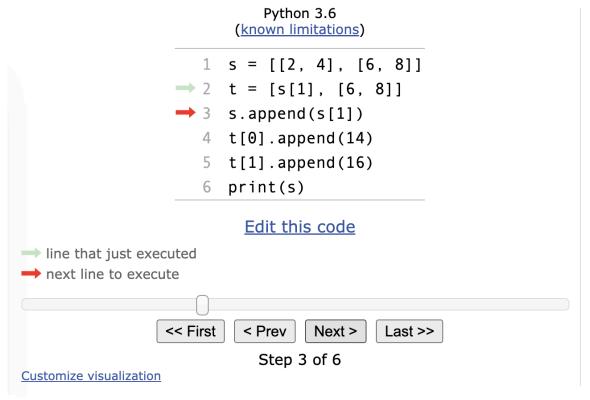


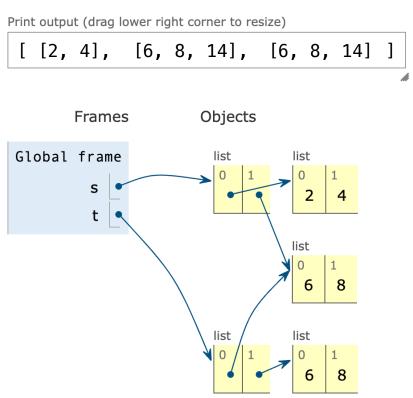


List Mutation

(Demo)

Nested Lists

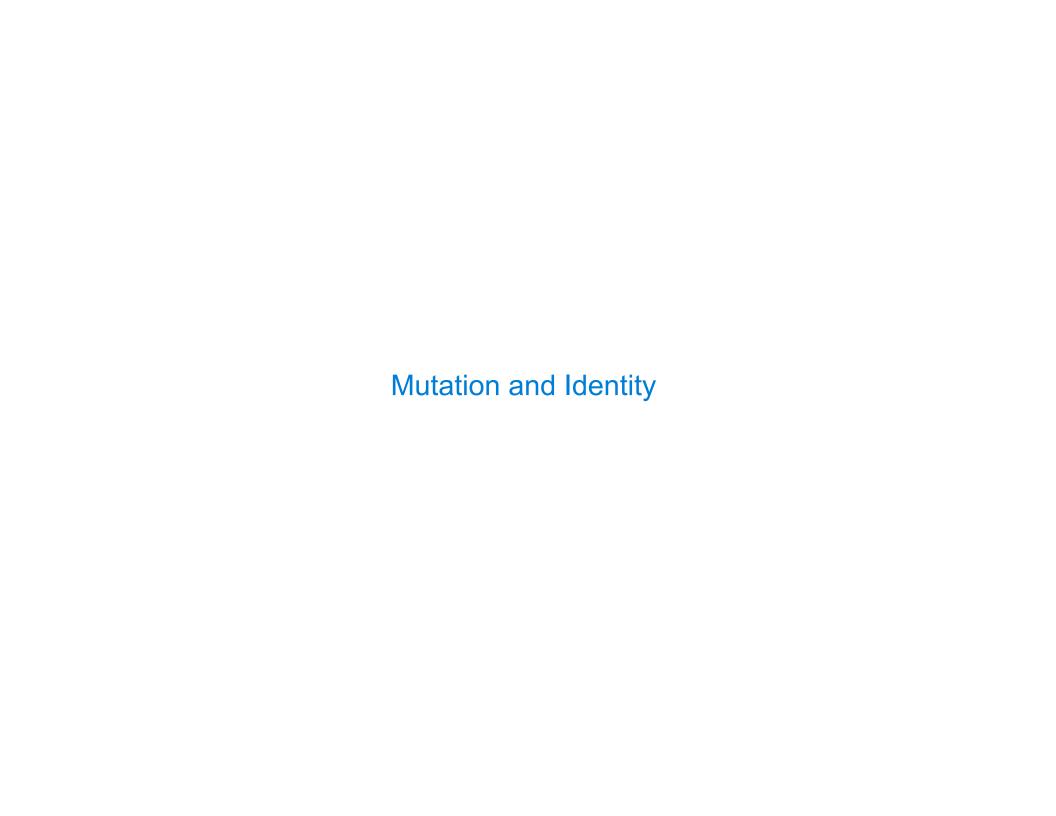




OC#GC

Building Lists Using Append

```
def sums(n, m):
    """Return lists that sum to n containing positive numbers up to m that
    have no adjacent repeats, for n > 0 and m > 0.
    >>> sums(5, 1)
    >>> sums(5, 2)
    [[2, 1, 2]]
    >>> sums(5, 3)
    [[1, 3, 1], [2, 1, 2], [2, 3], [3, 2]]
    >>> sums(5, 5)
    [[1, 3, 1], [1, 4], [2, 1, 2], [2, 3], [3, 2], [4, 1], [5]]
    >>> sums(6, 3)
    [[1, 2, 1, 2], [1, 2, 3], [1, 3, 2], [2, 1, 2, 1], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
    for k in range(1, \frac{\min(m+1, n)}{\text{sums}(n-k, m)}): # k is the first number of a list
            if rest[0] != k:
                result_append([k] + rest)
                                             # build a list out of k and rest
    if n <= m:
        result_append([n])
    return result
```



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- This view is no longer valid in the presence of change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]
                                    >>> a = [10]
                                    >>> b = [10]
>>> b = a
>>> a == b
                                    >>> a == b
True
                                    True
>>> a append(20)
                                    >>> b_append(20)
>>> a
                                    >>> a
[10, 20]
                                    [10]
>>> b
                                    >>> b
[10, 20]
                                    [10, 20]
                                    >>> a == b
>>> a == b
True
                                    False
```

Identity Operators

Identity

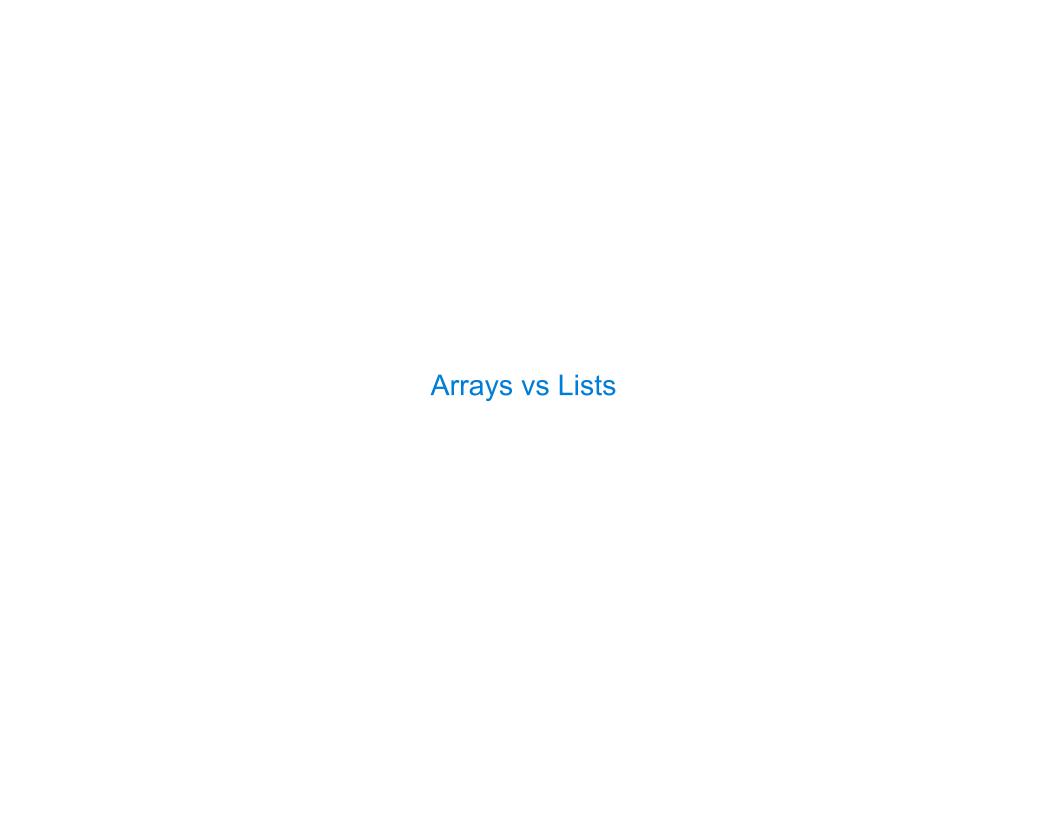
evaluates to True if both <exp0> and <exp1> evaluate to the same object

Equality

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

Identical objects are always equal values

(Demo)



Numpy Arrays Represent Fixed-Length Sequences of Numbers

$$t = [x + 1 \text{ for } x \text{ in } s]$$

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)

s = [3, 4, 5, 6]

Guidance:

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