1. What exactly is []?

ANS - In Python, [] is an empty list. It is a collection data type that is ordered and changeable. Lists are written with square brackets and can have any number of items separated by commas. Here’s an example of how to create an empty list in Python:

my\_list = []

You can also create a list with items in it like this:

my\_list = [1, 2, 3]

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.)

ANS - To assign the value ‘hello’ as the third value in a list of values stored in a variable called spam, you can use the following code:

spam = [2, 4, 6, 8, 10]

spam[2] = 'hello'

This will replace the third value (which is 6) with ‘hello’. The result will be:

[2, 4, 'hello', 8, 10]

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)]?

ANS - The value of spam[int(int('3' \* 2) / 11)] is 'd'. Here’s how it works:

'3' \* 2 is the string '33'.

int('33') is the integer 33.

33 / 11 is the integer 3.

spam[3] is the fourth item in the list, which is 'd'.

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

ANS - The value of spam[-1] is 'd'. This is because -1 refers to the last item in the list.

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

ANS - The value of spam[:2] is ['a', 'b']. This is because the slice notation [:2] means “give me all the items from the beginning of the list up to (but not including) the item at index 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')?

ANS - The value of bacon.index('cat') is 1. This is because the first occurrence of 'cat' in the list is at index 1.

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

ANS- After calling bacon.append(99), the list value in bacon will be [3.14, 'cat', 11, 'cat', True, 99]. This is because the append() method adds the argument passed to it (in this case, the integer 99) to the end of the list.

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

ANS- After calling bacon.remove('cat'), the list value in bacon will be [3.14, 11, 'cat', True]. This is because the remove() method removes the first occurrence of its argument (in this case, the string 'cat') from the list.

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

ANS - In Python, the list concatenation operator is +. It is used to combine two or more lists into a single list. Here’s an example:

list1 = [1, 2, 3]

list2 = [4, 5, 6]

list3 = list1 + list2

print(list3)

Output:

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

The list replication operator is \*. It is used to create a new list by repeating an existing list a certain number of times. Here’s an example:

list1 = [1, 2, 3]

list2 = list1 \* 3

print(list2)

Output:

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

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

ANS- In Python, the append() method is used to add an item to the end of a list. Here’s an example:

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list)

Output:

[1, 2, 3, 4]

The insert() method is used to add an item to a specific position in a list. Here’s an example:

my\_list = [1, 2, 3]

my\_list.insert(1, 'hello')

print(my\_list)

Output:

[1, 'hello', 2, 3]

So the main difference between the two methods is that append() adds an item to the end of the list, while insert() adds an item to a specific position in the list.

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

ANS - In Python, there are two methods for removing items from a list: remove() and pop().

The remove() method is used to remove the first occurrence of an item from a list. Here’s an example:

my\_list = [1, 2, 3]

my\_list.remove(2)

print(my\_list)

Output:

[1, 3]

The pop() method is used to remove an item from a specific position in a list. Here’s an example:

my\_list = [1, 2, 3]

my\_list.pop(1)

print(my\_list)

Output:

[1, 3]

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

ANS - In Python, both list values and string values are ordered sequences of values. This means that they can be indexed and sliced using the same syntax. For example:

my\_list = [1, 2, 3]

my\_string = 'hello'

print(my\_list[0]) # Output: 1

print(my\_string[0]) # Output: 'h'

print(my\_list[1:]) # Output: [2, 3]

print(my\_string[1:]) # Output: 'ello'

Both list values and string values can also be iterated over using a for loop. For example:

my\_list = [1, 2, 3]

my\_string = 'hello'

for item in my\_list:

print(item)

for char in my\_string:

print(char)

Output:

1

2

3

h

e

l

l

o

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

ANS - In Python, the main difference between tuples and lists is that tuples are immutable (i.e., their values cannot be changed), while lists are mutable (i.e., their values can be changed).

Here’s an example of how to create a tuple:

my\_tuple = (1, 2, 3)

And here’s an example of how to create a list:

my\_list = [1, 2, 3]

To change a value in a list, you can use indexing and assignment:

my\_list[0] = 4

But if you try to do the same thing with a tuple, you’ll get an error:

my\_tuple[0] = 4 # TypeError: 'tuple' object does not support item assignment

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

ANS - To create a tuple value that only contains the integer 42, you can use the following syntax:

my\_tuple = (42,)

Note the comma after the integer value. This is necessary to indicate that this is a tuple with only one value.

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

ANS - In Python, you can convert a list to a tuple using the tuple() function:

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

And you can convert a tuple to a list using the list() function:

my\_tuple = (1, 2, 3)

my\_list = list(my\_tuple)

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

ANS - In Python, variables that “contain” list values are actually storing references to the list objects in memory. This is true for all mutable data types in Python

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

ANS - In Python’s copy module, copy.copy() creates a shallow copy of an object, while copy.deepcopy() creates a deep copy of an object.

A shallow copy creates a new object that stores references to the same memory locations as the original object. This means that if you modify one of the objects, the other will also be modified.

A deep copy creates a new object with its own memory locations. This means that if you modify one of the objects, the other will not be modified.