For Loops and Intro to Lists

Review: **while** loops run while a condition is True, **for** loops iterate over a “collection”--a list, a string, a set of integers, or more

6 kinds of collections in Python:

1. Set: unordered group of any values
2. Range: ordered group of integers, in ascending order, index
3. String: ordered group of characters, index, immutable
4. Tuple: ordered group of any type of values, index, immutable
5. List: ordered group of any type of values, index, mutable
6. Dictionary (dict): unordered and index-less group of any values

0	set	unordered	any values	unindexed		mutable
1	range	ordered	integers	indexed	(start, end, step)	immutable
2	string	ordered	characters	indexed	“Letters and spaces”	immutable
3	tuple	ordered	any values	indexed	(“various”, 1, [‘items’])	immutable
4	list	ordered	any values	indexed	[‘items’, 1, [‘go’, ‘here’]]	mutable
5	dictionary (dict)	unordered	any values	unindexed		mutable

Indexes start at zero and let us call specific values by position, even if we don’t know what they are; for example, the above table of collection types is

What collections look like:

```
print(range(5))
# starts at zero
# stops right before 5
# prints every integer from 0 up to but not including 5
for i in range(5):
   print(i)
# does the exact same thing

for i in range(2, 10):
   print(i)
# now, the range starts at 2 instead of 0, and goes up to but doesn't include 10
```

```
for i in range(2, 10, 2):
   print(i)
# the third parameter is a "step", meaning instead of going up 1 each time, it goes up 2
# it prints 2, 4, 6, 8

for i in range(10, 2, -3):
   print(i)
# can step backwards
# prints 10, 7, 4

string = "welcome back from the break"
for i in string:
   print(i)
# prints every character, including spaces, on its own line

print(string[0:7])
# "print string from 0 to 7
# the colon indicates where you start and stop
# prints "welcome", i.e. the characters is positions 0, 1, 2, 3, 4, 5, and 6
# [ :7] means beginning to 7
# [10: ] means 10 to end
# we call this "splicing"

words = string.split()
print(words)
# every time there is a space, it recognizes a word break
# it then puts each word in a list
# this list is ['welcome', 'back', 'from', 'the', 'break']
string.split(";")
# splits every time there is a colon, instead of every time there is a space

tuple = (3, "hello")
# can have anything in it, even functions
print(tuple[0])
# prints the item at index 0 of the tuple
print(tuple[:2])
# prints both items
print(tuple[:1] + tuple[1:])
# concatenates the two separate items in the tuple back together

list = [3, "hello", [1,2,3]]
# this list has another list in it
print(list[0])
# can call items from a list the same as with a tuple
del list[1]
# removes the item at index 1 from the list
# list is now [3, [1,2,3]]
```

An application of collections in a function:

```
def mcdonald(animals, sounds):
    """
    Sings old mcdonald for every animal and sound pair given, using a for loop
    :param animals: the animals we want sung
    :type animals: list
    :param sounds: the sounds those animals make
    :type sounds: list
    :return: None
    """
    if len(animals) <= len(sounds):
        max = len(animals) + 1
    else:
        max = len(sounds) + 1
    for i in range(0, max):
        print("old mcdonald had a farm E I E I O")
        print("and on his farm, he has a", animals[i], "E I E I O")
        print("with a", sounds[i], sounds[i], "here, and a", sounds[i], sounds[i],
"there")
        print("here a", sounds[i], "there a", sounds[i], "everywhere a", sounds[i],
sounds[i])
        print("old mcdonald had a farm E I E I O")

animals = ["pig", "horse", "chicken", "dinosaur", "cat"]
sounds = ["oink", "neigh", "cluck", "roar"]

def mcdonald2(animals, sounds):
    """
    Sings old mcdonald for every animal and sound pair given, using a while loop
    :param animals: the animals we want sung
    :type animals: list
    :param sounds: the sounds those animals make
    :type sounds: list
    :return: None
    """
    i = 0
    if len(animals) <= len(sounds):
        max = len(animals) + 1
    else:
        max = len(sounds) + 1
    while i < max:
        print("old mcdonald had a farm E I E I O")
        print("and on his farm, he has a", animals[i], "E I E I O")
        print("with a", sounds[i], sounds[i], "here, and a", sounds[i], sounds[i],
"there")
        print("here a", sounds[i], "there a", sounds[i], "everywhere a", sounds[i],
sounds[i])
        print("old mcdonald had a farm E I E I O")
        i += 1
```

Iterating over a collection can be done with a **for** or a **while** loop; is generally easier with a **for** loop

Retrieve an item using its “index” (location 0 to list size) in brackets