1. ***What exactly is []?***
   Ans:- The empty list value, which is a list value that contains no items.

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'

   -
     ```
     spam = [2, 4, 6, 8, 10]
     spam[2] = 'hello'
     spam
     ```
   - **o/p** - `[2, 4, 'hello', 8, 10]`

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:- ‘d’

4. ***What is the value of spam[-1]?***
   Ans:- ‘d’

5. ***What is the value of spam[:2]?***
   Ans:- ['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')?***
   Ans:- 1

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

   o/p : [3.14, 'cat', 11, 'cat', True, 99]

8. ***How does bacon.remove('cat') change the look of the list in bacon?***
   Ans:- bacon.remove('cat')
   bacon
   o/p : [3.14, 11, 'cat', True, 99]

9. ***What are the list concatenation and list replication operators?***
   Ans:- The "+" operator is used for concatenating lists, while the "*" operator is used for replicating lists.

1. **List Concatenation (+):**

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
concatenated_list = list1 + list2
print(concatenated_list)
```

Output:

```
[1, 2, 3, 4, 5, 6]
```

The "+" operator combines the elements of list1 and list2 into a single list, concatenated_list.

2. **List Replication (*):**

```
original_list = [1, 2, 3]
replicated_list = original_list * 3
print(replicated_list)
```

Output:

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

The "" operator replicates the elements of original_list three times, creating a new list replicated_list with the repeated elements.*

10. ***What is difference between the list methods append() and insert()?***
    Ans:- The append() method adds an item to the end of a list, whereas. insert() method inserts an item in a specified position in 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:
    1. Using the remove() method: This method removes the first occurrence of the specified item from the list.

    2. Using the pop() method: This method removes the item at the specified index from the list and returns it.

12. ***Describe how list values and string values are identical.***
    Ans:- Both list values and string values are sequences of values. They can be indexed, sliced, concatenated, and have a length.

13. ***What's the difference between tuples and lists?***
    Ans:- Lists are mutable; they can have values added, removed, or changed. Tuples are immutable; they cannot be changed at all. Also, tuples are written using parentheses, (and), while lists use the square brackets, [and].

14. ***How do you type a tuple value that only contains the integer 42?***
    Ans:- tuple=(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.
      ```
      my_list = [1, 2, 3, 4, 5]
      my_tuple = tuple(my_list)
      print(my_tuple)
      ```

      Output:
      ```
      (1, 2, 3, 4, 5)
      ```

    - To get a tuple value's list form, you can use the **list()** function.
      ```
      my_tuple = (1, 2, 3, 4, 5)
      my_list = list(my_tuple)
      print(my_list)
      ```

      Output:
      ```
      [1, 2, 3, 4, 5]
      ```

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

Ans:- Variables that "contain" list values actually contain a reference to the list object in memory.

17. ***How do you distinguish between copy.copy() and copy.deepcopy()?***
    Ans:-
    - copy.copy(): This function performs a shallow copy of an object. It creates a new object and then copies the references of the original object's elements to the new object.
    - copy.deepcopy(): This function performs a deep copy of an object. It creates a completely independent copy of the original object and recursively copies all the objects it references