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

**Ans.** [] is 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.)

**Ans.** spam=[2,4,6,8,10]

Spam[2]=’hello’ ## this will replace 6 by ‘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[int(int('3' \* 2) / 11)]

spam[int(int(‘33’)/11)]

spam[int(33/11)]

spam[3] = ‘d’

O/P – ‘d’

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

**Ans**. spam[-1] = ‘d’

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

**Ans.** spam[0:2] = [‘a’,’b’]

Let's pretend bacon has the list [3.14, 'cat', 11, 'cat', True] for the next three questions.

Bacon = [3.14, ‘cat’, 11, ‘cat’, True]

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

**Ans.** bacon.index(‘cat’) = 1

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

**Ans**. bacon = [3.14, ‘cat’, 11, ‘cat’, True, 99]

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

**Ans.** bacon = [3.14, 11, ‘cat’, True, 99]

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

**Ans.** operator for list concatenation is +, while the operator for replication is \*.

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

**Ans.** append() adds an item to the end of a list, whereas . insert() inserts an item in a specified position in the list.

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

**Ans.** Following are the methods

1. Using the remove() method.
2. Using the list object's pop() method.

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

**Ans**. While list values and string values are both data types in Python, they have some similarities, but they are fundamentally different in terms of their properties and behaviours. Here's a comparison between list values and string values:

1. Sequence data type: Both lists and strings are considered sequence data types in Python. This means that they can contain multiple elements arranged in a specific order.
2. Indexing: Both lists and strings support indexing. We can access individual elements of a list or a string by their position using square brackets []. Indexing starts from 0 for both data types.
3. Slicing: Lists and strings also support slicing, which allows you to extract sub-sequences from them. You can use the syntax list[start:end] or string[start:end] to extract a portion of the sequence between the start and end indices.
4. Iteration: You can iterate over the elements of a list or a string using loops in Python. Both data types can be used in for loops to process each element individually.
5. Mutable vs. Immutable: One important distinction between lists and strings is that lists are mutable, while strings are immutable. This means that you can modify individual elements of a list, add or remove elements, whereas you cannot modify individual characters of a string. Instead, you need to create a new string if you want to make any changes to it.
6. Different methods and operations: Lists and strings have different methods and operations associated with them. For example, lists have methods like append(), insert(), and remove(), which allow you to modify the list in various ways. On the other hand, strings have methods like upper(), lower(), and replace(), which operate on the string itself.

In summary, while lists and strings share some similarities as sequence data types, they differ in mutability, available operations, and methods. Lists are mutable and can be modified, whereas strings are immutable and cannot be changed directly.

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

**Ans.** Tuples are immutable as opposed to lists which are mutable. Therefore, it is possible to change a list but not a tuple.

**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**. We can use function tuple() to change list to tuple and can use function list() to change list to tuple.

tup = tuple(list)

list = list(tup)

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

**Ans**. Variables will contain references to list values rather than list values themselves.

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

**Ans.** copy() create reference to original object. If we change copied object - it changes the original object. deepcopy() creates new object and does real copying of original object to new one. Changing new deepcopied object doesn't affect original object.