Assignment 4

1. What exactly is []?

[] is a square bracket which represents an empty list. Index value must be defined inside [].

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.)

Snippet:

```
spam = [2,4,6,8,10]
print("Before changing the third value: ",spam)
spam[2]="hello"
print("After changing the third value: ",spam)
```

Output:

Before changing the third value: [2, 4, 6, 8, 10]

After changing the third value: [2, 4, 'hello', 8, 10]

3. What is the value of spam[int(int('3' * 2) / 11)]?

```
spam = [2,4,6,8,10,'a','b','c','d']
print(spam[int(int('3' * 2) / 11)])
```

Explanation:

Here 3 is a string and '3'*2 which will be converted into 33.Then 33/11 is 3

Spam[int(int(3))]=8—which represents the index value 3.

Output: 8

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

Snippet:

```
spam = [2,4,6,8,10,'a','b','c','d']
        print(spam[-1])
```

Explanation:

It will print the last value

If the index value is negative, it will consider from reverse.

Output: d

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

Snippet:

```
spam = [2,4,6,8,10,'a','b','c','d']
        print( spam[:2])
```

Output: [2,4]

Explanation: Here slice operator is used spam[:2] indicated that it will print from the index value 0 and 1. The last index value is excluded.

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

Snippet:

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

Output: 1

Explanation: Here in this list 'cat' has index value 1.

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

Snippet:

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

Output:

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

Explanation:

List will preserve insertion order. So,the value 99 will be added at the end of list.

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

Snippet:

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

Output:

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

[3.14, 11, 'cat', True]

Explanation:

Here in the bacon list 'cat' occurs at two indexes 1 and 3. But after bacon.remove() it will remove the first occurrence only.

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

List concatenation operator(+): It will concatenate the two or more lists.

Snippet:

```
list1=[1,2,3]

list2=[4,5,6]

print(list1+list2)
```

Output:

[1, 2, 3, 4, 5, 6]

List replication operators: It will replicate the lists by multiplying the lists with the given integer value.

Snippet:

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

print(list*2)
```

Output:

[1, 2, 3, 1, 2, 3]

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

The append and insert methods both are used to manipulate the objects in the list, it differs in how the elements are added to the list.

List.append() method will add the element at the end of the list. It may take a single value as an argument I.e. what value should be added to the end of the list.

Snippet:

```
my_list =[1,2,3,4]

my_list.append(5)

print(my_list)
```

Output:

[1,2,3,4,5]

List.insert() method will add the element at the specified location in the list. It takes two arguments where the elements should be added and what the elements should be added.

Snippet:

```
my_list=[1,2,3,4,5]
my_list.insert(1,5)
print(my_list)
```

Output:

[1, 5, 2, 3, 4, 5]

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

The two methods are remove and del method which removes the elements from the list.

- **Remove method – used to remove the first occurrence of the specified element in the list.**

  Snippet:

  ```
  my_list = [1,2,3,2,5]
  my_list.remove(2)
  print(my_list)
  ```

  Output:

  [1,3,2,5]

- **Del method – used to remove the elements from the list by specifying the index.**

Snippet:

```
my_list = [1,2,3,4,5]
del my_list[3]
print(my_list)
```

Output:

[1, 2, 3, 5]

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

In python list values and string values are identical by the following ways.

1. Indexing:

In python both lists values and string values can be accessed with the help of index values. They can be retrieved using the indices of the values.

Snippet:

```
my_num = [1,2,3,4,5]
        print(my_num[2])
        my_string = "hii , surya"
        print(my_string[2])
        Output:
        3
        i
```

2. Slicing:

In python both lists and strings can be sliced to extract the sequence.

Snippet:

```
my_num = [1,2,3,4,5]
        print(my_num[0:2])
        my_string = "hii , surya"
        print(my_string[2:5])
        Output:
        [1, 2]
        i ,
```

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

Lists	Tuples
• **Lists are used to store the collection of items.**	• **Tuples are also used to store collection of items.**
• **Lists are mutable.**	• **Tuples are immutable.**
• **It can be modified, add or delete items after the lists are created.**	• **It cannot be modified, add or delete the items after the tuples are created.**
• **Since it is mutable. It consumes a lot of memory space.**	• **It didn't consume a lot of memory space because it is immutable.**

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

Snippet:

my_num = 42,

print(my_num)

Output:

(42,)

Explanation:

From the above code ‘,’(comma) is used to indicate that it’s a tuple not an integer.

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

Snippet:

my_list = [1,2,3,4,5]

my_tuple = (1,2,3)

print(list(my_tuple))

print(tuple(my_list))

Output:

(1,2,3,4,5)

[1,2,3]

Explanation:

From the above snippet, in the first print statement list() function is used which converts tuple to list and in second statement tuple() function is used which converts list to tuple. It is important that only datatypes are changed not the values inside the lists or tuple.

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

Yes, in python variables that "contain" list values are not necessarily lists themselves. Instead, they contain reference to a list object in a memory. In python variables are names referred to the objects. When you assign a list to a variable, the variable holds a reference to the memory location where the list is stored. This means that the variable doesn't directly contain the list data, but rather a reference to that data

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

In Python, the copy.copy() and copy.deepcopy() functions are used to create copies of objects.

copy.copy() performs the shallow copy of the object. It will create the new object copy_list and refer to the orignal_list object. If the value in the copied_list changes it will affect the orginal list.

Snippet:

import copy

```
original_list = [1, [2, 3]]
print(orginal_list)
copied_list = copy.copy(original_list)
copied_list[1][0] = 4
print(original_list)
print(copied_list)
```

Output:

```
[1, [2, 3]]
[1, [4, 3]]
[1, [4, 3]]
```

copy.deepcopy() performs deep copy of the object. It will create an independent copy of the object. The changes made won't affect the orginal_list object.

Snippet:

```
import copy
original_list = [1, [2, 3]]
Print(orginal_list)
copied_list = copy.deepcopy(original_list)
copied_list[1][0] = 4
print(original_list)]
print(copied_list)
```

Output:

```
[1, [2, 3]]
    [1, [2, 3]]
    [1, [4, 3]]
```