(http://csmatters.org) 2 - 16

0b10 - 0b10000

Lists: Creation, Traversal, Insertion, and Removal

Unit 2. Developing Programming Tools

Revision Date: Jan 12, 2020
Duration: 6 50-minute sessions

Lesson Summary

Summary

This lesson answers the question "What is a list and what can I do with one?". Students will find the answer to this question by exploring list operations and methods, as well as investigating how lists are modeled in real-world situations.

Outcomes

- Students will create lists, access, and traverse elements within the list.
- Students will be introduced to list syntax on the exam reference sheet.
- Students will perform list operations including insertion, concatenation, repetition, slices, and deletion.
- Students will be introduced to list methods including append, insert, pop, sort, reverse, index, count, and remove.
- Students will explore the Python API.
- Students will use lists to model several real-world situations.

Learning Objectives

CSP Objectives

- *EU DAT-1 - The way a computer represents data internally is different from the way the data is interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.*
  - LO DAT-1.A - Explain how data can be represented using bits.
- *EU AAP-1 - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.*
  - LO AAP-1.C - Represent a list or string using a variable.
  - LO AAP-1.D - For data abstraction: a. Develop data abstraction using lists to store multiple elements. b. Explain how the use of data abstraction manages complexity in program code.
- *EU AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.*
  - LO AAP-2.N - For list operations: a. Write expressions that use list indexing and list procedures. b. Evaluate expressions that use list indexing and list procedures.
  - LO AAP-2.O - For algorithms involving elements of a list: a. Write iteration statements to traverse a list. b. Determine the result of an algorithm that includes list traversals.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.
- MP7: Look for and make use of structure.

Common Core Math:

- F-IF.1-3: Understand the concept of a function and use function notation
- F-IF.4-6: Interpret functions that arise in applications in terms of the context

Common Core ELA:

- WHST 12.5 - Develop and strengthen writing as needed by planning, revising, editing, rewriting
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 2. Developing and using models

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- "Warm-Up Get Ready for Lists"
- "LinkyListy Role Play"

Other:

- Python API for Built-in Types (https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range) - https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range (https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range)
- http://interactivepython.org/runestone/static/thinkcspy/Lists/toctree.html (http://interactivepython.org/runestone/static/thinkcspy/Lists/toctree.html)
- Original collegeboard horse barn question #3: http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf (http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf)

Lesson Plan

Session 1

1.1 Getting Started (5 min)

Journal:

- When did you last make a list?
- What did you call the list?
- How long was the list?

Solicit some responses. Point out that different lists contain different kinds of information.

Say: In programming, we use data collections like lists for much the same reason as you use lists. A list is a sequential collection of data values that can be referenced using a single name and each value in the list is identified by an index. In some other programming languages, lists can be referred to by different names, such as an array. Today we are going simulate the way Python implements lists.

1.2 Guided Activities (40 min)

Optional: Review the string commands. Use the Lists.docx found in the Lesson Resources folder.

1.2.1 List Role Play [20 min]

See the file called LinkyListy Role Play in the Lesson Resources folder. Note: You can change student names to students within your class.

Create a human “list of students”. See the file called LinkyListy Role Play in the Lesson Resources folder. Note: The directions below are an example. Change student names to students within your class.

Create a List of students by starting with an empty list and modify the list by following program below.

Designate a section of the board or a poster to act as the console/printer for output and a volunteer student to act as the printer driver.

Designate an area at the front of the room for the computer memory (and future students).

As students are called by the program, have each student come up to stand in front of the room where memory resides.
Have each student point to the student that follows them to form a linked list as they come to the front.

LinkyListy Role Play Program:

```
students = []
students.append('Joe')
students.append('Pat')
students.append('Alea')
students.append('Marta')
print(students)
#add additional commands to append or insert 4 or 5 more students
print(students)
print(len(students))
print(students[3])
students.reverse()
print(students)
print('David' in students)
students.sort()
print(students)
more = ['Tom', 'Laverne']
students = students + more
print(students)
pets = ['fish'*3,'dog']
del students[1]
del students[2:4]
students.insert(0, 'Jennifer')
students = students + pets
print(students)
print(students.index('Marta'))
```

1.2.2 Check for understanding:

Ask students to give an example and then explain the effect of several of the List methods. For example, students.append(“Zoe”) would add Zoe to the end of the list of students. students.insert(4, “Larry”) would add Larry at index position 4 and slide everyone else down one slot. This could be done as a placemat activity. Use different colored markers for each student to write the example. Turn placement and ask the next student to explain the effect.

Notes:

- Length of a `list: len(students)`
- Accessing Elements: `print(students[2])`
- List Membership (in, not in): `print(“Mary” in students)`
- Concatenation and repetition (+ *):Make a list of more Python commands on sentence strips. On back of the sentence strip, write the output or trace of the statement after it has executed onto the back of the card. Have one student hold the deck of cards and cycle through them. Ask students to read a card and predict the output. Meanwhile, students who are in the list will

change positions to demonstrate the behavior within the list. The holder of the card provides feedback or congratulatoins in checking the correctness of classmate's responses.
- Include cards that model these behaviors:

```
more = ["Tom", "Laverne"]
students = students + more
pets =['fish'*3,'dog']*2          # creates a list of ['fishfishfish','dog','fishfishfish','dog']
```

- List deletion using the delete operator: `del students[2], del students[2:4]`
- List deletion using the List method pop: `stu = students.pop(2)` , `stuLast = students.pop()` . The pop method pops (deletes and returns) an element at a given index or the last element if no index is provided.
- Extension: additional list methods that may be useful and interesting: sort, reverse, index, count.

1.3 Access Python Documentation [10 min]

- Check out the Python docs for Built-in Types.
- Go to Section 4.6 Sequence types- list, tuple, range, and check out the sections on (4.6.1) Common Sequence Operations, (4.6.3) Mutable Sequence Types, and (4.6.4) Lists.
- Students need to realize that Application Programmer Interfaces (APIs) are an important resource for programming.
- Identify list methods found in the API. (Link to resource: Python API for Built-in Types (https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range))

1.4 List Slices [10 min]

- Quickly resurrect your list of students.
- This time, write names on the board.
- To simulate list slices, add each student's age and hair color after each student's name.
- Yes, lists can contain different kinds of objects!
- Ask students to copy the list and write the code to pull out a sublist that contains the second student's name and info.

Exam Reference Sheet

Say:

- List procedures are implemented in accordance with the syntax rules of each programming language. Textual list procedures on the exam reference sheet use a similar syntax to that used by Python.
- The exam reference sheet describes a list structure whose index values are 1 through the number of elements in the list, inclusive.
- For all list operations on the exam reference sheet, if a list index is less than 1 or greater than the length of the list, an error message is produced and the program will terminate.

aList ← [value1, value2, value3,...] creates a new list that contains the values value1, value2, value3, and … at indices 1, 2, 3, and … respectively and assigns it to aList.
aList ← [ ] creates a new empty list and assigns it to aList.
aList ← bList assigns *a copy of the list bList* to the list aList.
APPEND(aList, value) increases the length of aList by 1, and value is placed at the end of aList.
REMOVE(aList, i) removes the item at index i in aList and shifts to the left any values at indices greater than i. The length of aList is decreased by 1.
LENGTH(aList) evaluates to the number of elements currently in aList.
List traversal - all elements:

FOR EACH item IN aList {

<block of statements>

}

The variable *item* is assigned the value of each element of aList sequentially, in order, from the first to last.

1.5 Wrap Up (5 min)

Discussion: How are lists the same and how are they different from Strings?

Homework

Have the students do the Runestone lab activities for Lists to reinforce the above concepts.
(http://interactivepython.org/runestone/static/thinkcspy/Lists/toctree.html
(http://interactivepython.org/runestone/static/thinkcspy/Lists/toctree.html))

Session 2

2.1 Getting Started (5 min)

- Create a list that contains the integers from 1 to 10.
- Create the same list from an empty list using a loop.
- Check for understanding: create a list that counts down from 10 to 1 and then adds “Blastoff”.
- Reverse a given list.

2.2 Guided Activities (40 min)

2.2.1 Lists are Mutable [20 min]

Demonstrate that lists are mutable with the following activity

- Create a list representing a horse barn containing 4 horses in 6 stalls.
- Use the built-in constant `None` for the empty stalls.
- Find an empty stall.
- If there is an empty stall, place “Toby” in the Barn in that slot.
- Also. see if a particular horse is in the barn or if he is out to pasture.
- Exchange 2 horses in the barn. (Adapted from 2012 APCS exam question 3, page 13 of
  http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf
  (http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf))

2.2.2 Paired Programming [20 min]

- In a group of 2-3, make a zoo of animals.
- Demonstrate the use of at least 6 different list operations and methods.
- For a bonus, try to make a story with your code.

2.3 Wrap Up (5 min)

Discuss what kind of information can be acquired when a program processes data.
Describe some possible tables, diagrams, text, or other visual tools that could be used to communicate insight and knowledge gained from data.

Homework

Create a list of 5 students that contain the students' name, age, and hair color. Use a loop to extract the information for each student and print it out.

Teacher Note: Consider using the YumYumCupcake problem (see Formative Assessment) as part of tomorrow's opening exercises.

Session 3

3.1 Getting Started (5 min)

Introduction

The first of two sessions creating and traversing lists in Python and EarSketch.

Discussion

Discuss: Lists are collections of values combined into a single entity. What are some lists or other data collections used by music apps? How do the lists make the programs more appealing to their users?

3.2 Guided Activities (40 min)

Say: Today we will be learning to implement lists like those we acted out or use in our music apps.

6. Chapter Six: Lists (http://earsketch.gatech.edu/itec2120/book.html#_chapter_six_lists)

- 6.1. Creating Lists - PythonTutor Asg. 6.1
- 6.2. Indexes

- 6.3. Accessing and Modifying List Values
    - PythonTutor Asg. 6.2
    - EarSketch Example 6.1
    - EarSketch Asg. 6.1
    - EarSketch Example 6.2
    - EarSketch Asg. 6.2
- 6.4 List Traversal
    - PythonTutor Asg. 6.4 (http://earsketch.gatech.edu/itec2120/book.html#_pythontutor_asg_6_4)
    - EarSketch Example 6.3 (http://earsketch.gatech.edu/itec2120/book.html#_earsketch_example_6_3)
    - PythonTutor Asg. 6.5 (http://earsketch.gatech.edu/itec2120/book.html#_pythontutor_asg_6_5)
    - EarSketch Example 6.4 (http://earsketch.gatech.edu/itec2120/book.html#_earsketch_example_6_4)

3.3 Wrap Up (10 min)

Modify EarSketch Asg. 6.2 (http://earsketch.gatech.edu/itec2120/book.html#_earsketch_asg_6_2) by adding a fourth music clip to the drums list. Modify the call to the makebeat function so it always uses the last music clip in drums. Test your program by adding a fifth music clip to drums.

Session 4

4.1 Getting Started (5 min)

Introduction

The second of two sessions creating and traversing lists in Python and EarSketch.

Discussion

Say: Lists are collections of values combined into a single entity. The collections of data in our lists yesterday were unchanging and we learned to access individual elements of the lists. Today we will use Python to add elements to a list and to obtain a part or slice of a list while the program is running.

Show the video: List appending and slicing. (https://www.youtube.com/watch?v=TE6gg1p1eV4&feature=youtu.be)

4.2 Guided Activities (40 min)

6. Chapter Six: Lists (http://earsketch.gatech.edu/itec2120/book.html#_chapter_six_lists)

- 6.4. List Appending and Slicing
    - PythonTutor Asg. 6.3
- 6.5. For Loops with Lists
    - PythonTutor Asg. 6.4
    - EarSketch Example 6.3
    - EarSketch Asg. 6.3
- 6.6. For Loops with Indexes
    - PythonTutor Asg. 6.5
    - EarSketch Example 6.4

4.3 Wrap Up (10 min)

Modify EarSketch Asg. 6.4 (http://earsketch.gatech.edu/itec2120/book.html#_earsketch_example_6_4) by adding another music clip to the clips list. Discuss with an elbow partner how the pan and vol lists change to correspond to the addition of the music clip. Make the appropriate changes to these lists in EarSketch Asg. 6.4 (http://earsketch.gatech.edu/itec2120/book.html#_earsketch_example_6_4).

Review Session 5:

5.1 Discussion:

In this session, we will review two topics required by the Create Performance Task - User Input and User Defined Functions

5.2 Topic 1 - User Input

3.6. Console User Interaction (http://earsketch.gatech.edu/itec2120/book.html#_console_user_interaction)

- PythonTutor Asg. 3.3

- EarSketch Example 3.4
- EarSketch Asg. 3.3

5.3 Topic 2 - User-Defined Functions

Say, we review defining our own custom functions. Once defined, we can use them like any other function.

The fundamental approach to solving large programming problems is breaking them down into small function definitions and then calling those functions to solve the original complex problem. You will use the strategy of breaking a problem down into smaller parts and implementing each part separately throughout your programming experience.

7. Chapter Seven: Defining Your Own Functions
(http://earsketch.gatech.edu/itec2120/book.html#_chapter_seven_defining_your_own_functions)

- 7.3. Defining custom functions
  - PythonTutor Asg. 7.1
  - EarSketch Example 7.2
  - EarSketch Asg. 7.1
- 7.4. Return Statements
  - PythonTutor Asg. 7.2
  - Return Statement Evaluation
  - EarSketch Asg. 7.2
- 7.5. Combining Functions
  - PythonTutor Asg. 7.3

Session 6

6.1 Introduction:

Today we will define functions to respond to user input with dynamic lists in EarSketch.

6.2 Warm-up:

Brainstorm some ways a user of an EarSketch program might want to affect the output produced by an Easketch program.

6.3 Guided Practice:

Choose either topic from the brainstormed list or the sample provided in User Input and Programmer Defined Functions below for students to implement. If you choose to implement suggestions brainstormed by students you may want to provide User Input and Programmer Defined Functions as an example.

Say: The exam reference sheet operations on lists include means of accessing and assigning values to a list and procedures for inserting, appending and removing list values.

1. accessing an element by index. aList[i] accesses the element of aList at index i. The first element of aList is at index 1 and accessed using the notation aList[1].
2. assigning a value of an element of a list to a variable. x ← aList [i] assigns the value of aList[i] to the variable x.
3. assigning a value to an element of a list. aList[i] ← x assigns the value of x to aList[i] and aList[i] ← aList[j] assigns the value of aList[j] to aList[i].
4. inserting elements at a given index. INSERT(aList, i, value) shifts to the right any values in aList at indices greater than or equal to i. The length of the list is increased by 1, and value is placed at index i in aList.
5. adding elements to the end of the list. APPEND(aList, value) increases the length of aList by 1, and value is placed at the end of aList.
6. removing elements. REMOVE(aList, i) removes the item at index i in aList and shifts to the left any values at indices greater.

Say: Students will have a copy of the entire exam reference sheet while they take the end of course exam from the College Board. The exam reference sheet also includes the FOR EACH item IN aList { }.

1. The variable item is assigned the value of each element of aList sequentially, in order, from the first element to the last element.
2. The code in block of statements is executed once for each assignment of item.

6.4 Wrap up:

Have students move the user input section to a fourth user defined function. Students are to describe the algorithm used in their user input function in their own words.

Options for Differentiated Instruction

- Use a canvas shoe bag to demonstrate a list.
- Use a parking lot with numbered spaces as an example of a list.

Evidence of Learning

Formative Assessment

- Create a quiz similiar to the role play. Use the answer chart provided as a template for the quiz.
- Check for understanding: Ask students to explain what the effect of the List methods append, insert, etc. For example, students.append “Zoe” would add Zoe to the end of the list of students? Students.insert(4, “Larry”). Would add Larry at index position 4 and slide everyone else down one slot?
- Check for understanding: Create a list that counts down from 10 to 1 and then adds “Blastoff”.
- Let's visit the YumYum Shoppe: Create a list that has different kinds of cupcakes and simulate the actions that might take place during a typical day. For example, `yumYumCupcakes = ["chocolate mousse" *3, "vanilla creme", "strawberry fluff", "chocolate mousse"*2].` Have a customer purchase a vanilla creme cupcake if there are any, check how many chocolate mouse cupcakes are in the display case, bake some more, and add them in. Drop one cupcake on the floor and throw it away.

Summative Assessment

Paired programming: Make a zoo of animals and demonstrate the use of at least 6 different list operations and methods. Try to make a story with your code.

(http://www.umbc.edu/)


(http://www.umd.edu/)


(http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)