1. What exactly is []?

2. In a list of values stored in a variable called spam, how would you assign the value &#39;hello&#39; as the

third value? (Assume [2, 4, 6, 8, 10] are in spam.)

Let&#39;s pretend the spam includes the list [&#39;a&#39;, &#39;b&#39;, &#39;c&#39;, &#39;d&#39;] for the next three queries.

3. What is the value of spam[int(int(&#39;3&#39; \* 2) / 11)]?

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

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

Let&#39;s pretend bacon has the list [3.14, &#39;cat,&#39; 11, &#39;cat,&#39; True] for the next three questions.

6. What is the value of bacon.index(&#39;cat&#39;)?

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

8. How does bacon.remove(&#39;cat&#39;) 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&#39;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&#39;s tuple form? How do you get a tuple value&#39;s list form?

16. Variables that &quot;contain&quot; list values are not necessarily lists themselves. Instead, what do they

contain?

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

SoLUTIONS

1. [] represents an empty list in Python.
2. To assign the value 'hello' as the third value in a list called spam, you would do the following:

spam[2] = 'hello'

1. The value of **spam[int(int('3' \* 2) / 11)]** is **'d'**.
2. The value of **spam[-1]** is **'d'**.
3. The value of **spam[:2]** is **['a', 'b']**.
4. The value of **bacon.index('cat')** is **1**.
5. **bacon.append(99)** changes the look of the list value in bacon by adding the integer **99** to the end of the list. The new list would look like this: **[3.14, 'cat', 11, 'cat', True, 99]**.
6. **bacon.remove('cat')** changes the look of the list in bacon by removing the first occurrence of the string **'cat'**. The new list would look like this: **[3.14, 11, 'cat', True]**.
7. The list concatenation operator is **+** and the list replication operator is **\***.
8. The difference between the list methods **append()** and **insert()** is that **append()** adds an item to the end of a list, while **insert()** adds an item at a specified index in the list.
9. The two methods for removing items from a list are **remove()** and **pop()**.
10. List values and string values are identical in that they are both sequences of values. However, lists are mutable (i.e., their values can be changed), while strings are immutable (i.e., their values cannot be changed).
11. Tuples and lists are similar in that they are both sequences of values. However, tuples are immutable (i.e., their values cannot be changed), while lists are mutable (i.e., their values can be changed).
12. To type a tuple value that only contains the integer 42, you would do the following:

my\_tuple = (42,)

Note the comma after the integer 42, as this is necessary to specify that it is a tuple with a single value.

1. To get a list value's tuple form, you would use the **tuple()** function. To get a tuple value's list form, you would use the **list()** function. For example:

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

my\_tuple = (1, 2, 3)

my\_list = list(my\_tuple)

2 / 2

1. [] represents an empty list in Python.
2. To assign the value 'hello' as the third value in a list called spam, you would do the following:

pythonCopy code

spam[2] = 'hello'

1. The value of **spam[int(int('3' \* 2) / 11)]** is **'d'**.
2. The value of **spam[-1]** is **'d'**.
3. The value of **spam[:2]** is **['a', 'b']**.
4. The value of **bacon.index('cat')** is **1**.
5. **bacon.append(99)** changes the look of the list value in bacon by adding the integer **99** to the end of the list. The new list would look like this: **[3.14, 'cat', 11, 'cat', True, 99]**.
6. **bacon.remove('cat')** changes the look of the list in bacon by removing the first occurrence of the string **'cat'**. The new list would look like this: **[3.14, 11, 'cat', True]**.
7. The list concatenation operator is **+** and the list replication operator is **\***.
8. The difference between the list methods **append()** and **insert()** is that **append()** adds an item to the end of a list, while **insert()** adds an item at a specified index in the list.
9. The two methods for removing items from a list are **remove()** and **pop()**.
10. List values and string values are identical in that they are both sequences of values. However, lists are mutable (i.e., their values can be changed), while strings are immutable (i.e., their values cannot be changed).
11. Tuples and lists are similar in that they are both sequences of values. However, tuples are immutable (i.e., their values cannot be changed), while lists are mutable (i.e., their values can be changed).
12. To type a tuple value that only contains the integer 42, you would do the following:

makefileCopy code

my\_tuple = (42,)

Note the comma after the integer 42, as this is necessary to specify that it is a tuple with a single value.

1. To get a list value's tuple form, you would use the **tuple()** function. To get a tuple value's list form, you would use the **list()** function. For example:

scssCopy code

my\_list = [1, 2, 3] my\_tuple = tuple(my\_list) my\_tuple = (1, 2, 3) my\_list = list(my\_tuple)

1. Variables that "contain" list values actually contain references to the lists, not the actual lists themselves.
2. The difference between **copy.copy()** and **copy.deepcopy()** is that **copy.copy()** creates a shallow copy of a list, meaning that if the original list contains references to other objects (e.g., lists), then the copied list will contain references to the same objects as the original list. On the other hand, **copy.deepcopy()** creates a deep copy of a list, meaning that if the original list contains references to other objects