1. What exactly is []?

2. In a list of values stored in a variable called spam, how would you assign the value 'hello' as the third value? (Assume [2, 4, 6, 8, 10] are in spam.)

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.

3. What is the value of spam[int(int('3' \* 2) / 11)]?

4. What is the value of spam[-1]?

5. What is the value of spam[:2]?

Let's pretend bacon has the list [3.14, 'cat,' 11, 'cat,' True] for the next three questions.

6. What is the value of bacon.index('cat')?

7. How does bacon.append(99) change the look of the list value in bacon?

8. How does bacon.remove('cat') change the look of the list in bacon?

9. What are the list concatenation and list replication operators?

10. What is difference between the list methods append() and insert()?

11. What are the two methods for removing items from a list?

12. Describe how list values and string values are identical.

13. What's the difference between tuples and lists?

14. How do you type a tuple value that only contains the integer 42?

15. How do you get a list value's tuple form? How do you get a tuple value's list form?

16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?

17. How do you distinguish between copy.copy() and copy.deepcopy()?

Answers:

1. [] is an empty list in Python.

2. To assign the value 'hello' as the third value in the list variable spam, you can use the following code: `spam[2] = 'hello'`.

3. The value of `spam[int(int('3' \* 2) / 11)]` is 'd'. The expression `int('3' \* 2)` evaluates to 33, and then dividing it by 11 gives the index 3, which corresponds to the value 'd' in the list.

4. The value of `spam[-1]` is 'd'. Negative indices in Python start counting from the end of the list, so -1 refers to the last element in the list.

5. The value of `spam[:2]` is ['a', 'b']. It uses slicing to select the elements from index 0 to 1 (excluding index 2) in the list.

6. The value of `bacon.index('cat')` is 1. The index() method returns the index of the first occurrence of the specified element in the list.

7. The `bacon.append(99)` adds the value 99 to the end of the list bacon. After appending, the list bacon would become [3.14, 'cat,' 11, 'cat,' True, 99].

8. The `bacon.remove('cat')` removes the first occurrence of the value 'cat' from the list bacon. After removal, the list bacon would become [3.14, 11, 'cat,' True].

9. The list concatenation operator is '+', which combines two lists into a single list. The list replication operator is '\*', which repeats a list a specified number of times to create a new list.

10. The `append()` method adds an element to the end of a list, whereas the `insert()` method inserts an element at a specific position in the list, shifting the existing elements to the right.

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

12. List values and string values are identical in the sense that both can be indexed and sliced. They can also be iterated over using loops. However, lists can contain multiple values of different types, while strings are sequences of characters.

13. Tuples and lists are both sequence data types in Python, but they have some differences. Lists are mutable, meaning their elements can be modified, added, or removed. Tuples, on the other hand, are immutable, and their elements cannot be changed once defined. Tuples are typically used to represent collections of related values that should not be modified.

14. To create a tuple value that only contains the integer 42, you can use the following syntax: `(42,)`. The comma is necessary to indicate that it's a tuple with a single element.

15. To get a list value's tuple form, you can use the `tuple()` function, passing the list as an argument. For example, if `my\_list` is the list, you can convert it to a tuple with `tuple(my\_list)`. To get a tuple value's list form, you can use the `list()` function in a similar way.

16. Variables that "contain" list values actually store references to the list objects. In other words, the variables point to the memory location where the list is stored. The variable itself is not the list, but it holds the reference to it.

17. `copy.copy()` creates a shallow copy of a list, which means it

creates a new list object but the individual elements are still references to the original objects. `copy.deepcopy()` creates a deep copy of a list, which means it creates a new list object as well as new copies of all the objects within the list, recursively.