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

Answer-> [] is an empty list in Python. It is a data structure that can hold an ordered collection of items. It is denoted by square brackets and can contain elements of different data types.

**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.)**

Answer-> To assign the value 'hello' as the third value in the list stored in the variable **spam**, you can use the indexing operator and assign the value directly to the desired index.

Here's an example:

spam = [2, 4, 6, 8, 10]

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)]?**

Ansswer-> The expression int('3' \* 2) evaluates to the integer 33 2 because this express is printing the 3 two times . Dividing 33 by 11 results in 3. When accessing the value at index 3 in the spam list, the value returned is 'd'. So, the value of spam[int(int('3' \* 2) / 11)] is 'd'.

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

Answer-> The index -1 in a list refers to the last element. In the spam list, the last element is 'd'. Therefore, the value of spam[-1] is 'd'.

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

Answer-> The value of spam[:2] is ['a', 'b']. This is known as slicing in Python, where spam[:2] selects all elements from index 0 up to, but not including, index 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')?

Answer-> index of the cat is 1 because index start from 0,1,2…… .

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

Answer-> The bacon.append(99) method call adds the value 99 to the end of the bacon list. After the operation, the list will look like this:

[3.14, 'cat', 11, 'cat', True, 99]

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

Answer-> The bacon.remove('cat') method call removes the first occurrence of the value 'cat' from the bacon list. After the operation, the list will look like this:

[3.14, 11, 'cat', True]

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

Answer-> The list concatenation operator is +, which is used to combine two lists into a single list. The list replication operator is \*, which is used to repeat a list a specified number of times

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

Answer-> The append() method is used to add an item to the end of a list, while the insert() method is used to insert an item at a specific index within the list.

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

Answer-> The two methods for removing items from a list are remove() and pop(). The remove() method removes the first occurrence of a specified value from the list, while the pop() method removes and returns an item at a specified index (if no index is provided, it removes and returns the last item).

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

Answer-> List values and string values are similar in several ways. Both are sequences that can contain multiple items. They can be indexed and sliced to access specific elements or subsequences. Additionally, they both support common operations such as concatenation (+) and replication (\*). Both lists and strings are iterable, meaning you can iterate over their elements using loops or comprehensions.

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

Answer-> Tuples and lists are both sequence data types in Python, but they have some distinctions:

Lists are mutable, allowing modification, addition, and removal of elements. Tuples, on the other hand, are immutable and cannot be changed once created.

Lists are represented using square brackets ([]), while tuples use parentheses ().

Lists have various methods like append(), remove(), and sort() that can modify the list in-place. Tuples have fewer methods and are typically used when immutability is desired.

Lists are commonly used for collections of items where order and mutability are important. Tuples are often used for grouping related values together or returning multiple values from a function.

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

Answer-> my\_tuple = (42,) , the comma at end is important to make it tupple,otherwise it would be interpreted as just the value 42.

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

To convert a list value to its tuple form

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

print(my\_tuple)

To convert a tuple value to its list form

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

print(my\_tuple)

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

Answer-> Variables that "contain" list values actually store references to the list objects. In Python, variables are like labels or references that point to the actual list stored in memory. When you assign a list to a variable, the variable holds a reference to the list rather than the list itself. This allows you to access and manipulate the list using the variable.

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

Answer-> copy.copy() creates a shallow copy of an object. For a list, it creates a new list object, but the elements of the new list are references to the same objects as the original list. If any of the elements are mutable objects, changes made to those objects will be reflected in both the original list and the copied list.

copy.deepcopy() creates a deep copy of an object. It recursively copies all the objects contained within the original object, including nested lists and objects, creating new independent objects. Changes made to the copied list or its elements will not affect the original list.