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

**Answer –**

[] represents an empty list. It denotes a list that does not contain any 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 –**

spam[2] = 'hello'

**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)]?**

**Answer – 8**

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

**Answer – 10**

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

**Answer –**

[2, 4]

**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 – 1**

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

**Answer –**

[3.14, 'cat', 11, 'cat', True, 99]

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

**Answer –**

[3.14, 11, 'cat', True, 99]

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

**Answer –**

The list concatenation operator in Python is **+,** and the list replication operator is **\*.**

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

**Answer –**

**Append**() adds an element to the end of a list, while **Insert**() allows you to specify the index at which the element should be inserted.

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

**Answer –**

remove() & pop()

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

**Answer –**

Both list values and string values are sequences that support indexing, slicing, and iteration.

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

**Answer –**

Tuples are immutable and defined with parentheses(), while lists are mutable and defined with square brackets[].

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

**Answer –**

tuple = (42,)

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

**Answer –**

**list value's tuple form-**

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

**tuple value's list form**

my\_tuple = (1, 2, 3)

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 actually contain references or pointers to the list objects.

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

**Answer –**

**copy.copy()** creates a shallow copy of an object, which means that it creates a new object with references to the original elements. Modifying the elements in the copy will affect the original object.

**copy.deepcopy()** creates a deep copy of an object, which means it creates a new object with copies of all the nested objects and their elements. Modifying the elements in the copy will not affect the original object.