Lesson Guide

© 2021 HackerUSA Inc. d/b/a ThriveDX

Cybersecurity Professional Program

Introduction to Python for Security

File System & Error Handling

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

This lesson focuses on how the Python programming language can be used for error handling and file system management.

Lesson Overview

The module teaches how Python can manipulate files, manipulate directories, and handle errors in the system.

Learner Level

Learners are familiar with Python and the way it operates. They will now be introduced to how Python works with more specific aspects of the operating system.

Lesson Notes

When explaining file permissions and error handling during the lesson, let the learners practice commands and functions on their laptops.

Environment & Tools

- Windows, macOS, Linux
  - Python 3
  - PyCharm

Error Handling

Explain what error handling is and demonstrate how it works. Make sure learners understand the exception types.

How Python Handles Errors

Explain the main idea of error handling, what it is meant to do, and how it can be useful. Point out that in line 3 of the example above, the program will crash because the variable is divided by zero. Such cases can be managed by error handling.

Try* & *Except

Demonstrate how to use ***try*** and ***except***. The exception types will be explained on the following slides. Explain the idea of organized code in the ***try*** and ***except*** methods.

Common Exception Types

Point out that there are 63 built-in exceptions in Python that follow a tree hierarchy. Make sure learners understand how to work with exception types. Explain the difference between the exception types listed in the table and other types.

Let learners know that the ***keyboardinterrupt*** is in Python. You would have an out for the interrupt, so the user/programmer has a visual that it was pressed whether it was accidental or intentional.

Error Handling Loops

Demonstrate how to use error handling in loops. Explain ***finally*** and ***raise*** statements. Build full ***try*** and ***except*** methods using ***finally***, ***raise***, ***else***, and ***exception as error***. Show the flow for each. Note that ***raise*** will go to the base exception.

Lab PY-04-L1: *Try* & *Except* Practice

This lab should be completed in class.

Lab PY-04-L2: Error Handling

This lab should be assigned as homework.

File Manipulation

Explain file modes and demonstrate how they work. Make sure learners understand the differences among the modes.

Python & Files

Mention that modules will be explained later. Tell learners that having built-in modules means it is not necessary to import libraries. For example, show how to use the ***date*** function without importing ***datetime***.

File Access Modes

Point out the differences among the modes and how they can be used in combinations. It is important to note that every file assigned a ***w*** will overwrite the existing file. If the file does not exist, it will be created and written to. Music files and images contain binary or hexadecimal information. Please note that every file assigned an ***a*** points to the end of the file if it exists. If the file does not exist, it will be created and written to.

Opening Files

Add a reminder about the file permission modes. Explain that files must be closed after they are used.

Writing to Files

Explain the ***write*** function and when it is used. Add a reminder about the file permission modes. This slide presents the ***close()*** function, which will be discussed in more detail on the following slides. Explain to learners that we need to close a file after using it. Demonstrate the following:

1. Open a file for writing and write to it.
2. Open the file again for appending and write another sentence.
3. Open the file for reading and print the file.
4. Show that ***append*** added the sentence at the end of the file.

Reading Files

Describe the general purpose of the built-in function ***open***. A more detailed description will be presented later. Demonstrate how the ***read()*** and ***readline()*** functions are used in the code. For example, ***readline(5)*** will read the first five bytes of the line.

Exercise:

- Open a file and use ***read()*** to read all file content.
- Open a file, insert five ***readlines(),*** and print them separately.

Closing Files

Mention the importance of ***close()*** and demonstrate how it is used. Use the example of opening a file and then deciding whether to save it or not.

Note: An open file will close automatically when the program terminates (unless an exception occurs).

Exercise:

1. Open a file and use ***write()*** to write to it.
2. Do not close the file, and try to read it. *You will get an exception error!*

Flush

Explain the importance of ***flush()*** and demonstrate how it is used.

Note that flush works the same as when you save a doc file without closing it.

Exercise:

1. Open a file and use ***write()*** to write to it.
2. Flush it and write to it again, or open it again for reading without closing the file.
3. *Close it only when you are done using it!*

With Open **Statement**

Demonstrate how to use ***with()*** in the code. Explain the difference between ***with open*** and ***open***. The main difference is that for ***with open***, the ***close()*** function executes automatically. When a script opens and closes a file many times, using ***with open*** will remind the script to close the file at the end.

Lab PY-04-L3: Handling Files

This lab should be completed in class.

Lab PY-04-L4: Extracting Lines

This lab should be assigned as homework.

Module Definition & Usage

Explain the benefits of working with modules in Python. Discuss the need to import the modules, and make sure learners understand how to work with them.

What Are Modules?
Talk about the concept of modules and how they are imported. Python modules include predefined functions and are imported using the ***import*** keyword. Show the libraries that are not built in (like file permissions). Some modules must be downloaded from the interpreter.

Python *OS* Module
Explain the usage of the ***OS*** module and where it can be used. Point out that OS includes Windows, Linux, and macOS. For example, If we want to build a script that will get the network configuration in Linux OS, we can use the ***OS*** module to run the ***ifconfig*** command and get the output.

***OS* Functions**
Demonstrate how to use ***OS*** functions and mention why they are useful. In the example on the slide, the ***os.system()*** function executes ***ping*** in the operation system and outputs to the console.

Methods
Ensure learners understand the difference between functions and methods.

Listing Directories
The ***os.listdir(path)*** function lists all the files in the specified directory. The ***r*** before the directory instructs the interpreter to regard the characters that follow as a raw string.

In the example on the slide, the function lists the contents of the current working directory. If no path is specified in the function, it will execute the command on the current directory.

For example, to get a list of directories in C:\, use the following command:
os.listdir(r"c:\ ")

Advanced *OS* Module 1
Demonstrate how to use the ***OS*** functions. Describe ***os.rmdir()*** and mention that if the directory is not empty, an OS Error will be issued.
Note: Each function will get a full path for a directory or file. If a full path was not mentioned, the command would be executed in the current working directory.

Advanced *OS* Module 2
Demonstrate how to use the ***OS*** functions. Show how ***os.stat()*** operates and explain the return values. Note that ***os.walk()*** returns three values used by the loop to obtain the output, and ***os.stat()*** returns an object from ***os.stat_result***, which is used to obtain the status.

Datetime
Demonstrate how to work with the ***datetime*** module. ***Datetime*** can print output as a date and time structure or provide the present time.
Exercise:
Use ***datetime.datetime.now()*** to get the current time and print it in the following format: ***[hour]:[minute]:[sec]***

Lab PY-04-L5: *OS* Module & *Open* Function
This lab should be completed in class.

Lab PY-04-L6: *OS* Module
This lab should be assigned as homework.

***Platform* Module**
Demonstrate how to use the ***platform*** module. Explain where it can be effective and why.
Exercise:
Use the ***platform*** module to get the OS system and running scripts. Then use ***os.system()*** to execute a command such as ***ifconfig*** or ***ipconfig***.

***Random* Module**

Demonstrate how to use the ***random*** module. Explain where it can be effective and why. Point out that the module can run on strings and other data types.

Exercise:

Use the ***random*** function to retrieve numbers from 1 to 10. Use it twice and demonstrate a dice throw.

Log Parsing

Explain what parsing is and demonstrate how it works. Make sure learners understand its benefits and how to work with it.

Log File Parsing 1
Explain why parsing files is important and how it works. Describe how it works with loops for file objects. Using the example on the slide, explain that Python can remove information that exists in log files by parsing .***txt*** files.
Exercise:

1. Set a **string** with **“hello world”**
2. Use ***split()*** to remove the **“ “** and get a list of **[“hello”, “world”]**
3. Set a ***for*** loop to cover the list.
4. Explain that every row that ends with **\n** is an index for a **LIST** in files.

Log File Parsing 2
Demonstrate how to use ***split*** to parse logs. Remind the class about using ***file.close()***
The slide example shows how to achieve perfect parsing.
Notes:

- Each line in the loop represents a string.
- Each line sets a new variable for ***split()***
- Use string concatenation for a line.
- Use list **indexes** for new variables.

Lab PY-04-L7: Encoding & Decoding the Secret
This lab should be completed in class.

Lab PY-04-L8: Copying Files
This lab should be assigned as homework.

Lab PY-04-L9: Tasks & Questions
This lab should be assigned as homework.

TDX Arena

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

Homework Lab Assignment: Opening & Analyzing Files
This assignment asks the learner to use TDX Arena for additional practice opening and analyzing files in Python.

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

Homework Lab Assignment: Regular Expressions
This assignment asks the learner to use TDX Arena to better understand regular expressions in the context of Python.

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