Objects

In Python, every piece of data is an object. Examples are integers, strings, tuples, lists, images, robots, etc.

An object
- stores data (has state), and
- provides methods to access or manipulate its state.

Each object has a type. The type of the object determines what operations it supports.

We can define our own types using the class keyword. Think about a class as a blueprint for objects. You can create objects from the blueprint by just writing the class name.

Variables

A variable is a name for an object. One object can have multiple names, and its names can change during the course of a program.

A variable may also be unused. In that case we say that it has the value None.

All objects live on the heap. Names (variables) refer to objects on the heap.

Mutable and immutable objects

If the state of an object cannot change after the object has been constructed, it is immutable. In Python, number types, strings, tuples, and `frozenset` are immutable.

If the state of an object can change, it is mutable. Lists and all user-defined objects are mutable.

```
>>> A = [1, 2, 3, 4]
>>> B = A
>>> A
[1, 2, 3, 4]
>>> B
[1, 2, 3, 4]
>>> A[2] = 99
>>> A
[1, 2, 99, 4]
>>> B
[1, 2, 99, 4]
```

We need to be careful when a mutable object has several names.

Local variables

Objects live on the heap. But where do the variable names live?

If it is a field of an object, it lives inside the object on the heap. In particular, the elements of a list live inside the list object.

The local variables of a method live inside the method's activation record (also called stack frame).

Four local variables:

```
def test(m):
  k = m + 27
  s = "Hello World"
  A = [ len(s), k, m ]
```

Garbage collection

Many objects are used only briefly, and not needed afterwards. So after some time, the heap becomes full.

At that point, Python performs garbage collection: It checks all objects on the heap, and determines if there is any reference from a variable on some stack frame leading directly or indirectly to this object. If not, the object is destroyed.

You cannot easily predict when garbage collection happens. It can also be performed incrementally.

Python programs do not have to worry about memory leaks.