## LESSON: Introduction to Python

## This is the beginning of the Python Course (1 of 2), and instructors should ensure that the students understand the basics of Python, and provide demonstrations of how variables, data types, work in Python, and common mathematical operators. 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

* Check-in with the students to see if they have any questions or need further clarification from any subject from the last class and self-study module.
* 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 and student engagements of the labs are 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 explore Python's dynamic and versatile nature, known for rapid development and user-friendly learning. They'll discover its pivotal role in cybersecurity, including malware analysis and OS functions. Python's emphasis on code readability, highlighted as a primary implementation of conditions through the 'if' statement, will be covered. The lesson introduces operators for data manipulation and comparison, providing a solid foundation for learners to harness Python's capabilities in various domains, including cybersecurity.

### Objectives

* Define Python.
* Recognize the role of Python in cybersecurity.
* Explain what an interpreter is and compare it to a compiler.
* Define variables in Python.
* Identify the main data types and their syntaxis.
* Explain the main rules of code execution.
* Illustrate the print() and input() functions.

### Lesson Activities and Teaching Strategies

|  |  |  |
| --- | --- | --- |
| Estimated Time | Lesson Portion | Directions |
| 2 min | **Lesson Opener:**  Introduction to Python | * Introduce learners to the importance of Python 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:**  Python Fundamentals​ | * Introduce Python as a dynamic programming language known for its rapid development capabilities and ease of learning and development. * Discuss its versatility in developing various applications. * Explain Python's widespread use in cybersecurity due to its powerful capabilities, including malware analysis, OS functions, shell command execution, and sniffing and spoofing techniques. * Emphasize its design principles that prioritize code readability. * Describe how penetration testers utilize Python's IDE (Integrated Development Environment) for executing custom-made scripts in cybersecurity, such as vulnerability testing and malware analysis. * Highlight the advantages of using an IDE over a command-line interface (CLI) for software development. * Explain that Python is an interpreted language, where the interpreter converts source code to machine code one statement at a time during runtime. * Clarify that the Interpreter can be run directly from the command line using the 'py' command or installed within an IDE. * Introduce the concept of a compiler as a software tool used in programming to translate high-level code into lower-level machine code. * Explain the process of compilation, where high-level source code is converted into a format that a computer's central processing unit (CPU) can execute directly. * Compare and contrast interpreters and compilers based on key aspects, such as execution, error detection, speed, debugging, and portability. * Highlight the advantages and disadvantages of each approach, emphasizing the role of the Python Interpreter in the context of this lesson. * **Give learners a 5 minute break.** * Be prepared to discuss the implication of the real world scenario presented at the beginning of class on network types and devices. There are specific prompts that you should ask learners to reflect on to apply this concept to the real world scenario. |
| 2 | **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. |
| **5 min Break** | | |
| 20 min | **Cyber Uncovered:**  Python Data Types​ | * Explain that the purpose of variables in computer programming is to store information for reference and manipulation. * Emphasize how variables provide descriptive names to data, making programs more understandable. * Discuss Python's ability to automatically detect data types based on how values are specified. * Explain that different data types have different syntax and how Python handles them. * Cover the rules for naming variables, including starting with a letter or underscore, disallowing numbers as the first character, and avoiding special characters. * Describe the process of defining variables in Python, which involves creating symbolic names associated with specific values or data objects. * Provide an example using variables like "name" and "age" to store a string and an integer. * Provide a demonstration on how to assign a variable, storing data in a variable and print the information using a print() function * Highlight the importance of variables in storing and managing data within Python programs. * Explain that variables enable the performance of calculations, decision making, and efficient data manipulation. * Discuss Python's dynamic typing feature, which eliminates the need to explicitly declare data types for variables. * Explain how data types are determined automatically based on assigned values. * Introduce the primary data types in Python, including integers, floats, booleans, and strings. * Provide explanations and examples for each data type. * Cover essential code execution rules, including the use of comments to clarify code, Python's case sensitivity, and the role of colons, indentations, and keywords in statements. * Explain the purpose of the print() function in Python, which serves to display information during program execution. * Introduce the concept of f-strings and how they allow embedding variable values into strings. * Discuss the importance of user input and the input() function for interactive programs, along with the need to convert input data to the appropriate numerical type when necessary. |
| 20 min | **Lab:**  Python Data Types​ | * 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:**  Conditions and Arithmetic Operations​​ | * Start the lesson by explaining the significance of conditions in Python and how they control program flow based on certain criteria, either evaluating as True or False. * Introduce the if statement as the primary tool for implementing conditions, emphasizing its role in executing code blocks when conditions are met. * Use visual aids and examples to illustrate condition structures, making it clear how conditions are formulated and when they lead to specific actions. * Transition into discussing Python operations, highlighting that operations involve operators and operands to produce results. Provide clear definitions for operators, operands, and operations. * Present the variety of Python mathematical operators, such as +, -, \*, /, //, \*\*, and %, and explain their functions and typical use cases. Use real world examples to make concepts relatable. * Offer hands-on practice for learners to experiment with different mathematical operations in Python, encouraging them to calculate and display results. * Introduce Python comparison operators, explaining that they assess relationships between variables or values and return either True or False. * Provide clear explanations and examples for each comparison operator, including >, <, <=, >=, ==, and !=. * Be prepared to discuss the implication of the real world scenario presented at the beginning of class onConditions and Arithmetic Operations. 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:**  Conditions and Arithmetic Operations | * 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. |
| 20 min | **Lab:**  Conditions and Arithmetic Operations​​ | * 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   + Provide a summary of the key takeaway for Python Cybersecurity emphasizing on its various frameworks and OS functions   + Summary of comparison between interpreted and compiled programming language   + Key takeaway of Python Data Types such as variables, int, float, boolean, string and the use cases   + Provide an overview of F-string, input function, and and indentation in Python   + Main takeaways for mathematical operations, comparison operators, conditional statements such as if, else, and how they used to make decisions in code based on whether certain criteria are met. |
|  | 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 |

### 