Finding the unique items in a sequence

Sets make it trivial to find the unique items in a sequence.

```
a_list = ['The', 'sixth', 'sick', "sheik's", 'sixth', "sheep's", 'sick']
print (set(a_list) )                                    #①
#{'sixth', 'The', "sheep's", 'sick', "sheik's"}

a_string = 'EAST IS EAST'
print (set(a_string) )                                  #②
#{'A', 'E', 'S', ' ', 'T', 'I'}

words = ['SEND', 'MORE', 'MONEY']
print (''.join(words))                                  #③
#SENDMOREMONEY

print (set(''.join(words))  )                           #④
#{'R', 'M', 'E', 'S', 'Y', 'O', 'N', 'D'}
```

① Given a list of several strings, the `set()` function will return a set of unique strings from the list. This makes sense if you think of it like a for loop. Take the first item from the list, put it in the set. Second. Third. Fourth. Fifth — wait, that's in the set already, so it only gets listed once, because Python sets don't allow duplicates. Sixth. Seventh — again, a duplicate, so it only gets listed once. The end result? All the unique items in the original list, without any duplicates. The original list doesn't even need to be sorted first.

② The same technique works with strings, since a string is just a sequence of characters.

③ Given a list of strings, '' `.join(a_list)` concatenates all the strings together into one.

④ So, given a list of strings, this line of code returns all the unique characters across all the strings, with no duplicates.

The alphametics solver uses this technique to build a set of all the unique characters in the puzzle.

```
unique_characters = set(''.join(words))
```

This list is later used to assign digits to characters as the solver iterates through the possible solutions.