1. What exactly is []?

**Ans)** It is used to represent a list. It is a dynamic data structure which can store data of different data types and the values in it can be accessed by indexing.

1. 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)** We can use indexing method to assign ‘hello’ as third values in the list spam.

**Syntax**:

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.

1. What is the value of c?

**Ans)** ‘d’

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

**Ans)**  The value will be ‘d’.

1. 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.

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

**Ans)** 1. It returns the first occurrence index.

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

**Ans)**  The list will change as [3.14, 'cat,' 11, 'cat,' True, 99]. The function append( ) will add the value at the end.

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

**Ans)** [3.14, 11, 'cat', True]. It removes the first occurrence of the word ‘cat’.

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

**Ans) List Concatenation Operator (+):** The + operator can be used to concatenate two or more lists, resulting in a new list that contains all the elements from the concatenated lists in the specified order.

list1 = [1, 2, 3]

list2 = [4, 5, 6]

concatenated\_list = list1 + list2

print(concatenated\_list)

**List Replication Operator (\*):** The **\*** operator can be used to replicate a list by a specified number of times, creating a new list that contains multiple copies of the original list.

list1 = [1, 2, 3]

replicated\_list = list1 \* 3

print(replicated\_list) # Output: [1, 2, 3, 1, 2, 3, 1, 2, 3]

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

**Ans)** append( ) is used to add element at end of the list where insert( ) is used to add element at a specific position.

Example :

Append:

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list) # Output: [1, 2, 3, 4]

Insert :

my\_list = [1, 2, 3]

my\_list.insert(1, 4)

print(my\_list) # Output: [1, 4, 2, 3]

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

**Ans) remove( )** : It is used to remove the first occurrence of the specified element.

**pop( ) :** It is used to remove the element by indexing.

1. Describe how list values and string values are identical.

**Ans)** List values and string values in Python share some similarities, but they are fundamentally different data types with distinct characteristics. Here are some ways in which list values and string values are similar:

**Both are ordered sequences:** Both lists and strings are ordered sequences of elements. The elements in both data types have a specific order, and you can access individual elements by their index.

**Both support indexing and slicing:** Lists and strings can be accessed using indexing and slicing operations. You can use square brackets and an index or a range of indices to retrieve specific elements or sub-sequences from both data types.

**Both support iteration:** You can iterate over both lists and strings using loops, such as for loops. In each iteration, you can access individual elements within the list or string.

**Both support len() function:** Both lists and strings support the len() function, which returns the number of elements in the list or the number of characters in the string.

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

**Ans)**  Lists are mutable data types where as tuples are immutable data types. list is represented as [ ]. Tuples are represented as ( ). The values in tuples cannot be change once they are assigned.

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

**Ans)** To create a tuple value that only contains the integer 42, you can enclose the integer within parentheses.

**Syntax :**

my\_tuple = (42,)

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

**Ans)** To get list values in the form of a tuple we use **tuple( ).** To get tuple values in list form we can use **list( ).** Which converts one datatype to another.

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

**Ans)** Variables that "contain" list values in Python do not actually contain the list directly. Instead, they contain a reference or a pointer to the list object stored in memory. In other words, the variable holds the memory address where the list is stored.

When you assign a list to a variable, the variable becomes a reference to that list. It points to the memory location where the list data is stored. This is known as a reference or a reference variable.

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

**Ans)** copy.copy() creates a shallow copy where the top-level object is copied, but the nested objects are referenced. copy.deepcopy() creates a deep copy, recursively copying all objects, including nested objects, resulting in independent copies. The choice between the two depends on whether you need a shallow copy or a deep copy, considering the behavior of nested objects.