Current Technologies
Computer Learning Centers

CISCO Partner
Platinum Learning

# Programming for Network Engineers (PRNE) V2.0

***WHERE GREAT TRAINING HAPPENS EVERYDAY!***

# Programming for Network Engineers (PRNE) V2.0

## Course Duration

4 days

## Course Price

$3,595.00
36 CLCs

## Methods of Delivery

In-Person ILT
Virtual ILT
Onsite ILT

## About this Class

The Programming for Network Engineers (PRNE) V2.0 course is designed to equip you with fundamental skills in Python programming. Through a combination of lectures and lab experience in simulated network environments, you will learn to use Python basics to create useful and practical scripts with Netmiko to retrieve data and configure network devices. Upon completion of this course, you should have a basic understanding of Python, including the knowledge to create, apply, and troubleshoot simple network automation scripts.

# Programming for Network Engineers (PRNE) V2.0

## How you will benefit

This class will help you:

- Explain the need for network engineers to learn how to program
- Explain how programming relates to the journey into network automation and programmability
- Create useful and practical scripts to retrieve data and configure network devices
- Create, apply, and troubleshoot simple network automation scripts
- Gain hands-on experience with Python programming

## Why Attend with Current Technologies CLC

- Our Instructors are the top 10% rated by Cisco
- Our Lab has a dedicated 1 Gig Fiber Connection for our Labs
- Our Labs run up to Date Code for all our courses

# Programming for Network Engineers (PRNE) V2.0

## Who Should Attend

The job roles best suited to the material in this course are:

- Network Administrators
- Network Engineers with little or no programming or Python experience
- Network Managers
- Systems Engineers

## Objectives

After taking this course, you should be able to:

- Create a Python script
- Describe data types commonly used in Python coding
- Describe Python strings and their use cases
- Describe Python loops, conditionals, operators, and their purposes and use cases
- Describe Python classes, methods, functions, namespaces, and scopes
- Describe the options for Python data manipulation and storage
- Describe Python modules and packages, their uses, and their benefits
- Explain how to manipulate user input in Python
- Describe error and exception management in Python
- Describe Python code debugging methods

# Programming for Network Engineers (PRNE) V2.0

## Course Outline

**Module 1: Introducing Programmability and Python for Network Engineers**

**Module 2: Scripting with Python**

**Module 3: Examining Python Data Types**

**Module 4: Manipulating Strings**

**Module 5: Describing Conditionals, Loops, and Operators**

**Module 6: Exploring Classes, Methods, Functions, Namespaces, and Scopes**

**Module 7: Exploring Data Storage Options**

**Module 8: Exploring Python Modules and Packages**

**Module 9: Gathering and Validating User Input**

**Module 10: Analyzing Exceptions and Error Management**

**Module 11: Examining Debugging Methods**

# Programming for Network Engineers (PRNE) V2.0

## Lab Outline

- **Execute Your First Python Program**
- **Use the Python Interactive Shell**
- **Explore Foundation Python Data Types**
- **Explore Complex Python Data Types**
- **Use Standard String Operations**
- **Use Basic Pattern Matching**
- **Reformat MAC Addresses**
- **Use the if-else Construct**
- **Use for Loops**
- **Use while Loops**
- **Create and Use Functions**
- **Create and Use Classes**
- **Use the Python main() Construct**
- **Traverse the File Structure**
- **Read Data in Comma-Separated Values (CSV) Format**
- **Read, Store, and Retrieve Data in XML Format**

# Programming for Network Engineers (PRNE) V2.0

## Lab Outline Cont.

- Read, Store, and Retrieve Date in JavaScript Object Notation (JSON) Format
- Read, Store, and Retrieve Data in a Raw or Unstructured Format
- Import Modules from the Python Standard Library
- Import External Libraries
- Create a Python Module
- Prompt the User for Input
- Use Command-Line Arguments
- Manage Exceptions with the try-except Structure
- Manage Exceptions with the try-except-finally Structure
- Use Assertions
- Use Simple Debugging Methods
- Use the Python Debugger
- Code a Practical Debugging Script