Answer:

1. What exactly is `[]`?

Answer: `[]` represents an empty list in Python. It is a data structure that can store a collection of items, and in this case, it contains no elements.

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`.)

Answer: To assign the value `'hello'` as the third value in the `spam` list, you can use the index notation and assign the value directly:

spam[2] = 'hello'

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

Answer: First, let's break it down:

`int('3' \* 2)` will result in the integer `33`.

`int('33' / 11)` will give us `3`.

So, the expression `spam[int(int('3' \* 2) / 11)]` is equivalent to `spam[3]`. Since Python lists are 0-indexed, the value at index 3 in the list `spam` is `'d'`.

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

Answer: The index `-1` in Python represents the last element of a list. So, `spam[-1]` will give us `'d'`, which is the last element of the list.

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

Answer: Slicing a list with `[:2]` will give us a new list containing the elements from the beginning up to, but not including, index 2. So, `spam[:2]` will return `['a', 'b']`. 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')`?

Answer: The `index()` method is used to find the index of the first occurrence of a given element in the list. In this case, `bacon.index('cat')` will return `1`, as the first occurrence of `'cat'` is at index 1 in the list `bacon`.

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

Answer: The `append()` method is used to add an element to the end of a list. After calling `bacon.append(99)`, the list `bacon` will be updated to `[3.14, 'cat', 11, 'cat', True, 99]`.

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

Answer: The `remove()` method is used to remove the first occurrence of a specific element from a list. After calling `bacon.remove('cat')`, the list `bacon` will be updated to `[3.14, 11, 'cat', True]`.

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

Answer: The list concatenation operator is `+`, which is used to combine two or more lists into a new list. The list replication operator is `\*`, which is used to repeat a list a certain number of times to create a new list.

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

Answer: `append(item)`: The `append()` method adds the given item to the end of the list. `insert(index, item)`: The `insert()` method inserts the given item at the specified index, shifting other elements to the right.

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

Answer: The two methods for removing items from a list are:

1. `remove(item)`: Removes the first occurrence of the specified item from the list.

2. `pop(index)`: Removes and returns the item at the specified index. If no index is provided, it removes and returns the last item in the list.

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

Answer: Both lists and strings are ordered sequences of items. They can be indexed and sliced using the same syntax. They support the use of the `len()` function to determine their length. They can be iterated over using loops.

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

Answer: Lists and tuples are both used to store collections of items, but they have some key differences:

* Lists are mutable, which means their elements can be changed after creation, while tuples are immutable, and their elements cannot be changed once the tuple is created.
* Lists use square brackets `[]` for representation, while tuples use parentheses `()`.
* Lists support various methods to modify, append, and remove elements, while tuples have fewer methods due to their immutability.

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

Answer: To create a tuple containing the integer `42`, you can use parentheses `()` and separate the value with a comma:

my\_tuple = (42,)

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

Answer: To convert a list to a tuple, you can use the `tuple()` function:

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

To convert a tuple to a list, you can use the `list()` function:

my\_tuple = (4, 5, 6)

my\_list = list(my\_tuple)

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

Answer: Variables that "contain" list values in Python are actually references to the list objects in memory. When you assign a list to a variable, the variable contains the memory address of the list in the computer's memory.

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

Answer: `copy.copy()` (shallow copy) creates a new list and inserts references to the objects found in the original list. This means that changes to the original objects will also be reflected in the copied list, and vice versa, as they share the same references.

`copy.deepcopy()` (deep copy) creates a completely independent copy of the original list and all the objects it contains. Changes to the objects in the original list will not affect the objects in the copied list, and vice versa. It creates a new list and recursively copies the elements and their nested elements.