1. What exactly is []?

**Ans:** In Python, [] represents an empty list. It is a literal notation used to create a list with no elements.

Eg: my\_list = []

print(my\_list) # Output: []

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.)

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.

**Ans:** To assign the value 'hello' as the third value in the list stored in the variable spam, you can use the index notation and assign the value directly.

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

spam[2] = 'hello'

3.What is the value of spam[int(int('3' \* 2) / 11)]?

**Ans:** To determine the value of **spam[int(int('3' \* 2) / 11)]**, let's break it down step by step:

**'3' \* 2** multiplies the string **'3'** by **2**, resulting in **'33'**.

**int('33')** converts the string **'33'** to an integer, resulting in the value **33**.

**int('33') / 11** performs integer division, dividing **33** by **11**, resulting in **3**.

**spam[3]** retrieves the value at index **3** from the list or string **spam**.

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

**Ans:** In Python, the expression spam[-1] is used to access the last element of a list or the last character of a string.

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

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

**Ans:** In Python, the expression spam[:2] is used to slice a list named spam and retrieve the elements from index 0 up to, but not including, index 2. The value of spam[:2] would be a new list containing the first two elements of the original list spam.

Now, considering the list bacon = [3.14, 'cat', 11, 'cat', True], let's evaluate bacon[:2]

Eg:

bacon[:2] = [3.14, 'cat']

output:

Therefore, the value of bacon[:2] is a new list [3.14, 'cat'] containing the first two elements of the bacon list.

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

**Ans:** Cannot be determine the exact index of 'cat' within it.

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

**Ans:** bacon = [1, 2, 3, 4, 5]

bacon.append(99)

print(bacon)

output:

[1,2,3,4,5,99]

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

**Ans:** bacon=['cat','lion','tiger','cat']

bacon.remove('cat')

print(bacon)

output:

cat,lion,tiger

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

**Ans:** List Concatenation Operator (+): The plus operator is used to concatenate two or more lists together, creating a new list that contains all the elements from the operands.

list1 = [1, 2, 3]

list2 = [4, 5, 6]

concatenated\_list = list1 + list2

print(concatenated\_list)

output:

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

List Replication Operator (\*): The asterisk operator is used to replicate a list by multiplying it with an integer. It creates a new list by repeating the elements of the original list a specified number of times.

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:** append() adds an element to the end of a list, while insert() allows you to add an element at a specific position within the list.

Append eg:

my\_list = [1, 2, 3]

my\_list.append(4)

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

insert() eg:

my\_list = [1, 2, 3]

my\_list.insert(1, 4)

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

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

**Ans:** 1) Using the remove() method:

my\_list = [1, 2, 3, 4, 5]

my\_list.remove(3)

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

2) Using the del statement:

my\_list = [1, 2, 3, 4, 5]

del my\_list[2]

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

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

**Ans:** my\_list = [1, 2, 3, 4, 5]

my\_string = "Hello"

print(my\_list[0]) # Output: 1

print(my\_string[1]) # Output: 'e'

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

**Ans:** Tuples are immutable and typically used for collections of related values, while lists are mutable and commonly used for dynamic collections that can be modified.

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

**Ans:** tuple\_value = (42,)

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

**Ans:** To convert a list value into its tuple form, you can use the tuple() function in Python

my\_list = [1, 2, 3, 4, 5]

my\_tuple = tuple(my\_list)

print(my\_tuple)

output:

(1, 2, 3, 4, 5)

On the other hand, to convert a tuple value into its list form, you can use the list() function.

my\_tuple = (1, 2, 3, 4, 5)

my\_list = list(my\_tuple)

print(my\_list)

output:

[1, 2, 3, 4, 5]

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

**Ans:** # Creating a list and assigning it to two variables

list1 = [1, 2, 3]

list2 = list1

# Modifying the list through one variable

list1.append(4)

# Accessing the modified list through the other variable

print(list2) # Output: [1, 2, 3, 4]

In this example, list1 and list2 both refer to the same list object. Modifying the list through **l**ist1 by appending an element also affects list2, as they both point to the same list in memory.

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

**Ans:** copy.copy() creates a new object with references to the original object's child objects, while copy.deepcopy() creates a new object with recursively copied child objects, resulting in a completely independent copy.