Lesson Guide

Cybersecurity Professional Program

Introduction to Python for Security

Introduction to Programming

Course Path

- **PY-01: Introduction to Programming**
- PY-02: Data Types and Conditions
- PY-03: Loops
- PY-04: File System and Error Handling
- PY-05: Functions
- PY-06: Network Communication
- PY-07: Python for Security

Lesson Objectives

Python is a dynamic programming language developed for network manipulations and computing operations. This lesson is an introduction to the Python programming language and development environment. Explain the main goals of the lesson, including what computer programming is, why Python should be installed on a PC and how, what the PyCharm environment is, and understanding basic syntax principles.

Lesson Overview

This lesson introduces the concept of programming. Learners will learn the basic methodology of language-agnostic code. They will begin to work with Python and learn the correct usage of Python syntax. The lesson includes explanations of machines and source code.

Learner Level

Learners come from a non-programming background and have no prior knowledge of this subject. Make sure they understand the basic programming concepts since a lot of what follows builds upon this foundation. It is crucial that learners feel comfortable with the basics.

Lesson Notes

Every programming concept taught in the module is first explained in general, with no specific language. The Python implementation is then explained. Make sure that learners first understand the basic concept and only then study the Python version. Make sure learners understand the rationale for using specific conditions and loops to control the program flow at every stage. Always ask learners how the code would be written without that flow, and then present the correct solution. After each Python implementation, learners should practice on their own using PyCharm. For example, when explaining a Python command in class, let learners practice it on their laptops throughout the lesson.

Environment & Tools

- Windows, macOS, Linux
  - PyCharm
  - Python 3.x.x

Introduction to Programming

Explain basic programming concepts. After completing the relevant topics, let learners explain them again in their own words to sharpen their understanding.

Computer Instructions

Explain how the CPU handles instructions and understands the code. Explain that every source code ever written was ultimately converted to machine code for execution. Mention that machine code is a numerical language that enables fast execution. It can be regarded as the lowest level source code.

What Is Programming?

Explain what a programming language is and the concept of instructing the CPU to execute the code. You can compare programs to recipes, which require ingredients, quantities, steps, order, and testing.

What Is Python?

Briefly explain the following:

- Python as a high-level programming language
- Why it is easier to learn than other languages
- The advantage of readable code
- Python's efficient, high-level data structure
- Python's simple but effective approach to object-oriented programming
- Python includes structures that enable the expression of complex programs in concise and clear ways.
- Explain Python's syntax and its use of indentation to define blocks of code.

Python vs. Other Languages

Describe the following:

- Python's versatility
- How easy it is to learn
- The fact that it can work in any environment
- Python's large set of tools is also commonly used by hackers.

Tell learners that Python is suited for cybersecurity scripting for these reasons and more.

Python Interpreter

Explain the following:

- How does the interpreter language work in contrast with compiled languages, such as C++?
- Python is known as an interpreter language, a program that reads and executes code.
- Can an interpreter be run from a CLI using the ***py*** command or installed in the IDE?
- What are the multiple implementations of Python?

Compiler vs. Interpreter

Explain the main differences between a compiler and an interpreter. For example, compiling takes more time to process the code. After compiling code, the compiler will only run in the designated environment, while an interpreter can run in any environment. In addition, after compiling code, the code can no longer be read as plain code, while interpreter languages can be read as source code.

Python 2 vs. Python 3

Explain the differences between the last two versions of Python (2 and 3), such as syntax, how strings are stored, and calculation processes.

Discuss the following:

- Python 2: Rounded calculation (5/2 will return 2)
- Python 3: Not automatically rounded (5/2 will return 2.5)
- Syntax in Python 2 does not use parentheses in commands.
- Syntax in Python 3 treats commands as functions.

Python Types

Briefly explain how Python language types can be integrated with other programming languages.

Explain the following:

- CPython was written in C and Python and is defined as an interpreter and a compiler.
- Jython can import and use any Java class.
- All types are open-source.

Python Installation

Perform a clean installation of Python 3 and PyCharm, and explain each step of the process.

Python Website

Explain that in some cases, such as Windows, Python is not provided as part of the OS and must be downloaded and installed. Point out the binaries and installation files available on the Python website. Explain that binaries are available for different operating systems and Python versions.

Installation Process

Mention the following:

- Python installation is the same as any other program installation in Windows OS.
- Python 2 was installed in C:\, unlike Python 3.
- Without adding Python to the PATH, you will not be able to access it from the CLI.
- It is recommended to select ***Add Python 3.x.x to PATH*** so that it can be launched from any location via a CLI.

Python in CLI

Link this slide with the previous slide and demonstrate how Python can be launched from any location.

Python IDE

This section demonstrates the usage of Python IDLE and PyCharm.

Working with IDE

Briefly point out why it is better to use IDE instead of CLI. IDE is considered easier to use and includes many functions that help the code development process. Mention that IDE is often the preferred choice by programmers. Explain that PyCharm is a useful IDE program used for Python programming.

IDLE

Mention that IDLE is installed together with Python. Explain that IDLE is used to create, modify, and execute Python scripts or a single statement, similar to a Python Shell.

PyCharm

Introduce learners to PyCharm IDE and point out how it provides an overview of the code structure, suggestions about improving the code, and debugging capability. Mention that although Python comes with a preinstalled IDE, more advanced IDEs are available.

Python Environment & PyCharm

This section demonstrates the Python interpreter and how to change themes and fonts in PyCharm.

New Project

Mention that PyCharm creates a new environment for each project. Projects are not affected by each other's environment.

PyCharm Settings

Describe the PyCharm Settings feature, which enables:

- Changing themes, font styles, and font sizes
- Interpreter configuration
- Adding libraries

Interpreter Configuration

Describe how to add an interpreter to a project in PyCharm. Explain that if the PyCharm installation is not detected, it can be added manually.

PyCharm Virtual Environment

Explain why Python should work with an interpreter rather than a compiler. By using an interpreter, projects will run in a virtual environment with the selected Python version and cannot be downgraded to earlier versions, which would ruin the entire project. Explain the advantage of working with a virtual environment so that projects will not be affected by each other's environment. Show how to add a Python interpreter: Select **Add Python Interpreter** and click **Add Interpreter**.

PyCharm Code Execution

Explain how a project's code can be executed using various methods, such as right-clicking -> ***Play***, ***Shift*+*F10***, or clicking the *Play* icon at the top. Explain that the console inspects the syntax and provides the results.

Lab PY-01-L1: Installing Python

This lab should be completed in class.

Basic Syntax

Demonstrate actual code in Python, which will be the first encounter learners have with Python programming. Describe the various Python data types and programming attributes (case sensitivity, statements, whitespace, keywords, and indentation).

Python Data Types

Explain each data type and how *float* differs from *integer*.
Point out the following:

- When declaring a string, the value should be included in quotation marks.
- Integer values can be positive or negative integer numbers.
- Float values can be positive or negative decimal numbers.
- Boolean values (0, 1) are used in logical operations and decision trees.

Python Syntax

Explain that comments convey information that is not part of the code.
Discuss the importance of using comments in programs and scripts to make the code more understandable. Describe Docstrings and mention that they are used when the length of a string exceeds more than one line. Point out that three quotation marks can be used in ***print()*** to print multi-line strings.

Standard Syntax Principles

Explain the following:

- The case of a character does not depend on its location in a statement.
- Case-sensitive and case-insensitive languages (provide examples)
- Python is a case-sensitive programming language, and variables such as “John” and “john” will be considered different.
- Some languages require a symbol to end a statement, such as a semicolon (**;**).
- While some languages require statement completion, Python requires completion only for condition statements.
- *Case-sensitive* means that **x** is considered different than **X**. The variable **John** is different than the variable **john**.

Code Handling

Explain the following:

- Python is a more readable language due to its whitespace and indentation.
- Code blocks can be determined based on whitespace.
- Keywords cannot be used as variable names.
- Most languages ignore extra spaces or lines as long as they can recognize the different elements.
- Some languages recognize indentation.

Give examples of reserved keywords that cannot be used as variable names. Also, provide examples of these keywords: and, if, class, finally, print, return, raise.

Lab PY-01-L2: Comments and Print

This lab should be completed in class**.**

Lab PY-01-L3: PyCharm Personalization

This lab should be completed in class.

TDX Arena

Learners will need to navigate to the TDX Arena practice arena as described in the slide deck.

Homework Lab Assignment: Variables and Data Types
This assignment asks the learner to use the TDX Arena terminal for additional practice working with variables and data types.

To complete the assignment, learners will need to visit the outlined TDX Arena lab. This TDX Arena lab will be completed as a homework assignment before the next class.