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

Ans-‘[]’ represents an empty list in Python.

**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.**

Ans-You can assign the value 'hello' as the third value in the list stored in the variable spam by using the following code: **spam[2] = 'hello'**.

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

Ans-The value of spam[int(int('3' \* 2) / 11)] is 'd'. The expression int('3' \* 2) evaluates to 33, and 33 / 11 is 3. Therefore, the expression is equivalent to spam[3], which accesses the fourth element of the list.

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

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

**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.**

Ans-The value of spam[:2] is ['a', 'b'], which is a slice of the list containing the first two elements.

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

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

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

Ans-The bacon.append(99) method adds the integer value 99 to the end of the bacon list, so the new list would be [3.14, 'cat,' 11, 'cat,' True, 99].

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

Ans-The bacon.remove('cat') method removes the first occurrence of 'cat' from the bacon list, so the new list would be [3.14, 11, 'cat,' True].

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

Ans-The list concatenation operator is +, which combines two lists into a single list. The list replication operator is \*, which creates a new list by repeating a given list a specified number of times.

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

Ans-The append() method adds a new element to the end of a list, while the insert() method allows you to insert a new element at a specified index position within the list.

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

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

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

Ans-List values and string values are identical in that they are both ordered sequences of items. They differ in the types of items they contain (lists can contain any type of object, while strings contain only characters), and in the ways they can be modified (lists are mutable, while strings are immutable).

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

Ans-Tuples and lists are both sequences of values, but tuples are immutable while lists are mutable. This means that once a tuple is created, its values cannot be changed, whereas a list can be modified by adding, removing, or changing elements.

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

Ans-To create a tuple value that only contains the integer 42, you would use parentheses: (42,).

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

Ans-To get a list value's tuple form, you can use the tuple() function, and to get a tuple value's list form, you can use the list() function. For example, tuple([1, 2, 3]) returns (1, 2, 3) and list((4, 5, 6)) returns [4, 5, 6].

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

Ans-Variables that "contain" list values in Python are actually references to the list objects themselves. So, the variables contain references to the memory locations where the list objects are stored.

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

Ans-The copy.copy() function creates a shallow copy of a list, which means that the new list contains references to the same objects as the original list. The copy.deepcopy() function creates a deep copy of a list, which means that the new list contains copies of all objects in the original list, recursively copying any objects that are themselves mutable.