**1. What exactly is []?**

**[]** denotes an empty list in Python. Lists are ordered collections that can hold a variety of data types, and an empty list 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.)**

To assign the value **'hello'** as the third value in the list **spam** (i.e., replacing the value at index 2):

spam = [2, 4, 6, 8, 10]

spam[2] = 'hello'

**3. Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries. What is the value of spam[int(int('3' \* 2) / 11)]?**

First, evaluate the expression inside the list index:

* **'3' \* 2** results in the string **'33'**.
* **int('33')** converts the string to the integer **33**.
* **33 / 11** results in **3.0**.
* **int(3.0)** converts the float back to the integer **3**.

Thus, **spam[3]** accesses the fourth element of the list **['a', 'b', 'c', 'd']**, which is **'d'**.

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

The value of **spam[-1]** is **'d'**, as negative indices count from the end of the list.

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

The value of **spam[:2]** is **['a', 'b']**, which includes elements from the start of the list up to, but not including, index 2.

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

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

The value of **bacon.index('cat')** is **1**, as **'cat'** first appears at index 1.

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

After **bacon.append(99)**, the list **bacon** will be **[3.14, 'cat', 11, 'cat', True, 99]**.

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

After **bacon.remove('cat')**, the first occurrence of **'cat'** is removed, resulting in the list **[3.14, 11, 'cat', True, 99]**.

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

* List concatenation operator: **+**
* List replication operator: **\***

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

* **append()** adds an element to the end of the list.
* **insert(index, element)** inserts an element at a specified index.

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

* **remove(element)** removes the first occurrence of a specified element.
* **pop(index)** removes and returns the element at a specified index (or the last element if no index is specified).

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

Both lists and strings are sequences, meaning they:

* Have a length determined by **len()**.
* Support indexing and slicing.
* Can be iterated over with loops.

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

* Lists are mutable, meaning their elements can be changed after creation.
* Tuples are immutable, meaning their elements cannot be changed once assigned.

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

To create a tuple with a single integer 42, include a comma:

single\_element\_tuple = (42,)

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

* To convert a list to a tuple: **tuple(list\_variable)**
* To convert a tuple to a list: **list(tuple\_variable)**

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

Variables contain references to the list objects stored in memory.

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

* **copy.copy()** creates a shallow copy of an object, meaning it copies the object's structure but not the nested objects within it.
* **copy.deepcopy()** creates a deep copy of an object, meaning it recursively copies all objects, including nested ones, producing a fully independent clone.