Write Sublime Text 2 Packages with Python

Jenny JS Liang (jsliang)
PyConTW 2013

About 梁睿珊 / Jenny / jsliang

2006~2012

Student (undergraduate & graduate) @ NCTU CS

2012~present

Software Engineer @ IBM Taiwan

Joined Python user community since PyHUG Feb meeting, 2012

Why I like Sublime Text 2?

http://www.sublimetext.com/

1. Fuzzy match of...
   a. Goto Anything (Ctrl + P)
   b. Command Palette (Ctrl + Shift + P)
2. Multiple Selections/Edits
3. Cross Platform (OSX, Windows & Linux)
4. Python Plugin API

I'll use "ST2" for "Sublime Text 2" from now on.

Where do you place your packages?

- Menu bar > Preferences > Browse Packages...
- On package per folder
  - Packages/
    - *MyPackage*/
      - *.py
        - **Commands** or **EventListeners**
      - *.sublime-macro
      - *.sublime-menu
      - *.sublime-keymap
      - *.sublime-snippet
      - ...

FYR: check out files under Packages/Default/

Hello World - Your 1st Command

Menu bar > Tools > New Plugin...

```
import sublime, sublime_plugin

class ExampleCommand(sublime_plugin.TextCommand):
    def run(self, edit):
        self.view.insert(edit, 0, "Hello, World!")
```

Save to: Packages/HelloWorld/HelloWorld.py

Hello World - Executing Command

1. Restart ST2
2. Start Sublime Console by pressing **Ctrl + `**
3. Type in console:
   - view.run_command('example')
4. A "Hello World" string is inserted to the beginning of the view

Command Naming Rules

Each command is a subclass of **sublime_plugin.*Command**。When naming, use **CamelCase + "Command"**.

```
class HelloWorldCommand
(sublime_plugin.TextCommand): ...
```

To use the command, use *underscore_notation*:

```
view.run_command('hello_world')
```

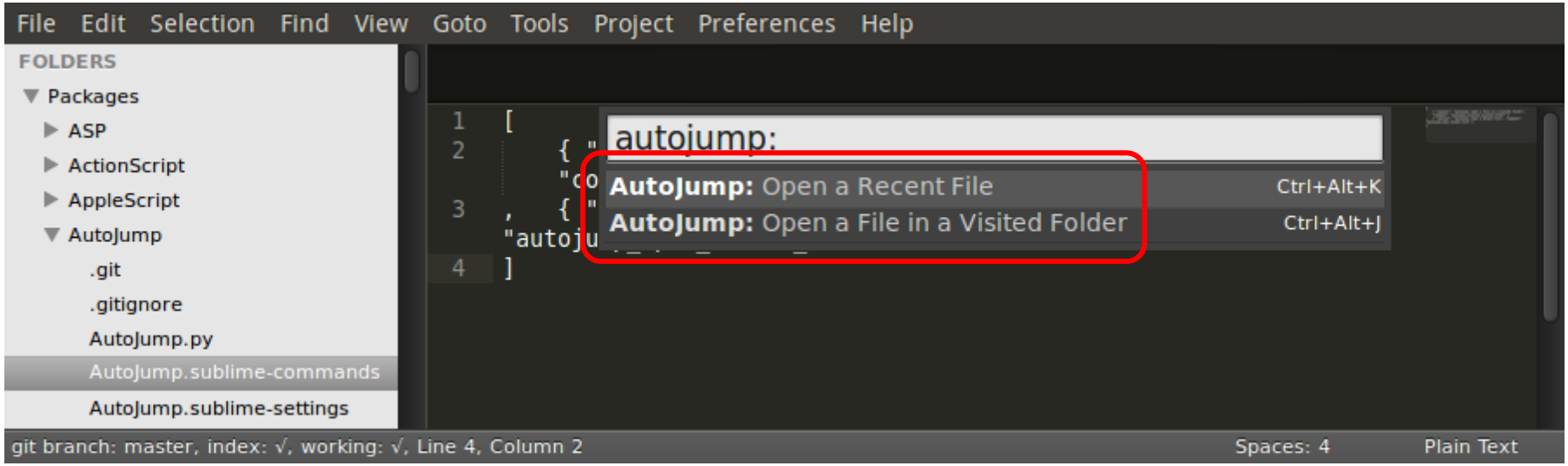
Types of Commands

1. Class sublime_plugin.**Application**Command
   - run(<args>)
   - institiated when ST2 is launched
2. Class sublime_plugin.**Window**Command
   - run(<args>)
   - instantiated once per window
   - The Window object may be retrieved via self.window.
3. Class sublime_plugin.**Text**Command
   - run(**edit**, <args>)
   - instantiated once per view
   - The View object may be retrieved via self.view.

Hierarchy: Application > Window > Text

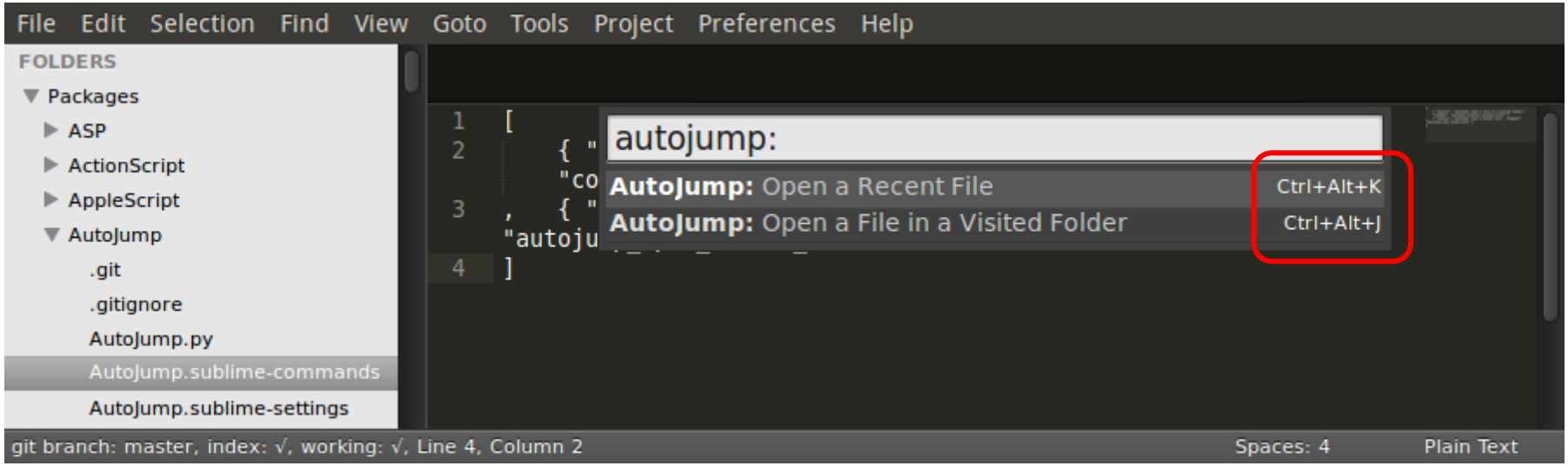
Make Your Command more Accessible (*.sublime-commands)

[
{ **"caption":** "AutoJump: Open a File in a Visited Folder",
"command": "autojump_traverse_visited_folder" }
,{ **"caption":** "AutoJump: Open a Recent File",
"command": "autojump_open_recent_file" }
]



Key Binding (Default.sublime-keymap)

```
[
{"keys": ["ctrl+alt+j"],
 "command": "autojump_traverse_visited_folder"}
, {"keys": ["ctrl+alt+k"],
 "command": "autojump_open_recent_file"}
]
```



Key Binding on different OS

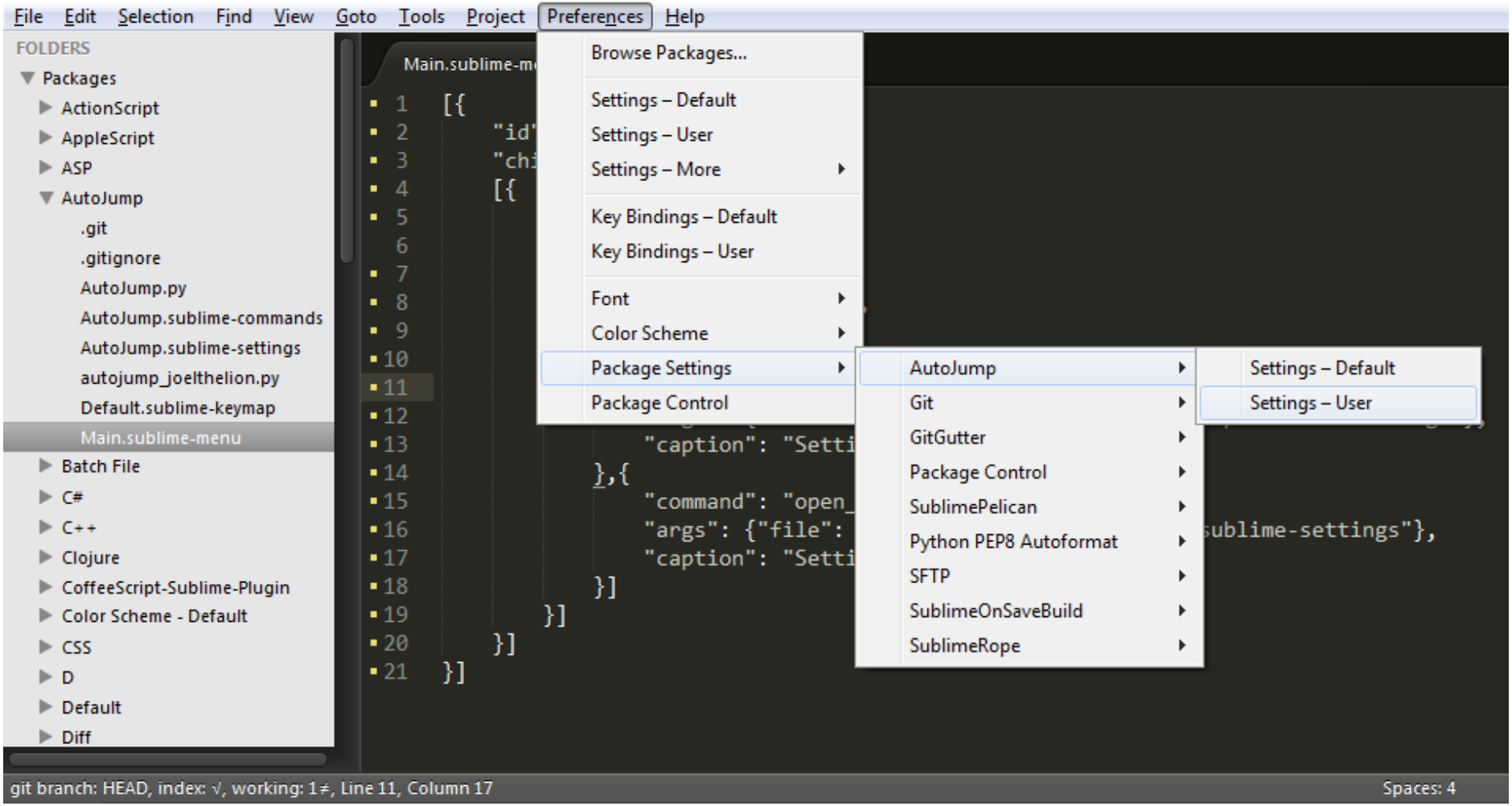
Default.sublime-keymap
Default (Linux).sublime-keymap
Default (OSX).sublime-keymap
Default (Windows).sublime-keymap

Menu Entries

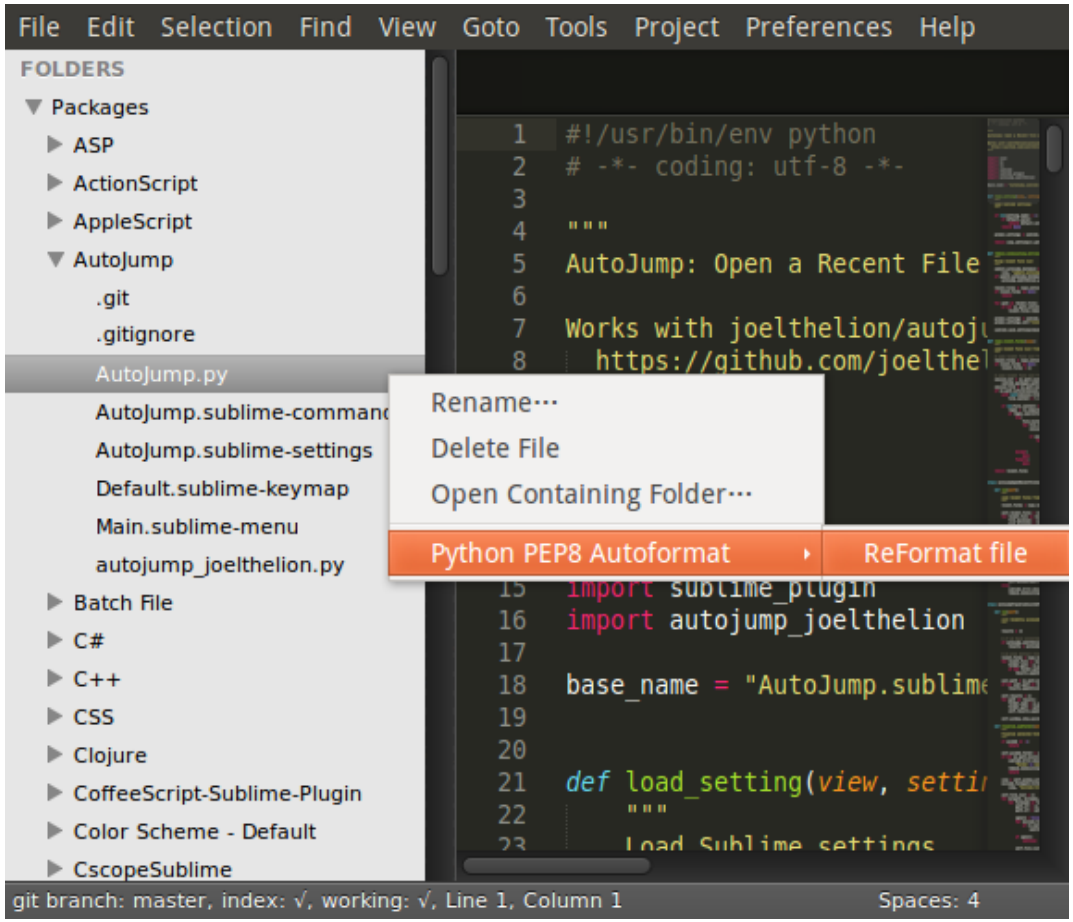
- Main.sublime-menu
  - Main program menu
- Context.sublime-menu
  - Context menu (right clicking on a file)
- Side Bar.sublime-menu
  - Side bar menu (right clicking on a file/folder in sidebar)

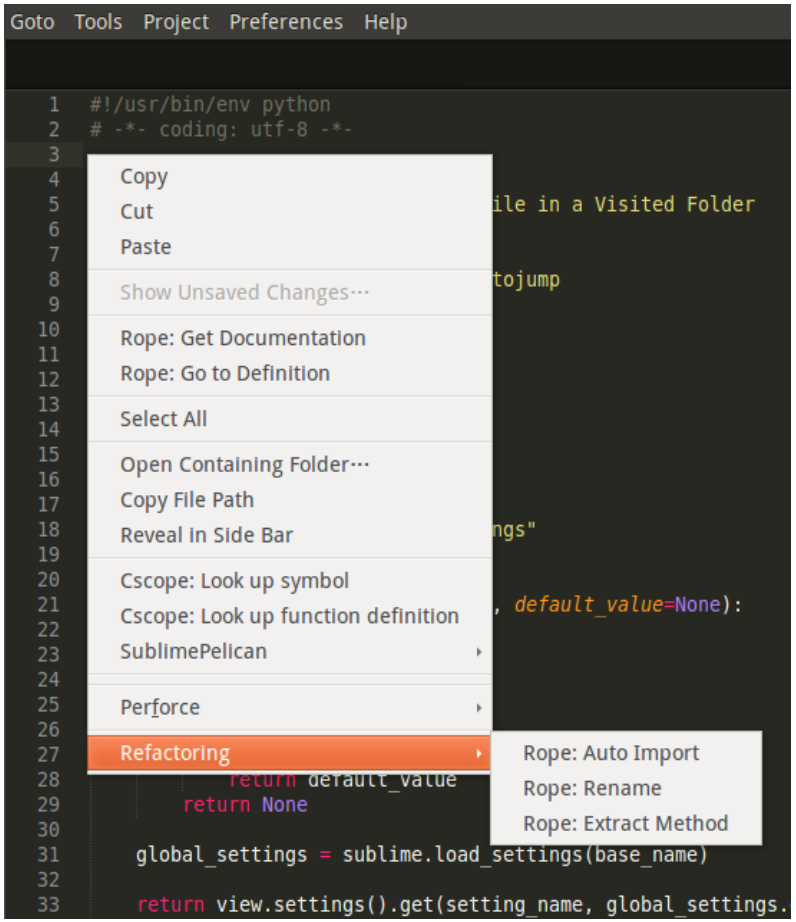
Each menu is a ***list*** of dicts, and each ***dict*** describes a *command* or *separator*.

Main Menu


File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
Packages
ActionScript
AppleScript
ASP
AutoJump
.git
.gitignore
AutoJump.py
AutoJump.sublime-commands
AutoJump.sublime-settings
autojump_joelthelion.py
Default.sublime-keymap
Main.sublime-menu
Batch File
C#
C++
Clojure
CoffeeScript-Sublime-Plugin
Color Scheme - Default
CSS
D
Default
Diff
Browse Packages...
Settings – Default
Settings – User
Settings – More
Key Bindings – Default
Key Bindings – User
Font
Color Scheme
Package Settings
Package Control
AutoJump
Git
GitGutter
Package Control
SublimePelican
Python PEP8 Autoformat
SFTP
SublimeOnSaveBuild
SublimeRope
Settings – Default
Settings – User
git branch: HEAD, index: √, working: 1≠, Line 11, Column 17
Spaces: 4

Side Bar Menu & Context Menu





Add Command to Menu Entries

Follow the structure in Packages/Default/*.sublime-menu and insert your entry.

```
[{
    "id": "edit", "children": [
        {"id": "wrap"}, {
            "command": "hello", "mnemonic": "h",
            "caption": "Hello Command below Wrap"
        }
    ]
}]
```

Event Listener

Each event listener is a subclass of **sublime_plugin.EventListener**.

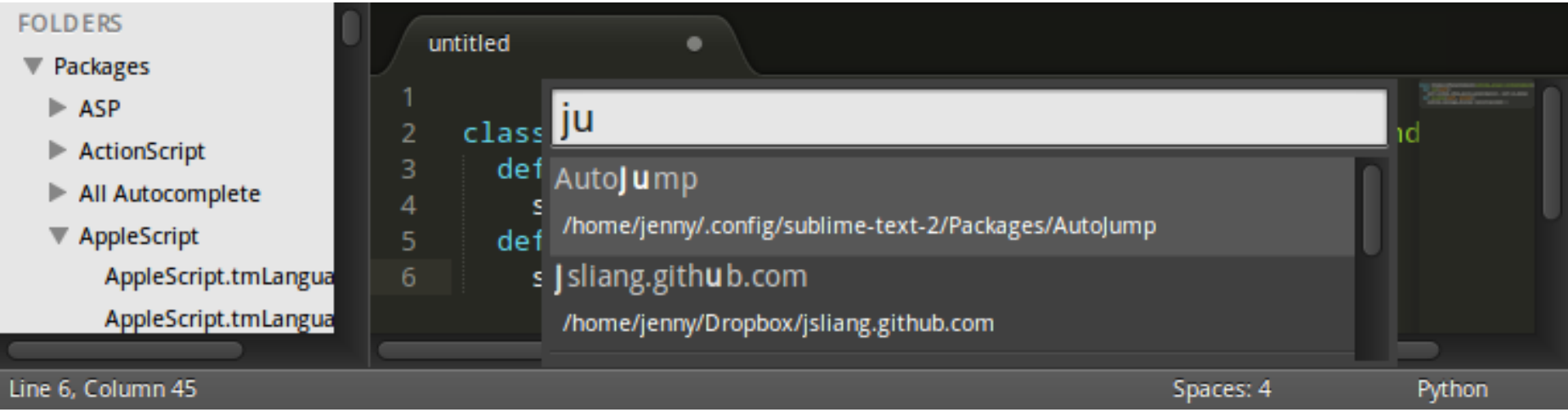
```
import sublime, sublime_plugin

class ViewClose(sublime_plugin.EventListener):
    def on_close(self, view):
        sublime.message_dialog("View closed.")
```

Event Listener Methods

- on_**new**(view)
- on_**clone**(view)
- on_**load**(view)
- on_**close**(view)
- on_**pre_save**(view)
- on_**post_save**(view)
- on_**modified**(view)
- on_**selection_modified**(view)
- on_**activated**(view) - on focus
- on_**deactivated**(view) - on blur
- on_**query_context**(view, key, operator, operand, match_all)

Quick Panel

- Similar to command palette
- Triggered by a ST2 window



Quick Panel

```
class ShowQuickPanelCommand
(sublime_plugin.WindowCommand):

    def run(self):
        self.window.show_quick_panel(mylist,
        self.on_done)

    def on_done(self, picked):
        sublime.message_dialog( mylist[picked] )
```

Package Settings

```
# Packages/MyPackage/MyPackage.sublime-settings
base_name = "MyPackage.sublime-settings"

pkg_settings = sublime.load_settings(base_name)

myvar = pkg_settings.get("myvar", "default_value")

pkg_settings.set("myvar", "new value")
sublime.save_settings(base_name)
```

Package Setting Files

Packages/

- MyPackage/
  - MyPackage.sublime-settings # default settings
  - Main.sublime-menu
  - ...
- User/
  - MyPackage.sublime-settings # user-customized settings
  - ...

Add Package Setting Option to Main Menu

- "id": "preferences", "children":
  - "id": "package-settings", "children":
    - "caption": "MyPackage",
    - "children":
      - "command": "**open_file**",
        "args": {"file": "**${packages}/MyPackage/MyPackage.sublime-settings**"},
        "caption": "Settings – Default"
      - "command": "open_file",
        "args": {"file": "**${packages}/User/MyPackage.sublime-settings**"},
        "caption": "Settings – User"

Manipulating Selections / Regions

```
sel_regionset = view.sel()
# sel_regionset is a RegionSet object

visible_region = view.visible_region()
# visible_region is a Region object
```

substr() / erase() / replace() / line() / split_by_newlines() / word() / show() / show_at_center() / ...

Example Plugins

- **Packages/Default/delete_word.py**
  - Deletes a word to the left or right of the cursor
- **Packages/Default/duplicate_line.py**
  - Duplicates the current line
- **Packages/Default/goto_line.py**
  - Prompts the user for input, then updates the selection
- **Packages/Default/font.py**
  - Shows how to work with settings
- **Packages/Default/mark.py**
  - Uses add_regions() to add an icon to the gutter
- **Packages/Default/trim_trailing_whitespace.py**
  - Modifies a buffer just before its saved

How to share my ST2 packages?

1. Compress your package folder to a file and let other people download it
   - do not forget to add a README telling users the extracting destination
2. Similar to 1, put your package on GitHub/Gitorious so that others can clone it.
3. If you think the above methods are too geekish...
   You must try Will Bond's Sublime Package Control

Sublime Package Control by wbond

http://wbond.net/sublime_packages/package_control

- a ST2 package that manages your installed ST2 packages
- search for and install ST2 packages
  - http://wbond.net/sublime_packages/community
  - http://wbond.net/sublime_packages/package_control/usage
- If you want your ST2 package to be found by Sublime Package Control...
  - http://wbond.net/sublime_packages/package_control/package_developers

References

http://www.sublimetext.com/docs/2/

http://www.sublimetext.com/docs/2/api_reference.html

http://net.tutsplus.com/tutorials/python-tutorials/how-to-create-a-sublime-text-2-plugin/