Advanced Network Automation Techniques Using Python(5 days)

By Dr. Vishwanath Rao

Course Objectives:

By the end of this course, participants should:

- Understand and apply fundamental Python concepts.
- Have experience working with essential data structures (lists, dictionaries, tuples, sets).
- Be able to write functions, handle errors, and manage files in Python.
- Understand object-oriented programming basics.
- Have hands-on experience solving problems and building simple Python projects.
- Understand and implement CI/CD workflows for network automation.
- Leverage NAPALM, Nornir, and vendor-specific APIs to automate multi-vendor networks.
- Build scalable, secure, and efficient network automation solutions.

1. Introduction to Python

- **What is Python?**
  - Python history and versions
  - Why Python is popular
  - Python use cases: web development, automation, data science, machine learning, etc.
- **Setting Up Python**
  - Installing Python (Windows, macOS, Linux)
  - Introduction to the Python interpreter
  - Setting up an Integrated Development Environment (IDE) like VS Code, PyCharm
- **Running Python Scripts**
  - Running Python scripts in the terminal/command line
  - Introduction to Jupyter Notebooks (optional)

2. Python Syntax Basics

- **Basic Syntax and Structure**
  - Python syntax rules (indentation, comments, line breaks)
  - Variables and assignment (x = 10)
  - Input and Output (input(), print())
- **Data Types**
  - Primitive types: integers, floats, strings, booleans

    - Type conversion (int(), float(), str(), bool())

3. Operators and Expressions

- **Arithmetic Operators** (+, -, *, /, //, %, **)
- **Comparison Operators** (==, !=, >, <, >=, <=)
- **Logical Operators** (and, or, not)
- **Assignment Operators** (+=, -=, *=, /=)
- **Operator Precedence**
- **Expressions and Evaluation**
    - Understanding expressions
    - Using expressions in conditions

4. Control Flow

- **Conditional Statements**
    - if, elif, and else statements
    - Nested conditions
    - Ternary operator (x if condition else y)
- **Loops**
    - for loops
        - Iterating over ranges and collections (e.g., range(), list)
    - while loops
        - Infinite loops and breaking out of loops
    - **Loop control** (break, continue, pass)
- **Best Practices in Writing Loops and Conditions**

5. Python Data Structures

- **Lists**
    - Creating and manipulating lists
    - List methods (append(), remove(), sort(), pop(), len())
    - List comprehensions
- **Tuples**
    - Creating tuples, immutability
    - Accessing elements in tuples
- **Dictionaries**
    - Key-value pairs, creating dictionaries
    - Dictionary methods (get(), items(), keys(), values())
    - Iterating through dictionaries
- **Sets**

    - Creating sets, set operations (union(), intersection(), difference())
    - Set methods
- **Common Data Structure Operations**
    - Checking membership (in, not in)
    - Length, min/max, sorting, slicing

6. Functions in Python

- **Defining and Calling Functions**
    - Using def to define functions
    - Arguments and return values
- **Function Arguments**
    - Positional arguments
    - Keyword arguments
    - Default values for arguments
    - Variable-length arguments (*args, **kwargs)
- **Return Statement**
    - Returning multiple values (tuples)
- **Scope and Lifetime of Variables**
    - Local vs global variables
    - Using the global keyword
- **Lambda Functions**
    - Writing anonymous functions
    - Using lambda functions in Python (with map(), filter(), etc.)

7. Error Handling and Exceptions

- **Introduction to Error Handling**
    - Types of errors (syntax errors, runtime errors, logical errors)
- **Using try, except Blocks**
    - Catching specific exceptions
    - Using multiple except blocks
- **finally and else**
    - Ensuring final code execution with finally
    - Code that runs only when no exception is raised using else
- **Raising Exceptions**
    - Using raise to manually trigger exceptions
- **Best Practices in Error Handling**

8. File Handling

- **Opening and Closing Files**
  - open(), read(), write(), close()
  - Using with to automatically close files
- **Reading from Files**
  - Reading the entire file, line by line, or into a list
  - Using loops to process file data
- **Writing to Files**
  - Writing strings and formatted data to files
  - Appending data to existing files
- **Working with Different File Formats**
  - Working with CSV, JSON, and text files (optional advanced)

9. Modules and Packages

- **Using Built-in Python Modules**
  - Importing modules (import, from ... import)
  - Common built-in modules (math, random, datetime, os)
- **Creating Your Own Modules**
  - Defining and importing custom modules
- **Introduction to Python Packages**
  - Installing external packages with pip
  - Using popular Python packages (e.g., requests, pandas)

10. Introduction to Object-Oriented Programming (OOP)

- **Basic Concepts of OOP**
  - Classes and Objects
  - Defining a class in Python
- **Attributes and Methods**
  - Instance variables and class variables
  - Writing methods (including __init__ for initialization)
- **Inheritance**
  - Creating subclasses and using inheritance
  - Method overriding
- **Encapsulation and Abstraction**
  - Using private variables and methods
  - Getters and setters (optional)

11. Working with Libraries

- **Understanding the Python Ecosystem**
  - Introduction to third-party libraries and modules
- **Using Popular Libraries**
  - Introduction to libraries like numpy, pandas, and matplotlib (basic overview)
- **Building Simple Projects with Libraries**
  - Building a basic data analysis script using pandas
  - Visualizing data using matplotlib

12. Debugging and Testing

- **Debugging Techniques**
  - Using print statements for debugging
  - Using the Python Debugger (pdb)
- **Writing Tests**
  - Importance of testing
  - Writing basic unit tests with the unittest module
- **Test-Driven Development (TDD) (Optional Advanced)**
  - Writing tests before writing the actual code

13. Final Projects

- **Project 1: Simple Banking Application**
  - Using conditions, loops, and functions to create a basic banking system that allows for deposits, withdrawals, and checking balances.
- **Project 2: Text-based Game**
  - Building an interactive text-based game using control flow, loops, and data structures.
- **Project 3: CSV File Data Processor**
  - Write a script that processes data from a CSV file and performs operations like searching, sorting, and generating summary statistics.
-

14. Introduction to Network Automation

- **Overview of Network Automation**
  - Benefits of automation in network management
  - Challenges in traditional network management

- **Network Automation Use Cases**
  - Config management, monitoring, troubleshooting
- **Python for Network Automation**
  - Why Python? (libraries, simplicity, community support)
  - Basic Python review (for non-programmers)

15. Python Networking Libraries

- **Introduction to Python Libraries for Networking**
  - Overview of commonly used libraries:
    - **Netmiko**
    - **Paramiko**
    - **NAPALM** (Network Automation and Programmability Abstraction Layer with Multivendor support)
    - **Nornir** (automation framework)
- **Network Device Connectivity**
  - SSH connections to network devices using **Netmiko**
  - Executing show commands, gathering outputs
  - Working with device configuration templates (e.g., Jinja2)

16. Network Device Configuration Automation

- **Automating Configuration Changes**
  - Push configuration changes using Python scripts
  - Generate configurations dynamically using **Jinja2** templates
- **Working with NAPALM**:
  - Multi-vendor support overview (IOS, JunOS, NX-OS, etc.)
  - Retrieving and pushing configurations
  - Configuration validation and rollback capabilities
- **Automating Device Backups**
  - Backup and restore configurations across multiple devices
  - Use of version control for configurations

17. Network API Integration

- **Introduction to APIs**
  - Overview of REST APIs and SOAP APIs
  - Using **HTTP** libraries in Python (requests module)
- **Vendor-specific APIs**
  - **Cisco's RESTCONF** and **NETCONF**
  - **Arista eAPI**, **Juniper JunOS PyEZ**

- **Working with APIs in Python**:
  - Querying network devices for information
  - Pushing configurations via APIs
- **Handling Authentication**:
  - Basic and OAuth2 authentication in APIs
  - Securely storing and retrieving credentials

18. Continuous Integration / Continuous Delivery (CI/CD) in Networking

- **Overview of CI/CD for Network Automation**
  - How CI/CD applies to network automation
  - Benefits of version control and automated pipelines
- **Git for Network Engineers**
  - Using Git for version control of network configurations
  - Creating and managing branches
  - Working with Pull Requests and code reviews
- **Using CI/CD tools**
  - **Jenkins** for automating network tasks
  - Building pipelines for network configuration updates
- **Automated Testing of Network Configurations**
  - Automated testing frameworks (e.g., **PyTest**, **Ansible Molecule**)
  - Validating configuration changes before deployment
- **Deploying Changes via Pipelines**
  - Integrating network automation with Jenkins/GitLab CI
  - Post-deployment validation and rollback processes

19. Advanced Automation with NAPALM

- **Advanced NAPALM Features**
  - Working with napalm.getters for retrieving network information
  - Automating compliance checks with NAPALM
- **Network Health Checks**
  - Automating regular health checks (e.g., OSPF, BGP status, interface monitoring)
  - Using Python scripts and NAPALM for automated troubleshooting
- **NAPALM's Merge vs Replace Configurations**
  - Differences between merge and replace operations
  - Safely updating configurations with minimal disruption

20. Automating Network Operations with Nornir

- **Introduction to Nornir**
  - Overview and advantages of Nornir over traditional automation tools
  - Building inventory and configuring Nornir
- **Parallel Execution with Nornir**
  - Running tasks across multiple devices in parallel
  - Task groups and workflows in Nornir
- **Nornir Plugins and Task Automation**
  - Using built-in plugins to gather information and configure devices
  - Writing custom plugins for specific network tasks
- **Data Processing and Reporting**
  - Processing and formatting network device outputs
  - Generating reports (CSV, JSON) for operational insights

21. Event-Driven Network Automation

- **Event-Driven Automation Concepts**
  - Monitoring network events (e.g., SNMP traps, syslog messages)
  - Triggering actions based on network events
- **Integration with Monitoring Tools**
  - Using Python to interact with monitoring tools like **Nagios**, **Prometheus**, **Grafana**
- **Automating Responses to Network Events**
  - Automating configuration changes based on specific triggers (e.g., link failures)
  - Self-healing networks using event-driven automation
- **Using Python for Automated Remediation**
  - Detecting network anomalies
  - Automating troubleshooting workflows (e.g., re-routing traffic, interface resets)

22. Network Infrastructure as Code (IaC)

- **IaC Principles for Network Automation**
  - Version controlling network configurations
  - Automated infrastructure provisioning
- **Tools for Network Infrastructure Automation**
  - **Terraform**: Automating network provisioning on cloud and physical devices

    - **Ansible**: Automated configuration management for networking
- **Creating Network Topologies Dynamically**
    - Automating the creation of network topologies with code
    - Automated network testing in sandbox environments

23. Network Automation Frameworks and Orchestration

- **Using Automation Frameworks**
    - Orchestration tools (e.g., **Ansible**, **SaltStack**, **Puppet**, **Chef**)
    - Integrating Ansible playbooks into Python scripts
- **Configuration Orchestration with Ansible**
    - Using Ansible for automating tasks across network devices
    - Writing custom modules for network-specific automation
- **Integrating Multiple Tools in a Workflow**
    - Combining **Ansible**, **NAPALM**, **Netmiko**, and **Nornir** for end-to-end automation
- **Automation Dashboards and Reports**
    - Generating real-time dashboards for network operations using **Grafana** and **Prometheus**

24. Security and Best Practices in Network Automation

- **Secure Coding Practices**
    - Handling sensitive information (e.g., credentials, API keys)
    - Encryption and securing API requests
- **Auditing and Logging**
    - Setting up logging for automation scripts
    - Auditing changes to network devices and maintaining an audit trail
- **Network Automation in Production Environments**
    - Safely automating production networks
    - Rollback strategies and disaster recovery
- **Best Practices for Managing Large-Scale Network Automation Projects**
    - Modularity and scalability in automation scripts
    - Version control and collaboration