Red Hat Python Lab - Lesson 2: Dictionaries, Objects

Daniel Mach <dmach@redhat.com>
Martin Sivák <msivak@redhat.com>

Useful links

- Python Quick Reference: http://rgruet.free.fr/#QuickRef
- Python docs: http://www.python.org/doc/
- PEP 8: http://www.python.org/dev/peps/pep-0008/
- pep8.py: http://pypi.python.org/pypi/pep8/
- pylint: http://www.logilab.org/project/pylint
- Epydoc: http://epydoc.sourceforge.net/
- Python in Python: http://pypy.org/

Interesting links:

- Letajici cirkus on root.cz: http://www.root.cz/serialy/letajici-cirkus/
- py.cz: http://www.py.cz/
- Unladen swallow: http://code.google.com/p/unladen-swallow/

Don't forget

- Batteries included -> read docs, find existing function, use it
- Follow PEP 8
- Pylint is your friend
- Write unit tests
- Objects are not everything, don't make your code over-complicated
- Return from functions as soon as possible -> more readable code, better indentation
- import this - python "cornerstones"

Command: Karma

Implement text filter to handle name++ and name-- input, with the exception of c++.
It will increase or decrease a numeric value assigned to name (starts at 0).
Also implement a command "karma name" which will print the current karma value.
Use dictionary as a data structure which holds the data and work with lowercase keys.

You **do not** have to **modify parse** function. Only add the processing functions and register them to FILTERS and COMMANDS.

Example follows:

```
> karma spam
spam has no karma
> spam++
> karma spam
spam's karma is 1
> spam--
> spam--
> karma spam
spam's karma is -1
> spam++
> karma spam
spam has no karma
```

Objects

Python supports objects with all three keystones of OOP: encapsulation, polymorphism and (multiple) inheritance.

```
class ClassName(object):
    pass
```

The basic (and "empty") class you can see above. The *object* mentioned in parentheses specifies the parent class. For all new style classes, the common parent has to be the object class. Python 2.6 and 3 also support interfaces, you can read more about them at Python website.

```
class ClassName2(ClassName):
    def method(self, argument):
        return argument
```

What you can see here is an inherited class with one method. All methods receive (automatically) reference to the instance as the first argument. The usual name for it is *self*.

```
# example of creating an instance and method invocation
cl = ClassName2()
cl.method("Hello World")
```

Objects - special methods and variables

- object.__init__(self, ...) # *it's not a constructor, only initializes attributes, doesn't return anything*
- object.__del__(self) # *if you use __del__, call `del obj` manually; garbage collector doesn't release objects with defined __del__.*
- object.__dict__ # *contains all attributes of an object*
- dir(object)

```
class ClassName(object):
    def __init__(self, spam):
        # assign value to an attribute
        self.spam = spam

        # protected attributes start with '_'; shouldn't be written from outside of the object
        self._spam = spam

        # private attributes start with '__'; do not use unless you know what you do !!!
        # acessible as self.__NAME within the object
        # acessible as obj._ClassName__NAME from the outside
        self.__spam = spam
```

For the complete list look at the documentation.

Project: Encapsulate IO interface

```
class BotInterface(object):
    def __init__(self):
        pass

    def read(self):
        """
        Read input and return it.

        @return: read data
        @rtype: str
        """


    def write(self, arg):
        """
        Write (send) argument to output.

        @param arg: text to be written to the output
        @type arg: str
        """

        pass
```

Project: Encapsulate IrcBot

```
class IrcBot(object):
    # nedavejte do teto tridy nic jineho, nez jsme tu uvedli
    # implementace prikazu i filtru budou mimo tridu, stejne tak
    # jako jejich pripadne globalni promenne
    COMMANDS = {}
    FILTERS = []

    def __init__(self, interface):
        self._if = interface

    def parse(self, msg):
        pass

    def run(self):
        while True:
            msg = self._if.read()
            msg = self.parse(msg)
            if msg:
                self._if.write(msg)

if __name__ == "__main__":
    intf = BotInterface()
    bot = IrcBot(intf)
    bot.run()
```

Logging

```
import logging

log = logging.getLogger("mujlog")
log.warn("varovani...")
```

Project: Log all incoming messages

Next lesson

- Modules
- Sockets
- Parsing IRC protocol