

## super(...).\_\_init\_\_() vs Base.\_\_init\_\_(self)

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2006-02/msg01325.html>

---

- *From:* Kent Johnson <[kent@xxxxxxxxxxxxxxxxxxxx](mailto:kent@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 09 Feb 2006 10:25:16 -0500
- 

Are there any best practice guidelines for when to use

`super(Class, self).__init__()`

vs

`Base.__init__(self)`

to call a base class `__init__()`?

The `super()` method only works correctly in multiple inheritance when the base classes are written to expect it, so "Always use `super()`" seems like bad advice. OTOH sometimes you need `super()` to get correct behaviour. ISTM "Only use `super()` when you know you need it" might be the best advice. Is there any conventional wisdom on this?

The question arises from a naive use of `super()` in a post on the tutor list. This code gives an `AttributeError` because `Base.__init__()` is never called:

```
import threading

class Base(object):
    def __init__(self):
        self.x = 1

class Derived(threading.Thread, Base):
    def __init__(self):
        super(Derived, self).__init__()

d=Derived()
d.x
```

If the order of base classes is reversed, the reference to `d.x` works but of course `threading.Thread.__init__()` is never called.

1. One way to fix the code is to call `Base.__init__()` and `threading.Thread.__init__()` explicitly in `Derived.__init__()`.
2. Another fix is for `Base.__init__()` to call `super(Base, self).__init__()` and to list `Base` first in the list of base classes. This is fragile – it depends on the order of base classes and adding another base class would break it.
3. A third fix might be to change both `Base` and `threading.Thread()` to call `super(...).__init__()`. This might break existing code that is written in the style of fix 1 (calling both base class `__init__()` methods explicitly).

## `super(...).__init__()` vs `Base.__init__(self)`

I prefer the first fix, it is explicit and fairly robust – it works if the order of bases is changed, and it's pretty clear from the body of `Derived.__init__()` that if you add another base class, you should change `__init__()`.

Any other opinions? Any consensus about the "best" way to do this?

BTW I understand what `super()` does, I know why the original code is broken, I'm not asking for help with that. I'm wondering what others think best practices are.

Thanks,  
Kent

.