Assignment - 4

1. What exactly is []?

[] is an empty list. Square braces used to represent the specific datatype called as list.

```
In [1]: l=[]

In [2]: type(l)
Out[2]: 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.)

```
In [3]: spam = [2,4,6,8,10]

        print(spam[2])    #identifying the third value

        spam[2] = "hello" #assigning the string to third value

        spam

        6

Out[3]: [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)]?

```
In [5]: spam = ['a', 'b', 'c', 'd']
        spam[int(int('3' * 2) / 11)]
Out[5]: 'd'
```

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

```
In [6]: spam = ['a', 'b', 'c', 'd']
        spam[-1]
Out[6]: 'd'
```

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

```
In [7]: spam = ['a', 'b', 'c', 'd']
        spam[:2]
Out[7]: ['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')?

```
In [10]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.index('cat')

Out[10]: 1
```

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

```
In [12]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.append(99)
         bacon

Out[12]: [3.14, 'cat', 11, 'cat', True, 99]
```

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

```
In [13]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.remove('cat')
         bacon

Out[13]: [3.14, 11, 'cat', True]
```

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

Plus(+) is a list concatenation operator.

```
In [14]: spam = ['a', 'b', 'c', 'd']
         bacon = [3.14, 'cat', 11, 'cat', True]
         # plus(+) is a list concatenation operator
         spam + bacon

Out[14]: ['a', 'b', 'c', 'd', 3.14, 'cat', 11, 'cat', True]
```

Asterisk(*) is a list replication operator.

```
In [15]: spam = ['a', 'b', 'c', 'd']

         # asterisk(*) is a list replication operator
         spam * 2

Out[15]: ['a', 'b', 'c', 'd', 'a', 'b', 'c', 'd']
```

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

Append() is used to add the element at the end of the list.

```
In [16]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.append(99)
         bacon

Out[16]: [3.14, 'cat', 11, 'cat', True, 99]
```

Insert() is used to add an element at the specific index of the list.

```
In [17]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.insert(3,99)
         bacon

Out[17]: [3.14, 'cat', 11, 99, 'cat', True]
```

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

1. pop

```
In [18]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.pop(1)
         bacon

Out[18]: [3.14, 11, 'cat', True]
```

```
bacon = [3.14, 'cat', 11, 'cat', True]
bacon.pop(1)
bacon
```

```
Signature: bacon.pop(index=-1, /)
Docstring:
Remove and return item at index (default last).
```

2. remove

```
In [19]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.remove(3.14)
         bacon

Out[19]: ['cat', 11, 'cat', True]
```

```
bacon = [3.14, 'cat', 11, 'cat', True]
bacon.remove(3.14)
bacon
```

```
Signature: bacon.remove(value, /)
Docstring:
Remove first occurrence of value.
```

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

List values and string values have similar indexing.

```
_list = ["a","b","c","d","e"]
for i in _list:
    print(i, _list.index(i))
```

```
a 0
b 1
c 2
d 3
e 4
```

```
_string = "abcde"
for i in _list:
    print(i, _string.index(i))
```

```
a 0
b 1
c 2
d 3
e 4
```

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

Tuple	List
Tuple is immutable	List is mutable
Implication of iterations is comparatively faster	Implication of iterations is time-consuming
Tuple does not have many built-in methods	Lists have several built-in methods
Tuples consume less memory as compared to list	The list is better for performing operations such as insertion and deletion

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

```
In [23]: _tuple = (42)
         _tuple
```

```
Out[23]: 42
```

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

```
In [24]: #tuple converted to list
         _tuple = ("a", "b", "c", "d")
         list(_tuple)

Out[24]: ['a', 'b', 'c', 'd']

In [25]: #list converted to tuple
         _list = ["a", "b", "c", "d"]
         tuple(_list)

Out[25]: ('a', 'b', 'c', 'd')
```

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

```
In [26]: variable = 'a', 'b', 'c', [1,2,3], 3.14, 'cat', 11, 'cat', True
         print(type(variable))
         print(type([1,2,3]))

         <class 'tuple'>
         <class 'list'>
```

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

copy.copy() is a Shallow Copy. In Shallow copy, the changes made to the Copied Object will get reflected to the Original Object.

copy.deepcopy() is a Deep Copy. In Deep copy, the changes made to the Copied Object will not get reflected to the Original Object.