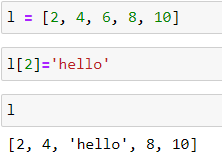
**1. What exactly is []?**

**Ans.** The square brackets are used to create a list in python. Eg. L = [1,2,3,4,5,6]

**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.** ‘d’

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

**Ans.** ‘d’

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

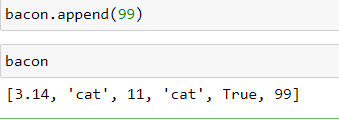
**Ans.** [‘a’,’b’,’c’]

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

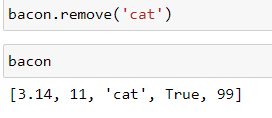
**6. What is the value of bacon.index('cat')?**

**Ans.** 1

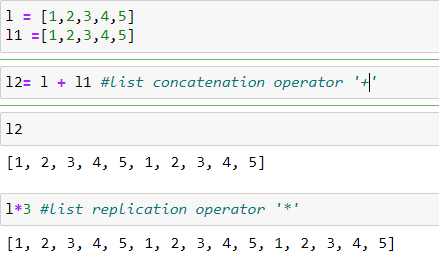
**7. How does bacon.append(99) change the look of the list value in bacon?**

**Ans. **

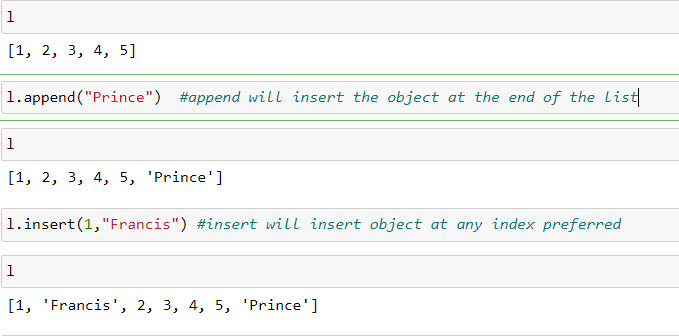
**8. How does bacon.remove(‘cat’) change the look of the list in bacon?**

**Ans. **

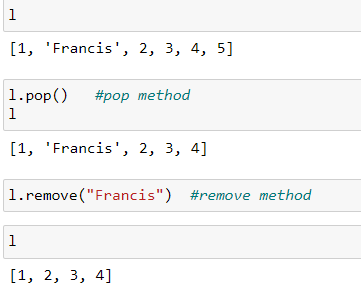
**9. What are the list concatenation and list replication operators?**

**Ans. **

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

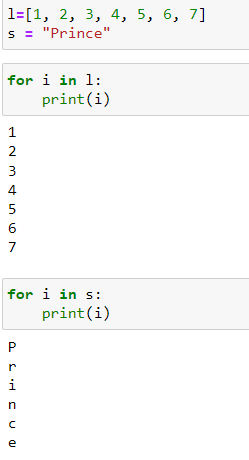
**Ans. **

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

**Ans. **

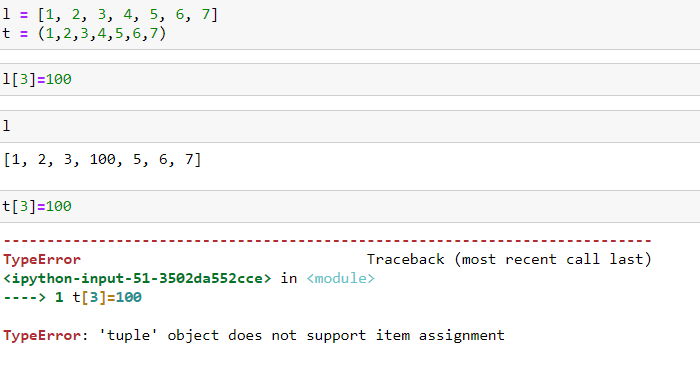
**12. Describe how list values and string values are identical.**

**Ans.** list and string values can be iterated through. They are iterables.

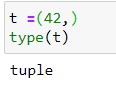


**13. What’s the difference between tuples and lists?**

**Ans.** Tuples are immutable whereas lists are mutable.

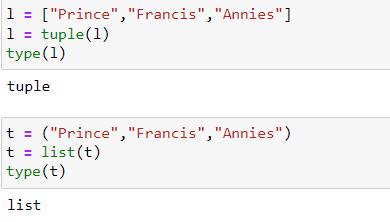


**14. How do you type a tuple value that only contains the integer 42?**

**Ans. **

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

**Ans.** We have to use Type casting to do that.

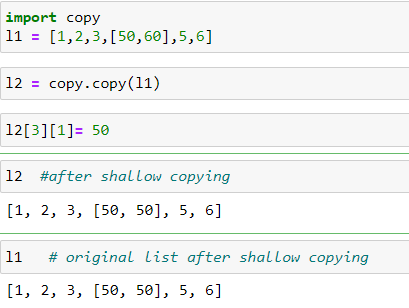


**16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?**

**Ans. **

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

**Ans.** shall copy creates a new compound object however it will not copy the the child objects to the new compound object instead will reference the child objects to the new object. A change in the new compound objects will make a change in the original objects.



Deep copy creates a new compound object and one by one copies each child object to the new compound object. A change in new child objects will not impact or change the values in the original object.

