Python Basics

- Variables and Data Types
- Comments and Print Statements
- Arithmetic and Logical Operators
- String Manipulation
- Type Conversion

Control Flow

- if, elif, else Statements
- for Loops
- while Loops
- break, continue, pass

Functions

- Defining and Calling Functions
- Function Parameters and Return Values
- Scope (Global vs Local Variables)

Error Handling

- **`try`**, **`except`**, **`else`**, **`finally`**
- Custom Exceptions with **`raise`**

Data Structures

- Lists
- Tuples (Read-only Lists)
- Dictionaries (Key-Value Pairs)
- Sets
- Looping Through Data Structures
- Nested Data (Dicts Inside Lists, etc.)

File and Log Handling

- Reading Text Files
- Writing to Files
- Working with File Paths
- Handling Large Log Files

JSON and CSV Handling

- Reading and Writing JSON (json module)
- Parsing JSON Logs

- Reading and Writing CSV (csv module)

Regular Expressions

- re.search()
- re.findall()
- Writing Patterns to Match Log or Payload Content

Using External Libraries

- Installing with pip
- Importing Modules
- Using Common Modules: os, sys, time, random

Debugging and Logging

- Using `pdb` for Step-by-Step Debugging
- Logging with `logging` **Module (vs. print)**

Virtual Environments

- Creating Isolated Environments with venv
- Activating/Deactivating Virtual Environments
- Installing Dependencies with pip in venv

Script Structure and Best Practices

- `if __name__ == "__main__":` Guard
- Organizing Code into Functions and Modules

Concurrency (Optional but Useful)

- Running Threads with `threading`
- Asynchronous Code with `asyncio`
- Running Multiple Processes with `multiprocessing`

API Interaction

- Using requests.get() and requests.post()
- Parsing API Responses (JSON)
- Sending Headers and Authentication Tokens

Command-Line and System Interaction

- Using sys.argv for Command-Line Arguments
- Running Shell Commands with subprocess
- Capturing Output from Terminal Tools

Network Scripting (Optional but Powerful)

- Using socket to Build Basic Clients/Servers
- Parsing and Sending Raw Packets with scapy
- Packet Sniffing with scapy
- Port Scanning Scripts

Security/CTF-Specific Tools (Optional)

- Using pwntools for Exploit Writing
- Encoding/Decoding with base64, hashlib
- Working with Binary Data (struct, bytearray)

Python for Malware Analysis (optional advanced topic):

- Bytecode disassembly with dis
- Working with PE files using libraries like pefile

Log and Alert Parsing from Tools:

- Automating Suricata/Zeek/Wazuh alert parsing
- Normalizing log formats with Python

Basic Fuzzing Concepts:

- Using Python to generate malformed input for fuzzing

Binary Exploitation Support

- Python scripting for:
  - ROP chain creation (ropgadget, pwntools)
  - Shellcode crafting and encoding
  - ELF/PE parsing
- Libraries: capstone, pwntools, angr (for symbolic execution)

Memory & Process Interaction

- Interacting with running processes:
  - psutil (process info, system metrics)
  - ptrace wrappers or frida (dynamic instrumentation)
  - Memory dumping/parsing

Web App Testing Scripts

- Automate testing of forms, sessions, and authentication
- Detecting XSS, SQLi patterns with payload injection
- Useful tools: requests, BeautifulSoup, mechanize, selenium

MITM/Proxying Tools

- Write simple proxies/sniffers
- Modify traffic in real time
- Build or interact with:
    - mitmproxy
    - Custom DNS spoofers

Interfacing with Security Tools

- Programmatic access to:
    - Metasploit RPC
    - Shodan or VirusTotal APIs
    - Wazuh API for rule tuning and alerting
    - Parsing PCAPs with pyshark or dpkt