

Mutability

Announcements

Mutation and Identity

Sameness and Change

- As long as we never modify objects, a compound object is just the totality of its pieces
- This view is no longer valid in the presence of change
- A compound data object has an "identity" in addition to the pieces of which it is composed
- A list is still "the same" list even if we change its contents
- Conversely, we could have two lists that happen to have the same contents, but are different

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a
[10, 20]
>>> b
[10, 20]
>>> a == b
True
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
>>> a
[10]
>>> b
[10, 20]
>>> a == b
False
```

Identity Operators

Identity

`<exp0> is <exp1>`

evaluates to `True` if both `<exp0>` and `<exp1>` evaluate to the same object

Equality

`<exp0> == <exp1>`

evaluates to `True` if both `<exp0>` and `<exp1>` evaluate to equal values

Identical objects are always equal values

(Demo)

Arrays vs Lists

Numpy Arrays Represent Fixed-Length Sequences of Numbers

```
import numpy as np
a = np.array([3, 4, 5, 6])
b = a + 1
```

vs

```
s = [3, 4, 5, 6]
t = [x + 1 for x in s]
```

Numpy array advantages:

- Much faster repeated arithmetic
- More concise expressions
- Handles 2+ dimensions (matrix, etc.)

Numpy disadvantages:

- Fixed size: appending makes a new array
- Fixed type: [3, 4] and [[3, 4], [5, 6]]
but not [3, [4, 5]]

(Speed Test Demo)

Guidance:

- Repeated calculations over long lists of numbers should use array operations
- Collecting results as they are generated should use a list