Lesson Guide

Cybersecurity Professional Program

Introduction to Python for Security

# Data Types & Conditions

## Course Path

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

## Lesson Objectives

Learners will be taught how to work with user input, variables, casting, comparison, mathematical operators, conditions, and advanced data structures.

## Lesson Overview

The lesson introduces data types, arithmetic operators, casting, conditions, and data structures.

## Learner Level

Learners should already know basic Python syntax and structure.

## Lesson Notes

During the lesson, when explaining a Python command, let learners practice the command on their own to understand the topic better.

## Environment & Tools

- Windows, macOS, Linux
    - PyCharm
    - Python3

## *Variables & User Input*

In this part of the lesson, learners will learn how variables work and how they can be created and modified through user input. After explaining variables and user input, let learners explain them in their own words.

**Variables**

Mention that variables store information and explain how variables help label data with meaningful names for easy storing and retrieving data. Explain the importance of labeling data with descriptive names to help make programs clear. Clarify to the learners that variables are a type of information container.

**Variable Types**

Demonstrate variable types, print them, and make sure learners understand each type. In addition, explain the following:

- Unlike in other languages, there is no need to declare a variable's type in Python.
- Variable names cannot start with an integer or contain special characters.
- Python automatically detects the type by the way the value is specified.
- Integers can have any length and are limited based on the available memory.

**User Input**

Demonstrate how to use input and explain that every input is treated as a string unless converted. Tell learners they will learn how to perform this conversion later in the module.

In addition, explain the following:

- Bind each input with a variable for later manipulation.
- Show how to define input types that meet the program's requirements.
- Point out the syntax in the example used to bind the input to a variable.

**Printing to the Console**

Demonstrate how to use the print function and let learners try it in class.

Explain the ***print()*** function, which prints strings and contents of a variable.

Mention that it can also print a variable's type using the ***type()*** function.

**String Formatting**

Demonstrate the options used to perform string formatting. Explain the methods of combining a string with a variable to produce predefined dynamic strings. Show how to use the percentage symbol with a variable, such as ***%d(digit)*** or ***%s(string)***.

Show how to use ***format()*** and explain that the curly brackets represent variables within the string. Show how to use the ***f string***, where **f** is placed before the string.

## *Operators & Casting*

Introduce comparison and mathematical operators and how they are implemented in Python. In addition, explain typecasting and how variables can be converted.

**Logical Operations**

Explain **AND** statements in which all conditions must be met. For example, print '**Hey**' only if the sky is blue **AND** there are clouds. There's a blue sky without clouds today, so the program will not print **Hey**. It doesn't matter if the sky is blue; there needs to be a cloud.

**Conditional Operations**

Explain the following:
**OR**: Only one of the conditions needs to be true.
Print '**Hey**' if the sky is blue **OR** there are clouds in the sky. There's a blue sky without clouds today, so that the program will print **Hey**.
**NOT**: Reverses the result of the condition.
**XOR** (**Exclusive OR**): Only one condition can be true. Suppose the sky is blue **OR** has clouds, but not both.

**Python Comparison Operators**

Point out some comparisons among integers and the results in the console as Boolean true or false. Mention that comparison operators are also known as relational operators.

**Python Mathematical Operators**

Demonstrate different kinds of mathematical operations in Python. Discuss how the results can be stored in variables for later use or printed to the console.

**Python Mathematical Operations**

Demonstrate different kinds of mathematical operations in Python.
Discuss how the results can be stored in variables for later use or printed to the console.
Point out the following:
***+*** adds the values on both sides of the operator.
**-** subtracts the operand on the right from the operand on the left.
******* multiplies the values on both sides of the operator.
/ divides the operand on the left by the operand on the right.
***%*** (modulo) divides the operand on the left by the operand on the right and returns the remainder.

**Python Type Casting**

Demonstrate how to check the type using the "type" function and convert types.
In addition, explain the following:
How to manipulate variables and convert from one type to another.

- An ***int()*** constructs an integer number casted from a string or a float.
- A ***float()*** constructs a number from an integer, a float, or a string.
- A ***str()*** constructs a string from various data types, including strings, integers, and floats.

You can also demonstrate a casting error.

**Lab PY-02-L1: Working with User Input**

The learners should complete this lab in class.

# *Conditions*

Describe conditions and how they can control program flow.

**Conditions**

Explain why conditions are necessary for programming and how a program is dynamic based on conditions chosen by the user. Writing useful programs require constant review of the conditions and updating them accordingly. Mention that developers can use as many conditions as they want in a program.

**Conditions in Python**

Demonstrate how to use conditions with different results. Explain the main idea of "if-else" statements and their flow in different scenarios. Use examples, such as "If x is true, execute y; if x is false, execute z." Mention that the "else" command is used only once at the end of a statement.

**Lab PY-02-L2: Basic Conditions**

The learners should complete this lab in class.

**Advanced Conditions**

Explain that the use case for "elif" is in cases where two choices are not enough. Mention that "elif" can be used multiple times, unlike the one-time use of "else." Point out that there is no limit to the number of conditions used in a program.

**Lab PY-02-L3: Advanced Conditions**

The learners should complete this lab in class.

# *Advanced-Data Structures*

Discuss data structures, their types (dictionary, tuple, list), and manipulations.

**Data Structure Benefits**

Explain the main idea of working with data structures, organizing code, and making it more efficient. Mention that data structures in Python include Dictionary, Tuple, and List. Provided is an explanation for each structure in the following slides.

**List**

Mention that a list is the most versatile data type in Python. Demonstrate how to create a list. Point out that lists are created by writing values (items) within square brackets. To create a list, use: ***listName = [“value1”,”value2”]***
Describe a list as sequenced values that count from left to right, beginning with 0.

**List Manipulation**

Demonstrate how to update and delete variables from a list. Mention how easy it is to update and edit lists by specifying the list name and the position of the value you want to change. Point out that the count in a list index begins with 0 (not 1).

**List Manipulation - Second Slide**

Demonstrate how the functions ***append()*** and ***remove()*** affect the lists. Point out that there are also other functions not mentioned in this module that can manipulate lists, such as ***pop()***, ***insert()***, ***sort()***, and others. Consider demonstrating the functions; however, mention to the learners that the labs or exams will not include them.

**Tuple**

Demonstrate how to create tuples. Emphasize that tuples are included in parentheses, read-only, and not edited. Also, mention that sequences in tuples are similar to those in lists.

**Dictionary**
Explain the purpose of a dictionary and its uses. Demonstrate how to create a dictionary and its key: value structure. Mention that dictionaries in Python are included in curly brackets and are unordered.

**Dictionary Manipulation**
Demonstrate how to update and delete variables from a dictionary. Explain how dictionaries can be modified by adding or removing content. Point out that deleting an entire dictionary is done using ***diction1.clear()***

**Nested Data Structures**
This part should be covered if there is any spare time during the lesson. Demonstrate how to use ***.append()***. Describe nested lists and point out that they can include other lists and be modified almost the same way as regular lists.

**Lab PY-02-L4: Dictionary Lab**
This lab should be completed in class.

**String**
Explain that developers sometimes use string manipulation to cut, delete, or print a specific value from a list.

**String Concatenation**
Define concatenation, substrings, and splits. Point out that only the + sign works with string concatenation, and other arithmetic signs do not.

**String Sub-String**
Point out that a split is done using whitespace by default or a predefined separator. Mention that the specified end index isn't displayed and is considered the endpoint of the slicing.

## *TDX Arena*

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

**Lab PY-02-L5: IP Address to Binary**
This assignment asks the learner to separate an IP address after receiving it from a user and convert it to a binary value.
To complete the assignment, learners will need to visit the outlined TDX Arena lab.
This TDX Arena lab should be started in class, and if there isn't time in class, it should be assigned as homework.

**Lab PY-02-L6: Create and Use Data Structures**
This assignment asks the learner to practice working with different types of data structures.
To complete the assignment, learners will need to visit the outlined TDX Arena lab.
This TDX Arena lab should be started in class, and if there isn't time in class, it should be assigned as homework.

**Homework Lab Assignment: String Manipulations and Conditions**
This assignment asks the learner to use the TDX Arena terminal for additional practice working with conditions and manipulating strings.
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.

**Homework Lab Assignment: Password Strength Checker**
This assignment asks the learner to use the TDX Arena terminal to develop a tool to test password strength.
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.