Lightning Talks

Properties vs. Decorators

Philipp von Weitershausen

Decorator syntax

- At EuroPython 2004 I voted against it
- Today I like it (or I just got used to it)

```
>>> class JamesBrown(object):
...     @property
...     def feel(self):
...         return self._feel
```

One wart

- Writable (and delable) properties

```
>>> class JamesBrown(object):
...     def _getFeel(self):
...         return self._feel
...     def _setFeel(self, feel):
...         self._feel = feel
...     feel = property(_getFeel, _setFeel)
```

- Leaves setters and getters behind in class
- Decorator spelling is more compact and easer to read

There's a way to use a decorator...

- ... if you really want

```
>>> class JamesBrown(object):
...     @apply
...     def feel():
...         def get(self):
...             return self._feel
...         def set(self, feel):
...             self._feel = feel
...         return property(get, set)
```

- Doesn't leave setters and getters behind
- Still not as compact and easy on the eyes
- Python 3k won't have apply

My first idea

- Read & write properties
  - http://codespeak.net/svn/user/philikon/rwproperty

```
>>> from rwproperty import getproperty, setproperty
>>> class JamesBrown(object):
...     @getproperty
...     def feel(self):
...         return self._feel
...     @setproperty
...     def feel(self, feel):
...         self._feel = feel
```

Goofing around at PyCON

- Jim Fulton + Ian Bicking + me + Texan beer

```
>>> from classproperty import classproperty
>>> class JamesBrown(object):
...     class feel(classproperty):
...         def __get__(self):
...             return self._feel
...         def __set__(self, feel):
...             self._feel = feel
...
```

http://codespeak.net/svn/user/philikon/classproperty

Thank you

```
>>> i = JamesBrown()
>>> i.feel = "good"
>>> i.feel
'good'
```

Questions?