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

**Answer:** The symbol [] is represented as an 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.)**

**Answer:** spam= [2, 4, 6, 8, 10]  
spam[2]='hello'  
spam

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)]?**

**Answer:** ‘d’

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

**Answer:** ‘d’

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

**Answer:** ['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')?**

**Answer:** 1, it takes the 1st occurrence of ‘cat’ from the given list

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

**Answer:** [3.14, 'cat', 11, 'cat', True, 99]

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

**Answer:** [3.14, 11, 'cat', True, 99]

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

**Answer:** List concatenation is denoted by symbol ‘+’ where items of 2 lists are added one after the other. Ex: l1=[2,3,4,5]   
l2=[6,7,8]  
l3=l1+l2  
Output: [2,3,4,5,6,7,8]

List replication is denoted by symbol ‘\*’ where the items of a list are repeated as per the number given by the user.  
Ex: l4=l2\*2  
Output: [6,7,8,6,7,8]

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

**Answer:** Append: It is used to permanently add an item to the end of a list. It takes exactly one parameter.   
Ex: l1=[1,2,3]  
l1.append(‘abc’)  
Output of l1: [1,2,3,’abc’]

Insert: It is used to add an item to the specified index of the list. It takes 2 parameters, one will be the index and other be the item  
Ex: l1=[1,2,3,4,5]  
l1.insert (2, “abc”)  
Output of l1= [1, 2, 'abc', 3, 4, 5]

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

**Answer:** Two methods of removing items from a list are:  
pop(): By default it removes the last item of the list, but we can give an input as the index of the item to be removed as well.   
remove(): It always removes the 1st occurrence of an element from the list.

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

**Answer:** Index: Both list and string elements can be accessed using indexing  
Divide: Both list and strings can be sliced to get a specific portion of elements.  
Loop: Both list and strings can be iterated over without any issue

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

**Answer:** The main difference between lists and tuples is that tuples are immutable whereas lists are mutable.  
Tuple, once created, cannot be modified whereas lists can be modified using append(), insert()

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

**Answer:** tuple1 = (23,)  
Here the comma after 23 is very much important as it indicates the tuple is having a single element in it.

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

**Answer:** We can do the same by type casting as   
Ex: list1 = [1, 2, 3, 4, 5]  
tuple1 = tuple(list1)  
print(tuple1)  
here the list is converted to tuple

tuple1 = (1, 2, 3, 4, 5)  
list1 = list(tuple1)  
print(list1)  
Here the tuple is converted to list

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

**Answer:** Variables that "contain" list values are not necessarily lists themselves, they are referenced as objects. These variable point to the memory location where the list objects are present.

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

**Answer:** copy.copy() creates a new object and copies the references of the original object’s elements into it. Changes made to the original object do reflect in the copied object as well.  
Ex of copy.copy()  
import copy  
original\_list = [1, [2, 3]]  
copied\_list = copy.copy(original\_list)  
original\_list[0] = 10  
original\_list[1].append(4)  
print(original\_list)  
print(copied\_list)

Output: [10, [2, 3, 4]]  
[1, [2, 3, 4]]

copy.deepcopy() creates a deep copy where a completely independent copy of the object and its nested objects is created. Any modification of the original list does not make changes to the copied list objects  
Ex of copy.deepcopy()  
import copy  
original\_list = [1, [2, 3]]  
copied\_list = copy.deepcopy(original\_list)  
original\_list[0] = 10  
original\_list[1].append(4)  
print(original\_list)  
print(copied\_list)

Output: [10, [2, 3, 4]]  
[1, [2, 3]]