1.1 The Python Memory Model: Introduction

DATA

- Data is stored in objects
- Objects have three components : **id, type ,value**

ID

- Unique identifier of the object
- Its like a unique memory address
- See object's id by calling id function

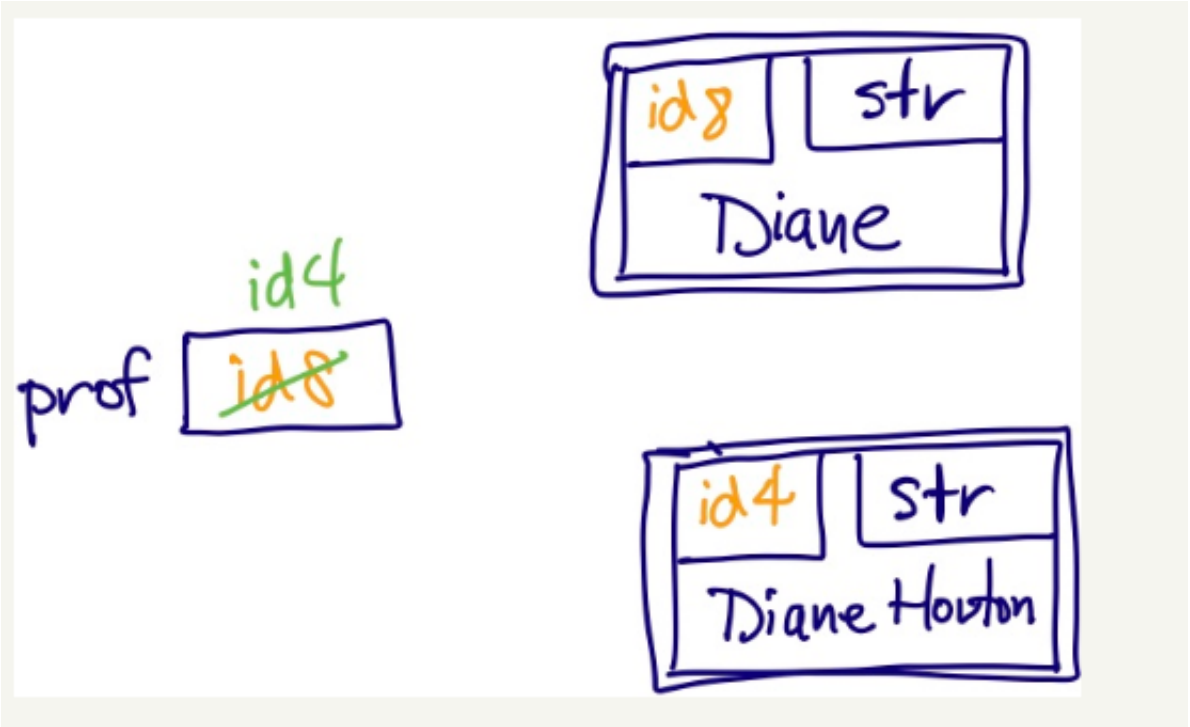
TYPE

- Determines what functions and python operators can operate on it
- See object's type by calking type function

VARIABLES

- Is not an object, doesn't store data
- REFERS to an object that stores data (has an id)
- When the variable is called: Variable -> Refers to the object through the ID

- Id : upper-right
- Type: upper left
- Object center



Objects have a type, but variables don't!

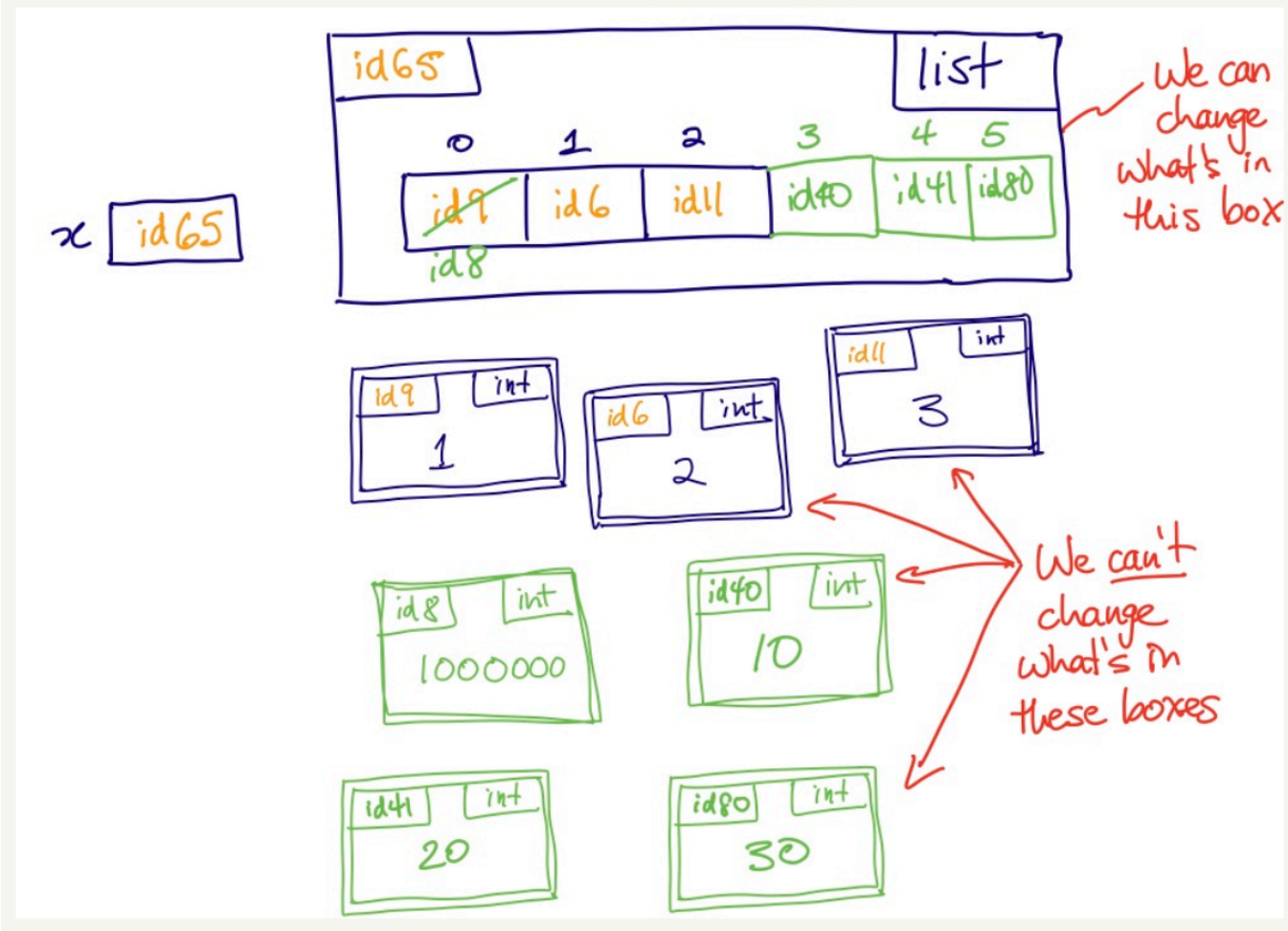
MUTABILITY AND ALIASING

IMMUTABLE DATA TYPES:

- **"Building Blocks"**
- Value stored in an object of that type cannot change
- Intergers, strings, booleans, tuples
- For example: once you concatenate a string, a new ID is created :)

MUTABLE DATA TYPES:

- More Complex, use the building blocks
- Lists, dictionaries, user-defined classes
- The ID doesn't change when mutating the object.



ALIASING:

- When two variables refer to the same object
- Have the same ID
- Even though two lists might have the same elements, they have different memory addresses.
- "Action at a distance":
  - Modifying a variable's value w/o explicitly mentioning that value
  - Mutable data types -> side effect of action @ a distance
  - Immutable data types - . Don't change, unaffected!
- Sometimes making a copy so changes can be made w/o affecting the original is good : often unnecessary though.

Changing a reference is not the same as mutating a value!

TWO TYPES OF EQUALITY

VALUE EQUALITY:

- **"=="**
- Compares the values stored in the objects they reference

IDENTITY EQUALITY:

- **"is"**
- Compares the ids of the objects they reference