1. What exactly is []?

Ans🡪 [] is used to denote an empty list. A list is a built-in data type in Python that allows you to store and organize multiple values in a sequence.

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

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

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

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

Ans🡪 [3.14, 11, 'cat', True, 99]

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

Ans🡪 The list concatenation operator in Python is the + symbol. It is used to combine two or more lists into a single list.

Code-

list1 = [1, 2, 3]

list2 = [4, 5, 6]

concatenated\_list = list1 + list2

print(concatenated\_list)

Output-

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

The list replication operator in Python is the \* symbol. It allows you to create a new list by repeating an existing list a certain number of times.

Code-

list1 = [1, 2, 3]

replicated\_list = list1 \* 3

print(replicated\_list)

Output-

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

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

Ans🡪 The append() method is used to add an element to the end of a list.

Code-

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list)

Output-

[1, 2, 3, 4]

The insert() method is used to add an element at a specific position in a list.

Code-

my\_list = [1, 2, 3]

my\_list.insert(1, 'hello')

print(my\_list)

Output-

[1, 'hello', 2, 3]

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

Ans🡪The two methods for removing items from a list in Python are remove() and pop().

The remove() method is used to remove the first occurrence of a specified value from a list.

The pop() method is used to remove an element from a specific index in a list.

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

Ans🡪 Sequential Data: Both lists and strings are sequential data types. They store a collection of items that can be accessed by their indices.

Indexing and Slicing: Both lists and strings allow indexing and slicing operations. Individual elements or subsequence can be accessed using square brackets [] and the corresponding indices or slices.

Iteration: Lists and strings can be iterated over using loops or comprehensions to access each element one by one.

Length: Both lists and strings have a length that can be determined using the len() function. It returns the number of elements (items or characters) present in the list or string.

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

Ans🡪 Mutability: Tuples are immutable, meaning their elements cannot be modified after creation. Once a tuple is created, its elements and their order remain fixed. On the other hand, lists are mutable, allowing for modifications to their elements.

Syntax: Tuples are defined using parentheses () or without any enclosing symbols, separating elements with commas. Lists, on the other hand, are defined using square brackets [].

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

Ans🡪 my\_tuple = (42,)

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

Ans🡪

Code for List to Tuple-

my\_list = [1, 2, 3, 4]

my\_tuple = tuple(my\_list)

print(my\_tuple)

Output-

(1, 2, 3, 4)

Code for Tuple to List-

my\_tuple = (1, 2, 3, 4)

my\_list = list(my\_tuple)

print(my\_list)

Output-

[1, 2, 3, 4]

16. 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 itself. Instead, they contain a reference or a pointer to the list object in memory.

Code-

my\_list = [1, 2, 3]

another\_list = my\_list

my\_list.append(4) # Modifying the list through the 'my\_list' variable

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

In this example, both my\_list and another\_list refer to the same list object. Modifying the list by appending an element through the my\_list variable also affects the another\_list variable because they point to the same list object in memory.

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

Ans🡪 copy.copy() code-

import copy

original\_list = [1, 2, [3, 4]]

copied\_list = copy.copy(original\_list)

copied\_list[2].append(5)

print(original\_list) # Output: [1, 2, [3, 4, 5]]

print(copied\_list) # Output: [1, 2, [3, 4, 5]]

copy.deepcopy() code-

import copy

original\_list = [1, 2, [3, 4]]

deepcopied\_list = copy.deepcopy(original\_list)

deepcopied\_list[2].append(5)

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

print(deepcopied\_list) # Output: [1, 2, [3, 4, 5]]