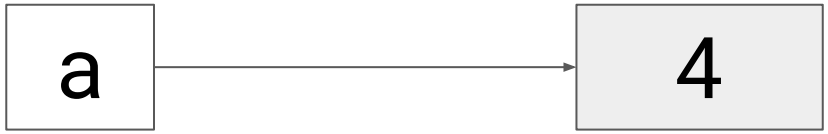
Data model

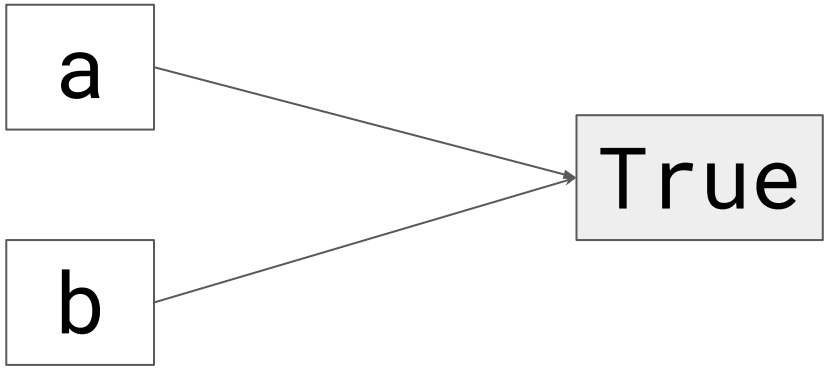
All objects in python:
numbers, sequences, mappings are
containers.

All data in a Python program is represented by objects or by relations between objects.

```
>>> a = 4
```



```
>>> a = True
>>> b = a
```

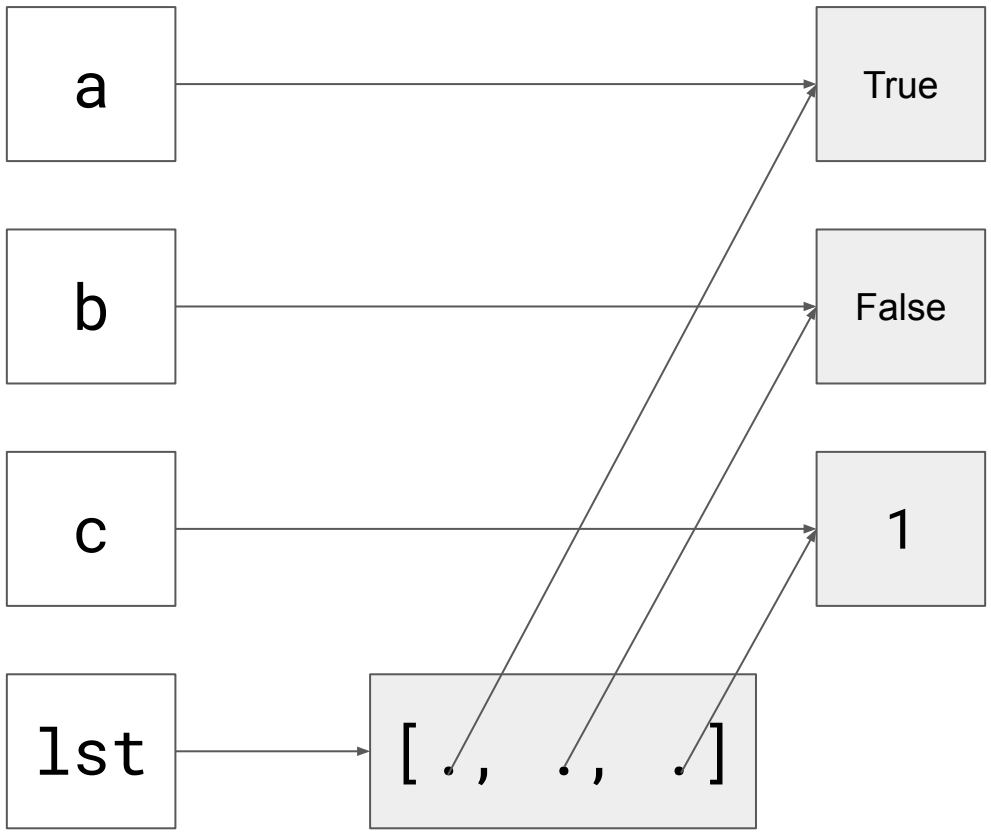


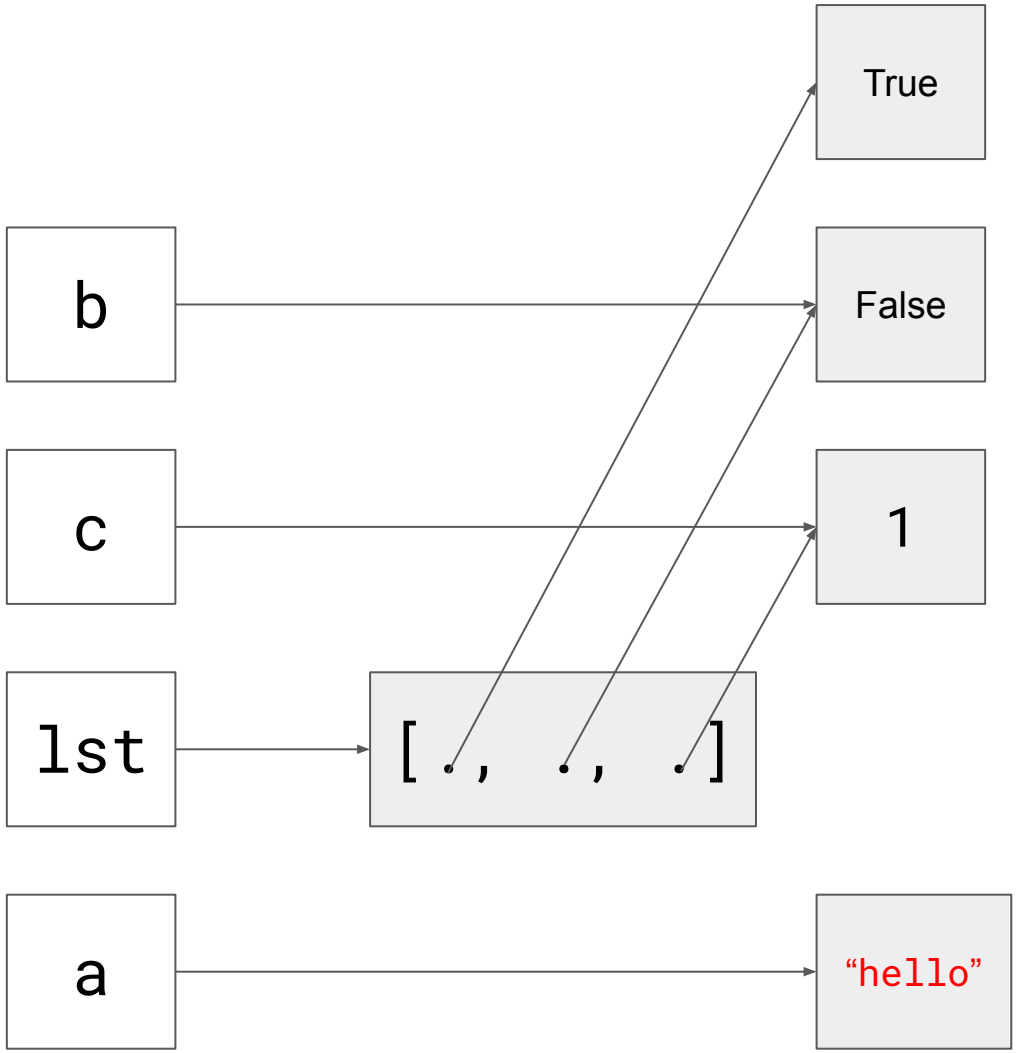
```
>>> a = True
>>> b = a
>>> a is b
True

>>> id(a) == id(b)
True
```

```
>>> a = True
>>> b = False
>>> c = 1

>>> lst = [a, b, c]
>>> lst
[True, False, 1]
```

```
>>> a = True
>>> b = False
>>> c = 1

>>> lst = [a, b, c]
>>> lst
[True, False, 1]

>>> a = 'hello'
>>> lst # ?
```

```
>>> a = True
>>> b = False
>>> c = 1

>>> lst = [a, b, c]
>>> lst
[True, False, 1]

>>> a = 'hello'
>>> lst
[True, False, 1]
```


a
b
c
lst
[., ., .]
True
False
1


True
b
False
c
1
lst
[., .,]
a
“hello”

```
>>> a = []
>>> b = [1]
>>> c = [1, 2]
>>> lst = [a, b, c]

>>> lst
[[], [1], [1, 2]]
```

```
>>> a = []
>>> b = [1]
>>> c = [1, 2]
>>> lst = [a, b, c]

>>> lst
[[], [1], [1, 2]]

>>> # Now let's append something
>>> a.append('hello')
```

```
>>> a = []
>>> b = [1]
>>> c = [1, 2]
>>> lst = [a, b, c]

>>> lst
[[], [1], [1, 2]]

>>> # Now let's append something
>>> a.append('hello')

>>> lst
[['hello'], [1], [1, 2]]
```

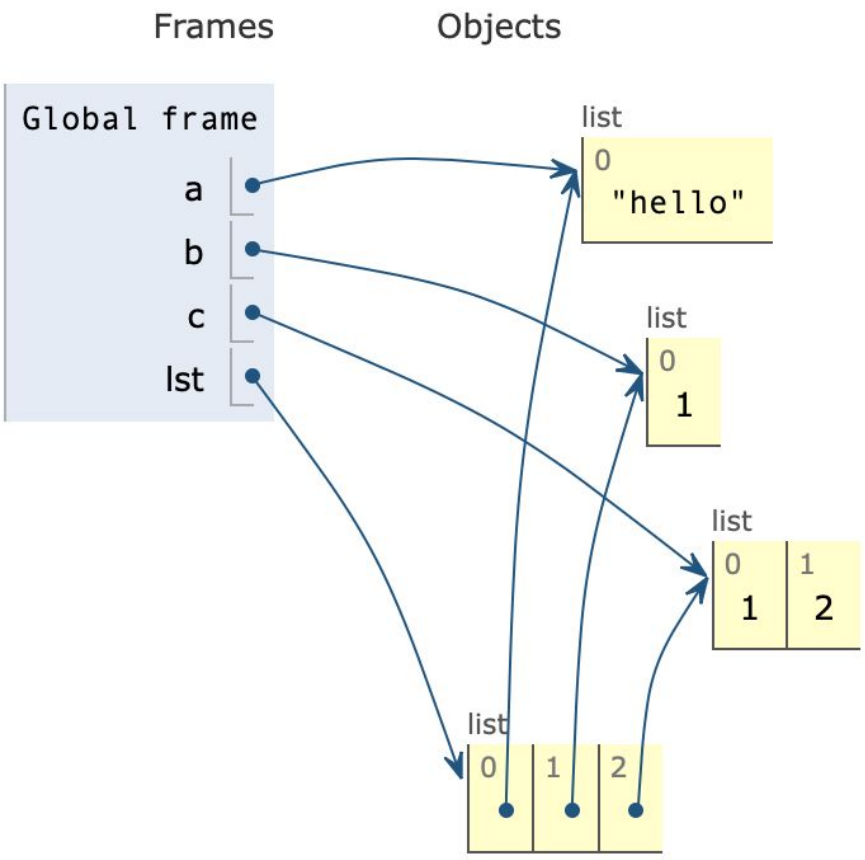
Visualisation

For visualisation we can use great online instrument

http://www.pythontutor.com/

Print output (drag lower right corner to resize)

```
[[], [1], [1, 2]]
[['hello'], [1], [1, 2]]
```



```
>>> a = []
>>> b = [1]
>>> c = (a, b)

>>> c[0] = 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item
assignment

'tuple' object does not support item assignment
```

```
>>> a = []
>>> b = [1, 2]
>>> c = (a, b)

>>> c[0].append(1)
>>> c
([1], [1, 2])
```

Always think about mutability of an object

Mutability

Immutable: int, float, complex, bool, str, frozenset, tuple, Bytes

Mutable: dict, list, set

Object

There is builtin class **object** in python which is **base** for all classes.

It has the **methods** that are common to all instances of Python classes.

Everything is an object in python

Object

There is builtin class **object** in python which is **base** for all classes.

It has the **methods** that are common to all instances of Python classes.

Everything is an object in python

Object

```
>>> isinstance(object, object)
True

>>> def a(): return 1
>>> isinstance(a, object)
True

>>> isinstance(type, object)
True
```

Object

```
>>> isinstance(object, object)
True

>>> def a(): return 1
>>> isinstance(a, object)
True

>>> isinstance(type, object)
True
```

dir([*object*])

With an argument, attempt to return a list of valid attributes for that object.

dir([*object*])

```
>>> dir(object)
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__le__',
'__lt__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__']
```

dir([*object*])

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__',
'__delitem__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__',
'__lt__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
'__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count',
'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
```

Magic methods

```
item in list      ->     list.__contains__(item)

list_1 == list_2 ->      list_1.__eq__(list_2)

...
```

Magic methods

https://docs.python.org/3.8/reference/datamodel.html

Conclusion

- Everything is an *object*
- Each object has an *identity*,a *type*,and a *value*
- *id*(obj) returns the object's *identity*
- Use mutable and immutable types wisely