Lists

We can compose simple data types (e.g., strings, integers, etc) into more complex data structures.

A list, for example, lets us store a sequence of values (often referred to as elements):

```
pet_names = ['Yogi', 'Boomer', 'Gunther', 'Banana']
```

You can access and reassign individual positions (or indexes) in a list using square brackets `[]`. NOTE: lists in python always start at 0 (e.g., 'Yogi' in our example is at position 0 of the `pet_names` list).

```
print(pet_names[0]) # will print Yogi
print(pet_names[1]) # will print Boomer
print(pet_names[3]) # will print Banana

# We can access a list with negative numbers, too...
print(pet_names[-1]) # will print Banana
```

CHALLENGE QUESTION: how might we get the second-to-last value in the list?

We can modify the value of something in the list:

```
pet_names[1] = "BOOMER"
print(pet_names[1]) # will print BOOMER
```

A word of warning: if you try to index into a position beyond the size of the list, python will yell at you: `IndexError: list index out of range`.

```
pet_names[4] # will give you an error
pet_names[100] # will give you an error
pet_names[-4] # will give you an error
```

We can loop over a list using a for loop:

```
for name in pet_names:
    print(name)
```

It's often useful to check to see if a particular value is in the list:

```
print("'Alex' in the list? " + str('Alex' in pet_names))         # no
print("'Yogi' in the list? " + str('Yogi' in pet_names))         # yes
print("'BANANA' in the list? " + str('BANANA' in pet_names))     # nope - remember, py
print("'gunther' in the list? " + str('gunther' in pet_names))   # nope
```

We can append things to the end of lists:

```
pet_names.append("Doggo")
# We can print the entire list
print(pet_names)
```

We can insert things into the list at a particular position:

```
pet_names.insert(1, "Fishy")
print(pet_names)
```

We can remove a particular value from the list:

```
pet_names.remove("Gunther")
print(pet_names)
```

We can remove something from the list by position:

```
pet_names.pop(1)
print(pet_names)
```

Wait, how long is our list? We can use the `len()` instruction to check:

```
list_len = len(pet_names)
print(list_len)
# We could also:
print(len(list_len))
```

We can sort a list:

```
sorted_pet_names = sorted(pet_names)
print(sorted_pet_names)

# What if we want reverse order?
```

```
reverse_sorted_pet_names = sorted(pet_names, reverse=True)
print(reverse_sorted_pet_names)
```

We can remove everything from a list:

```
pet_names.clear()
print(pet_names)
```

For more things you can do with lists see: https://docs.python.org/3/tutorial/datastructures.html#more-on-lists