Aside: \texttt{\_\_str\_}\_

- We saw that constructing calls \texttt{\_\_init\_}\_
- Similarly, \texttt{print(x)} calls \texttt{x.\_\_str\_()}\_
- These underscored method names are called \texttt{special method names}.
- We’ll see more of these later.
Method/attribute look-up

• When we say `thingee.stuff` or `thingee.do_something()`, Python must:
  1. Find the name `thingee`.
  2. Follow the reference in it, to get to an object.
  3. Look inside the object to find attribute `stuff` or method `do_something`.

• Suppose `thingee` is both a `PencilCase` and a `Container`, because of inheritance.
  • There may be more than one definition of `stuff` and `do_something`!
How Python does it

• Python starts looking in the most specific part of the object.
  • If not found, it goes “up” as needed.

• Suppose a method in a parent class calls a helper method.
  • Python still starts looking in the most specific part of the object.

• Example: next slide.

• Trace it in the visualizer.
class A:
    def g(self, n):
        return n
    def f(self, n):
        return self.g(n)

class B(A):
    def g(self, n):
        return 2 * n

a = A()
b = B()

print('a.f(1): {}'.format(a.f(6)))
print('b.f(1): {}'.format(b.f(6)))