1. What exactly is []?

In Python, square brackets ([]) are used to create a list. A list is a mutable data type that can store a collection of objects. The objects in a list can be of any type, including strings, integers, floats, and objects.

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)]? → spam[3] → ‘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]

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

In Python, the list concatenation operator is the plus sign (+), and the list replication operator is the asterisk (*).

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

The main difference between the append() and insert() methods is that append() adds an item to the end of a list, while insert() adds an item to a specific index in a list.

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

The pop() method is used to remove the last element from a list.

The remove(item) method is used to remove the specified item from a list.

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

1. List values are mutable, while string values are immutable. This means that you can change the contents of a list after it has been created, but you cannot change the contents of a string after it has been created.

2. List values can contain any type of object, while string values can only contain characters. This means that you can store a list of integers, strings, objects, or any other type of object in a list, but you can only store characters in a string.

3. List values are iterable, while string values are also iterable. This means that you can iterate over the items in a list using a for loop, and you can also iterate over the characters in a string using a for loop.

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

1. Tuples and lists are both data structures in Python that can be used to store collections of data. However, there are some key differences between the two.

2. Tuples are immutable, while lists are mutable. This means that once you create a tuple, you cannot change its contents. However, you can change the contents of a list after it has been created.

3. Tuples are typically used to store data that does not need to be changed, such as the names of people in a group or the dates of important events. Lists are typically used to store data that may need to be changed, such as a shopping list or a list of tasks to complete.

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

tuple=42,

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

list(tuple)

tuple(list)

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

Variables that "contain" list values are not necessarily lists themselves. Instead, they contain references to lists. A reference is a pointer to an object in memory. When you assign a list to a variable, the variable is assigned a reference to the list, not the list itself. This means that if you change the list, the variable will still point to the new list.

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

The Python copy module provides two functions for copying objects: copy.copy() and copy.deepcopy(). The main difference between the two functions is that copy.copy() creates a shallow copy of the object, while copy.deepcopy() creates a deep copy of the object.

A shallow copy creates a new object that contains the same values as the original object. However, the new object does not contain any references to the objects that are contained in the original object. This means that if you change the original object, the changes will be reflected in the new object.

A deep copy creates a new object that contains the same values as the original object, and also contains references to the objects that are contained in the original object. This means that if you change the original object, the changes will not be reflected in the new object.