Lecture 10: Introduction to Computer Programming Course - CS1010

DEPARTMENT OF COMPUTER SCIENCE | 02/19/2019

Rensselaer

Announcements

- Homework 5 Posted
- Exams will be graded by end of week
  - There is a make-up exam tomorrow
- Wednesday 12:00 pm: Make-up Exam

Goals for today

- Lists

1)Creating lists

2) Indexing and Slicing Lists

3) Basic List Methods

4) Nesting Lists

- Problems

Object Types (Lecture 2)

Name	Type (representation)	Example
Integers	int	Whole Numbers: 1, 5 , 7500
Floating Point	float	Decimal: 2.3, 4.6, 23.15
Strings	str	Ordered sequence of characters: “hello” “Sam” “2000”
Lists	list	Ordered sequence of objects: [10, “hello”, 500.5]
Dictionary	dict	Unordered Key Value pairs: {“mykey”:”Value”, “place”: “New York”}
Tuples	tup	Ordered immutable sequence of objects: (100,”Hello”, 20.5)
Sets	set	Unordered collection of unique objects: {“a”,”b”}
Booleans	bool	Logical Value: True, False

Lists

With Strings and Tuples we introduced the concept of a sequence in Python.

Lists are the most generalized form of sequence in Python.

In Python, list is a type of container, which is used to store multiple data types at the same time.

A single list may contain DataTypes like Integers, Strings, as well as Objects.

List Object Type

- Lists are ordered sequences that can hold variety of object types.
- Syntax: Square brackets []; elements separated by commas.
- [1,2,3,4]
- Can use indexing and slicing
- Are mutable: Elements inside a list can be changed
- There are a variety of methods we can use on Lists.
- Let’s explore

Indexing and Slicing

- Indexing and slicing work just like in strings or Tuples
- Python indexes from element 0 to n-1
- Slice operation is performed on Lists with the use of colon(:).
- To return elements from beginning to a range use [:Index],
- To return elements from end use [:-Index],
- To return elements from specific Index to the end use [Index:],
- To return elements within a range, use [Start Index:End Index]
- To print whole List with the use of slicing operation, use [:].
- Finally to return whole List in reverse order, use [::-1].

Basic Methods

- Lists are similar to ‘arrays’ from other languages.
- However, Python lists are more flexible because:
  - They have no fixed type
  - They have no fixed size

Some Python Methods for Lists

Function	Description
Append()	Add/Append an element to the end of the list
Extend()	Add all elements of a list to the another list
Insert()	Insert an item at a given index
Remove()	Remove an item from the list
Pop()	Remove and return an element at the given index
Clear()	Removes all items from the list
Index()	Returns the index of the first matched item
Count()	Returns the count of number of items passed as an argument
Sort()	Sort items in a list in ascending order
Reverse()	Reverse the order of items in the list
copy()	Returns a copy of the list

Nesting Lists

- One advantage of Python Data Structures is that they support ‘Nesting’
- Can have a list within a list.
- Let’s try in Spyder.

Built-In Functions with Lists

- Lists can use many built-in functions in python
- Some of the important functions are:
- sum, max, min, len
- Let’s practice in Spyder.

Problem 1

- Given an array of ints, return True if 6 appears as either the first or last element in the array. The array will be length 1 or more.
- first_last6([1, 2, 6]) → True
  first_last6([6, 1, 2, 3]) → True
  first_last6([13, 6, 1, 2, 3]) → False

Solution

- Check for length greater than 1
- Check if 6 is in the list
- Test for first and last element equal to 6

Problem 2

- Given 2 arrays of ints, a and b, return True if they have the same first element or they have the same last element. Both arrays will be length 1 or more.
-

  common_end([1, 2, 3], [7, 3]) → True
  common_end([1, 2, 3], [7, 3, 2]) → False
  common_end([1, 2, 3], [1, 3]) → True

Solution

- Check if length of both lists is atleast 1
- Compare the first or last elements

Problem 3

- Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array.
- Test cases
  big_diff([10, 3, 5, 6]) → 7
  big_diff([7, 2, 10, 9]) → 8
  big_diff([2, 10, 7, 2]) → 8

Solution

- Use built in functions min and max

Problem 4

- Given an array of ints length 3, return an array with the elements "rotated left" so {1, 2, 3} yields {2, 3, 1}.
- Test cases
  rotate_left3([1, 2, 3]) → [2, 3, 1]
  rotate_left3([5, 11, 9]) → [11, 9, 5]
  rotate_left3([7, 0, 0]) → [0, 0, 7]

Solution

- Create another list of 3 elements
- Assign the rotating elements to the new list
- Return the new list

Next Class

- Problems on Lists
- In-Class Submission