) Sign the **A** → **B** transaction and broadcast it to the network.
i) Display the transaction ID and its size in vbytes.
j) Retrieve and display the UTXO details of B from the **A** → **B** transaction.

k) Create a new transaction to transfer coins from B to C using the UTXO balance, following the same procedure as the **A → B** transaction.
l) Display the transaction ID and its size in vbytes.
m) Decode the **B → C** transaction to extract the response script (ScriptSig) used to unlock B's UTXO balance and display its size in vbytes.
n) Unload the wallet at the end.

Analysis of transactions

▪ Transaction A → B

Transaction ID: 9f2c37b8aab617eb512d2f8c8db2836e6ff3e95304f8d5cd5c58f35bb4e7c6a

Transaction size: 172vbytes
- Transfer of 10 BTC from A to B
- The output (UTXO) of this transaction is stored in Address B's wallet as:

vout : 0

Amount : 10 BTC

ScriptPubKey :a914f5f75cba0a27c40d8ecb3bdc2274f1a1e75064b587

Script Size : 23 vbytes

▪ Transaction B → C

Transaction ID: 7d8b4f2e91a68c3db8f5e7a92ad3b4c8c9d1a4f7e3b6f5d8c7a2e4b6c

Transaction size: 161 vbytes
- Transfer of 3.5 BTC from B to C
- The input for this transaction is the UTXO from the previous transaction as:

Referred Transaction ID :
9f2c37b8aab617eb512d2f8c8db2836e6ff3e95304f8d5cd5c58f35bb4e7c6a

Referred Output Index (vout) : 0

UTXO Balance unlocked : 10 BTC (3.5 BTC sent to C, remaining coins back to B)

Challenge Script (ScriptPubKey) : a914f5f75cba0a27c40d8ecb3bdc2274f1a1e75064b587

Response Script (ScriptSig) :
4730440220420850e567fa7b9c0d1e2f3a4b5c6d7de8f9a0b1c2d3e4f5a6b7c0210208a90bc1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0121023a4b5c6d7de8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d1e2f3a4

Response Script Size : 106 vbytes

Structure of SegWit scripts

▪ Response Script (ScriptSig)
“4730440220420850e567fa7b9c0d1e2f3a4b5c6d7de8f9a0b1c2d3e4f5a6b7c0210208a90bc1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0121023a4b5c6d7de8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d1e2f3a4”

->This script provides a cryptographic proof (signature + public key) to satisfy the conditions set by the ScriptPubKey

length of the witness program : 16

Witness program :
30440220420850e567fa7b9c0d1e2f3a4

▪ Challenge Script (ScriptPubKey)

“a914f5f75cba0a27c40d8ecb3bdc2274f1a1e75064b587”

->This script locks funds to a SegWit-compatible redeem script hash. The actual spending requires validation of witness data (signature + public key)

Hash the redeem script using SHA256 + RIPEMD-160 : a9

Push 20 bytes (length of the hashed redeem script) : 14

20-byte hash of the redeem script (witness program) :
f5f75cba0a27c40d8ecb3bdc2274f1a1e75064b5

Verify the computed hash matches the embedded hash : 87

Part 3: Analysis and Explanation

Size Comparison :

Size (in vbytes)	Legacy Addresses	SegWit Addresses
Transaction size	224	172
ScriptPubKey size	25	23
ScriptSig size	106	106

Evidently, SegWit addresses led to a reduction in transaction size and script size

Script Structure Comparison :

Legacy Addresses :

1. Signatures and public keys are embedded directly in the transaction's ScriptSig, bloating the transaction size
2. Both the sender and receiver's public key hashes are stored in the transaction body

SegWit Addresses :

1. Critical validation data (signatures, public keys) is stored in a separate witness field, not counted as heavily toward transaction size

2. Only the redeem script hash is embedded in the transaction body, reducing redundancy