Lesson Plan

# Lesson 04: Data Structures 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Lesson Overview** | | | |
| **Lesson Length** | 3 hours (180 minutes) | | |
| **Overview** | This session will familiarize students with data structures used within the Python scripting language. | | |
| **Objectives**   * **Domain:** Cognitive * **Level:** Applying | Using conditionals, loops, Python dictionaries, local data, the CSV Library, and the Glob Library, students will be able to:   * TLO 1: Examine the implications of using computation to solve a problem   + ELO 1.1: Discuss best practices for using computation to solve a problem   + ELO 1.2: Suggest types of problems that can be solved through computation   + ELO 1.3: Show how computation can solve a problem * TLO 2: Recognize key computer science concepts   + ELO 2.2: Identify data structures used in Python scripting | | |
| **Instructional Methods** | Informal lecture, demonstration, guided discussion, practical exercises | | |
| **Assessment Strategies** | Informal: Guided and practical exercises  Formal: N/A | | |
| **Materials and Equipment Needed** | Required:   * SBU * Jupyter Notebook * Python   Optional:   * N/A | | |
| **Background Resources** | Resource:   * NGA SME * Technical facilitators   Subject matter/content questions may be referred to:   * Jeremy DeBrow, Course Manager   [Jeremy.R.Debrow@nga.mil](mailto:Jeremy.R.Debrow@nga.mil)  [Jeremy.R.Debrow@coe.ic.gov](mailto:Jeremy.R.Debrow@coe.ic.gov)  National Geospatial-Intelligence College (NGC) HDNPE Branch  Unclassified: 571-557-7583 | | |
| **Comments** | ELOs 1.1, 1.2, and 1.3 are meant to be covered throughout the entire lesson (informal lecture and assessment). Instructors will be expected to facilitate classroom discussion that identifies problems best suited to be solved computationally, best practices for solving those problems, and potential solutions. | | |
| **Lesson Sequence** | | | |
| **Lesson Topic** | **Instructional Method** | | **Time**  **(mins)** |
| Introduction | Informal lecture, guided discussion | | 05 |
| Review | Informal lecture, guided discussion, demonstration | | 10 |
| Lesson: Data Structures | Informal lecture, guided discussion, demonstration | | 60 |
| Assessment | Guided exercise, practical exercises | | 100 |
| Conclusion | Informal lecture | | 05 |
| **Lesson Outline** | | | |
| **Introduction** | | | |
| Introduction | * **Attention** (to be personalized by instructor) * **Motivation** (to be personalized by instructor) * **Overview** (to be personalized by instructor)   + Learning objectives   + Lesson topics/main points * **Rules of Engagement** (to be personalized by instructor) | | |
| **Body** | | | |
| ***Lesson Topic*** | ***Main Points/Notes*** | ***Personalization*** | |
| Review | **Informal Lecture/Guided Discussion/Demonstration (10 minutes)**  **Lesson 4: Data Structures**  **4.1. Objectives**  **4.2. Overview**  *(Facilitator Notes:*   * *Have students load the U\_CSCI2011\_L04\_Data\_Structures\_1\_SG\_V3.0.ipynb file to begin the lesson.)* * *See instructor’s notebook for instructor guidance for discussion points and guided exercises.* * *Refer back to Lesson 1 and relate the four steps of problem-solving using Computational Thinking (Decomposition, Pattern Recognition, Abstraction, & Algorithm Design) to lessons, exercises, examples, student questions/comments, etc., as appropriate throughout this lesson.)*   **4.3. Review**   * 4.3.1. Boolean Expressions * 4.3.2. If Statements * 4.3.3. String Manipulations * 4.3.4. Numbers and Casting |  | |
| Lesson: Data Structures | **Informal Lecture/Guided Discussion/Demonstration (60 minutes)**  **4.4 Lesson: Ordered Sequences**   * 4.4.1. Sequences * 4.4.2. Lists   + 4.4.2.1. Creating Lists   + 4.4.2.2. List Methods and Operations * 4.4.3. Tuples   + 4.4.3.1. Creating Tuples   + 4.4.3.2. Mutable vs. Immutable * 4.4.4. Indexing and Slicing * 4.4.4.1. Indexing * 4.4.4.2. Slicing * 4.4.5. Common Operations Across Data Structures |  | |
| Assessments | **Guided Exercise/Practical Exercise (100 minutes)**  *(Facilitator Note: All assessments should incorporate a facilitator directed discussion on computational thinking techniques as they relate to the assigned problem. Utilize student handouts, performance support tools, or projected code cells to capture facilitator/student discussion.)*  **4.5. Guided Exercise: Employee Hours**  *(Facilitator Note: Refer back to Lesson 1 and relate the four steps of problem-solving using Computational Thinking (Decomposition, Pattern Recognition, Abstraction, & Algorithm Design) as appropriate throughout these exercises.)*   * You manage a team of employees, and store vital business data in Python data structures. For example, you have a list of employees and how many hours they have worked so far this year. That list is below.What dates do we have data for? * Problem 1: You want to sum up the total number of ours your team has worked this year. This means you need to access the hour numbers. Access the hours for the first employee in the list. * Problem 2: Create an empty list to hold on to all the hours numbers. Then, add the number of hours for each employee to the list. * Problem 3: Use the sum() function to find the total number of hours your team worked. * Problem 4: Use the len() function to count how many employees are on your team, and use that number and the total number of hours you found in problem 4 to calculate the average number of hours worked per employee. * Problem 5: The company is going through some employee turnover. Rasel and Emilia have quit and need to be debriefed. Create a new list to hold the names of employees who need debriefing, then remove them from the current employee list and add them to the debriefing list.   **4.6. Practical Exercises**  *(Facilitator Note:*   * *Refer back to Lesson 1 and relate the four steps of problem-solving using Computational Thinking (Decomposition, Pattern Recognition, Abstraction, & Algorithm Design) as appropriate throughout these exercises.* * *The practical exercises deemed most important due to content and/or a cumulative result, which should be completed first in the interest of maximum training value in relation to time are Practical Exercises 1, 3, and 5. Ensure you go over the exercise solutions and (as necessary) the processes to arrive at the solutions with the students.* * *Follow-up questions are designed to be asked by the facilitators individually as each student completes the task and has it looked at by a facilitator.*   **4.6.1. Practical Exercise 1: Sequence Questions**   * Problem 1: List the types of sequences * Problem 2: What is an index? What is a negative index? When would you use it? * Problem 3: What is slicing in your own words? * Problem 4: What are the actions you can do with lists? * Problem 5: What is the difference between the append and extend functions in your own words? * Problem 6: What is the difference between immutable and mutable? * Problem 7: What data types can lists and tuples hold?   **4.6.2. Practical Exercise 2: Practice Lists**   * Problem 1: Create two lists of strings and save them to variables. * Problem 2: Use list methods to combine the lists of strings to create one, big list of strings. * Problem 3: Find how many elements are in the list. * Problem 4: Retrieve one item from the list using its index location. * Problem 5: Now that we have a list of strings, combine that into one string with spaces in between each word.   **4.6.3. Practical Exercise 3: Practice Tuples**   * Problem 1: It's common to store coordinates in tuples because they are made up of two elements: a latitude and a longitude. How many coordinates are in the list below? Don't count them, use len() to find the answer. * Problem 2: Create your own coordinate and store it in a tuple. Then, add your new coordinate to the list of coordinates above. * Problem 3: Dates are also made up of multiple elements (year, month, day). Thus, tuples are good data structures to store date information. Below is a tuple that holds a date. Create your own to store a different date, storing the year as the first element, the month as the second, and the day as the third.   **4.6.4. Practical Exercise 4: Negative List Index**   * Write a script that takes a list and an index (int) as inputs, and outputs the corresponding negative index. HINT: The negative index will depend on the list's length. * Bonus: Before you find the negative index, first check if the input index is valid. If not, print a helpful error message.   **4.6.5. Practical Exercise 5: Dates and Tuples**   * Problem 1: Write some code to determine whether two tuples that hold date information come from the same year. Assume the year data is the first element in the tuple. * Problem 2: Extend the code you wrote above to print the number of months between the two dates if the dates come from the same year (simply find the difference between the two month numbers). If the dates are from different years, print a message saying that.   **4.7. Appendix** |  | |
| Administrative Notes | N/A |  | |
| **Assessment** | | | |
| ***Assessment Type*** | ***Instructions/Prompts/Notes*** | | |
| Guided Exercise | See the facilitator notes located above for additional guidance. All exercises will be conducted inside the Jupyter Notebook lesson file. | | |
| Practical Exercise | See the facilitator notes located above for additional guidance. All exercises will be conducted inside the Jupyter Notebook lesson file. | | |
| **Conclusion** | | | |
| Conclusion | * **Final Summary** (to be personalized by instructor)   + Review learning objectives   + Review lesson topics/main points * **Remotivation** (to be personalized by instructor) * **Closure** (to be personalized by instructor) * **Next Lesson Introduction** (to be personalized by instructor) | | |