

# Text Editors

P. Chambers, R. Entwistle

Next Generation Computational Modelling CDT  
University of Southampton

FEEG6003 Advanced Computational Methods II

# Outline

- 1 Overview
  - IDEs vs Text Editors
  - Common Text Editors

- 2 Sublime
  - Main Features
  - Snippets
  - Build Systems

# Preface

- Ensure you have downloaded the `.ova` appliance from <http://www.southampton.ac.uk/~ngcmbits/>
- When prompted, follow the exercises on the blog post: <http://computationalmodelling.bitbucket.org/tools/sublime.html>

# Outline

## 1 Overview

- IDEs vs Text Editors
- Common Text Editors

## 2 Sublime

- Main Features
- Snippets
- Build Systems

# Integrated Development Environments

(IDE's)

- Program development tool, typically including:
  - ▶ Graphical User Interface (GUI)
  - ▶ Source code editor
  - ▶ Automated build and run procedures
  - ▶ Output display
  - ▶ Debugging options
  - ▶ Variable Explorers
- Examples:
  - ▶ Spyder (Python)
  - ▶ Quincy (C)
  - ▶ TeXmaker (LaTeX) ...
- Many tools, user friendly
- Usually only work for one (or few) Computing Languages

# Integrated Development Environments

(IDE's)

- Program development tool, typically including:
  - ▶ Graphical User Interface (GUI)
  - ▶ Source code editor
  - ▶ Automated build and run procedures
  - ▶ Output display
  - ▶ Debugging options
  - ▶ Variable Explorers
- Examples:
  - ▶ Spyder (Python)
  - ▶ Quincy (C)
  - ▶ TeXmaker (LaTeX) ...
- Many tools, user friendly
- Usually only work for one (or few) Computing Languages

# Text Editors

- Used to edit text/code (intuitive)
  - ▶ Bare functionality 'out of the box'
  - ▶ Fast operation through keystrokes
  - ▶ Can be customisable
  - ▶ Often cross-platform
  - ▶ Users need learn only one editor
- Open source:
  - ▶ Emacs, Vim, Nano ...
- Commercial:
  - ▶ Sublime Text<sup>1</sup>, UltraEdit, Textmate...

---

<sup>1</sup>Sublime Text offers an untimed free trial

# Text Editors

- Used to edit text/code (intuitive)
  - ▶ Bare functionality 'out of the box'
  - ▶ Fast operation through keystrokes
  - ▶ Can be customisable
  - ▶ Often cross-platform
  - ▶ Users need learn only one editor
- Open source:
  - ▶ Emacs, Vim, Nano ...
- Commercial:
  - ▶ Sublime Text<sup>1</sup>, UltraEdit, Textmate...

---

<sup>1</sup>Sublime Text offers an untimed free trial



# Outline

## 1 Overview

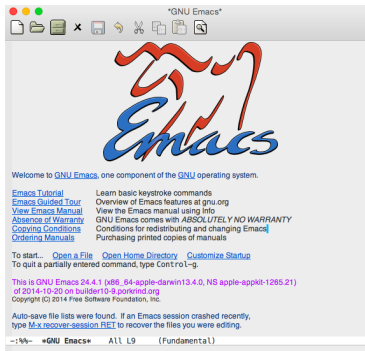
- IDEs vs Text Editors
- **Common Text Editors**

## 2 Sublime

- Main Features
- Snippets
- Build Systems

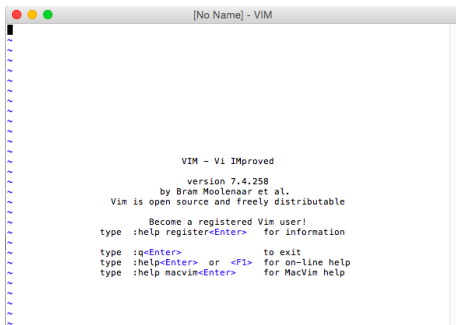
# Emacs

- Free. GNU General Public Licensed, 'GPL'
- Entirely cross-platform
- Highly customisable via Lisp language and package control
- Installed on most Linux Servers
- Steep (!) learning curve



# Vim

- Free. Open Source 'Charityware' - 'GPL' compatible
- Entirely cross-platform
- Customisable
- Based on Vi (default UNIX editor)
- Also a steep learning curve



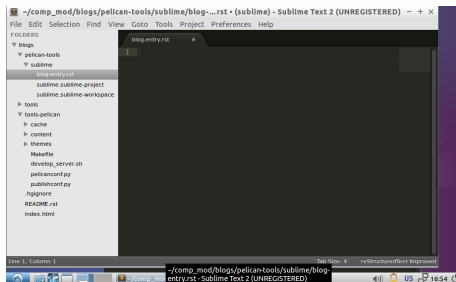
The screenshot shows a window titled "[No Name] - VIM". The window contains the following text:

```
VIM - Vi IMproved
      version 7.4.258
    by Bram Moolenaar et al.
Vim is open source and freely distributable

  Become a registered Vim user!
type :help register<Enter>  for information
type :q<Enter>              to exit
type :help<Enter> or <F1>   for on-line help
type :help macvim<Enter>   for MacVim help
```

# Sublime

- Commercial user licensed - untimed free trial
- Windows, OSX, Linux
- Customisable via JSON settings files (simple!)
- Written in Python
- GUI only
- Not currently available on most servers



# Outline

1

## Overview

- IDEs vs Text Editors
- Common Text Editors

2

## Sublime

- **Main Features**
- Snippets
- Build Systems

# Sublime Features: General

- Multiple document tabs
- Side bar (Projects)
- 'GoTo anything'
- Smart auto-complete
- Snippet expansion and suggestions
- Build Systems
- Keybindings and macros - Efficient
- Simple & instant customisation
  - ▶ Extensive package control
  - ▶ Plugins written in Python
  - ▶ JSON user settings - "attribute": "value"

# Sublime Features: Projects

- Add/remove directories to the workspace: 'Project' tab
- Recover workspace in another session

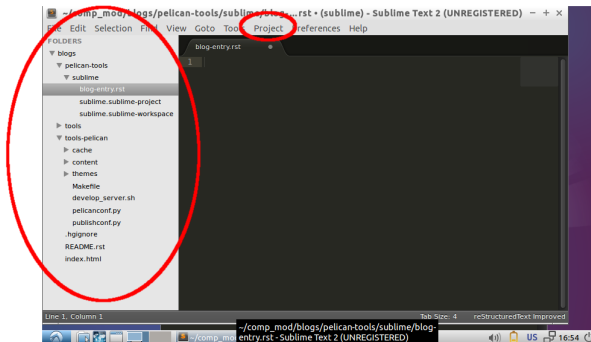
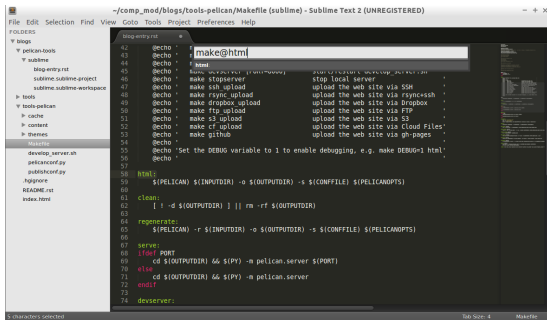


Figure: Projects in Sublime

# Sublime Features: GoTo Anything

- Search file names and content in the current project
  - ▶ # - Fuzzy search document
  - ▶ : - Go to line
  - ▶ @ - Go to definition



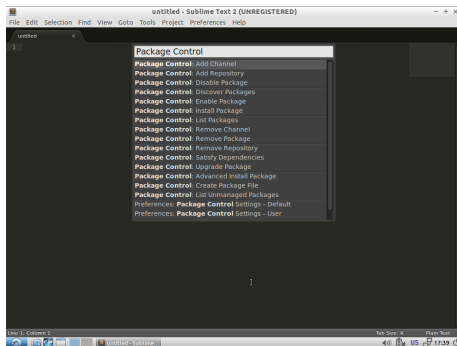
```
42 @echo -
43 @echo -
44 @echo -
45 @echo -
46 @echo -
47 @echo -
48 @echo -
49 @echo -
50 @echo -
51 @echo -
52 @echo -
53 @echo -
54 @echo -
55 @echo -Set the DEBUG variable to 1 to enable debugging, e.g. make DEBUG=1 html-
56 @echo -
57
58 html:
59 $(PELICAN) $(INPUTDIR) -o $(OUTPUTDIR) -s $(CONFFILE) $(PELICANOPTS)
60
61 clean:
62 [ ! -d $(OUTPUTDIR) ] || rm -rf $(OUTPUTDIR)
63
64 regenerate:
65 $(PELICAN) -r $(INPUTDIR) -o $(OUTPUTDIR) -s $(CONFFILE) $(PELICANOPTS)
66
67 serve:
68 lftp PORT
69 cd $(OUTPUTDIR) && $(PX) -m pelican.server $(PORT)
70 else
71 cd $(OUTPUTDIR) && $(PX) -m pelican.server
72 endif
73
74 devserver:
```

Figure: GoTo 'html' definition in Makefile in the current project



# Sublime Features: Package Control

- Install packages via the 'Command Palette'
- Packages available instantly (no restart needed)
- Automatic updates for installed packages



# Targets for Practical (Part 1)

- Optimise ST2 for writing blog entries (reStructuredText)
  - ▶ Install ST2
  - ▶ Become familiar with the layout
  - ▶ Set up package control and install some packages

## Blog Page:

<http://computationalmodelling.bitbucket.org/tools/sublime.html>



# Outline

- 1 Overview
  - IDEs vs Text Editors
  - Common Text Editors

- 2 **Sublime**
  - Main Features
  - **Snippets**
  - Build Systems

# Snippets

- Smart templates
- Improve your efficiency

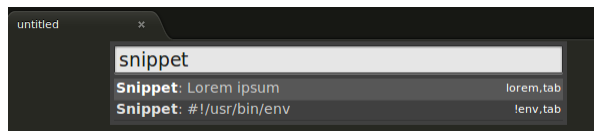


Figure: Using the Command Palette to find the list of snippets

# New Snippet: Format

- *Tools > New Snippet...*
- XML file format
- Opens basic Hello snippet template

```
<snippet>
<content><![CDATA[
Hello, ${1:this} is a ${2:snippet}.
]]></content>
  <!-- Optional: Set a tabTrigger to define how to trigger the snippet -->
  <!-- <tabTrigger>hello </tabTrigger> -->
  <!-- Optional: Set a scope to limit where the snippet will trigger -->
  <!-- <scope>source.python </scope> -->
</snippet>
```

# New Snippet: Tab key field markers

- Cycle through the fields you wish to update.

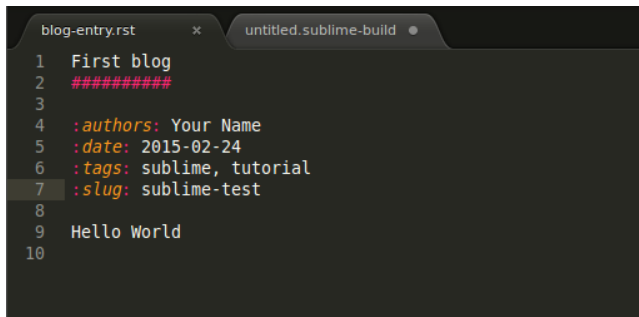
```
Hello , ${1:this} is a ${2:snippet}.
```

```
${#:abc}
```

- `${...}` - denotes a new field
- 1, 2 - the cycle order
- `abc` - the text you will replace (use as a definition of the feild)

# New Snippet: Pelican Blog Title

- Snippet: create blog title metadata quickly
- User defined snippets in *Packages > User*



```
blog-entry.rst x  untitled.sublime-build ●
1  First blog
2  #####
3
4  :authors: Your Name
5  :date: 2015-02-24
6  :tags: sublime, tutorial
7  :slug: sublime-test
8
9  Hello World
10
```

Figure: Blog entry title generated by the snippet

# Outline

1

## Overview

- IDEs vs Text Editors
- Common Text Editors

2

## Sublime

- Main Features
- Snippets
- **Build Systems**



# Build Systems

- Run files through external programs
- Displays the outputs



The screenshot shows a code editor window titled 'hello.c'. The code is as follows:

```
1  #include <stdio.h>
2
3  int main(void){
4      /* Function that prints "Hello World" followed by a new line
5       to the standard output and returns a interger */
6
7      printf("Hello World\n");
8
9      return 0;
10 }
11
```

Below the code editor, the output is displayed:

```
Hello World
[Finished in 0.1s]
```

The status bar at the bottom of the editor shows 'Line 11, Column 1', 'Tab Size: 4', and 'C'.

Figure: Execution of a build system

# New Build System: Format

- *Tools > Build System > New Build System*
- JSON file format
- Opens basic `make` JSON Build System

```
{  
  "cmd": ["make"]  
}
```

- See existing build-systems for more advanced settings

# New Build System: Pelican

- Need Build System to:
  - ▶ Navigate to Makefile
  - ▶ Run `make html`
  - ▶ Open/view output

```
{
  "cmd": ["bash", "-c", "cd ~/comp-mod/blogs/pelican-tools && make html"],
  "selector": "text.restructuredtext"

  "variants":
  [
    {
      "name": "Run",
      "cmd": ["bash", "-c", "cd ~//comp-mod/blogs/pelican-tools &&
              make html && cd ~/comp-mod/blogs/tools && firefox index.
              html"]
    }
  ]
}
```

# New Build System: Blog Preview

The image displays a workflow for a static site generator. On the left, the Sublime Text editor shows the source file `blog-entry.rst` with the following content:

```
1 First Blog
2 #####
3
4 :authors: Your Name
5 :date: 2015-02-24
6 :tags: sublime, tutorial
7 :slug: sublime-test
8
9 Hello World!
```

On the right, the Mozilla Firefox browser shows the rendered output. The page title is "Computational Modelling Tools Workshops". The main heading is "First Blog". The content area displays "Hello World!". To the right of the content, the following metadata is shown:

Published: Tue 24 February 2015  
By [Your Name](#)  
In [my-blog](#).  
tags: [sublime](#) [tutorial](#)

Below the main content, there is a section titled "Other articles" with a link to [Installing Pelican and Mercurial on lubuntu](#).

## Targets for Practical (Part 2)

- Create a new snippet for blog title and metadata
- Create a build system for making the html page and viewing it locally

### Blog Page:

`http://computationalmodelling.bitbucket.org/tools/sublime.html`



# Summary

- ST2 is an intuitive text editor with many tools which work ‘out of the box’
- We have demonstrated ways to customise ST2 for writing Pelican blog entries
- Techniques presented should be transferable to other languages