1. What exactly is []?

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, 4, 6, 8, 10]

spam[2] = 'hello'

print(spam)

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 = ['a', 'b', 'c', 'd']

int('3' \* 2) / 11 = 3

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]?

spam = ['a', 'b', 'c', 'd']

first\_two\_elements = spam[:2]

print(first\_two\_elements)

prints ['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')?

bacon = [3.14, 'cat', 11, 'cat', True]

index = bacon.index('cat')

print(index)

1.

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

bacon = [3.14, 'cat', 11, 'cat', True]

bacon.append(99)

print(bacon)

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

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

bacon = [3.14, 'cat', 11, 'cat', True]

bacon.remove('cat')

print(bacon)

[3.14, 11, 'cat', True]

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

List concatenation is the process of combining two or more lists into a single list. The + operator is used to concatenate lists. For example, the following code will create a new list that is the concatenation of the lists a and b:

a = [1, 2, 3]

b = [4, 5, 6]

c = a + b

print(c)

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

List replication is the process of creating a new list that is a copy of an existing list repeated a specified number of times. The \* operator is used to replicate lists. For example, the following code will create a new list that is a copy of the list a repeated 3 times:

a = [1, 2, 3]

b = a \* 3

print(b)

[1, 2, 3, 1, 2, 3, 1, 2, 3]

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

The append() and insert() methods are used to add elements to a list in Python. The main difference between the two methods is that append() adds an element to the end of the list, while insert() adds an element to a specific position in the list.

append()Adds an element to the end of the list.

insert()Adds an element to a specific position in the list.

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

remove() method: This method removes the first occurrence of the specified value from the list.

pop() method: This method removes and returns the item at the specified index from the list.

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

List values and string values are not identical in Python. They are both sequences of data, but they have different types and can contain different data.

A list is a sequence of objects, where each object can be any type (an integer, a float, a string, etc.). A string is a sequence of characters.

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

Feature Tuple List

Mutability Immutable Mutable

Speed Faster Slower

Memory usage Less More

Use cases Data structures that need to be immutable Data structures that need to be mutable

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

tuple\_value = (42,)

The trailing comma is necessary to indicate that the tuple contains only one element. Otherwise, Python would interpret the code as an attempt to create a list with two elements, the first of which is the integer 42 and the second of which is an empty tuple.

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

Operation Syntax

Convert a list to a tuple tuple(list\_value)

Convert a tuple to a list list(tuple\_value)

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

Variables that "contain" list values do not actually contain lists directly. Instead, they contain references to list values. A reference is a pointer to an object in memory. When you assign a list value to a variable, Python creates a reference to the list object and stores the reference in the variable.

This is an important distinction because it means that changes to the list object will be reflected in the variable that refers to the list. For example, if you have a variable called list\_value that contains a list, and you then add an element to the list, the variable list\_value will also be updated to reflect the change.

This is in contrast to immutable objects, such as strings. When you assign an immutable object to a variable, Python creates a copy of the object and stores the copy in the variable. Changes to the original object will not be reflected in the variable.

list\_value = [1, 2, 3]

list\_value.append(4)

print(list\_value)

[1, 2, 3, 4]

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

copy.copy() creates a shallow copy of an object. This means that the copy contains a reference to the same underlying data as the original object. Any changes made to the original object will be reflected in the copy.

copy.deepcopy() creates a deep copy of an object. This means that the copy contains a new reference to each of the object's data elements. Changes made to the original object will not be reflected in the copy.