Information Infrastructure II

INFO I211 – Spring 2014 – Sections 18530 & 22519

Lecture 9 – 2014.02.12 & 2014.02.13

Instructor:
Mitja Hmeljak,
http://mypage.iu.edu/~mitja
mitja@indiana.edu

From **python.org** official documentation:

- Private Variables

  <http://docs.python.org/release/2.6/tutorial/classes.html#private-variables>

- Naming Styles:

  _single_leading_underscore,

  __double_leading_underscore, ...

  <http://www.python.org/dev/peps/pep-0008/#descriptive-naming-styles>

- Class Methods

  <http://docs.python.org/2/library/functions.html#classmethod>

Group Task 1:

Design a class to keep information about bicycles (class Bicycle, file bicycle_semiprivate.py)

In the object, remember:

- the type (street, racing, mountain, etc.)
- the brand
- the price (make the price semi-private)

In your class code, remember the total number of bicycles

Provide a class method to find the number of bicycles

Provide object methods to find out type, brand, & price and to *print* the bicycle's attributes

Instantiate 3 bicycles and print the number of bicycles and each of the bicycles

Print the type, brand, and price of one bicycle using the object methods you designed to obtain these individual attributes

Using Properties

```
class Critter(object):
...
    name = property(get_name, set_name)
```

Property: An *interface* that allows indirect access to an attribute by wrapping access methods around dot notation

property() function

- Takes accessor methods and returns a property
- Supply with *get* and *set* methods for controlled access to private attribute
- Supply only *get* method for "read-only" property

Using Properties (continued)

```
>>> print crit.name
Randolph
>>> crit.name = "Sammy"
Name change successful.
>>> print crit.name
Sammy
>>> crit.name = ""
Critter's name can't be empty string.
```

property_critter.py

critter_caretaker.py

Group Task 2:

Re-implement a class to keep information about bicycles (class Bicycle, file bicycle_properties.py)

In the object, remember:

- the type (street, racing, mountain, etc.)
- the brand
- the price (turn all instance/object attributes into *properties*)

In your class code, remember the total number of bicycles, as well as the *last bicycle* accessed by client code.

Provide a class method to find the number of bicycles.

Provide object methods to find out type, brand, & price and to *print* the bicycle's properties.

Provide an object method to find out the *last bicycle* accessed by client code.

Instantiate 3 bicycles and print the number of bicycles and each of the bicycles

Print the type, brand, and price of one bicycle using the object methods you designed to obtain these individual properties.