

LING115 Lecture Note

Session#3: Vim Text Editor

1. Introduction

We will write many programs in Python. This involves writing one or more lines of code and then invoking the Python interpreter to run what we wrote. For example, we first write the following line of code:

```
print 'Hello, LING115'
```

We save this program as, say, `hello.py` and then run it by entering the following in the command prompt:

```
$ python hello.py
```

This session is about the first part: using a text-editor called *Vim* to write code.

2. Text-editors in Unix-based Systems

There are several text-editors people use in Unix-based systems such as *vi*, *emacs*, and *pico*. Which editor you use is entirely your choice. I am teaching you how to use Vim, which is a particular version of the *vi* text editor, because I know more about it than other text-editors. If you are already familiar with any text-editor for use in Unix-based systems, you do not have to read this lecture note.

3. Modality

Vim is a text-editor, like Notepad in Windows. If you're new, you need to learn to use it because you need to remember the right keystrokes to get it to work.

The main difference between Vim and Notepad is this. In Notepad, we type on the white screen and use the menu bar on top to run any commands that Notepad comes with, like saving or loading a file. When we work with Vim, however, we need separate key strokes to switch from one mode to another.

We run Vim by entering `vim` in the command prompt optionally with a filename as its argument. That is,

```
$ vim
```

or

```
$ vim hello.py
```

Initially, we are in the *normal* mode, or the mode in which individual keystrokes have special meaning in that they each execute a particular command in Vim. This is like working with the top menu bar in Notepad.

So to type anything, we need to switch to the *insert* mode. This is done by hitting the 'i' key. We can switch back to the normal mode by hitting the ESC key.

4. Commands in Normal Mode

We can type away as long as we want once we are in the insert mode and I believe this is the part that I do not have to explain. Below is a short list of essential commands in normal mode for editing your text (e.g. save, undo, and so forth). For more, google 'vim commands'.

| Key(s) | Description |
|---------------|---|
| i | Enter insert mode. |
| ESC | Switch to normal mode. |
| u | Undo the last action. |
| Ctrl + r | Redo. |
| dd | Delete current line. |
| v | Start highlighting characters of your text. This is like pressing the left mouse button for the first time before you hold it to highlight in GUI. Every time you hit a right/left arrow key, Vim highlights the next/previous character. |
| V | Start highlighting lines of your text. This is just like hitting the 'v' key except the highlighted portion increases/decreases by a single line rather than by a character. |
| y | Yank (copy) the highlighted portion. |
| x | Cut the highlighted portion. |
| d | Delete the highlighted portion. |
| p | Paste yanked/cut portion AFTER the cursor. |
| :<number> | Jump to <number>th line from the top. |
| :w <filename> | Save as <filename>. If <filename> is not specified, Vim writes over the current file. |
| :q | Quit. Vim will not exit if there are unsaved changes. |
| :wq | Save and exit. |
| :q! | Exit Vim without saving. |