1. What exactly is []?

[ ] is a empty list, like a =[ ]

# empty list

my\_list = []

In Python, a list is created by placing elements inside square brackets [], separated by commas.

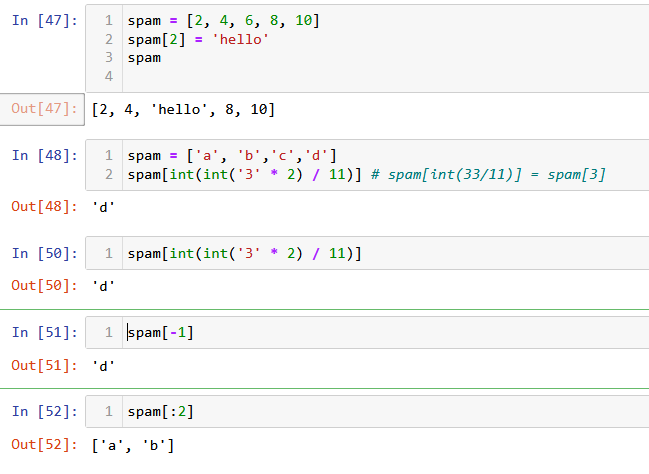
A list can have any number of items and they may be of different types (integer, float, string, etc.).

# list with mixed data types

my\_list = [1, "Hello", 3.4]

A list can also have another list as an item. This is called a nested list.

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.)

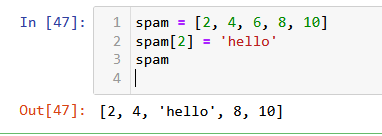


*spam = [2, 4, 6, 8, 10]*

*spam[2] = 'hello'*

*spam*

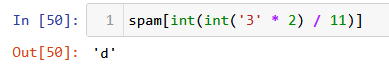
*[2, 4, 'hello', 8, 10]*



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

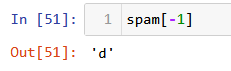
3. What is the value of spam[int(int('3' \* 2) / 11)]?

**Output : d**



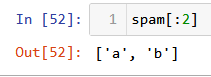
4. What is the value of spam[-1]?

**Output : d**



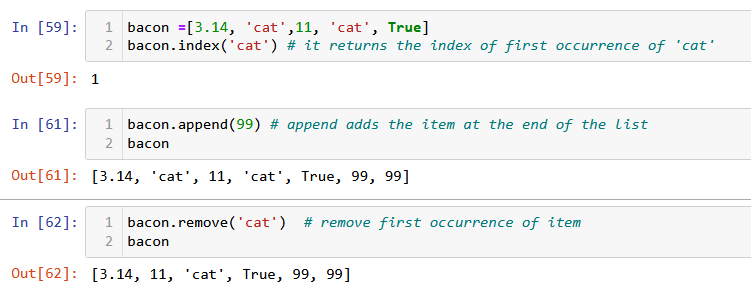
5. What is the value of spam[:2]?

**Output : [‘a’,’b’]**

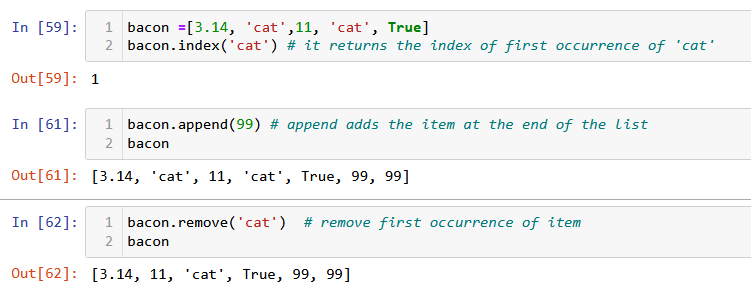


Let's pretend bacon has the list [3.14, 'cat,' 11, 'cat,' True] for the next three questions.

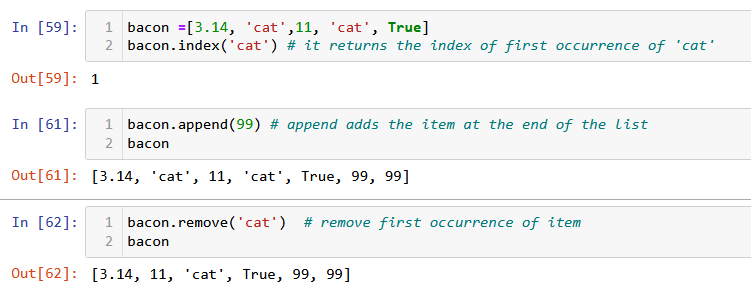
6. What is the value of bacon.index('cat')?



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?

Concatenation is done by **+** operator. Concatenation is supported by sequence data types(string, list, tuple). Concatenation is done between the **same data types**only.

Concatenating two lists will result in a new list object, but calling extend method will update the original list itself. Its return type is None.

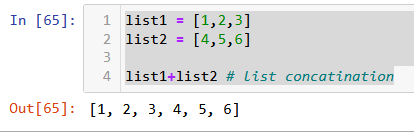
The most conventional method to perform the list concatenation, the use of “+” operator can easily add the whole of one list behind the other list and hence perform the concatenation.

In python, ( \* ) is list replication operator ( + ) is list concatenation operator

list1 = [1,2,3]

list2 = [4,5,6]

list1+list2 # list concatenation

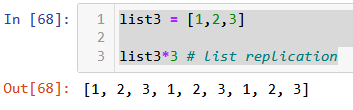


Sequences datatypes (both mutable and immutable) support a **repetition**operator \* The repetition operator **\*** will make multiple copies of that particular object and combines them together. When \* is used with an integer it performs multiplication but with list, tuple or strings it performs a repetition

Items in the sequence*s* are not copied but are referenced multiple times.

list3 = [1,2,3]

list3\*3 # list replication



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

|  |  |
| --- | --- |
| .append() | insert() |
| The .append() method adds an additional element to the **end** of an already existing list.  When using .append(), the original list gets modified. No new list gets created. | If you don't want to just add items to the end of a list, you can specify the position you want to add them with .insert(). |
| .append() adds an item to the **end** of a list | .insert() inserts and item **in a specified position** in the list. |
| Length of the list increases by 1 | Length of the list increases by 1 |
| * list\_name is the name you've given the list. * .append() is the list method for adding an item to the end of list\_name. * item is the specified individual item you want to add. | * list\_name is the name of the list. * .insert() is the list method for inserting an item in a list. * position is the first argument to the method. It's always an integer - specifically it's the index number of the position where you want the new item to be placed. * item is the second argument to the method. Here you specify the new item you want to add to the list. |
| Syntax:  list\_name.append(item) | Syntax:  list\_name.insert(position,item) |

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

In Python, there are many methods available on the list data type that helps you to remove an element from a given list. The methods are **remove(), pop()** and **clear().**

|  |  |
| --- | --- |
| Method | Description |
| remove() | It helps to remove the very first given element matching from the list. |
| pop() | The pop() method removes an element from the list based on the index given. |
| clear() | The clear() method will remove all the elements present in the list. |

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

The similarity between Lists and Strings in Python is that both are sequences. The differences between them are that firstly, Lists are mutable but Strings are immutable. Secondly, elements of a list can be of different types whereas a String only contains characters that are all of String type.

Strings and lists are similar, but they are not same. One simple difference between strings and lists is that lists can any type of data i.e. integers, characters, strings etc, while strings can only hold a set of characters.

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

|  |  |
| --- | --- |
| List | Tuple |
| Lists are mutable | Tuples are immutable |
| The implication of iterations is Time-consuming | The implication of iterations is comparatively Faster |
| The list is better for performing operations, such as insertion and deletion. | Tuple data type is appropriate for accessing the elements |
| Lists consume more memory | Tuple consume less memory as compared to the list |
| Lists have several built-in methods | Tuple does not have many built-in methods. |
| The unexpected changes and errors are more likely to occur | In tuple, it is hard to take place. |

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

(42,) (The trailing comma is mandatory.)

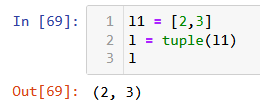
tuple = (42,)

tuple

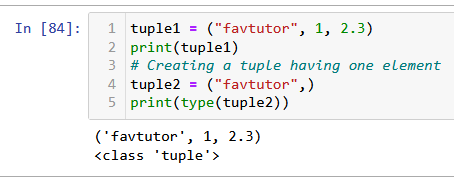
15. How do you get a list value's tuple form? How do you get a tuple value's list form?

The tuple() and list() functions, respectively

Get a list value's tuple form



Get a tuple value's list form



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

Variables will contain references to list values rather than list values themselves. But for strings and integer values, variables simply contain the string or integer value.

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

The copy.copy() function will do a shallow copy of a list, while the copy.deepcopy() function will do a deep copy of a list. That is, only copy.deepcopy() will duplicate any lists inside the list.