Cheat Sheet

Working with Lists

Object

In general, anything that can be assigned to a variable in Python is referred to as an object.

Strings, Integers, Floats, Lists etc. are all objects.

Examples

- "A"
- 1.25
- [1,2,3]

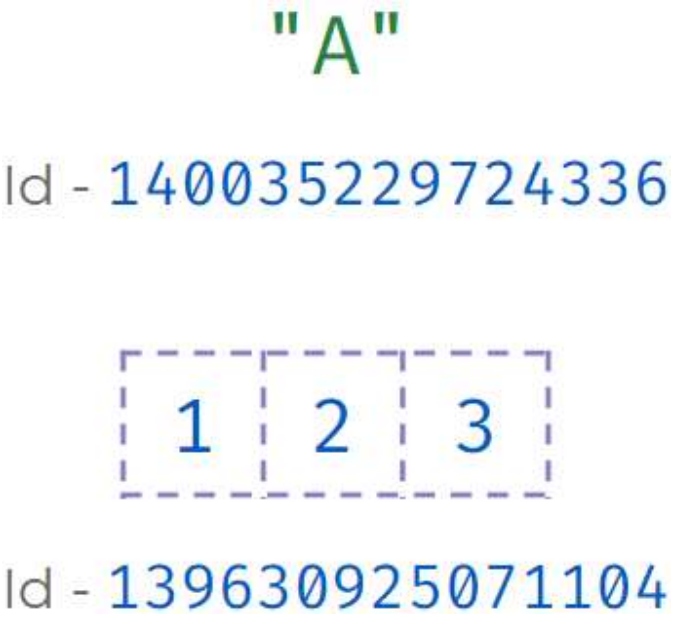
Identity of an Object

Whenever an object is created in Python, it will be given a unique identifier (id).

This unique id can be different for each time you run the program.

Every object that you use in a Python Program will be stored in Computer Memory.

The unique id will be related to the location where the object is stored in the Computer Memory.



Finding Id

We can use the

`id()` to find the id of a object.

Code

PYTHON

```python
print(id("Hello"))
```

Output

```
140589419285168
```

Id of Lists

PYTHON

```python
list_a = [1, 2, 3]
list_b = [1, 2, 3]
print(id(list_a))
print(id(list_b))
```

Output

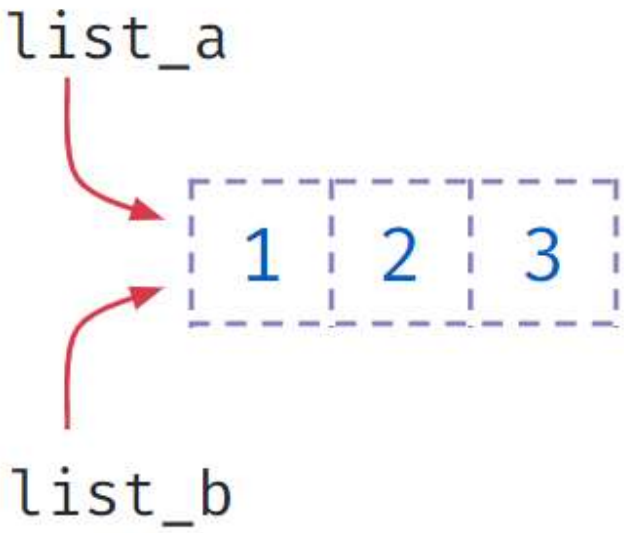
```
139637858236800
139637857505984
```

Modifying Lists

Modifying Lists - 1

When assigned an existing list both the variables

`list_a` and `list_b` will be referring to the same object.



Code

PYTHON

```
list_a = [1, 2, 3]
list_b = list_a
print(id(list_a))
print(id(list_b))
```

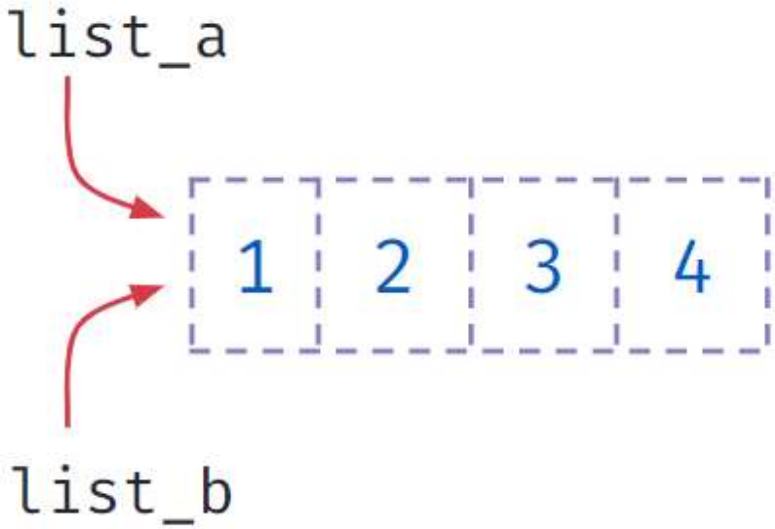
Output

```
140334087715264
140334087715264
```

Modifying Lists - 2

When assigned an existing list both the variables

list_a and list_b will be referring to the same object.



Code

PYTHON

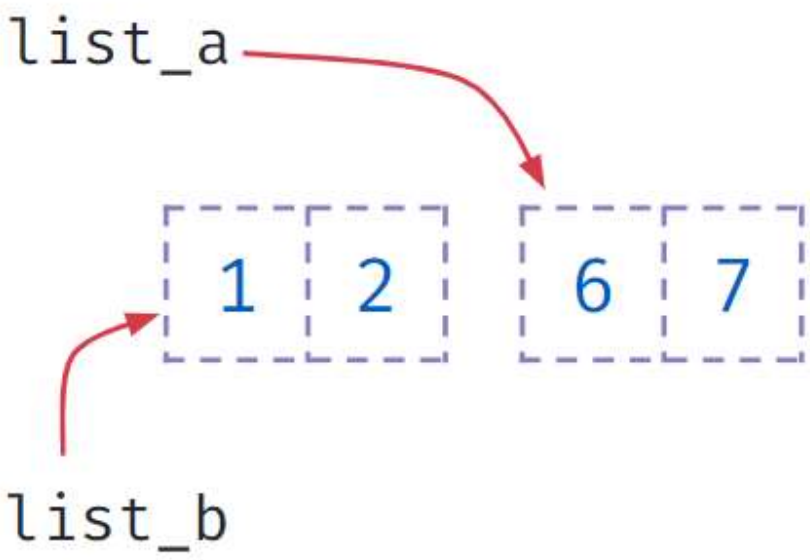
```
list_a = [1, 2, 3, 5]
list_b = list_a
list_b[3] = 4
print("list a : " + str(list_a))
print("list b : " + str(list_b))
```

Output

```
list a : [1, 2, 3, 4]
list b : [1, 2, 3, 4]
```

Modifying Lists - 3

The assignment will update the reference to new object.



Code

PYTHON

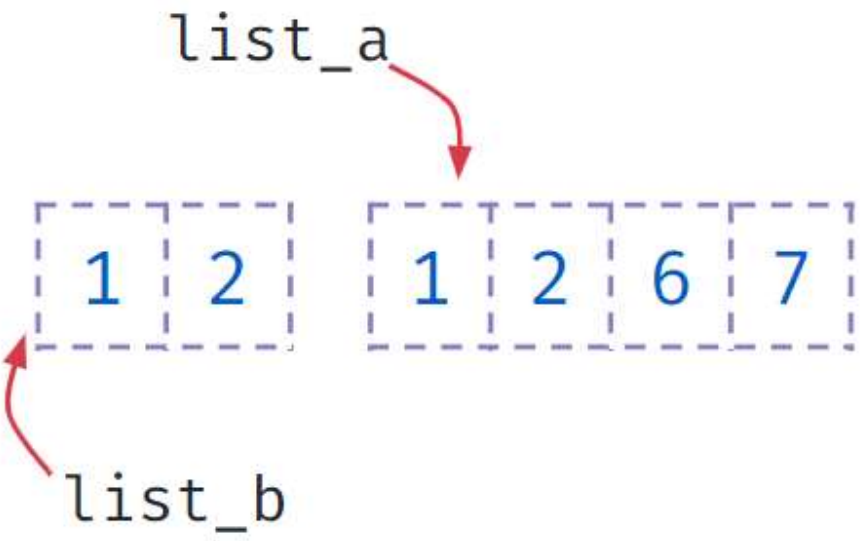
```
list_a = [1, 2]
list_b = list_a
list_a = [6, 7]
print("list a : " + str(list_a))
print("list b : " + str(list_b))
```

Output

```
list a : [6, 7]
list b : [1, 2]
```

Modifying Lists - 4

The assignment will update the reference to a new object.



Code

PYTHON

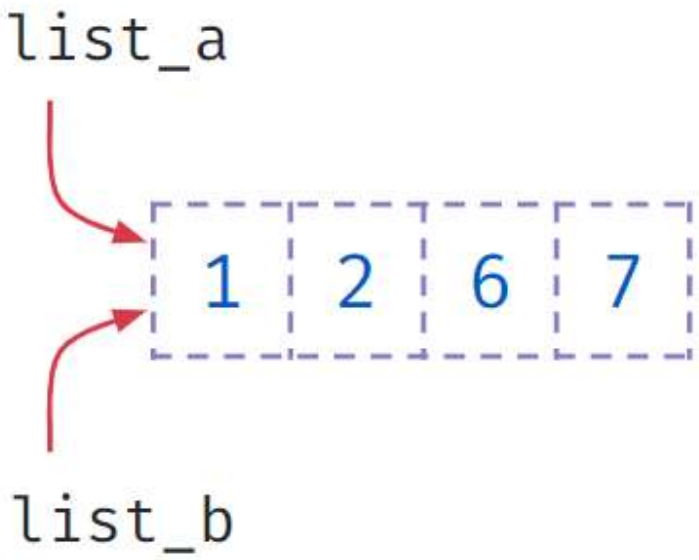
```
1  list_a = [1, 2]
2  list_b = list_a
3  list_a = list_a + [6, 7]
4  print("list a : " + str(list_a))
5  print("list b : " + str(list_b))
```

Output

```
list a : [1, 2, 6, 7]
list b : [1, 2]
```

Modifying Lists - 5

Compound assignment will update the existing list instead of creating a new object.



Code

PYTHON

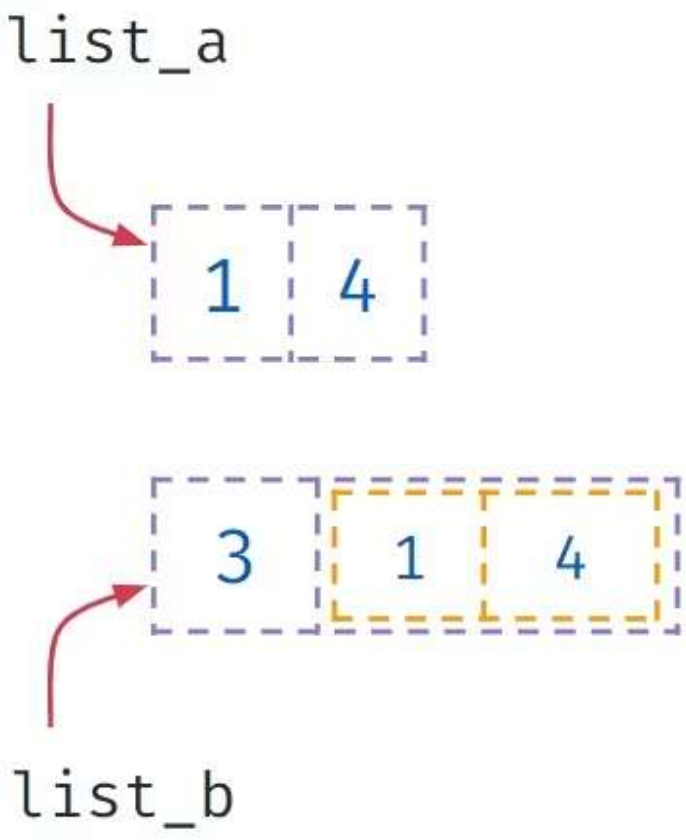
```
list_a = [1, 2]
list_b = list_a
list_a += [6, 7]
print("list a : " + str(list_a))
print("list b : " + str(list_b))
```

Output

```
list a : [1, 2, 6, 7]
list b : [1, 2, 6, 7]
```

Modifying Lists - 6

Updating mutable objects will also effect the values in the list, as the reference is changed.



Code

PYTHON

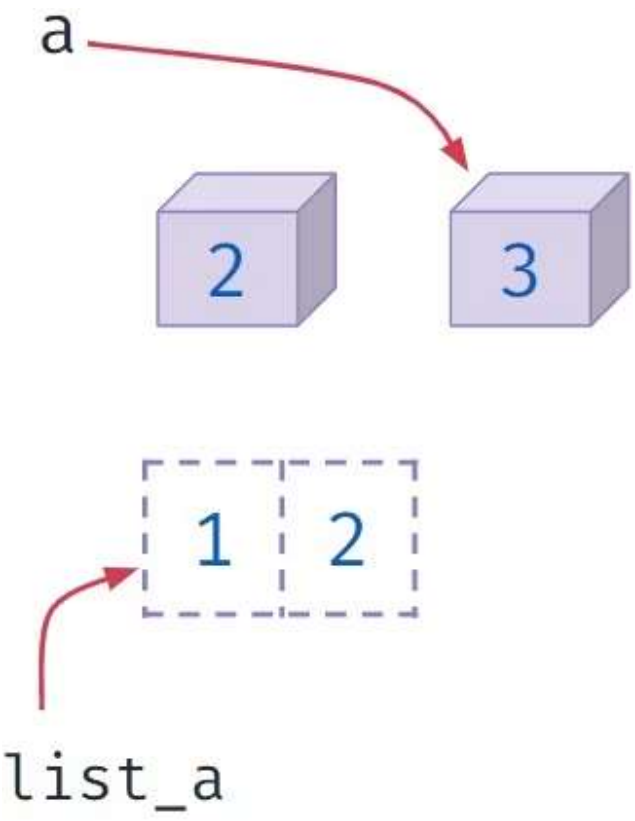
```
list_a = [1,2]
list_b = [3, list_a]
list_a[1] = 4
print(list_a)
print(list_b)
```

Output

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

Modifying Lists - 7

Updating immutable objects will not effect the values in the list, as the reference will be changed.



Code

PYTHON

```
a = 2
list_a = [1,a]
print(list_a)
a = 3
print(list_a)
```

Output

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

Submit Feedback