1. What exactly is []?

**Ans:** It creates an empty list.

1. 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:** The 3rd value i.e. ‘6’ can be changed with ‘hello’ by using the following commands:

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

print(spam) # It will print spam as [2, 4, 6, 8, 10]

spam[2]="hello"

print(spam) # It will print spam as [2, 4, 'hello', 8, 10]

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.

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

**Ans:** ‘d’

**Explanation:** int(‘3’\*2) is 33. int(int('3' \* 2) / 11) is 33/11 = 3. spam[3] = d.

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

**Ans:** ‘d’ (It will take last value of the list as the index of last value is -1)

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

**Ans:** It will return the first two elements i.e. [‘a’, ‘b’]

Let's pretend bacon has the list [3.14, 'cat,' 11, 'cat,' True] for the next three questions.

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

**Ans:** 1 (as it returns the index of 1st occurrence of ‘cat’, which is 1)

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

**Ans:** It will add another element as 99 at the end of the list bacon.

Output will be: [3.14, 'cat', 11, 'cat', True, 99]

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

**Ans:** It will remove the 1st occurrence of ‘cat’ in the list and gives the output as:

[3.14, 11, 'cat', True]

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

**Ans:** The operator for list concatenation is +, while that for replication is \*

Example of concatenation:

list1=[1, "Cat"]

list2=[-20, "Banana"]

list3=list1+list2

print(list3)

Output of the above code is: [1, 'Cat', -20, 'Banana']

Example of replication:

list1=[1, "Cat"]

print(list1\*2)

Output of the above code is: [1, 'Cat', 1, 'Cat']

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

**Ans:** append() will insert element at the end of a list, while insert will add element before the desired index.

Example of append():

list1=[1, "Cat"]

list1.append("Tiger")

print(list1)

Output of the above code is: [1, 'Cat', 'Tiger']

list1=[1, "Cat"]

list1.insert(1,"Tiger")

print(list1)

Output of the above code is: [1, 'Tiger', 'Cat']

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

**Ans:** There are two methods for removing items from a list: (i) using remove() and (ii) pop() functions.

Example 1: remove() function, removes a particular (specified) item from a list.

list1=[1, "Cat", "Car", -20, "Bus"]

list1.remove(-20)

print(list1)

Output of the above code is: [1, 'Cat', 'Car', 'Bus']

Example 2: pop() function removes the last item from the list

list1=[1, "Cat", "Car", -20, "Bus"]

list1.pop()

print(list1)

Output of the above code is: [1, 'Cat', 'Car', -20]

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

**Ans:** Following are the similarity between list values and string values in Python:

* Lists and strings both are sequence types in Python i.e. the elements are arranged in a specific order.
* Both lists and strings support indexing. We can access the required elements of a list or string by using the positions of the elements.
* Desired elements from both lists and strings can be extracted by slicing the lists or strings.
* It is possible to iterate both lists and strings using different loops.
* len() function can be used to determine the number of characters in a string or the number of elements in a list.

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

**Ans:** Following are the differences between tuples and lists:

* Tuples can be defined using parentheses (), whereas lists can be defined using square brekets [].
* Tuples are immutable i.e. the elements of a tuple cannot be changed once the tuple is created. Whereas, lists are mutable i.e. the elements of a list can be modified.
* Tuples are generally used for indicating a collection of related items. Whereas, lists can contain any type of data/items.

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

**Ans:**

tup=(42,)

print(tup)

type(tup)

Output of the above code is:

(42,)

tuple

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

Ans: To get list value’s tuple form:

list1=[1, "Cat", "Car", -20, "Bus"]

print(list1)

tup1=tuple(list1) # To convert list to tuple.

print(tup1)

Output of above code is:

[1, 'Cat', 'Car', -20, 'Bus']

(1, 'Cat', 'Car', -20, 'Bus')

To get tuple value’s list form

tup1=(1, "Cat", "Car", -20, "Bus")

print(tup1)

list1=list(tup1) # To convert tuple to list.

print(list1)

Output of the above code is:

(1, 'Cat', 'Car', -20, 'Bus')

[1, 'Cat', 'Car', -20, 'Bus']

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

**Ans:** They contain references to the items in the list.

list2=[10, -30, "Battery"]

list3=list2

print(list3)

list3.append("Apple") # It will append 'apple' in both list2 as well as in list3

print(list2)

Output of the above code is:

[10, -30, 'Battery']

[10, -30, 'Battery', 'Apple']

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

**Ans:**

copy.copy(): It creates a new object and inhabits it with references to the same nested objects as they are in the original.

copy.deepcopy(): It creates an entirely independent copy of the object and all of its nested objects.

Example:

import copy

list1=[[1, "Cat", "Car"], [-20, "Bus"]]

list2=copy.copy(list1) # copy the references of list1

list3=copy.deepcopy(list1) # copy the items of list1

list1[0][0]=200 # Changes the 1st item of the list1 and list2 to 200 i.e. 1 changes to 200.

print(list1)

print(list2)

print(list3)

Output of the above code is:

[[200, 'Cat', 'Car'], [-20, 'Bus']]

[[200, 'Cat', 'Car'], [-20, 'Bus']]

[[1, 'Cat', 'Car'], [-20, 'Bus']]