IT Management & Audits

Practical Lab Manual

Crypto Wallet Explorer

Practical P13

Learning Domain

Cryptocurrency Wallet Security & Blockchain Exploration

Course Learning Outcomes

CLO13: Understand cryptocurrency wallet mechanics and blockchain data structures

Unit

Unit VIII: Blockchain & FinTech Innovation

Time Allocation: 3 hours

Learning Mode: Hands-on (80%) + Theory (20%)

Difficulty Level: Intermediate

Crypto Wallet Explorer

Practical P13

Quick Reference

Practical Code	P13
Practical Name	Crypto Wallet Explorer
Slot	T/P-13
Duration	3 hours
CLO Mapping	CLO13
Unit	Unit VIII: Blockchain & FinTech Innovation
Delivery Mode	Hands-on Lab
Target Audience	Intermediate Level
India Integration	MEDIUM
Screenshot Count	5 Required

CRITICAL SECURITY DISCLAIMER

NEVER use wallets generated in this lab for storing real cryptocurrency or real funds. All wallets created during this practical are for **educational and testnet purposes only**. Private keys and mnemonic phrases generated in a lab environment should be considered compromised.

Rules to follow:

- Use **testnet networks only** (Sepolia, Goerli) — never mainnet for transactions
- **Do not send real ETH or tokens** to any address generated in this lab
- **Do not reuse** lab-generated mnemonic phrases for personal wallets
- Treat all generated keys as **publicly exposed** material
- Delete all generated key material after completing the lab

Failure to follow these rules may result in **permanent loss of funds**.

Prerequisites

- Basic cryptography concepts (hashing, public/private keys, digital signatures)
- P12 (Blockchain Fundamentals) recommended as prior completion
- Understanding of public-key cryptography and key pairs
- Python programming fundamentals and CLI familiarity
- Internet connection for blockchain API queries

Tools Required

Tool	Version	Free Tier	Notes
Python	3.8+	✓	Core runtime
eth-account	Latest	✓	Ethereum account management
web3.py	Latest	✓	Ethereum interaction library
mnemonic	Latest	✓	BIP-39 word list generation
Rich	Latest	✓	Terminal formatting
Click	Latest	✓	CLI framework
pip	Latest	✓	Python package manager

Learning Objectives

- ✓ Generate hierarchical deterministic (HD) wallets using BIP-39 mnemonic phrases
- ✓ Understand key derivation paths (BIP-32/BIP-44) and Ethereum address generation
- ✓ Build and sign raw Ethereum transactions with proper field structure
- ✓ Explore on-chain transactions and decode their components
- ✓ Query blockchain data including balances, blocks, and token holdings
- ✓ Apply wallet security best practices and understand threat models
- ✓ Analyze India's cryptocurrency regulatory landscape (VDA taxation, RBI stance)

What You Will Learn

By the end of this practical, you will:

1. Understand the cryptographic foundations underlying cryptocurrency wallets
2. Generate HD wallets from mnemonic phrases and derive child keys
3. Visualize the key derivation path from mnemonic to Ethereum address
4. Construct, sign, and analyze raw Ethereum transactions
5. Query the Ethereum blockchain for transaction details, balances, and block data
6. Evaluate wallet security practices and identify common vulnerabilities
7. Understand India's VDA tax framework and regulatory environment

Real-World Application

Cryptocurrency wallets are the gateway to blockchain-based financial systems. Indian exchanges such as **WazirX**, **CoinDCX**, and **ZebPay** manage millions of user wallets, each requiring robust key management, transaction signing, and regulatory compliance. With India's 30% VDA tax and 1% TDS framework, auditors must understand how transactions flow from wallet creation through on-chain settlement to tax reporting.

Hands-On Procedure

Part A: Environment Setup

Step 1: Clone Repository and Explore Project Structure

Objective: Set up the Crypto Wallet Explorer project and understand its architecture.

1. Open your terminal and navigate to your working directory
2. Clone the crypto-wallet-explorer repository
3. Explore the project structure to understand the codebase layout

```
# Clone the repository
git clone https://github.com/itma-practicals/crypto-wallet-explorer.git
cd crypto-wallet-explorer

# View the project structure
tree src/
# src/
# |-- wallet_generator.py      # HD wallet generation (BIP-39/32)
# |-- key_visualizer.py        # Key derivation path visualization
# |-- transaction_builder.py   # Raw TX construction & signing
# |-- transaction_explorer.py  # On-chain transaction lookup
# |-- block_explorer.py        # Block data exploration
# |-- balance_checker.py       # ETH & ERC-20 balance queries
# |-- cli.py                   # Click-based CLI entry point
# |-- utils.py                 # Shared utilities & formatting

# View CLI help
python src/cli.py --help
```

Clone and Explore Repository

Expected Output

```
Usage:  cli.py [OPTIONS] COMMAND [ARGS]...
Commands:
generate-wallet    Generate a new HD wallet
show-derivation    Visualize key derivation path
build-tx           Build and sign a raw transaction
explore-tx         Explore an existing transaction
check-balance      Check address balance
explore-block      Explore block details
```

Step 2: Create Virtual Environment and Install Dependencies

Objective: Set up an isolated Python environment and configure API credentials.

```
# Create and activate virtual environment
python -m venv venv
source venv/bin/activate     # Linux/Mac
# venv\Scripts\activate      # Windows

# Install dependencies
pip install eth-account web3 mnemonic rich click python-dotenv

# Verify installation
pip list | grep -E "eth-account|web3|mnemonic|rich|click"

# Configure API keys
cp .env.example .env
# Edit .env:
# ETHERSCAN_API_KEY=your_etherscan_api_key_here
# INFURA_PROJECT_ID=your_infura_project_id_here

# Verify CLI loads
python src/cli.py --help
```

Environment Setup

You need free API keys from two services:
Etherscan: Register at `https://etherscan.io/apis` for transaction/block data.
Infura: Register at `https://infura.io/` for Ethereum node access.
Both offer free tiers sufficient for this lab. **Never commit your .env file to version control.**

Screenshot 1

What to paste: Terminal showing the cloned repository structure (`tree src/` output) and the CLI help output displaying all available commands.

Paste your screenshot here

Part B: Wallet Generation & Key Derivation

Step 3: Generate an HD Wallet from Mnemonic Phrase

Objective: Generate an HD wallet using BIP-39 and understand each output component.

```
# Generate a new HD wallet
python src/cli.py generate-wallet
# Output: mnemonic (12 words), seed (hex), master private key,
#         master public key, Ethereum address

# Generate with 24-word mnemonic (more entropy)
python src/cli.py generate-wallet --strength 256

# Generate multiple accounts from same mnemonic
python src/cli.py generate-wallet --accounts 5
```

Generate HD Wallet

Expected Output

```
===== HD Wallet Generated =====
Mnemonic Phrase (12 words):
abandon ability able about above absent absorb abstract
absurd abuse access accident
Seed (hex):  5eb00bbddcf069084889a8ab9155568165f5c453...
Master Private Key:  0x4c0883a6958...
Master Public Key:  0x04bfcab28162...
Ethereum Address:  0x71C7656EC7ab88b098defB751B7401B5f6d8976F
```

The mnemonic shown above is an **example only**. Your output will be different. **REMINDER:** Do not use this wallet for real funds — this is testnet-only. Anyone with your mnemonic can derive all keys and steal funds.

BIP-39 Mnemonic Generation Process:

1. **Entropy:** 128 bits of random data generated (for 12-word mnemonic)
2. **Checksum:** SHA-256 hash of entropy; first 4 bits appended (132 bits total)
3. **Word Selection:** 132 bits split into 12 groups of 11 bits; each maps to a word from the 2048-word BIP-39 list
4. **Seed Derivation:** Mnemonic + optional passphrase processed through PBKDF2-HMAC-SHA512 (2048 iterations) producing a 512-bit seed

Screenshot 2

What to paste: Terminal output showing the generated HD wallet. **IMPORTANT:** Redact or blur the mnemonic phrase and private key — leave only partial values visible.

Paste your screenshot here

Step 4: Visualize Key Derivation Path

Objective: Understand the hierarchical key derivation from mnemonic to Ethereum address using BIP-32/BIP-44.

```
# Show the derivation path visualization
python src/cli.py show-derivation

# Show derivation for a specific path
python src/cli.py show-derivation --path "m/44'/60'/0'/0/0"

# Show multiple derived addresses (indices 0-4)
python src/cli.py show-derivation --indices 5
```

Visualize Key Derivation Path

Expected Output

```
===== Key Derivation Path Visualization =====
Mnemonic (12 words)
|  PBKDF2-HMAC-SHA512 (2048 rounds)
v
Seed (512 bits)
|  HMAC-SHA512 with key "Bitcoin seed"
v
Master Key (m)
+- m/44'    (Purpose:  BIP-44 Multi-Account)
|   +- m/44'/60'    (Coin:  Ethereum)
|   |   +- m/44'/60'/0'    (Account 0)
|   |   |   +- m/44'/60'/0'/0    (External chain)
|   |   |   |   +- m/44'/60'/0'/0/0    (Index 0)
|   |   |   |       |  secp256k1 + Keccak-256
|   |   |   |       v
|   |   |   |   Ethereum Address:  0x71C7...
```

BIP-44 Path Breakdown:

Level	Value	Meaning
m	Root	Master key (derived from seed)
44'	Purpose	BIP-44 multi-account hierarchy
60'	Coin Type	Ethereum (60 = ETH, 0 = BTC)
0'	Account	Account index (hardened)
0	Change	0 = external, 1 = internal (change)
0	Index	Address index within the chain

The apostrophe (') indicates **hardened derivation** — the child key cannot be used to derive the parent key. This is a critical security property.

Screenshot 3

What to paste: Terminal output showing the complete key derivation path ASCII visualization, from mnemonic through seed, master key, BIP-44 path levels, to the final Ethereum address.

Paste your screenshot here

Part C: Transaction Building & Exploration

Step 5: Build and Sign a Raw Transaction

Objective: Construct a raw Ethereum transaction, understand each field, and sign it.

```
# Build a testnet transaction (Sepolia)
python src/cli.py build-tx \
  --to 0x742d35Cc6634C0532925a3b844Bc9e7595f2bD18 \
  --value 0.01 --network sepolia

# Build with custom gas parameters
python src/cli.py build-tx \
  --to 0x742d35Cc6634C0532925a3b844Bc9e7595f2bD18 \
  --value 0.01 --gas-price 20 --gas-limit 21000 \
  --network sepolia
```

Build and Sign a Raw Transaction

Expected Output

```
===== Transaction Builder (Sepolia Testnet) =====
Transaction Fields:
nonce:      0     (sender's tx count)
gasPrice:   20 Gwei  (fee per gas unit)
gasLimit:   21000    (max gas for simple transfer)
to:         0x742d35Cc6634C053...
value:      0.01 ETH (10000000000000000 wei)
data:       0x (empty for simple transfer)
chainId:    11155111 (Sepolia)
Signature (ECDSA):
v:  28  r:  0x3a4b5c...  s:  0x1a2b3c...
Raw Signed TX: 0xf86c0a8502540be400...
TX Hash:  0x9fc76417374aa880d4449a1f7f31ec597...
```

Transaction Field Reference:

Field	Description
`nonce`	Sequential counter preventing replay attacks
`gasPrice`	Price in wei per unit of gas
`gasLimit`	Maximum gas units (21000 for simple ETH transfer)
`to`	Recipient address (20 bytes)
`value`	Amount of ETH to transfer (in wei)
`data`	Input data for contract calls; empty for transfers
`v, r, s`	ECDSA signature; `v` encodes chain ID (EIP-155)

This command builds and signs a transaction **locally** — it does **not** broadcast to any network. The transaction exists only in your terminal output.

Step 6: Explore Existing On-Chain Transactions

Objective: Fetch and decode a real Ethereum mainnet transaction using the Etherscan API.

```
# Explore a mainnet transaction by hash
python src/cli.py explore-tx \
  --hash 0
      x5c504ed432cb51138bcf09aa5e8a410dd4a1e204ef84bfed1be16dfba1b22060
       \
  --network mainnet

# Explore with verbose output
python src/cli.py explore-tx \
  --hash 0
      x5c504ed432cb51138bcf09aa5e8a410dd4a1e204ef84bfed1be16dfba1b22060
       \
  --network mainnet --verbose
```

Explore an Existing Transaction

Expected Output

```
===== Transaction Explorer (Mainnet) =====
TX Hash:  0x5c504ed432cb51138bcf09aa5e8a410d...
Status:        Success
Block Number:  46147
Timestamp:     2015-08-07 03:30:33 UTC
From:  0xA1E4380A3B1f749673E270229993eE55F35663b4
To:    0x5DF9B87991262F6BA471F09758CDE1c0FC1De734
Value: 31337 wei (0.000000000000031337 ETH)
Gas Used:      21000 (100.0%)
Gas Price:     50 Gwei
TX Fee:        0.00105 ETH
```

The hash `0x5c504ed...` is the **first-ever Ethereum transaction** on mainnet, sent in block 46147 on August 7, 2015. Exploring well-known transactions builds intuition for on-chain data.

Screenshot 4

What to paste: Terminal output showing decoded transaction details from the `explore-tx` command, including sender, receiver, value, gas details, block number, and status.

Paste your screenshot here

Part D: Blockchain Exploration

Step 7: Check Address Balances

Objective: Query ETH and ERC-20 token balances and understand wei/gwei/ether conversions.

```
# Check balance of a mainnet address
python src/cli.py check-balance \
  --address 0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe \
  --network mainnet

# Check with token details
python src/cli.py check-balance \
  --address 0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe \
  --network mainnet --show-tokens
```

Check Address Balances

Expected Output

```
===== Balance Checker (Mainnet) =====
Address:  0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe
ETH Balance:
Wei:   123456789000000000000
Gwei:  123456789.000000000
Ether: 123.456789 ETH
ERC-20 Tokens:
USDT: 10,500.00  USDC: 5,200.50  DAI: 1,000.00
```

Step 8: Explore Block Details

Objective: Query block-level data including timestamp, validator, transaction count, and gas usage.

```
# Explore the latest block
python src/cli.py explore-block --number latest --network mainnet

# Explore a specific block
python src/cli.py explore-block --number 17000000 --network mainnet

# Explore the genesis block
python src/cli.py explore-block --number 0 --network mainnet
```

Explore Block Details

Expected Output

```
===== Block Explorer (Mainnet) =====
Block Number:      19,500,000
Block Hash:        0xa1b2c3d4e5f6...
Timestamp:         2024-03-15 14:22:47 UTC
Miner/Validator:   0x95222290DD7278Aa3Ddd389...
TX Count:          187
Gas Used:          12,450,000 / 30,000,000 (41.5%)
Base Fee:          25.3 Gwei
Difficulty:        0 (Proof of Stake)
```

Since Ethereum's Merge (September 2022), the "miner" field shows the block **proposer** (validator). Difficulty is 0 and nonce is zeroed — these fields were only relevant under Proof of Work.

Screenshot 5

What to paste: Terminal output showing both balance check results (ETH balance with unit conversions) and block exploration results (block details with transaction count, gas usage, validator).

Paste your screenshot here

Conceptual Background

Cryptographic Foundations

Hash Functions

- **SHA-256:** 256-bit digest used in Bitcoin for block hashing and Merkle trees. Properties: deterministic, pre-image resistant, collision resistant.
- **Keccak-256:** Ethereum's primary hash function (SHA-3 variant). Used for address derivation, transaction hashing, and state root computation.

Elliptic Curve Cryptography (secp256k1)

Both Bitcoin and Ethereum use the **secp256k1** curve ($y^2 = x^3 + 7$ over $\mathbb{F}_p$). A private key k (256-bit integer) generates public key $K = k \times G$ via scalar multiplication. Computing K from k is efficient; deriving k from K is computationally infeasible (discrete logarithm problem).

Digital Signatures (ECDSA)

The sender creates signature (r, s) using their private key and the transaction hash. Anyone can verify using the sender's public key. Ethereum adds recovery parameter v allowing public key recovery from the signature alone.

HD Wallets

- **BIP-39:** Defines a 2048-word list for mnemonic generation. 12 words = 128-bit entropy + 4-bit checksum. Optional passphrase adds extra protection.
- **BIP-32:** A single seed generates an entire tree of key pairs via HMAC-SHA512 derivation. Enables millions of addresses from one backup.
- **BIP-44:** Standardized path `m/purpose'/coin'/account'/change/index` enables multi-currency support and wallet interoperability across MetaMask, Ledger, Trust Wallet, etc.

Ethereum Address Derivation

Private Key (256-bit) → **Public Key** (secp256k1, 512-bit) → **Keccak-256 Hash** → **Last 20 Bytes** → **Checksum Address** (EIP-55)

Transaction Anatomy

- **Nonce:** Prevents double-spending; increments per outgoing transaction
- **Gas Price/Limit:** Fee mechanism; 21,000 gas for simple transfers
- **To/Value/Data:** Recipient, amount (wei), and encoded contract calls
- **Signature (v, r, s):** ECDSA proof; v includes chain ID (EIP-155) for replay protection

Wallet Security Best Practices

1. **Seed Phrase Storage:** Paper or metal backup; never store digitally in plaintext
2. **Hardware Wallets:** Ledger/Trezor store keys in secure element chips
3. **Cold Storage:** Air-gapped computers; transactions signed offline
4. **Multi-Sig:** Require m-of-n signatures (e.g., 2-of-3) for treasury management
5. **Passphrase:** Optional BIP-39 passphrase creates a different wallet from same mnemonic

India Context: Cryptocurrency Regulation

RBI Stance

The RBI banned banks from crypto transactions in 2018 (overturned by Supreme Court in 2020), expressed "serious concerns" in 2021, and is developing the Digital Rupee (e-INR) CBDC as a regulated alternative. Private cryptocurrencies are not banned but face heavy regulation and taxation.

SEBI Proposals

SEBI has proposed classifying certain crypto assets as securities, requiring exchange registration, mandating investor protection, and enforcing KYC/AML compliance.

VDA Tax Framework (Finance Act 2022)

- **30% Tax:** Flat rate on all VDA gains (no deductions except acquisition cost)
- **1% TDS:** On crypto transactions above INR 10,000 (INR 50,000 for specified persons)
- **No Loss Set-Off:** Crypto losses cannot offset other income
- **Gift Tax:** Receiving crypto as gift is taxable income for recipient

Indian Exchange Wallet Security

- **WazirX:** Multi-sig cold storage for 95%+ of funds, KYC mandatory
- **CoinDCX:** BitGo custody, ISO 27001 certified, FIU-IND registered
- Exchanges use layered architecture: **Hot** (5-10%, immediate), **Warm** (10-20%, delayed), **Cold** (70-85%, air-gapped) wallets
- All exchanges must register with FIU-IND under PMLA regulations

Assessment & Deliverables

Assessment Questions

Q1. Explain the relationship between mnemonic phrase, seed, and private key in an HD wallet. Why is the mnemonic the "master backup"?

Q2. What does each level of the BIP-44 path `m/44'/60'/0'/0/0` represent? How does changing the last index generate a new address?

Q3. Describe five key fields of an Ethereum transaction. What happens if you submit a transaction with an incorrect nonce?

Q4. Why does Ethereum use Keccak-256 instead of SHA-256 for address derivation? What security properties does it provide?

Q5. Compare hot, warm, and cold wallets. Which would you recommend for an Indian exchange holding INR 500 crore in crypto, and why?

Q6. Explain how ECDSA signatures authenticate transactions without revealing the private key. What is the role of v?

Q7. Describe India's VDA tax framework. How does the 30% flat tax and 1% TDS impact crypto traders and exchanges?

Q8. If a hardware wallet is lost, how can the owner recover funds? What security risks exist during recovery?

Deliverables Checklist

Item	Description	Type	Status
Screenshot 1	Repo structure + CLI help	Paste	☐
Screenshot 2	Generated wallet (redacted)	Paste	☐
Screenshot 3	Key derivation visualization	Paste	☐
Screenshot 4	Transaction exploration	Paste	☐
Screenshot 5	Balance + block exploration	Paste	☐
Q1–Q8	Assessment answers	Text	☐
Wallet Output	Generated wallet (redacted)	Text	☐
TX Analysis	Transaction field analysis	Text	☐

Verification Checklist

- ☐ Repository cloned and project structure verified
- ☐ Virtual environment created and all dependencies installed
- ☐ API keys configured in `.env` (Etherscan + Infura)
- ☐ HD wallet generated with mnemonic, seed, and keys
- ☐ Key derivation path visualization displayed and understood
- ☐ Raw transaction built and signed on Sepolia testnet
- ☐ Existing mainnet transaction explored and decoded
- ☐ Balance checked with unit conversions displayed
- ☐ Block details explored for latest and/or specific blocks
- ☐ All 5 screenshots captured (sensitive data redacted)
- ☐ All 8 assessment questions answered
- ☐ All generated key material deleted after completion
- ☐ No real funds sent to any lab-generated address

Grading Rubric

Criteria	Description	Points	Score
Environment Setup	Repo, venv, deps, API keys	10	___/10
Wallet Generation	HD wallet with valid mnemonic, seed, keys	15	___/15
Key Derivation	Path visualized, BIP-32/44 explained	15	___/15
TX Building	Raw TX built, fields explained, signed	15	___/15
Blockchain Exploration	TX explored, balance checked, block retrieved	15	___/15
Security Awareness	Key handling, testnet usage, data redaction	10	___/10
Screenshots	All 5 clear and properly redacted	10	___/10
Assessment Answers	Q1–Q8 accurate with depth	10	___/10
	TOTAL	**100**	___/100

Appendix A: BIP-39 Word List & Entropy Reference

Entropy (bits)	Checksum (bits)	Total (bits)	Words
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

Key properties: each word uniquely identifiable by first 4 characters; 2048 words total (index 0 = "abandon", index 2047 = "zoo"); final word contains checksum bits; word lists exist in multiple languages (English most common).

Appendix B: Key Derivation Path & Transaction Reference

Path	Cryptocurrency	Standard
`m/44'/0'/0'/0/0`	Bitcoin	BIP-44
`m/44'/60'/0'/0/0`	Ethereum	BIP-44
`m/44'/501'/0'/0'`	Solana	BIP-44 (modified)
`m/44'/714'/0'/0/0`	Binance Chain	BIP-44
`m/44'/118'/0'/0/0`	Cosmos	BIP-44

Derivation types: Hardened (index $\geq 2^{31}$, denoted '): uses parent private key, more secure. **Normal** (no apostrophe): uses parent public key, allows watch-only derivation.

Appendix C: Wei / Gwei / Ether Conversion Table

Amount	Wei	Gwei	Ether
1 Wei	1	0.000000001	0.000000000000000001
1 Gwei	1,000,000,000	1	0.000000001
1 Ether	10^{18}	10^{9}	1
21,000 gas at 20 Gwei	4.2×10^{14}	420,000	0.00042
0.01 ETH	10^{16}	10^{7}	0.01

Appendix D: Wallet Security Checklist

☐ Mnemonic phrase written on paper/metal, stored securely (never digitally)

- ☐ Private keys never shared or transmitted over the internet
- ☐ Hardware wallet firmware updated to latest version
- ☐ Recovery seed tested on a separate device before reliance
- ☐ Multi-sig configured for high-value wallets (2-of-3 minimum)
- ☐ Hot wallet contains only funds needed for immediate use
- ☐ Transaction amounts and addresses double-checked before signing
- ☐ Wallet software downloaded only from official sources (verify checksums)
- ☐ 2FA enabled on all exchange accounts (TOTP preferred over SMS)

Appendix E: Troubleshooting

Problem: `ModuleNotFoundError: No module named 'eth_account'`
Solutions:

1. Verify venv is activated (check for `(venv)` prefix)
2. Reinstall: `pip install -force-reinstall eth-account web3 mnemonic`
3. Check Python version: `python -version` (must be 3.8+)
4. On Windows: ensure activated with `venv\Scripts\activate`

Problem: `HTTPError 403` or `Invalid API Key`
Solutions:

1. Verify `.env` file exists with valid keys (no trailing spaces/quotes)
2. Check Etherscan key at `https://etherscan.io/myapikey`
3. Check Infura ID at `https://app.infura.io/dashboard`
4. Free tier: Etherscan allows 5 calls/sec; add delays if rate-limited

Problem: Signing fails with `ValueError` or invalid raw TX
Solutions:

1. Verify `-to` address is valid checksummed Ethereum address
2. Ensure `-value` is a decimal number (e.g., 0.01)
3. Provide explicit `-gas-limit 21000` for simple transfers
4. Ensure wallet generation (Step 3) was completed first

Appendix F: Resources & Tools

Documentation

→ Web3.py: `https://web3py.readthedocs.io/`

→ BIP-39 Spec: `https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki`

→ BIP-32 Spec: `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`

→ BIP-44 Spec: `https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki`

→ Ethereum Yellow Paper: `https://ethereum.github.io/yellowpaper/paper.pdf`

→ RBI on Virtual Currencies: `https://www.rbi.org.in/`

→ Income Tax Section 115BBH (VDA): `https://incometaxindia.gov.in/`

Tools Used in This Practical

Tool	Purpose	Cost
Python 3.8+	Core runtime environment	Free
eth-account	Ethereum key/account management	Free
web3.py	Ethereum blockchain interaction	Free
mnemonic	BIP-39 mnemonic generation	Free
Rich	Terminal output formatting	Free
Click	CLI framework	Free
Etherscan API	Transaction/block data queries	Free (5 req/sec)
Infura	Ethereum node access (JSON-RPC)	Free (100K req/day)

—END OF LAB MANUAL—

Document Version: 1.0
IT Management & Audits – Practical Lab Series