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

[] is the empty list

**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.)**

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)]?**

d

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

d

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

**[‘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')?**

1

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

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

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

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

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

Concatenation operators: Is used to combine or concatenate two or more lists.

List = list1 + list2

Replication operators: Used to replicate the list as per given no of times

List = list1\*n

N = integer

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

append() as element to the end of the list, whereas insert elements insert element at given index position. Append() takes only one argument , while insert takes two arguments viz. index, element to insert

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

* 1. Remove() method: removes first occurrence of given element. Takes one argument viz. element to be removed
  2. Pop() method: removes element at specified index and returns element removed. Takes one argument viz. index of element to be removed

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

Lists and strings share similarities with sequence of elements with indexing and slicing capabilities but they different in terms of mutability, where strings are immutable. Also list can contain any data type while strings only contains characters.

my\_l = [2,3,4]

my\_str = “Hello”

1. Accessing elements by index:

print(my\_l[0]) #o/p: 2

my\_str[1] #o/p: e

2. Slicing:

my\_l[1:]. #o/p: [3,4]

my\_str[:3]. #o/p: ‘Hel’

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

Mutability: Tuples are immutable while lists are mutable

Syntax: Tuples syntax is (), while lists is []

Usage: Tuples used majorly for data that do not change with time e.g coordinates, constants, database records, while lists used to store data that changes over time

Performance: Tuples are more memory efficient and faster to process compared to lists

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

my\_tuple = (42,)

, is essential to use after a number to tell python that it is tuple with single value, else python may interpret it as simple integer value.

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

* 1. **List values’s tuple form**

**#to convert list to tuple, tuple() function is used**

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

print(tuple(my\_list))

* 1. **Tuple values list form**

**#to convert tuple to list, list() function is used**

my\_tuple = (4,5,6,7)

print(list(my\_tuple))

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

It contains reference to the memory locations where the list objects are stored, hence if two variables refers to the same list object then changes made to one variable also reflect as changes in the other variable as both variables refer to the same list object.

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

**copy.copy():**

- creates shallow copies of object, where references to the original elements are maintained in the copies object.

- so changes made to one affect copies object also

- this is more memory efficient way of copying objects

**copy.deepcopy():**

- creates deep copy of object, deep copy ensures copied object is fully independent of the original object

- changes made to one do not affect other object

- less memory efficient than copy.copy()