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.)

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)]?

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

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

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')?

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

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

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

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

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

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

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

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

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

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

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

ANSWER

1. `[]` represents an empty list in Python. It is a data structure that can hold a collection of values, and it is mutable, meaning you can add, remove, or modify elements in the list.

2. To assign the value 'hello' as the third value in a list stored in the variable `spam`, you can use indexing like this:

```python

spam[2] = 'hello'

```

3. The value of `spam[int(int('3' \* 2) / 11)]` is `'d'`. Here's the breakdown:

- `'3' \* 2` creates the string `'33'`.

- `int('33')` converts it to the integer 33.

- `33 / 11` equals 3.

- So, `spam[3]` refers to the fourth element of the list, which is `'d'`.

4. The value of `spam[-1]` is also `'d'`. In Python, negative indices count from the end of the list, so `-1` refers to the last element of the list.

5. The value of `spam[:2]` is `['a', 'b']`. This is a list slicing operation that extracts elements from the beginning of the list up to, but not including, the element at index 2.

6. The value of `bacon.index('cat')` is 1. It returns the index of the first occurrence of the string 'cat' in the list `bacon`.

7. The `bacon.append(99)` method adds the integer 99 to the end of the list `bacon`. After this operation, the list will look like this: `[3.14, 'cat', 11, 'cat', True, 99]`.

8. The `bacon.remove('cat')` method removes the first occurrence of the string 'cat' from the list `bacon`. After this operation, the list will look like this: `[3.14, 11, 'cat', True, 99]`.

9. List Concatenation Operator: `+` is used to concatenate two or more lists, combining their elements into a new list.

List Replication Operator: `\*` is used to replicate a list by repeating it a specified number of times.

10. The `append()` method is used to add an element to the end of a list. The `insert()` method is used to insert an element at a specified position within the list, shifting existing elements to make room for the new one.

11. The two methods for removing items from a list are:

- `remove()`: Removes the first occurrence of a specified value.

- `pop()`: Removes an element at a specified index (or the last element if no index is specified) and returns the removed value.

12. Both list values and string values are ordered sequences of elements. They can be indexed and sliced, and they support iteration. However, list values can contain elements of different data types and are mutable (can be changed), while string values are immutable (cannot be changed after creation) and consist of characters.

13. Tuples and lists are both ordered collections of values in Python, but they have the following differences:

- Lists are mutable (can be modified), while tuples are immutable (cannot be modified after creation).

- Lists are defined using square brackets `[]`, while tuples use parentheses `()`.

- Lists are typically used for collections of items where the order and content may change, while tuples are used for fixed collections of items that should not change.

14. To create a tuple value containing only the integer 42, you can do it like this:

```python

my\_tuple = (42,)

```

The comma is required to indicate that it's a tuple with one element.

15. To convert a list to a tuple, you can use the `tuple()` constructor:

```python

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

```

To convert a tuple to a list, you can use the `list()` constructor:

```python

my\_tuple = (1, 2, 3)

my\_list = list(my\_tuple)

```

16. Variables that "contain" list values store references to the list objects in memory. They do not contain the actual list data but point to where the list data is stored. This means that if you assign the same list to multiple variables, they all refer to the same list in memory. Modifying the list through one variable will affect all references to that list.

17. `copy.copy()` is used to create a shallow copy of an object. In the context of lists, it creates a new list object but copies references to the elements within the original list. If the elements are mutable (e.g., lists within a list), changes to the elements in the new list will affect the original list and vice versa.

`copy.deepcopy()` is used to create a deep copy of an object. It creates a completely independent copy of the object and all its nested objects. Changes to the elements in the new copy will not affect the original object. This is useful when dealing with nested structures like lists of lists.