Assignment 4:

Submitted by Dr. Abhishek V. Hukkerikar

Note: The answers are in blue color

1. What exactly is []?

[] are used to enclose the contents of list. [] indicates an empty 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)]?

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

Basically, its 33/11 which is 3. Since 3 is in quotes, ('3'*2) means 3 after 3 or 33. As the indexing in python starts from 0, the 3rd element corresponds to 'd'.

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

spam[-1] also corresponds to 'd' as the using negative digits within square brackets indicates negative indexing. The first letter from the reverse order of the list is d. Therefore, spam[-1] is 'd'.

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

spam[:2] indicates the first three entries of the list. However, the last number is exclusive. Therefore, the third element is not added to the list. **Thus, the answer is ['a','b'].**

If the question was spam[0:2], the 0th element is inclusive while the third element (2nd) entry is exclusive.

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')?

The answer is 1. Though, cat has been repeated twice in the list, the index value of the first appearance will be returned.

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

The value of 99 will be added to the list at the last. The list will look like as following:

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

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

So, only the first appearance is removed. The list will look like the following:

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

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

Concatenation is the combining two or more lists into a single list whereas replication of the list is repeating the list two or more times. The following code explains the difference:

```
l1 = [1,3,'ineuron', 'fsds', 2.0]
l2 = [117, 'Abhishek', 'student', 2.0]
# Concatenation operation
conc_list = l1 + l2
# List replication
rep_list = l1*2
print(conc_list, rep_list)
```

```
[1, 3, 'ineuron', 'fsds', 2.0, 117, 'Abhishek', 'student', 2.0] [1, 3, 'ineuron', 'fsds', 2.0, 1, 3, 'ineuron', 'fsds', 2.0
]
```

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

The append method will add an element only at the end of the list while the insert method can add an element anywhere in the list depending on the index given.

```
l1 = [1,3,'ineuron', 'fsds', 2.0]
l2 = [117, 'Abhishek', 'student', 2.0]
l1.append('additional element')
l2.insert(3, 'iNeuron')
print(l1,l2)
```

```
[1, 3, 'ineuron', 'fsds', 2.0, 'additional element'] [117, 'Abhishek', 'student', 'iNeuron', 2.0]
```

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

Items could be removed by:

1. remove(): Specify the element to be removed and it will remove the first occurrence/appearance of the item.

```
list = [1, 2, 3, 4, 2]
list.remove(2)
print(list)
→ [1, 3, 4, 2]
```

2. pop (): Removes the item based on the index number. If nothing is given, it removes the last item of the list. It can also return the removed item.

```
list = [1, 2, 3, 4]
item = list.pop(1)
```

```
print(list)
# Output: [1, 3, 4]
print(item)
```
→ 2

3. del: Removes item with a specified index or even range.

```
list = [1, 2, 3, 4]
del list[1]
print(list)
```
→ [1, 3, 4]

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

Both the list and strings have indexes and slices. The len() function gives the len of both the types. Additionally, they can be used in loops, concatenated or replicated.

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

Lists are denoted by square brackets and are mutable. This means that elements can be added, removed or even changed. Whereas tuples are immutable. Once the data is added to tuples, no changes are allowed thereafter. Tuples are denoted by () common brackets.

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

t = (42), where t is the tuple comprising of 42

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

Kindly note that the question is a bit confusing. A better sentence would be *How to convert a list to tuple and vice versa?*

By using tuple() and list[] functions respectively.

```
list = [1, 2, 3, 4]
tuple_form = tuple(list)
print(tuple_form)
```
→ (1, 2, 3, 4)

Similarly,

```
tuple = (1, 2, 3, 4)
list_form = list(tuple)
print(list_form)
```
→ [1, 2, 3, 4]

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

The variables that have list values are actually references to the list objects. They do not contain list values themselves. When a new list is created and assigned to the variable, the variable would contain the reference to the list and not the list per se. This means any changes to the variable would also lead to the change in the list elements. Kindly take a look at the following code:

```
L1 = [1,2,3]
```

```
L2 = L1
L2[1] = 2.1
print(L1)
```
→ [1,2.1,3]

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

The difference is shallow copy by copy.copy() and deep copy by deepcopy(). Take a look at the following code:

```
L1 = [1,[2,3],4]
L2 = list(L1)
print(L1 is L2)
L2[1][0] = 'changed element'
print(L1)

# Deep copy
import copy
L4 = [1,[2,3],4]
L5 = copy.deepcopy(L1)
print(L1 is L2)
L4[1][0] = 'changed element'
print(L4)
```