**1. What exactly is []?**

[] is an empty list

[] is used to define "list literals," allowing us to declare a list and its contents in our program.

Index brackets are also used to write expressions that evaluate to a single item within a list, or a single character in a string.

**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"

print(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)]?**

spam=['a', 'b', 'c', 'd']

spam[int(int('3' \* 2) / 11)]

output will be ‘d’

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

d

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

['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')?**

1

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

[3.14, 'cat', 11, 'cat', True, 99]

It appends the item at the end of list as last element

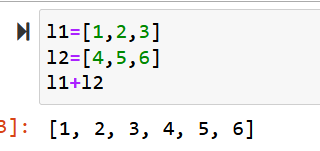
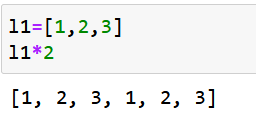
**8. How does bacon.remove('cat') change the look of the list in bacon?**

[3.14, 11, 'cat', True]

It removes the first occurrence of the element named cat

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

The operator for list concatenation is +, while the operator for replication is \*.

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

append() adds an item to the end of a list

insert() inserts an item in a specified position in the list

Graphical user interface, text, chat or text message

Description automatically generated Text, chat or text message

Description automatically generated with medium confidence

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

The methods are remove(), pop() and clear().

remove() helps to remove the very first given element matching from the list.

The pop() method removes an element from the list based on the index given.

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.
* a string's length is the number of characters in the string; a list's length is the number of items in the list. Each character in a string as well as each item in a list has a position, also called an index.
* Lists are similar to strings, which are ordered collections of characters, except that the elements of a list can be of any type.

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

The primary difference between tuples and lists is that tuples are immutable as opposed to lists which are mutable.

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

**t = (42,)**

Graphical user interface, text, application

Description automatically generated

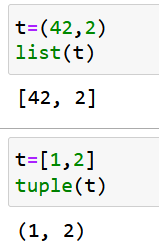
**15. How do you get a list value's tuple form? How do you get a tuple value's list form?**

get a list value's tuple form :

Using the tuple() built-in function, an iterable can be passed as an input to the tuple () function, which will convert it to a tuple object. If you want to convert a Python list to a tuple, you can use the tuple() function to pass the full list as an argument, and it will return the tuple data type as an output.

get a tuple value's list form :

To convert a tuple into list in Python, call list() builtin function and pass the tuple as argument to the function. list() returns a new list generated from the items of the given tuple.



**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() returns a shallow copy of the list, and deepcopy() returns a deep copy of the list.
* both have the same value but have different IDs.
* copy() create reference to original object. If you change copied object - you change the original object. . deepcopy() creates new object and does real copying of original object to new one. Changing new deepcopied object doesn't affect original object.

|  |  |
| --- | --- |
| * Shallow copy is faster. * Deep copy is comparatively slower. |  |

### Deep copy

**Syntax:** copy.deepcopy(x)

### Shallow copy

**Syntax:** copy.copy(x)

| **Shallow Copy** | **Deep Copy** |
| --- | --- |
| 1. Shallow Copy stores the references of objects to the original memory address. | 1. Deep copy stores copies of the object’s value. |
| 1. Shallow Copy reflects changes made to the new/copied object in the original object. | 1. Deep copy doesn’t reflect changes made to the new/copied object in the original object. |
| 1. Shallow Copy stores the copy of the original object and points the references to the objects. | 1. Deep copy stores the copy of the original object and recursively copies the objects as well. |
| 1. Shallow copy is faster. | 1. Deep copy is comparatively slower. |