1.What exactly is []?

[] is an empty list in Python. It is a data structure that can hold an ordered collection of items.

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.)

You can assign the value 'hello' as the third value in the list spam using the following code:

arduino

Copy code

spam[2] = 'hello'

3.What is the value of spam[int(int('3' \* 2) / 11)]?

The value of spam[int(int('3' \* 2) / 11)] is 'd'.

The string '3' is concatenated with itself to form '33'. This is then converted to an integer using int('33'), which is 33.

The expression int('33') / 11 results in 3.0.

Since we're using integer division (//), the result is truncated to 3.

spam[3] is 'd', so that is the final value.

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

The value of spam[-1] is the last item in the list spam, which is 'd'.

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

The value of spam[:2] is a new list containing the first two items of spam, which are ['a', 'b'].

5.What is the value of bacon.index('cat')?

The value of bacon.index('cat') is 1.

This method returns the index of the first occurrence of 'cat' in the list bacon.

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

bacon.append(99) adds the value 99 to the end of the list bacon.

The list bacon will now have the value [3.14, 'cat', 11, 'cat', True, 99].

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

bacon.remove('cat') removes the first occurrence of 'cat' from the list bacon.

The list bacon will now have the value [3.14, 11, 'cat', True].

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

The list concatenation operator is +, which combines two lists into a new list.

The list replication operator is \*, which creates a new list by repeating an existing list a specified number of times.

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

append() adds a new item to the end of the list.

insert() adds a new item to a specific position in the list by shifting the other items to make room.

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

The two methods for removing items from a list are remove() and pop().

remove() removes the first occurrence of a specified item from the list.

pop() removes and returns the item at a specified index in the list.

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

Both list values and string values are sequences of values that can be indexed and sliced.

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

Tuples and lists are both used to store collections of values, but there are some key

Sure, here are the questions and answers:

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

To type a tuple value that only contains the integer 42, you would write (42,), with a comma after the integer to indicate that it is a tuple and not just an integer.

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

To get a list value's tuple form, you can use the tuple() function, passing in the list as an argument. For example, if my\_list = [1, 2, 3], you could get its tuple form with my\_tuple = tuple(my\_list). To get a tuple value's list form, you can use the list() function, passing in the tuple as an argument. For example, if my\_tuple = (1, 2, 3), you could get its list form with my\_list = list(my\_tuple).

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

Variables that "contain" list values are typically references or pointers to a location in memory where the list is stored. This means that the variable does not actually contain the list itself, but rather a reference to its location in memory. When you modify the list through the variable, you are actually modifying the list in memory, not the variable itself.

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

copy.copy() creates a shallow copy of an object, meaning that it creates a new object with a new memory address, but the contents of the object still reference the same memory addresses as the original object. This means that if the contents of the original object are modified, the contents of the copied object will also be modified. copy.deepcopy() creates a deep copy of an object, meaning that it creates a new object with new memory addresses for all of its contents. This means that modifying the original object will not affect the copied object, and vice versa.