

Discussion 05

Mutability, Object Oriented Programming

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

Announcements

- Homework 2 due TODAY (7/7)
- CATS released!!!
 - Can have a partner
 - more time for this project
- Midterm a week from today
 - all content through today's lecture (inheritance) will be in scope
 - Logistics will come on Piazza
 - Start studying!

Mutability



List Mutation Functions

- `append(elem)`
 - box `elem` in list and add to end of list (can lead to nested lists)
- `extend(elem)`
 - unbox `elem` and add to end (have to use an iterable)
- `insert(index, elem)`
 - insert `elem` at `index` (don't replace existing elem)
- `remove(elem)`
 - remove first appearance of `elem` in list (error if not found)
- `pop(index)`
 - removes and return elem at `index` (default arg is end of list)

Mutating Lists

- List Mutation Functions modify **existing** list
- Slicing creates a **new** list
- `a = a + b` creates a **new** list
- `a += b` mutates **existing** list (basically `extend`)
- Indexing into list and changing values modifies **existing** list
 - `a = [1, 2, 3]`
 - `a[0] = 7`

Identity vs Equality

- `is`
 - Check if two objects are the same (point to same reference in memory)
- `==`
 - Check to see if content is the same
- Demo

```
>>> a = [7, 6, 4]
>>> b = [7, 6, 4]
>>> a is b
False
>>> a == b
True
```

Shallow Copy and Deep Copy

- Shallow Copy
 - What Python does most of the time
 - Copy top level of list
 - Point to same objects with nested list
- Deep Copy
 - Make completely new copy of list
 - Difficult to do this
- Whenever we copy a sequence, we are using a shallow copy

Object Oriented Programming (OOP)

Object Oriented Programming (OOP)

- What is OOP?
 - Use of classes to define our own data types
 - More abstraction
 - Reuse code with inheritance
 - MORE ABSTRACTION
- Those with prior experience in Java are familiar
- You have already used OOP!
 - `list.append`
 - `append` is a method belonging to the `list` class

Some Terminology

- Class
 - Template for creation of object
- Object
 - An instance of a class
- Variables
 - Instance Variables
 - property specific to an object
 - Class Variables
 - property shared between all instances of a class
- Method
 - Function that is bound to a class

Functions vs Methods

- Methods need to take in `self` as an object
- `self` argument tells which object to call method on
- Two methods of writing method calls
 - `Class.method(self, args)`
 - `object.method(args)`
 - `self` is automatically set as object
- Demo

Thank you!

Attendance Form -> <https://tinyurl.com/adit-disc05>

Anon Feedback -> <https://tinyurl.com/adit-anon>