PRNE

Programming for Network Engineers

32 horas | Automation, Programmability & DNA | Cisco

Cisco Continuing Education Credits

24 CE Credits

INTRODUÇÃO

Programming for Network Engineers (PRNE) is a 4-day course that equips network engineers withfundamental skills in Python programming. Version 2.0 puts an emphasis on Python Fundamentals focusing onusing Python basics to create useful and practical scripts with Netmiko to retrieve data and configure network devices.

After completing this course, students should have a basic understanding of Python and enoughknowledge to create, apply, and troubleshoot simple network automation scripts.

The learned programming skills are performed in simulated network environments giving students practical hands-on application and a sense of familiarity when learning programming.

OBJETIVO DO CURSO

Upon completing this course, students will be able to meet these objectives:

- Explain the need for Network Engineers to learn how to program and how it relates to the journey into NetworkAutomation and Programmability;
- Create a Python script;
- Describe data types that are commonly used in Python coding;
- Describe Python strings and their use cases;
- Describe Python loops, conditionals, operators, and their purpose 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.

PÚBLICO-ALVO

Professionals interested in knowing and implementing solutions using the automation tools.

This course is a automation introduction and pre-requesite for these trainings:

• Implementing Automation for Cisco Enterprise Solutions (ENAUI);

• Implementing Automation for Cisco Data Center Solutions (DCAUI);

• Implementing Automation for Cisco Security Solutions (SAUI);

• Implementing Automation for Cisco Service Provider Solutions (SPAUI);

• Implementing Automation for Cisco Collaboration Solutions (CLAUI).

PRÉ-REQUISITOS

The knowledge and skills that students should have before attending this course are as follows:

• Familiarity with Cisco IOS-XE software or other Cisco network device configuration and operation skills;

• Basic network management knowledge;

• Cisco CCNA or equivalent knowledge.

CONTEÚDO PROGRAMÁTICO

Course Introduction

Overview

Course Goal and Objectives

Course Flow

Your Training Curriculum

Learner Introductions

Introducing Programmability and Python for Network Engineers

Describe programmability basics

Continued Importance of the CLI

Describe the importance of Python

Describe and demonstrate a Python program

Scripting with Python

Create a Python script

Running a Python Script

Describe the procedure for initiating Python

Python Interactive Shell

Describe the Python interpreter

Editors and IDEs

Describe editors and IDEs

Examining Python Data Types

Describe data types that are commonly used in Python coding

Describe the string data type

Describe the integer data type

Describe the float data type

Describe the Boolean data type

Describe the Python type() function

Explain types of variables used in Python coding

Analyze the features of lists

Describe the tuple data type

Describe data sets

Describe the Python dictionary

Describe nested data types

Explain how to navigate complex data structures

Manipulating Strings

Describe Python strings and their use cases

Describe string manipulation

Describe string splitting

Describe string modification

Describe string concatenation

Describe whitespace stripping

Explain formatting and templating

Describe the use of escape characters

Describe regular expressions

Explain string methods

Describing Conditionals, Loops, and Operators

Describe Python loops, conditionals, operators, and their purpose and use cases

Describe whitespace and its properties as provided in the PEP 8 style guide

Describe conditional statements

Describe operators and their uses

Describe an example of using conditionals

Describe loops and their uses

Loops with Lists, Dicts, and Ranges

Exploring Classes, Methods, Functions, Namespaces, and Scopes

Describe Python classes, methods, functions, namespaces, and scopes

Practical Reuse of Code

Describe functions

Code Commenting

Describe namespaces and scopes

Define classes and methods

Using Inheritance to Extend Functionality

Describe the main construct

Exploring Data Storage Options

Describe the options for Python data manipulation and storage

Describe various data formats

Reading Data from a Simple Text and CSV File

Writing Data to a Simple Text and CSV File

Reading Data from a JSON Text File

Explain how to read data from a JSON text file

Writing Data to a JSON Text File

Data Access in Raw or Unstructured Format

Exploring Python Modules and Packages

Describe Python modules and packages, their uses, and their benefits

Python Standard Library Modules

Reusable Code with Modules and Packages

Gathering and Validating User Input

Explain how to manipulate user input in Python

Describe methods for obtaining user input

Command-Line Arguments

Describe the format of an argument

Analyzing Exceptions and Error Management

Describe error and exception management in Python

Describe common error types

Explain the uses for exceptions

Describe assertions

Examining Debugging Methods

Describe Python code debugging methods

Describe the Python code debugging process

Describe Python functions that can be used for debugging

Describe code debuggers

Describe the Python debugger

Describe common pdb commands

Course Summary

Provide a brief description of what you have learned in this course

Explain how to further develop your programming skills after having completed this course

Lab outline

Discovery 1: Execute Your First Python Program

Task 1: Execute a simple Python script to gather the device software version

Discovery 2: Use the Python Interactive Shell

Task 1: Launch the Python Interactive Shell

Task 2: Hello, Network!

Task 3: Exit the Python Interactive Shell

Task 4: Check Your Python Version

Discovery 3: Explore Foundational Python Data Types

Task 1: Experiment with String Data

Task 2: Experiment with Integer Data

Task 3: Experiment with Float (Floating-Point Integer) Data

Task 4: Experiment with Boolean Data

Discovery 4: Explore Complex Python Data Types

Task 1: Experiment with Lists

Task 2: Experiment with Tuples

Task 3: Experiment with Sets

Task 4: Experiment with Dictionaries

Task 5: Optimizing the Dictionary Code

Discovery 5: Use Standard String Operations

Task 1: Store and Read a String

Task 2: Split a String

Task 3: Concatenate a String

Task 4: Explore Formatting Outputs with f-Strings

Task 5: Use Escape Characters to Format a String

Task 6: Use String Methods to Modify a String

Discovery 6: Use Basic Pattern Matching

Task 1: Read and Store a String

Task 2: Split a String Based on RegEx Pattern Matching

Task 3: Concatenate a String Based on RegEx Pattern Matching

Task 4: Extract Specific Keyword Values Within a Line

Task 5: Extract Lines with Specific Keywords

Discovery 7: Reformat MAC Addresses

Task 1: Read a MAC Address and Store it as a String

Task 2: Manipulate and Concatenate MAC Addresses with RegEx

Task 3: Convert MAC Addresses From EUI-48 to EUI-64

Discovery 8: Use the if-else Construct

Task 1: Create a Simple Conditional Block Using if and else

Task 2: Create a Simple Conditional Block Using if, elif, and else Statements

Discovery 9: Use for Loops

Task 1: Use for Loops to Iterate Over Different Data Types

Discovery 10: Use while Loops

Task 1: Create a Simple Script Using a While Loop

Discovery 11: Create and Use Functions

Task 1: Create a Simple Script Using Functions

Task 2: Modify the Network Audit Tool to Include Functions for Repetitive Tasks

Discovery 12: Create and Use Classes

Task 1: Create a Class

Discovery 13: Use the Python main() Construct

Task 1: Create a Simple Script that Incorporates the main() Construct

Task 2: Modify the Network Audit Tool to Use a main() Function

Discovery 14: Traverse the File Structure

Task 1: Write a Simple Script to Navigate the File System

Discovery 15: Read Data in CSV Format

Task 1: Create a Simple Script to Read CSV Data

Task 2: Modify a Simple Script to Manipulate CSV Data

Task 3: Modify a Simple Script to Store CSV Data

Discovery 16: Read, Store, and Retrieve Data in XML Format

Task 1: Create a Simple Script to Read Raw XML Data

Task 2: Modify a Simple Script to Manipulate XML Data

Discovery 17: Read, Store, and Retrieve Data in JSON Format

Task 1: Create a Simple Script to Read JSON Data

Task 2: Modify a Simple Script to Manipulate JSON Data

Task 3: Modify a Simple Script to Store JSON Data

Task 4: Modify the Network Audit Tool to Retrieve Results and Store Them in a JSON Data Structure

Discovery 18: Read, Store, and Retrieve Data in a Raw or Unstructured Format

Task 1: Create a Simple Script to Read Raw Data

Task 2: Modify a Simple Script to Manipulate Raw Data

Task 3: Modify a Simple Script to Store Raw Data

Discovery 19: Import Modules from the Python Standard Library

Task 1: Explore How to Import Libraries such as sys and os from the Python Standard Library

Discovery 20: Import External Libraries

Task 1: Import Specific Functions from a Library

Task 2: Explore Renaming Packages When Imported

Task 3: Explore PIP for Package Management

Discovery 21: Create a Python Module

Task 1: Add Functions and Documentation to a Simple Script to Create a Module

Discovery 22: Prompt the User for Input

Task 1: Create a Script that Prompts the User for Input

Task 2: Modify the Script to Validate Input from the User

Discovery 23: Use Command-Line Arguments

Task 1: Accept Arguments that are Added to the Command Line

Task 2: Modify the Network Audit Tool to Accept Command-Line Options

Discovery 24: Manage Exceptions **with the try-except Structure**

Task 1: Create a Simple Script Using try-except

Task 2: Modify the Network Audit Tool to Include Exception Management Using try-except to AddressPotential Failures

Discovery 25: Manage Exceptions with the try-except-finally Structure

Task 1: Create a Simple Script Using try-except-finally

Task 2: Modify the Network Audit Tool to Include Exception Management Using try-except-finally to AddressPotential Failures

Discovery 26: Use Assertions

Task 1: Define a Valid Assertion Test and Observe the Results

Task 2: Define an Invalid Assertion Test and Observe the Results

Discovery 27: Use Simple Debugging Methods

Task 1: Create a Simple Script Using the Print Function to Debug a Variable

Task 2: Add the Print Function to the Network Audit Tool to Debug Steps in Script Execution

Task 3: Debug Code with the Python Interactive Interpreter

Discovery 28: Use the Python Debugger

Task 1: Import pdb into a Script and Insert a Breakpoint

Task 2: Run a Script in the Python Debugger Using the Command Line

Discovery 29: Code a Practical Debugging Script

Task 1: Resolve the Bugs in a Sample Script