## LESSON: Python Programming Essentials​

**Premier**

## This is the beginning of the Python Course (2 of 2), and instructors should ensure that the students understand the basics of Python, and provide demonstrations such as:

## Python data structures (list, types, and dictionaries)

1. Loops, error handling; try and except blocks
2. Python os. popen() function

## Python Tutor is a great resource that can help visualize this concept and how students can understand them well: https://pythontutor.com/render.html#mode=edit

For this lesson and upcoming lessons, instructors are required to ensure the following activities are completed for each lesson

* Review the “Lesson Opener” and “Real World Scenario” with the learners prior to starting the module.
* Throughout the module, you will find “Consider the Real World Scenario” slides. Review the questions found on these slides, tie the concepts back to the scenario discussed at the start of the lesson as well as content you are presenting, and encourage the learners to share their thoughts.
* For each lesson, you will find a “Pulse Check” slide which is the opportunity for instructors to open a poll to gather feedback from the learners. Leave the poll open for about 1 minute and after you close the poll, share the results with the learners. Encourage the learners to share their thoughts. This information will help the instructors as well as the learners better understand where they are with regards to the lesson.
* Labs are to be demonstrated live for each module. The demonstration of labs is the top priority for the lead instructor. While demonstrating each lab, encourage students to participate and explore.
* At the end of each lesson, it is important to take a few minutes to review the key concepts for the lesson, provide guidance on what the learners can do to prepare for the next lesson, and wrap up with Q&A.

### Summary

In this lesson, learners will discover Python's versatile data structures, including lists, tuples, and dictionaries, and grasp their fundamental role in organizing and storing data efficiently. They will explore the concept of the index, a numerical value used to locate specific elements within data structures, and understand how index slicing simplifies data extraction from sequences. The lesson covers essential operations with lists, such as adding and removing elements, and delves into dictionaries, highlighting their key-value pairs and manipulation. Learners will also gain proficiency in using loops, including for and while loops, to automate repetitive tasks and comprehend how the 'range()' function aids controlled repetition. Error handling techniques, common Python errors, and file handling, including permission modes and functions like 'read()' and 'write()', are also covered in detail. Additionally, learners will discover how to use the Python OS module for cross-platform system interactions and how to manage functions effectively within their code.

### Objectives

* Describe Python’s data structures: Lists, tuples, and dictionaries.
* Explain the index method within a data structure.
* Illustrate loops in Python.
* Explain error handling in Python along with the try and except blocks.
* Identify common error types.
* Describe Python’s file system structure.
* Illustrate the main file opening methods and permission modes in Python.
* Define import modules in Python.
* Illustrate the OS system and OS popen functions.
* Define functions in Python.
* Explain how to create and call a function.

### Lesson Activities and Teaching Strategies

|  |  |  |
| --- | --- | --- |
| Estimated Time | Lesson Portion | Directions |
| 2 min | **Lesson Opener:**  Python Programming Essentials​ | * Introduce learners to the importance of Python programming essentials​ in cybersecurity. |
| 5 min | **Real World Scenario:**  Python Programming Essentials​ | * Review the real world scenario challenge and inform learners that you will be constantly coming back to this scenario throughout the lesson to discover how to solve and apply concepts to this real situation. |
| 2 | **Lesson Companion:** Python Programming Essentials​ | * Review the lesson companion, and inform learners that you will be constantly coming back to this scenario throughout the lesson to discover how to solve and apply concepts to this real situation. |
| 20 min | **Cyber Uncovered:**  Data Types and Loops​ | * Introduce the concept of data structures as organizational tools for data storage and management. * Explain that Python offers primary data structures like lists, tuples, and dictionaries for various data storage needs. * Introduce lists as ordered collections of items and explain their flexibility in data management. * Discuss how items can be added, removed, or modified within lists. * Explain that tuples are similar to lists but immutable, making them suitable for storing constant data. * Highlight the importance of immutability in certain situations. * Describe dictionaries as collections of labeled items with keys and corresponding values. * Emphasize their efficiency in lookup operations. * Illustrate how data structures not only store data but also facilitate operations like searching, sorting, and iterating. * Discuss how these operations optimize algorithmic performance. * Explain the index method's dual functionality: Locating an item and retrieving an item. * Provide examples to demonstrate its practical use. * Introduce index slicing as a method to retrieve multiple items using the colon (:) notation. * Explain how negative index numbers access elements from the end of the data structure. * Dive deeper into lists by discussing their mutable nature and how they can store various data types. * Highlight the flexibility of lists for different data storage needs. * Explore various list operations, such as adding elements using append() and insert(), and removing elements using del() and remove(). * Provide an in-depth understanding of dictionaries, emphasizing their key-value pair structure and versatility. * Explain how dictionaries can store different data types. * Discuss adding, deleting, and updating key-value pairs within dictionaries. * Show examples of these operations for practical comprehension. * Introduce the concept of loops as control structures for automating repetitive tasks. * Differentiate between for and while loops. * Explain the for loop's versatility in automating code execution. * Describe how it can iterate over sequences and iterable objects. * Discuss the range() function for controlled repetition and provide an example. * Discuss the while loop's use in executing code as long as a specified condition remains true. * Emphasize the importance of updating loop conditions to avoid infinite loops. * Showcase the combination of loops and conditions for more efficient coding. * Explain how conditions can be used with both while and for loops. * Be prepared to discuss the implication of the real world scenario presented at the beginning of class on Data Types and Loops. There are specific prompts that you should ask learners to reflect on to apply this concept to the real world scenario. |
| 5 | **Real-World Scenario** | * Review the real world scenario challenge and inform learners that you will be constantly coming back to this scenario throughout the lesson to discover how to solve and apply concepts to this real situation. |
| 10 min | **Lab:**  Data Types and Loops​ (While you Wait) | * Remind learners to use this lab to practice and apply the concepts they have learned throughout the day. * Learners will receive direct feedback on their lab to properly assess their knowledge and determine where they might need additional assistance. * Note: This lab is referred as “While you Wait” in TDX Arena |
| **5 min Break** | | |
| 20 min | **Cyber Uncovered:**  File System and Error Handling​ | * Start by explaining the importance of error handling in Python. * Explain common types of errors in programming * Emphasize that errors are not just roadblocks but also provide valuable information to guide programmers. * Present the code snippet 'div = num1/num2' that triggers a ZeroDivisionError. * Discuss the consequences of such an error and the need to handle it appropriately to prevent program crashes. * Introduce the try block as the testing ground for potentially error-causing code and the except block as the safety net to catch and handle errors gracefully. * Explain that this prevents the entire program from crashing. * Describe common error types in Python, including SyntaxError, NameError, TypeError, ZeroDivisionError, FileNotFoundError, and IndentationError. * Explain what each error signifies and when it might occur in coding. * Explain how Python specifies which error to catch in the except block. * Discuss scenarios where targeted error handling is useful, enabling different actions based on the encountered error. * Introduce the except Exception block as the last line of defense that can catch any error, known or unknown. * Emphasize its usefulness when the types of potential errors are uncertain. * Provide an example where the code attempts to add an integer and a string, which raises a TypeError. * Show how the except Exception as e block catches this error and prints a general error message. * Transition to file handling and explain that Python provides built-in modules for working with files, eliminating the need for external libraries. * Differentiate between text and binary files. * Discuss the two primary file opening methods in Python: The traditional open-close method and the with-as method. * Explain the importance of closing files to free up system resources. * Describe common permission modes, such as 'r' for reading, 'w' for writing, and 'a' for appending. * Highlight the differences and potential exceptions that can occur when using these modes. * Explain the read(), write(), and append() permissions and their functions when working with text files. * Provide code snippets to demonstrate how to read, write, and append to files. * Be prepared to discuss the implication of the real world scenario presented at the beginning of class on File System and Error Handling​. There are specific prompts that you should ask learners to reflect on to apply this concept to the real world scenario. |
| 15 min | **Lab:**  File System and Error Handling​ | * Remind learners to use this lab to practice and apply the concepts they have learned throughout the day. * Learners will receive direct feedback on their lab to properly assess their knowledge and determine where they might need additional assistance. |
| 5 | **Pulse Check** | * Before you launch the pulse check, explain each section clearly, and encourage the learners to participate in the survey. * After administering the survey, share the poll results with learners and ask learners to provide feedback * Encourage learners to attend office hours with the associate instructor. |
| **5 min Break** | | |
| 20 min | **Cyber Uncovered:**  Import Modules and Functions | * Begin the lesson by explaining the concept of importing libraries or modules in Python. * Emphasize that modules contain prewritten code that simplifies specific tasks and functions, acting like specialized tools borrowed from a toolbox. * Discuss the significance of importing modules at the beginning of Python scripts to make their functions available throughout the code. * Introduce the Python OS module and its role in interacting with the operating system. * Highlight the versatility of the OS module, which supports commands for various operating systems, including Windows, Linux, and macOS. * Explain the os.system() function, which enables the execution of system commands and shell operations. * Caution students about potential security risks related to this function, such as privilege escalation. * Describe the os.popen() function, which is similar to os.system() but offers additional capabilities. * Explain how os.popen() allows for capturing the output of executed commands for further processing or analysis. * Shift the focus to Python functions, also known as methods, procedures, or subroutines. * Define functions as blocks of code designed to perform specific tasks and highlight their role in organizing and simplifying code. * Explain the process of creating a function using the def keyword, function name, parentheses, and colon. * Use an example to illustrate how to define a simple function, emphasizing that defining a function doesn't execute its code. * Describe how to execute a function by calling it with its name followed by parentheses in the main code. * Show how this triggers the function and runs the code within its body, thus performing the designated task. * Be prepared to discuss the implication of the real world scenario presented at the beginning of class on import modules and functions. There are specific prompts that you should ask learners to reflect on to apply this concept to the real world scenario. |
| 10 min | **Lab:**  OS Utility Library | * Remind learners to use this lab to practice and apply the concepts they have learned throughout the day. * Learners will receive direct feedback on their lab to properly assess their knowledge and determine where they might need additional assistance. |
| 15 | **Lesson Closure** | * Encourage learners to read ahead of time * Provide learners additional resources to read / practice and assign homework (e.g., future labs) before you demonstrate the labs during the next class * Spend some time to highlight what are the key takeaways from today’s lesson * Important topics covered during the class includes   + Explain the key takeaway of Python Data structures such as list, tuple, and dictionaries   + Provide high-level summary of index methods, index slicing, and various loops such as for loop, and while loop   + Provide key takeaway of error handling in Python using try and except block   + Provide a summary of error handling types such as ZeroDivisionError, NameError, FileNotFound Error, etc.   + Provide the use cases and takeaway for os.popen() function, import module, and os modules in python |
|  | Add Additional Time Filler | * Review using Kahoot or other similar platforms * Conduct interview preparation conversations * Continue discussions on real-world scenarios * Demonstrate how to create users in Linux and grant them permissions * Discuss different career paths in cybersecurity and highlight the roles that require Linux skills |

### 