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

Ans:

**[]** represents an 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] = '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:

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

Considered bacon as bacon = [3.14, 'cat', 11, 'cat', True]

**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]

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

Ans:

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

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

Ans:

The append() method adds an element to the end of a list, while the insert() method inserts an element at a specified position in a list.

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

Ans:

The two methods for removing items from a list are remove() and pop()

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

Ans:

List values and string values are both sequence types, allowing for indexing, slicing, and iteration. They also have a length and can be accessed element-wise.

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

Ans:

* Tuples are immutable and lists are mutable
* Tuples are typically used for unchangeable data, while lists are used for data that can be modified.
* Tuples are defined using parentheses **()** and lists are defined using square brackets **[]**

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

Ans:

a = (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, we can use the tuple() function. To get a tuple value's list form, we can use the list() function.

Eg:

a\_list = [1, 2, 3]

a\_tuple = tuple(a\_list)

b\_tuple = (1, 2, 3)

b\_list = list(b\_tuple)

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

Ans:

Variables that contain list values in Python are actually storing references or pointers to the list object in memory. The variables contain the memory address where the list is stored rather than directly holding the list itself.

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

Ans:

copy.copy() creates a shallow copy of an object, while copy.deepcopy() creates a deep copy of an object and its nested objects.

In copy.copy() the top-level object is copied, but the nested objects are not. copy.deepcopy() creates completely independent copy of the original object and all its nested objects.