Python Basic- Assignment- 04

1. What exactly is []?

Answer:

In Python, [] is an empty list. Lists are a type of data structure used to store and organize data, and are denoted by the use of square brackets.

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.)

Answer:

spam[2] = 'hello'

#Input:

```
spam = [2, 4, 6, 8, 10]
spam[2] = 'Hello'
print(spam)
```

#Output:

```
[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)]?

Answer:

#Input:

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

spam[int(int('3' * 2) / 11)]
```

#Output:

‘d’

Explanation: The value of spam[int(int('3' * 2) / 11)] is 'd'. This is because '3' * 2 evaluates to '33', which is then converted to an integer (33) and divided by 11 to get 3. Finally, 3 is used to index the list, so the value at index 3 is 'd'.

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

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

Answer:

#Input:

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

spam[-1]
```

#Output:

‘d’

Note: negative indexing starts from the end of a list and the first indexing value is -1.

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

Answer:

#Input:

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

spam[:2]
```

#Output:

['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')?

Answer:

#Input:

```
bacon = [3.14, 'cat', 11, 'cat', True]

bacon.index('cat')
```

#Output:

“1”

Done by- Md. Abdullah Al Mahmud;   Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

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

Answer:

#Input:

```
bacon = [3.14, 'cat', 11, 'cat', True]

bacon.append(99)

print(bacon)
```

#Output:

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

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

Answer:

#Input:

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

#Output:

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

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

Answer:

The list concatenation operator is "+" and is used to combine two or more lists together. For example:

```
list_a = [1, 2, 3]
list_b = [4, 5, 6]
list_c = list_a + list_b
print(list_c)
```

# Outputs:

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

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

The list replication operator is "*" and is used to repeat the elements in a list a certain number of times. For example:

```
list_d = [1, 2, 3]
list_e = list_d * 3
print(list_e)
```

# Outputs:

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

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

Answer:

The append() method adds an item to the end of the list.

#Input:

```
a = ['a','b','c']
a.append(1)
print(a)
```

#Output:

```
['a', 'b', 'c', 1]
```

The insert() method adds an item at the specified index. It takes two arguments(1, “Cat”), the first argument(1) for the index of the element, and the second argument(“Cat”) is the element to be inserted.

#Input:

```
a = ['a','b','c']
a.insert(1,"Cat")
print(a)
```

#Output:

```
['a', 'Cat', 'b', 'c']
```

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

Answer:

The two methods for removing items from a list in Python are the “del” statement and the “remove()” list method.

The “del” statement is used to remove an item from a list using its index. For example, del mylist[1] will delete the item in the 1st index(‘orange’) position of the list mylist.

#Input:

```
mylist = ['apple','orange','lemon','jackfruit']
del mylist[1]
print(mylist)
```

#Output:

```
['apple', 'lemon', 'jackfruit']
```

The remove() list method is used to remove an item from a list using its value. For example, mylist.remove('lemon') will remove the string 'lemon' from the list mylist.

#Input:

```
mylist = ['apple','orange','lemon','jackfruit']
mylist.remove('lemon')
print(mylist)
```

#Output:

```
['apple', 'orange', 'jackfruit']
```

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

Answer:

List values and string values are similar in that they are both data structures used to store and manipulate collections of data. Both lists and strings can contain any type of data, including numbers, strings, booleans, and objects. Additionally, both list values and string values can be indexed, which allows for elements within the data structure to be accessed using a numerical index. Finally, operations such as concatenation, slicing, and iteration can be performed on both list values and string values.

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

Answer:

Tuples and lists are both sequence data types in Python, but they differ in several ways.

Tuples are immutable, meaning they cannot be changed or modified. Once you assign values to a tuple, you cannot add, remove, or alter them in any way. On the other hand, lists are mutable, meaning they can be modified after they are created.

Tuples are typically created using parentheses (), while lists are created using square brackets [].

Additionally, tuples can be used as keys in dictionaries, while lists cannot. This is because dictionaries require keys to be immutable.

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

Answer:

A tuple value containing only the integer 42 can be written as follows:

```
my_tuple = (42,)
```

This is known as a singleton tuple, since it only contains one element. The comma at the end is important, as it tells python that the value is a tuple and not just a plain integer.

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

Answer:

To get a list value's tuple form, you can use the built-in tuple() function. This function takes in a list and returns a tuple containing all of the items from the list. For example:

```
list_val = [1,2,3]
tuple_val = tuple(list_val)
print(tuple_val)
```

# Output:

```
(1, 2, 3)
```

To get a tuple value's list form, you can use the built-in list() function. This function takes in a tuple and returns a list containing all of the items from the tuple. For example:

```
tuple_val = (1,2,3)
list_val = list(tuple_val)
print(list_val)
```

# Output:

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

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

Answer:

Variables that contain list values are references to the list. They do not contain the actual list values, but instead contain a reference to the memory location where the list is stored. This allows the variable to point to the same list, even if the list is modified.

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

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

Answer:

copy.copy() creates a shallow copy of an object. This means that it copies the reference to the object and its value. This means that the original object and the new object are linked, so if you make a change to the original object, the new object will also be affected.

Example:

```
original_list = [1,2,3]
shallow_copy = copy.copy(original_list)
shallow_copy[0] = 5
print(shallow_copy) # [5,2,3]
print(original_list) # [5,2,3]
```

copy.deepcopy() creates a deep copy of an object. This means that it copies the object's value and creates a new object that is not linked to the original object. This means that if you make a change to the original object, the new object will not be affected.

Example:

```
original_list = [1,2,3]
deep_copy = copy.deepcopy(original_list)
deep_copy[0] = 5
print(deep_copy) # [5,2,3]
print(original_list) # [1,2,3]
```

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007