Aliasing, Mutation, and Cloning

Aliasing

- Occurs when two or more variables reference the same object in memory.
- These variable are called “aliases” because are they are different names that you can use to reference the same object in memory.
- Aliases have the same id when you call the id() function.
- They can be a source of bugs because you can modify an object using one of the aliases and forget that the other ones point to the same object, so they will be modified too.

Mutation

- A **mutable** object can change after it has been defined.
- An advantage of mutable objects is that they be easily modified without additional memory usage.
- A disadvantage of mutable objects is that they are more prone to bugs due to the risk of aliasing.
- An **immutable** object cannot change after it has been defined.
- An advantage of immutable objects is that they are safer from bugs.
- A disadvantage of immutable objects is that they may require additional memory usage because a new object has to be created to create a modified copy of the original.
- Built-in mutable objects in Python include lists and dictionaries.
- Built-in immutable objects in Python include tuples, strings, floats, integers, and booleans.

Cloning

- Cloning is making a copy of the object in memory.
- The clone initially contains the same data as the original object, but it exists as a separate object in memory, not as an alias.
- You can modify the clone without modifying the original object. They are independent.

Fig. 1 Aliasing. Two variables reference the same object in memory.

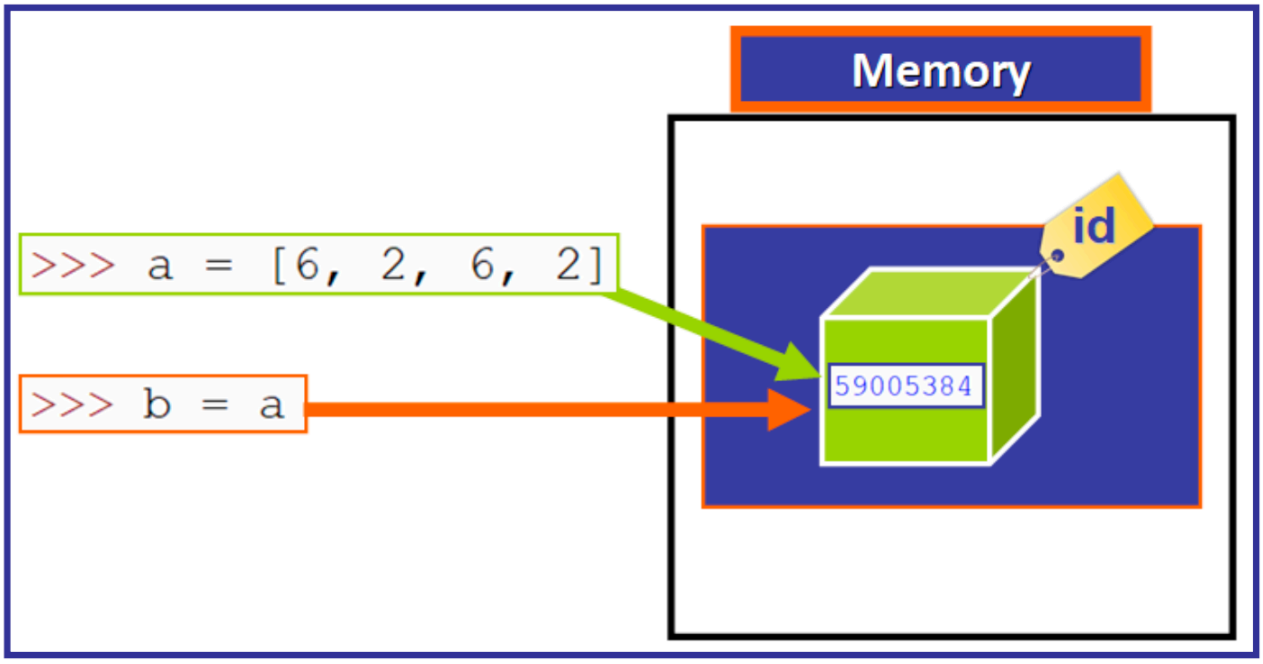

Fig. 2 Cloning.