Identity, Equality, and Mutable Values

---- is ----

- tests identity
- cannot be overloaded by any method
- checks whether objects are pointing to the same area in memory

---- == ----

- tests equality
- can be overloaded by implementing a __eq__ method for an object class
- usually used to test equality in mutable objects

```
>>> x = range(4)
>>> y = range(4)
>>> x == y

>>> x is y

>>> z = x
>>> z == x

>>> z is x

>>> z[2] = 7
>>> x == z

>>> x == y

>>> boom = “bam”
>>> bam = “boom”

>>> boom == bam

>>> boom = bam
>>> bam = boom
>>> boom is bam

>>> boom == bam
```

What would Python print?

```
def list_maker(a, b):
      return list(a) + list(b)

def list_copier(a):
      return a[:]

def nester(a, b):
      return b.append(a)

>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a == b

>>> a is b

>>> c = nester(a, b)
>>> a == b

>>> b[-1] is a

>>> d = nester(c, a)
>>> a

>>> d = a + b
>>> e = list_maker(a, b)
>>> d == e

>>> d is e
```

Write a function find_ratio(input, word) such that, given any string, will return you the ratio between one word and the rest of the string.

Note: calling VAR_NAME.split() where VAR_NAME is a string will return you a list of each word example:
>>> diane_young = "baby baby baby baby right on time"
>>> diane_young.split() = ["baby", "baby", "baby", "baby", "right", "on", "time"]

```
“””
>>> find_ratio(“baby baby baby baby right on time”, “baby”)
“4:7”
“””
def find_ratio(input, word):
```

Implement a function uniq, which takes in a *sorted* list of numbers that might contain duplicates. Mutate the list such that it only contains one instance of each number (i.e. remove the duplicates). Do not return or create any new lists in your implementation (remember, slicing a list will create a new list, so don't do it!).

```
def uniq(lst):
  """Takes a sorted list of numbers and mutatively removes all duplicates
  from the list.

  >>> lst = [1, 1, 2, 4, 4, 9, 10, 10, 10, 10]
  >>> uniq(lst)        # doesn't return anything!
  >>> lst
  [1, 2, 4, 9, 10]
  >>> empty = []
  >>> uniq(empty)
  >>> empty
  []
  """
```