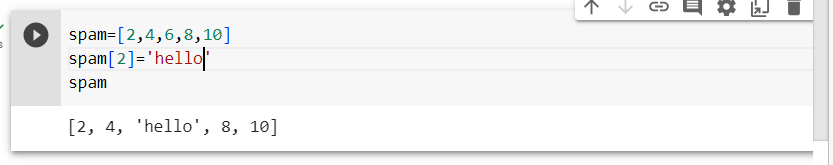
**1. What exactly is []?**

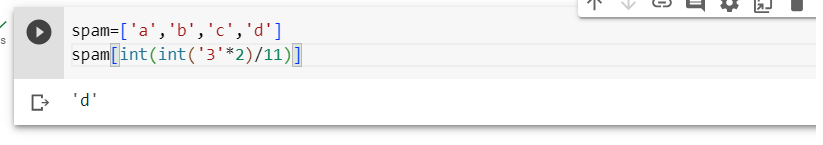
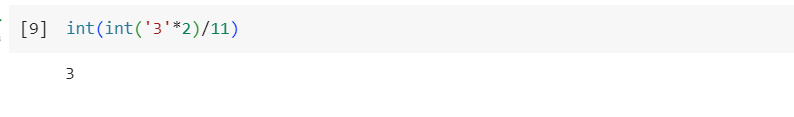
Ans.) [] is a empty list value, which is a list value that contains no items.

**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.) 

**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)]?**

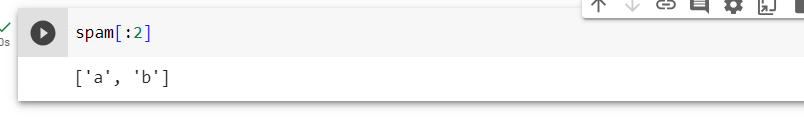
Ans.) 

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

Ans.) A picture containing text, screenshot

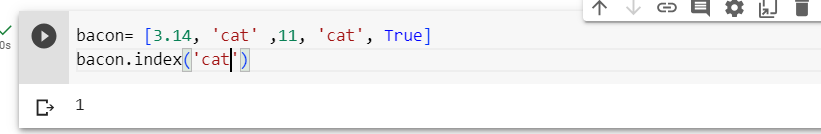
Description automatically generated

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

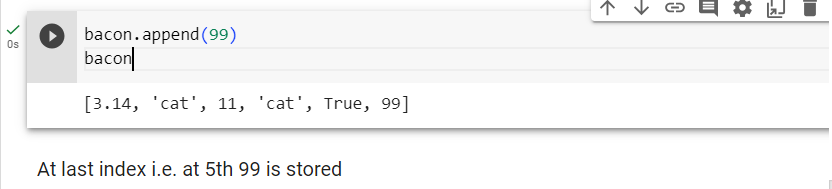
Ans.) 

**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.) ****

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

Ans.) 

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

Ans.)

A picture containing text, screenshot, font, line

Description automatically generated

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

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

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

Ans.) While append() will add values only to the end of a list, insert() can add them anywhere in the list.

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

Ans.) The del statement and the remove() list method are two ways to remove values from a list.

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

Ans.) Both lists and strings can be passed to len(), have indexes and slices, be used in for loops, be concatenated or replicated, and be used with the in and not in operators.

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

Ans.)

|  |  |
| --- | --- |
| **Tuples** | **Lists** |
| Tuples are immutable, meaning their elements cannot be modified once they are defined. | Lists are mutable, allowing their elements to be modified, appended, removed, or reordered. |
| Tuples are defined using parentheses ( ) or without any delimiters. | Lists are defined using square brackets [ ] |
| Tuples are generally more memory-efficient and provide slightly faster performance compared to lists. | As List is mutable,it require more memory due to potential resizing, and their operations can be slower than tuples. |
| Tuples have fewer built-in methods due to their immutability. However, they can be concatenated, sliced, and unpacked. | Lists provide a wider range of built-in methods for adding, removing, and manipulating elements, such as append(), remove(), insert(), and extend(). |
| Ex.)my\_tuple = (1, 2, 3) or my\_tuple = 1, 2, 3. | Ex.)my\_list = [1, 2, 3] |

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

Ans.) A tuple value that only contains the integer 42 can be created as follows:

my\_tuple = (42)

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

Ans.) To convert a list to a tuple, we can use the tuple() function.

Ex.) A picture containing text, font, screenshot, line

Description automatically generated

To convert a tuple to a list, you can use the list() function.

Ex.) A picture containing text, font, line, white

Description automatically generated

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

Ans.) Variables that "contain" list values in Python do not actually contain the list itself, but rather a reference to the list object in memory. This applies to all mutable objects, not just lists.

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

Ans.)

|  |  |
| --- | --- |
| **copy.copy()** | **copy.deepcopy()** |
| The copy.copy() function will do a shallow copy of a list | copy.deepcopy() function will do a deep copy of a list |
| Shallow copying is useful when you want to create a new object with the same values as the original but without affecting the original object if the mutable objects inside are modified. | Deep copying is useful when you need to create an independent and self-contained copy of an object hierarchy. Here's an example that demons |