1. What exactly is []?

**Solution:** [] is the notation for an empty list in Python programming language. A list is a data structure that holds a collection of items, which can be of any type. An empty list has no items, so it is represented by the brackets with nothing in between. For example, myList = [] declares an empty list named "myList".

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.)

**Solution:** 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)]?

**Solution:** “d”

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

**Solution:** ‘d’

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

**Solution:** ‘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')?

**Solution:** bacon.index('cat') -> 1

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

**Solution:** [3.14, 'cat', 11, 'cat', True, 99]

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

**Solution:** [3.14, 11, 'cat', True, 99]

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

**Solution**: In Python, the "+" operator is used for concatenating two or more lists, which means it combines the elements of the lists into a single list. For example, if you have two lists: list1 = [1, 2, 3] list2 = [4, 5, 6] You can concatenate them into a single list: list3 = list1 + list2

The "\*" operator is used for replicating a list a certain number of times. For example, if you have a list: list1 = [1, 2, 3] You can replicate it 3 times: list2 = list1 \* 3

Both of these operators can also be used with strings and other iterable objects.

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

**Solution**: In Python, both append() and insert() are methods that can be used to add elements to a list, but they work in slightly different ways.

The append() method is used to add an element to the end of a list. It only takes one argument, which is the item to be added.

On the other hand, the insert() method is used to add an element to a specific position in the list. It takes two arguments: the first is the index at which the element should be inserted, and the second is the item to be added.

So, the main difference between append and insert is, append() add element at the end of the list and insert() add element at specific position in the list.

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

**Solution:** In Python, there are two main methods for removing items from a list: remove() and pop().

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

**Solution:** In Python, lists and strings are both types of sequences. A sequence is an ordered collection of items, and both lists and strings are used to store and manipulate groups of related values.

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

**Solution:** One of the main differences is that tuples are immutable, while lists are mutable. This means that once a tuple is created, its contents cannot be changed. On the other hand, the contents of a list can be modified after it is created.

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

**Solution:** my\_tuple = (42,)

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

**Solution:** my\_list = [1,2,3,4,5]

Convert\_tuple = tuple(my\_list)

my\_tuple = (1,2,3,4,5)

convert\_list = list(my\_tuple)

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

Solution: Variables that "contain" list values actually contain references to list objects, rather than the list itself. In Python, when you assign a list value to a variable, the variable stores a reference to the memory location where the list is stored, rather than a copy of the list's contents.

For example, when you write my\_list = [1, 2, 3], a list object is created in memory containing the values 1, 2, and 3, and the variable my\_list is assigned a reference to that object.

This means that when you make changes to the list using the variable, you are actually modifying the original list object, and those changes will be reflected in any other variables that reference the same list.

Also, when you assign the same list to multiple variables, they will point to the same list object in memory and any change made to the list through one variable will be reflected in all the variables pointing to the same list.

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

**Solution: copy()** creates a shallow copy of an object. A shallow copy creates a new object with a new reference to the same objects that the original object contains. This means that any changes made to the contained objects will be reflected in both the original and the copied object.

**deepcopy()** creates a deep copy of an object. A deep copy creates a new object with new references to new objects that are copies of the objects contained in the original object. This means that any changes made to the contained objects will not be reflected in the copied object.