1. What exactly is []?
Ans: [] is an empty list in Python. A list is a data structure in Python that holds a collection of elements, which can be of any data type. An empty list is a list that contains no elements, and it is represented in Python by a pair of square brackets with nothing between them, as in [].

You can create an empty list in Python using the list() constructor or by simply using the empty brackets []. Empty lists are commonly used as placeholders that can be populated later with data, or as return values from functions that don't have any specific value to return.

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.)

Ans: 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)]?
Ans: Spam has the value 'd' (int(int('3' * 2) / 11).

The expression is assessed as follows:

The result of the expression '3' * '2' is the number '33'.

The int() method converts the text '33' to an integer and returns the value 33.

By employing the integer division operator (//), the number 3 is produced when the number 33 is divided by 11.

Since Python indexing begins at 0, the expression spam[3] refers to the fourth entry of the list spam.

The string "d" makes up the fourth element of ["a," "b," "c," and "d."

As a result, spam[int(int('3' * 2) / 11)] has the value 'd'.

4. What is the value of spam[-1]?
Ans: The value of spam[-1] is 'd'.

In Python, negative indexing allows you to access elements from the end of a list. Index -1 refers to the last element of the list, -2 refers to the second-last element, and so on.

In the given list ['a', 'b', 'c', 'd'], the last element has an index of -1. Therefore, spam[-1] refers to the last element of the list, which is the string 'd'.

5. What is the value of spam[:2]?
Ans: Spam[:2]'s value is ['a', 'b'].

Slices are a Python feature that let you take a chunk of an existing list and turn it into a new list. List[start:end] is the syntax for cutting a list, where start is the index of the first element to include and end is the index of the first element to exclude.

The slice spam[:2] in the supplied list ['a', 'b', 'c', 'd'] starts at the beginning of the list and extends up to the element at index 2, but does not contain it. 'a' and 'b', the first two items on the list, are thus included in the slice. Therefore, ['a', 'b'] is the value of spam[:2].

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')?
Ans: Without more details, it would be challenging to respond to this question. We must understand what bacon is in order to compute the value of bacon.index('cat').

In Python, the index() function is used to determine the position of a given value within a list. A ValueError is raised if the specified value cannot be found in the list.

Bacon.index('cat') would return the index of the first instance of "cat" in the list if bacon is a list with the value "cat". For instance, bacon.index('cat') would return 1 if bacon = ['dog', 'cat','mouse'].

However, bacon.index('cat') would throw a ValueError if it didn't already contain the value "cat" in it.

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

Ans: The append() method in Python is used to add a new element to the end of a list. If bacon is a list, then bacon.append(99) would add the value 99 to the end of the list bacon.

For example, if bacon = [1, 2, 3], then bacon.append(99) would change the list bacon to [1, 2, 3, 99]. The append() method modifies the list in place, so the original list is changed.

8. How does bacon.remove('cat') change the look of the list in bacon?
Ans: To remove the first instance of a specified value from a list, use Python's remove() function. If bacon is a list, the value 'cat' would be removed from the list bacon's first instance if bacon.remove('cat') is true.

In the case where bacon = ['cat', 'dog', 'cat','mouse'], bacon.remove('cat') would eliminate the first instance of 'cat' from the list bacon. ['dog', 'cat','mouse'] would be on the list bacon following the method call.

The delete() method would throw a ValueError if the value 'cat' wasn't found in the list bacon. The original list is altered using the delete() method, which modifies the list already in use.

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

Ans: List concatenation, or the process of joining two or more lists into one list, is performed in Python using the + operator. For instance, list1 + list2 would equal [1, 2, 3, 4, 5, 6] if list1 = [1, 2, 3] and list2 = [4, 5, 6].

List replication, or the act of producing a new list by iteratively repeating an existing list a certain number of times, uses the * operator. For instance, list1 * 3 would get [1, 2, 3, 1, 2, 3, 1, 2, 3] if list1 = [1, 2, 3].

The initial lists are not changed by the + or * operators. As a result of the actions, they instead produce new lists.

10. What is difference between the list methods append() and insert()?
Ans: In Python, the append() and insert() methods are used to add elements to a list. The main difference between these two methods is the way in which they add elements to the list:

- append() method adds an element to the end of the list, while insert() method inserts an element at a specified index in the list.
- append() method takes a single argument, which is the element to be added to the list, while insert() method takes two arguments: the first argument is the index where the element should be inserted, and the second argument is the element to be inserted.
- If you want to add an element to the end of the list, you can use either append() or insert() method with the index set to the length of the list. However, using append() is simpler and more readable.

For example, if my_list = [1, 2, 3], then my_list.append(4) would result in [1, 2, 3, 4], while my_list.insert(1, 'a') would result in [1, 'a', 2, 3, 4].

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

Ans: In Python, there are two methods for removing items from a list:

1. `remove()`: This method removes the first occurrence of the specified element from the list. If the element is not found in the list, a `ValueError` is raised.

2. `pop()`: This method removes and returns the element at the specified index from the list. If no index is specified, it removes and returns the last element in the list.

For example, if `my_list = [1, 2, 3, 4, 5]`, then `my_list.remove(3)` would result in `[1, 2, 4, 5]`, while `my_list.pop(2)` would result in `3` and the `my_list` would become `[1, 2, 4, 5]`.

12. Describe how list values and string values are identical.
Ans: As Python sequences of values that may be retrieved using an index, list values and string values are comparable. The + operator can be used to concatenate lists and strings, and loops can iterate over both types of data.

The methods len(), which returns the length of the sequence, max(), which returns the greatest element in the sequence, and min(), which gives the shortest element in the sequence, are also shared by lists and strings.

Lists and strings, however, differ significantly in a number of key ways. One of the key distinctions between lists and strings is that lists can have their values altered, whereas strings cannot have their values changed once they have been formed. Lists may additionally include components.

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

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

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

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

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