1. What exactly is []?

Ans: The square brackets tell Python that this is a list comprehension, producing a 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.)

Ans:

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

spam[2] = ‘hello’

output: [2, 4, 'hello', 8, 10]

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:

spam[int(int('3' * 2) / 11)] = spam[int(int('33') / 11)] = spam(int(33/11)) = spam(3) = **'d'**

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

Ans:

spam[-1] is the Value of the last index in the list. That is **'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

index() : gives the index of the first found element in the list

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 meaning in bacon?

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

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

Ans:

The operator for list concatenation is +

The operator for replication is *

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

Ans:

append() will add values only to the end of a list

insert() can add them anywhere in the list

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

Ans: del statement and the remove() list method are two ways to remove values from a list.

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

Ans :

Both Lists and Strings can be

1. Iterable
2. Indexed
3. Used in for loops
4. Split and concatenate
5. Used with in and not in operators

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

Ans:

List	Tuple
Lists are mutable	Tuple are immutable
Implication of iterations is Time-consuming	Implication of iterations is comparatively Faster
The list is better for performing operations, such as insertion and deletion.	Tuple data type is appropriate for accessing the elements
Lists consume more memory	Tuple consume less memory as compared to the list
Lists have several built-in methods	Tuple does no have must built-in methods

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

Ans: (42,) (The trailing comma is mandatory.)

a = (42)	a = (42,)
type(a)	type(a)
Output: int	Output: tuple

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

Ans: list() , tuple()

```
l = ['a','b']

t = (1,2)

tuple(l)

list(t)
```

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

Ans : They contain references to list values.

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

Ans :

Shallow Copy : **copy.copy()**

A shallow copy creates a new object which stores the reference of the original elements. So, a shallow copy doesn't create a copy of nested objects, instead it just copies the reference of **nested** objects. This means, a copy process does not recurse or create copies of nested objects itself.

1. **Normal List:** If we have **Normal list** then if we create another list using **shallow copy**: **These lists are going to refer Different memory Locations.**
2. **Nested List:** If we create a **nested list** then if we create another list using **shallow copy**: **These lists are going to refer Same memory Locations.**
   Item will not get copied, But if we're making the changes the object. Then it is going to change both the lists

Deep Copy : **copy.deepcopy()**

A deep copy creates a new object and recursively adds the copies of nested objects present in the original elements.

1. **Normal List:** In a normal list shallow copy equals to deep copy(**Different memory Locations**)
2. **Nested List: :** If we create a **nested list** then if we create another list using **shallow copy**: **These lists are going to refer Different memory Locations.**