Schnittmenge zweier Listen (ohne Datentyp set)

1. Schreiben Sie eine Fuktion `intersection(a,b)`, die zwei Listen `a, b` entgegennimmt und eine weitere Liste zurückgibt, die nur die Elemente enthält, die in beiden Listen enthalten sind (ohne Duplikate). Stellen Sie sicher, dass Ihr Programm mit zwei Listen unterschiedlicher Größe arbeiten kann.

2. Bonus: Erkundigen Sie sich über den Datentyp `set`. Wie lässt sich dieser Datentyp für die obige Aufgabe verwenden?

Solution:

```
#!/usr/bin/env python
# coding: utf-8
# <h1>Table of Contents<span class="tocSkip"></span></h1>
# <div class="toc"><ul class="toc-item"><li><span><a href="#Schnitt-zweier-Listen" data-toc-modified-id="Schnitt-zweier-Listen-1"><span class="toc-item-num">1  </span>Schnitt zweier Listen</a></span></li></ul></div>
# ### Schnitt zweier Listen
#
# Die unten angegebene Fuktion `set` wandelt Listen in Menge um. Dadruch enthalten Sie keine Duplikate mehr.
def intersection(lst1, lst2):
    return list(set(lst1) & set(lst1))
intersection([5,"4",3,1,3,2,7,"5",6,1,2,6,7,8], [1,"5",3,7,3])
def intersection(lst1, lst2):
    """
    Intersects two lists and returns an intersection without duplicates.

    :param lst1: list
    :param lst2: list
    _return: list
    """
    uniquelst1 = lst1#uniqueSorted(sort(lst1))
    uniquelst2 = lst2#uniqueSorted(sort(lst2))
    interslst = []
    for val in uniquelst1:
        #if val in uniquelst2: # auf sort kann man verzichten: if val in uniquelst2 and not in interslst !!
        if val in uniquelst2 and (val not in interslst):
            interslst.append(val)
    return interslst
intersection([5,"4",3,1,3,2,7,"5",6,1,2,6,7,8], [1,"5",3,7,3])
# Nur unter Verwendung von Stoff aus der VL ist es auch m glich Listen zu schneiden und Duplikate zu entfernen. Der Einfachheit halber k nnen wir die Liste zun chst sortieren ( das ist sicher nicht sehr effizient, aber oben steht ja wie man es eigentlich machen w rde ).
### brauchen wir nicht!
def maximum(inputList):
    """
    Returns the value and the index of the maximal value in a list.
    :param inputList: type list
    :return: value, index
    """
    currentIndex = 0
    currentVal = inputList[currentIndex]
    for i, v in enumerate(inputList):
        if v > currentVal:
```

```
            currentIndex = i
            currentVal = v
    return currentVal, currentIndex
def sort(inputList):
    """
    Sorts the inputList and returns the result.
    :param inputList: list
    :return: list
    """
    newList = []
    while inputList:
        value, index = maximum(inputList)
        inputList.pop(index)
        newList.append(value)
    return newList
### brauchen wir nicht!
def uniqueSorted(sortedlst):
    """
    Remove duplicates from a sorted list.

    :param sortedlst: list
    :return: list
    """
    uniquelst = []
    while sortedlst:
        uniquelst.append(sortedlst.pop(0))
        while sortedlst and (uniquelst[-1] == sortedlst[0]):
            sortedlst.pop(0)
    return uniquelst
```