



Laurent Gregoire

<http://www.vim.org/about.php>

*It is a poor craftsman
who blames his tools.*

CS 152: *Programming Language Paradigms*



Editor Plugins

Prof. Tom Austin

San José State University

Plugin architectures for different text editors / IDEs

- Emacs (*Emacs Lisp*)
- Vim (*Vimscript*)
 - *Learn Vimscript the Hard Way*, Steve Losh
- Eclipse (*Java, AspectJ*)
- Sublime Text (*Python*)

Python

- Invented by Guido van Rossum
 - benevolent dictator for life (BDFL)
- "Executable pseudocode"
- scripting language
- whitespace sensitive
 - don't mix tabs and spaces



Employee class

```
class Employee:  
    def __init__(self, name, sal, bon):  
        self.name = name  
        self.salary = sal  
        self.bonus = bon  
  
    def get_wages(self):  
        return self.salary + self.bonus
```

Manager class

```
class Manager(Employee):
    def __init__(self, n, s, b, subs):
        Employee.__init__(self, n, s, b)
        self.subordinates = subs

    def get_department_wages(self):
        wages = self.get_wages()
        for emp in self.subordinates:
            wages += emp.get_wages()
        return wages
```

Using Employee and Manager

```
alice = Employee("Alice", 125000, 200)
dilbert = Employee("Dilbert", 100000, 2000)
wally = Employee("Wally", 85000, 0)

phb = Manager("Pointy-haired boss", 136000,
             100000, [alice, dilbert, wally])

print("Alice makes " + `alice.get_wages()`)
print("The boss makes " + `phb.get_wages()`)
print("The whole department makes " +
      `phb.get_department_wages()`)
```

Executing system calls in Python

```
import subprocess

p = subprocess.Popen("ls -l",
                    shell=True,
                    stdout=subprocess.PIPE)
for bline in p.stdout:
    line = bline.decode('utf-8')
        .rstrip('\n')
    print(line)
```

Developing a new plugin

- Tools > Developer > New Plugin
- Save in Packages/User/ directory
 - (OSX): Under
/Users/<username>/Library/Application Support/Sublime Text 3/
 - (WIN7): Under
C:\Users\<username>\AppData\Roaming\Sublime Text 3

`sublimeplugin.TextCommand`

- Represents a command that is
 - bound to a key, or
 - placed in a menu
- override `run` method
- To execute in the console:
 - `view.run_command('example')`

Rot13 example

(in-class)

Adding menu options

- Save JSON file in same directory, named:
 - `Main.sublime-menu`
 - `Side Bar.sublime-menu`
 - `Context.sublime-menu`

Sample Main.sublime-menu

```
[{"id" : "sjsuTestApp1",  
  "caption" : "First Example",  
  "children" : [  
    { "caption" : "Enc Rot13",  
      "command" : "rot13"  
    }  
  ]  
}]
```

Key bindings

- Provide command shortcuts
- special keys
 - `control`
 - `shift`
 - `super` (for OSX command key)
- Save JSON file in same directory, named:
 - `"Default (Windows) .sublime-keymap"`
 - `"Default (OSX) .sublime-keymap"`
 - `"Default (Linux) .sublime-keymap"`

Sample "Default (OSX).sublime-keymap"

```
[ {  
  "keys": ["super+shift+r"],  
  "command": "rot13"  
}]
```

Duplicate line example (in-class)

sublime_plugin.EventListener

- Triggers actions on some event
- For available hooks, see
 - https://www.sublimetext.com/docs/3/api_reference.html#sublime_plugin.EventListener

Sample EventListener

```
import sublime, sublime_plugin
class EventDump(
    sublime_plugin.EventListener):

    def on_load(self, view):
        print(view.file_name()+" loaded")

    def on_new(self, view):
        print("New file created")
```

References

- "Creating Sublime Text 3 Plugins",
by Sam Mello
 - <https://cnpagency.com/blog/creating-sublime-text-3-plugins-part-1/>
- Sublime Text 3 API Reference
 - https://www.sublimetext.com/docs/3/api_reference.html

Lab: Create a plugin for MyScheme

- Use `vm.rb`, `compiler.rb` from previous lab
- Add a "My Scheme" menu item
 - child "Run" executes current `.byco` file
 - display output to the console
 - Add a key binding for this command
- When a `.myscm` file is saved,
 - compile to a `.byco` file in same dir