Lesson Plan

# Lesson 06: Loops

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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**  **(min)** | |
| Introduction | Informal lecture, guided discussion | | | 05 | |
| Review | Informal lecture, guided discussion, demonstration | | | 15 | |
| Lesson: Flow Control | Informal lecture, guided discussion, demonstration, guided exercise | | | 80 | |
| Assessment | Guided exercise, practical exercises | | | 75 | |
| 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 (15 minutes)**  **Lesson 4: Loops**  **4.1. Objectives**  **4.2. Overview**  *(Facilitator Notes:*  *Have students load the U\_CSCI2011\_L06\_Loops\_SG\_V3.0.ipynb file to begin the session.)*  *See instructor’s notebook for instructor guidance for discussion points and guided exercises.)*  **6.3. Review**   * 6.3.1. Lists * 6.3.2. Sets * 6.3.3. Dictionaries | |  | | |
| Lesson: Loops | **Informal Lecture/Guided Discussion/Demonstration (80 minutes)**  **6.4. Lesson: Loops**  **6.4.1. For Loops**   * 6.4.1.1. Temporary Variables * 6.4.1.2. Looping Through Other Collections * 6.4.1.3. Looping through dictionaries with .items() * 6.4.1.4. Counting with a Dictionary   **6.4.2. While Loops**   * 6.4.2.1. Infinite Loops   **6.4.3. Break and Continue** | |  | | |
| Assessment | **Guided Exercise/Practical Exercise (75 minutes)**  *(Instructor 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..)*  **6.5. Guided Exercise: Credit Card Data**   * We have been given data about credit card users in a dictionary. Each key is a user's name, and linked to that is another dictionary containing their credit card type and the number of transactions using their card. We want to accomplish the following.   + Problem 1: Determine what types of cards are in the data set.   + Problem 2: Create a dictionary that maps the card types to the number of people in the data set who use that card.   + Problem 3: Find the card type in the data set with the most users.   **6.6. Practical Exercises**  *(Facilitator Notes:*   * *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, 2, 3, and 4. 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.)*   **6.6.1. Practical Exercise 1: Iterating through a List**   * Given a list of the integers 1 through 100, iterate through the list and print only those integers divisible by 3 and 5 (e.g. 15, 30, 45...).   **6.6.2. Practical Exercise 2: Latitude or Longitude**   * Given the following list of geocoordinates in DMS format. Count how many latitudes and how many longitudes are in the list. HINT: You can determine whether the coordinate is a latitude or a longitude by its length or its direction character ('N', 'E', 'S', 'W')   **6.6.3. Practical Exercise 3: Lists in a List**   * Problem 1: Using indexing, access the value 'here' in the list of lists below. * Problem 2: Using indexing, access the value 'here' in the list of lists below. * Problem 3: Using indexing, access the value 'here' in the list of lists below. * Problem 4: Using a nested for loop, access each number in the list below. For each number, if it's a 4, print it out; otherwise do nothing. * Problem 5: Using a nested for loop, calculate the sum of all the numbers in the list below. HINT: The answer is 76.   **6.6.4. Practical Exercise 4: Sets in a List**   * You are given a list of sets below. Create the union of every set that has exactly 1 or exactly 2 elements.   **6.6.5. Practical Exercise 5: Crime Data**   * Problem 1: Loop over the dictionary below. Print out each date along with how many crimes were reported on that date. * Problem 2: Loop over the dictionary below. Count the number of times each coordinate appears in the data set. Store this information in a dictionary where the coordinate is the key and the appearance count is the value. * Problem 3: Using the dictionary you just created, programmatically find the coordinate that appeared most in the data set. HINT: The answer is (39.29, -76.65).   **6.6.6. Practical Exercise 6: Shortest and Longest Words**   * Problem 1: Use a loop to find the length of the shortest word in the set below. HINT: The answer is 2. * Problem 2: Use a loop to find the length of the longest word. Append all the words that have that length to a list. HINT: Your final list should have two words in it.   **6.6.7. [Challenge] Practical Exercise 7: Digit Degree**   * Let's define digit degree of some positive integer as the number of times we need to replace this number with the sum of its digits until we get to a one digit number. Write a script to find the digit degree of any positive integer.   **6.6.8. [Challenge] Practical Exercise 8: Common Character Count**   * Given two strings find the number of common characters between them (case sensitive). Count the total number of matches, not just the unique characters, for example if the two strings are 'aaa' and 'aa' the answer should be 2.   **6.6.9. Practical Exercise 9: Guessing Game**   * Create a game to guess the number that the computer randomly generates. Generate a random number between 1 and 9 inclusive using random.randint(). For more information, see Python: Functions for sequences. Enter your own number, then have computer print out if you were too high, low, or you got it right. The code below will get you started. It initializes a random number and saves a guess. | |  | | |
| 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) | | | | |