1. `[]` is an empty list in Python.

2. You can assign the value 'hello' as the third value in the list by using the index number and assignment operator: `spam[2] = 'hello'`

3. The value of `spam[int(int('3' \* 2) / 11)]` is 'd'. `'3'\*2` produces the string '33', which is converted to an integer and divided by 11 to give 3. The index of the third element in the list is 'd'.

4. The value of `spam[-1]` is 'd', which is the last element of the list.

5. The value of `spam[:2]` is `['a', 'b']`, which is a sublist containing the first two elements of the list.

6. The value of `bacon.index('cat')` is 1, which is the index of the first occurrence of 'cat' in the list.

7. `bacon.append(99)` adds the integer 99 to the end of the list, so the list becomes `[3.14, 'cat', 11, 'cat', True, 99]`.

8. `bacon.remove('cat')` removes the first occurrence of 'cat' from the list, so the list becomes `[3.14, 11, 'cat', True]`.

9. The list concatenation operator is `+` and the list replication operator is `\*`.

10. The `append()` method adds an item to the end of the list, while the `insert()` method inserts an item at a specified index position.

11. The two methods for removing items from a list are `remove()` and `pop()`. `remove()` removes the first occurrence of an item with a specified value, while `pop()` removes an item at a specified index position and returns it.

12. Both list values and string values are ordered sequences of items, can be indexed and sliced, and have a length.

13. Tuples and lists are both ordered sequences of elements, but tuples are immutable while lists are mutable. Once a tuple is created, you cannot change its values, while lists can be modified by adding, removing, or changing elements.

14. You can type a tuple value that only contains the integer 42 as `(42,)`. The comma is required to indicate that it is a tuple with a single element.

15. To get a list value's tuple form, you can use the `tuple()` function: `tuple([1, 2, 3])` returns `(1, 2, 3)`. To get a tuple value's list form, you can use the `list()` function: `list((1, 2, 3))` returns `[1, 2, 3]`.

16. Variables that "contain" list values are actually references to the list objects themselves. This means that multiple variables can reference the same list object, and modifying the list through one variable will affect all variables that reference the list.

17. `copy.copy()` creates a shallow copy of a list, which means that the new list references the same objects as the original list. Any changes made to the objects in the new list will affect the objects in the original list. `copy.deepcopy()` creates a deep copy of a list, which means that the new list contains new objects that are copies of the objects in the original list. Changes made to the objects in the new list will not affect the objects in the original list.