

Re: Question about classes

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-11/2971.html>

From: Steve Holden (steve_at_holdenweb.com)

Date: 11/22/04

Date: Mon, 22 Nov 2004 08:44:50 -0500

Steven Bethard wrote:

```
> Ben wrote:
>
>> class foo(xstart):
>> x = 5
>> if xstart != 0:
>> x = xstart
>>
>> a = foo(8)
>>
>> What I am curious about is why not? What am I missing about classes here?
>> Is the functionality delivered in some other fashion, or must I:
>>
>> class foo:
>> x = 5
>>
>> a = foo()
>> a.x = 8
>
>
> The parentheses after a class name do not indicate a parameter list;
> they indicate the list of base classes. So generally, they must be
> classes/types:
>
> >>> def make_class(base):
> .... class C(base):
> .... pass
> .... return C
> ....
> >>> make_class(object)
> <class '__main__.C'>
> >>> make_class(1)
> Traceback (most recent call last):
> File "<interactive input>", line 1, in ?
> File "<interactive input>", line 2, in make_class
> TypeError: Error when calling the metaclass bases
> int() takes at most 2 arguments (3 given)
```

```
>
> If you want to set a class or instance variable for an object at
> instantiation time, you probably want to supply the __init__ method:
>
> >>> class C(object):
> ... x = 5
> ... def __init__(self, x):
> ... if x != 0:
> ... C.x = x
> ...
> >>> c = C(0)
> >>> c.x
> 5
> >>> c = C(8)
> >>> c.x
> 8
>
> It seems strange to me that you want to set a *class* variable here, not
> an instance variable, though perhaps you have your reasons. Note that
> by doing so, every time you make a new instance, you'll change the x
> attribute for all objects:
>
> >>> C.x
> 5
> >>> c1 = C(0)
> >>> c1.x
> 5
> >>> C.x
> 5
> >>> c2 = C(8)
> >>> c2.x
> 8
> >>> c1.x
> 8
> >>> C.x
> 8
>
> If instead, you intended to set an instance variable, you might write it
> like:
>
> >>> class C(object):
> ... def __init__(self, x):
> ... if x != 0:
> ... self.x = x
> ... else:
> ... self.x = 5
> ...
> >>> c = C(0)
> >>> c.x
> 5
> >>> c = C(8)
```

comp.lang.python: Re: Question about classes

```
> >>> c.x  
> 8  
>  
> Steve
```

Don't forget, though, that due to the attribute resolution order, if an instance doesn't have a particular attribute but the class does then a reference to the attribute in a method will get the class attribute.

This has been used to implement class defaults in the past.

regards
another Steve

--
<http://www.holdenweb.com>
<http://pydish.holdenweb.com>
Holden Web LLC +1 800 494 3119