```python
# A list is another "sequence" type in Python.
# Whereas strings are sequences of characters,
# lists are sequences of any type.

# To create a list containing some "elements":
numbers = [4, 8, 15, 16, 23, 42]

# Like any variable, "numbers" is the name, and [2, 3, ...] is its value.

# Vocabulary:
#       - element
#       - index
#       - position

# Draw "numbers" in memory.









print(numbers)
# Printing the list prints all the elements, separated by commas.

# Many of the things we learned about strings are also true of lists.


# Length:
print(len(numbers))

# Indexing:
print(numbers[1])
print(numbers[4])
print(numbers[-1])

# (Remember, in computer science, we like to count starting at 0, not at 1.)
# We can also use another variable as an index.
i = 2
print(numbers[i])
i += 1
print(numbers[i])

# Or use an index in an expression
x = numbers[2] + numbers[3]
if x % numbers[4] == 2:
    print("Will this actually print?") # You tell me!


# We can assign to an index as well.
numbers[1] = 9
print(numbers) # What prints?
```

```
53
54
55
56 # We can also use a for loop using range and len, like with strings.
57 for i in range(len(numbers)):
58     print(numbers[i])
59
60
61
62
```

```
1  import random
2
3  # Roll a pair of dice 1,000,000 times.
4  # Record how many times each outcome occurs.
5
6  NUM_ROLLS = 1000000
7
8  # We want to remember how many times we roll a 2, 3, 4, etc.
9  # We need variables to remember things, but making a bunch of variables
10 # like:
11
12 # rolled_2 = ...
13 # rolled_3 = ...
14 # rolled_4 = ...
15
16 # is ugly and tedious, and requires also writing code like this:
17
18 # if current_dice_roll == 2:
19 #       rolled_2 += 1
20 # elif current_dice_roll == 3:
21 #       rolled_3 += 1
22 # etc.
23
24 # Ugh!
25
26 # We'll use a list instead
27
28 rolls = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
29
30 # This list has 11 elements (Why not 12?). Each element will track how many
31 # times we roll the corresponding dice value, where index 0 corresponds with
32 # a dice value of 2, index 1 with a dice value of 3, etc., and index 10
33 # with a dice value of 12.
34
35 for counter in range(NUM_ROLLS):
36     die_1 = random.randrange(1, 7)
37     die_2 = random.randrange(1, 7)
38     total = die_1 + die_2
39     rolls[total - 2] += 1
40
41 # Print the results, including the overall % of each roll
42 print("RESULTS after", NUM_ROLLS, "rolls:")
43 for i in range(len(rolls)):
44     print("{0}: {1} ({2:0.2f})".format(i + 2, rolls[i], rolls[i] / NUM_ROLLS *
       100))
45
46
47
```

```
1  # One form of "for" loop uses "range". But there are other forms.
2
3  # for can iterate directly over a list. In each iteration of the loop,
4  # the variable used is given the "next" value from the list we are
5  # looping over.
6
7  numbers = [5, 4, 3, 2, 1]
8  for n in numbers:
9      # In the first iteration, n is equal to 5
10     # In the second, it is equal to 4, etc.
11     print(n)
12
13
14 # This is functionally equivalent to:
15 for i in range(len(numbers)):
16     n = numbers[i]
17     print(n)
18
19
20 # So when should you use each one?
21
22
23
24
25
26
27 # Be careful here... in the first loop, "n" does not DIRECTLY represent
28 # each element of the list. It is actually a temporary variable pointing
29 # to each element. This is subtle but important.
30 for n in numbers:
31     n += 1
32
33 print(numbers) # What prints?
34
35
36
37
38 # Lists are very similar to strings, so it should not be surprising
39 # that we can do the same with a string
40 name = "Neal"
41 for letter in name:
42     print(letter)
43
44 # Which is the same as
45 for i in range(len(name)):
46     letter = name[i]
47     print(letter)
48
```