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()?

1. `[]` represents an empty list in Python.
2. `spam[2] = 'hello'`
3. The value is `'d'`.
4. The value is `'d'`.
5. The value is `['a', 'b']`.
6. The value is `1`.
7. `bacon` becomes `[3.14, 'cat', 11, 'cat', True, 99]`.
8. `bacon` becomes `[3.14, 11, 'cat', True]`.
9. The list concatenation operator is `+` and the list replication operator is `*`.
10. `append()` adds an element to the end of a list, while `insert()` inserts an element at a specific index.
11. The two methods are `remove()` and `pop()`.
12. Both list and string values can be indexed and sliced, and both support the use of loops and certain built-in functions in Python.
13. Tuples and lists are both used to store collections of items, but tuples are immutable (cannot be changed after creation) while lists are mutable.
14. `(42,)`

15. To get a list value's tuple form, you can use the `tuple()` function. To get a tuple value's list form, you can use the `list()` function.
16. Variables that contain list values are actually references to the list objects stored in memory.
17. `copy.copy()` creates a shallow copy of a list (i.e., a new list with references to the same objects as the original), while `copy.deepcopy()` creates a deep copy of a list (i.e., a new list with new copies of all the objects in the original).