**lazy attributes (Python recipe)**

How to create attributes with 'computed at first use' values.

*#*

*# first solution:*

*#*

**class** sample**(**object**):**

**class** one**(**object**):**

**def** \_\_get\_\_**(**self**,** obj**,** type**=**None**):**

**print** "computing ..."

obj**.**one **=** **1**

**return** **1**

one **=** one**()**

x**=**sample**()**

**print** x**.**one

**print** x**.**one

*#*

*# other solution:*

*#*

*# lazy attribute descriptor*

**class** lazyattr**(**object**):**

**def** \_\_init\_\_**(**self**,** fget**,** doc**=**''**):**

self**.**fget **=** fget

self**.**\_\_doc\_\_ **=** doc

**def** \_\_appoint\_\_**(**self**,** name**,** cl\_name**):**

**if** hasattr**(**self**,**"name"**):**

**raise** **SyntaxError,** "conflict between "**+**name**+**" and "**+**self**.**name

self**.**name **=** name

**def** \_\_get\_\_**(**self**,** obj**,** cl**=**None**):**

**if** obj **is** None**:**

**return** self

value **=** self**.**fget**(**obj**)**

setattr**(**obj**,** self**.**name**,** value**)**

**return** value

*# appointer metaclass:*

*# call the members \_\_appoint\_\_ method*

**class** appointer**(**type**):**

**def** \_\_init\_\_**(**self**,** cl\_name**,** bases**,** namespace**):**

**for** name**,**obj **in** namespace**.**iteritems**():**

**try:**

obj**.**\_\_appoint\_\_**(**name**,** cl\_name**)**

**except** **AttributeError:**

**pass**

super**(**appointer**,** self**).**\_\_init\_\_**(**cl\_name**,** bases**,** namespace**)**

*# base class for lazyattr users*

**class** lazyuser**(**object**):**

\_\_metaclass\_\_ **=** appointer

*# usage sample*

**class** sample**(**lazyuser**):**

**def** one**(**self**):**

**print** "computing ..."

**return** **1**

one **=** lazyattr**(**one**,** "one lazyattr"**)**

x**=**sample**()**

**print** x**.**one

**print** x**.**one

**del** x**.**one

**print** x**.**one

This recipe works only for Python >= 2.2.

When an attribute is expensive to compute, it's sometime desirable to delay this computing until the value is actually needed.

The first solution show how to create a lazy attribute "by hand"

The second solution allow to create this kind of attributes with a syntax similar to property attributes definition. In the second solution, lazyattr user classes must inherit from lazyuser. This is needed for the lazyattr \_\_appoint\_\_ initialization.

I hope that this recipe can also be a good sample of metaclass and descriptor definitions.