Emacs as a Python IDE

Noufal Ibrahim

Consulting software developer

PyCon India 2011 - Symbiosys, Pune

`http://nibrahim.net.in`
`@noufalibrahim`
`noufal@nibrahim.net.in`

Outline

Introduction

- Emacs

Programming

- General
- Major modes
- Utilites
- Power tools

Work tracking

- Introduction
- Task tracking
- Task collection

Finally

- References

What is Emacs?

- Programmers editor. Customisable using elisp.
  - Customisability is a feature (not an add on).
  - Hence lots of “applications” in Emacs.
- Widely ported.
- Older than many of us.
- Almost religious following.

What is Emacs?

- Programmers editor. Customisable using elisp.
  - Customisability is a feature (not an add on).
  - Hence lots of “applications” in Emacs.
- Widely ported.
- Older than many of us.
- Almost religious following.

What is Emacs?

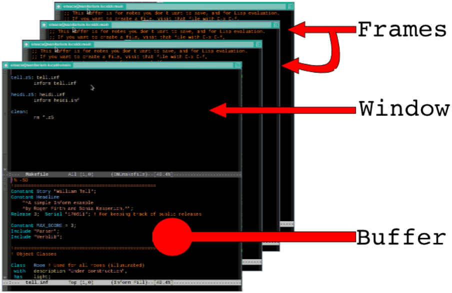
- Programmers editor. Customisable using elisp.
  - Customisability is a feature (not an add on).
  - Hence lots of “applications” in Emacs.
- Widely ported.
- Older than many of us.
- Almost religious following.

What is Emacs?

- Programmers editor. Customisable using elisp.
  - Customisability is a feature (not an add on).
  - Hence lots of “applications” in Emacs.
- Widely ported.
- Older than many of us.
- Almost religious following.

Basics

- Common terms have different meanings.



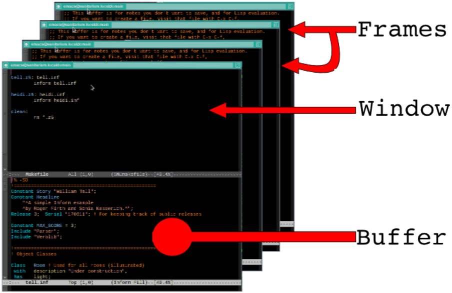
```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
```

```
tell.z5: tell.inf
        inform tell.inf

heidi.z5: heidi.inf
        inform heidi.inf

clean:
        rm *.z5
--:---  Makefile       All (1,0)      (GNUmakefile)--[49.4%]----------------
!% -SD
!============================================================
Constant Story "William Tell";
Constant Headline
    "^A simple Inform example
    ^by Roger Firth and Sonia Kesserich.^";
Release 3;  Serial "170611"; ! For keeping track of public releases

Constant MAX_SCORE = 3;
Include "Parser";
Include "Verblib";

!============================================================
! Object Classes

Class   Room ! Used for all rooms (illuminated)
 with   description "Under construction",
 has    light;
--:---  tell.inf       Top (1,0)      (Inform Fill)--[49.4%]----------------
```

- Major modes are “environments”.
- Minor modes are “utilities”.

Basics

- Common terms have different meanings.



```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
```

```
tell.z5: tell.inf
        inform tell.inf

heidi.z5: heidi.inf
        inform heidi.inf

clean:
        rm *.z5
```

```
--:---  Makefile       All (1,0)      (GNUmakefile)--[49.4%]----------------------
```

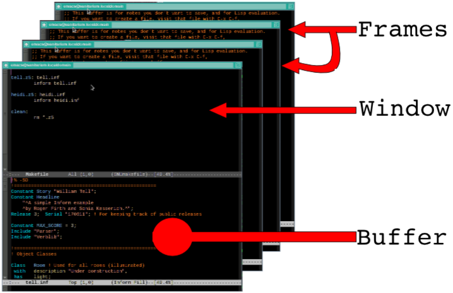
```
!% -SD
!===============================================
Constant Story "William Tell";
Constant Headline
    "^A simple Inform example
    ^by Roger Firth and Sonia Kesserich.^";
Release 3;  Serial "170811"; ! For keeping track of public releases

Constant MAX_SCORE = 3;
Include "Parser";
Include "Verblib";

!===============================================
! Object Classes

Class   Room ! Used for all rooms (illuminated)
 with   description "Under construction",
 has    light;
```

```
--:---  tell.inf       Top (1,0)      (Inform Fill)--[49.4%]----------------------
```

- `Major modes` are “environments”.
- `Minor modes` are “utilities”.

Basics

- Common terms have different meanings.



```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
```

```
tell.z5: tell.inf
        inform tell.inf

heidi.z5: heidi.inf
        inform heidi.inf

clean:
        rm *.z5
--:---  Makefile      All (1,0)      (GNUmakefile)--[49.4%]--------------------
!% -SD
!===============================================
Constant Story "William Tell";
Constant Headline
    "^A simple Inform example
    ^by Roger Firth and Sonia Kesserich.^";
Release 3;  Serial "170811"; ! For keeping track of public releases

Constant MAX_SCORE = 3;
Include "Parser";
Include "Verblib";

!===============================================
! Object Classes

Class   Room ! Used for all rooms (illuminated)
 with   description "Under construction",
 has    light;
--:---  tell.inf      Top (1,0)      (Inform Fill)--[49.4%]--------------------
```

- `Major modes` are “environments”.
- `Minor modes` are “utilities”.

Elisp

- Emacs' own dialect of lisp.
- Customisation and configuration language.
- All modes, utils etc. implemented using this.
- Quite old, not very fast but works.
- Arguably one of the most popular lisp dialects today.

Elisp

- Emacs' own dialect of lisp.
- Customisation and configuration language.
- All modes, utils etc. implemented using this.
- Quite old, not very fast but works.
- Arguably one of the most popular lisp dialects today.

Elisp

- Emacs' own dialect of lisp.
- Customisation and configuration language.
- All modes, utils etc. implemented using this.
- Quite old, not very fast but works.
- Arguably one of the most popular lisp dialects today.

Elisp

- Emacs' own dialect of lisp.
- Customisation and configuration language.
- All modes, utils etc. implemented using this.
- Quite old, not very fast but works.
- Arguably one of the most popular lisp dialects today.

Elisp

- Emacs’ own dialect of lisp.
- Customisation and configuration language.
- All modes, utils etc. implemented using this.
- Quite old, not very fast but works.
- Arguably one of the most popular lisp dialects today.

Useful features

- The kill ring
  - Each `copy` or `kill` (cut) is saved in a ring.
  - After doing a `yank` (paste), you can cycle through the ring.
- Keyboard Macros
  - Save a long sequence of keystrokes.
  - Replay them (sometimes with minor changes).
- Expansion
  - Expand abbreviations intelligently.

Useful features

- The kill ring
  - Each `copy` or `kill` (cut) is saved in a ring.
  - After doing a `yank` (paste), you can cycle through the ring.
- Keyboard Macros
  - Save a long sequence of keystrokes.
  - Replay them (sometimes with minor changes).
- Expansion
  - Expand abbreviations intelligently.

Useful features

- The kill ring
  - Each `copy` or `kill` (cut) is saved in a ring.
  - After doing a `yank` (paste), you can cycle through the ring.
- Keyboard Macros
  - Save a long sequence of keystrokes.
  - Replay them (sometimes with minor changes).
- Expansion
  - Expand abbreviations intelligently.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

Major modes

- There are 2 major modes for Python.
- `python.el` developed by the Emacs community.
- `python-mode.el` developed by the Python community.
- Both provide
  - Navigation.
  - Semantic selection.
  - Inferior interpreter process.
  - Intelligent indentation.
  - `PDBTrack` support.

python.el

- Distributed as part of Emacs.
- The newer of the modes.
- Less features than `python-mode.el`.
- Uses an `emacs.py` module to introspect buffer code.
- Inferior interpreter to evaluate buffers.
- Buffer examination using `pylint`.

python.el

- Distributed as part of Emacs.
- The newer of the modes.
- Less features than `python-mode.el`.
- Uses an `emacs.py` module to introspect buffer code.
- Inferior interpreter to evaluate buffers.
- Buffer examination using `pylint`.

python.el

- Distributed as part of Emacs.
- The newer of the modes.
- Less features than `python-mode.el`.
- Uses an `emacs.py` module to introspect buffer code.
- Inferior interpreter to evaluate buffers.
- Buffer examination using `pylint`.

python.el

- Distributed as part of Emacs.
- The newer of the modes.
- Less features than `python-mode.el`.
- Uses an `emacs.py` module to introspect buffer code.
- Inferior interpreter to evaluate buffers.
- Buffer examination using `pylint`.

python.el

- Distributed as part of Emacs.
- The newer of the modes.
- Less features than `python-mode.el`.
- Uses an `emacs.py` module to introspect buffer code.
- Inferior interpreter to evaluate buffers.
- Buffer examination using `pylint`.

python.el

- Distributed as part of Emacs.
- The newer of the modes.
- Less features than `python-mode.el`.
- Uses an `emacs.py` module to introspect buffer code.
- Inferior interpreter to evaluate buffers.
- Buffer examination using `pylint`.

python-mode.el

- Separately developed by the Python community.
- Older and more features in addition to `python.el`.
- Uses `pymacs` for code completion.
- Has an accompanying `doctest-mode`.
- Better syntax highlighting.

python-mode.el

- Separately developed by the Python community.
- Older and more features in addition to `python.el`.
- Uses `pymacs` for code completion.
- Has an accompanying `doctest-mode`.
- Better syntax highlighting.

python-mode.el

- Separately developed by the Python community.
- Older and more features in addition to `python.el`.
- Uses `pymacs` for code completion.
- Has an accompanying `doctest-mode`.
- Better syntax highlighting.

python-mode.el

- Separately developed by the Python community.
- Older and more features in addition to `python.el`.
- Uses `pymacs` for code completion.
- Has an accompanying `doctest-mode`.
- Better syntax highlighting.

python-mode.el

- Separately developed by the Python community.
- Older and more features in addition to `python.el`.
- Uses `pymacs` for code completion.
- Has an accompanying `doctest-mode`.
- Better syntax highlighting.

PyMacs

- PyMacs is a library that allows Emacs extensions to be written in Python.
- Runs a separate Python process.
- Communicates via. a lispy protocol.
- Many extensions use this.

PyMacs

- PyMacs is a library that allows Emacs extensions to be written in Python.
- Runs a separate Python process.
- Communicates via. a lispy protocol.
- Many extensions use this.

PyMacs

- PyMacs is a library that allows Emacs extensions to be written in Python.
- Runs a separate Python process.
- Communicates via. a lispy protocol.
- Many extensions use this.

PyMacs

- PyMacs is a library that allows Emacs extensions to be written in Python.
- Runs a separate Python process.
- Communicates via. a lispy protocol.
- Many extensions use this.

Virtualenv integration

- `virtualenv.el` allows setting virtualenv for Emacs.
- Simply specify the virtualenv and the major modes will use it.
- Never worked for me. :(
- Available at `https://github.com/aculich/virtualenv.el`

Virtualenv integration

- `virtualenv.el` allows setting virtualenv for Emacs.
- Simply specify the virtualenv and the major modes will use it.
- Never worked for me. :(
- Available at `https://github.com/aculich/virtualenv.el`

Virtualenv integration

- `virtualenv.el` allows setting virtualenv for Emacs.
- Simply specify the virtualenv and the major modes will use it.
- Never worked for me. :(
- Available at `https://github.com/aculich/virtualenv.el`

Virtualenv integration

- `virtualenv.el` allows setting virtualenv for Emacs.
- Simply specify the virtualenv and the major modes will use it.
- Never worked for me. :(
- Available at `https://github.com/aculich/virtualenv.el`

PDBTrack

- Following code as it is stepped through the debugger.
- Useful for the `import pdb; pdb.set_trace()` trick.
- When the interpreter enters the debugger, Emacs will track the active file.
- Works out of the box for both modes.

PDBTrack

- Following code as it is stepped through the debugger.
- Useful for the `import pdb; pdb.set_trace()` trick.
- When the interpreter enters the debugger, Emacs will track the active file.
- Works out of the box for both modes.

PDBTrack

- Following code as it is stepped through the debugger.
- Useful for the `import pdb; pdb.set_trace()` trick.
- When the interpreter enters the debugger, Emacs will track the active file.
- Works out of the box for both modes.

PDBTrack

- Following code as it is stepped through the debugger.
- Useful for the `import pdb; pdb.set_trace()` trick.
- When the interpreter enters the debugger, Emacs will track the active file.
- Works out of the box for both modes.

Syntax checkers

- Emacs has `flymake-mode` to run compliations and highlight errors.
- This can integrate with `pyflakes` or `pylint`.
- Highlights possible errors in your code as you type.

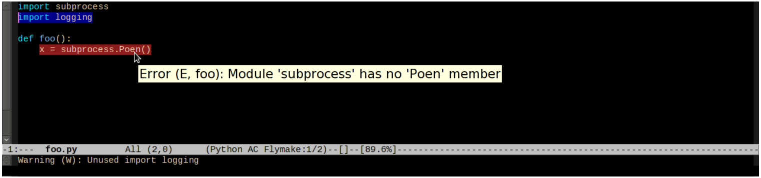
Syntax checkers

- Emacs has `flymake-mode` to run compliations and highlight errors.
- This can integrate with `pyflakes` or `pylint`.
- Highlights possible errors in your code as you type.

Syntax checkers

- Emacs has `flymake-mode` to run compliations and highlight errors.
- This can integrate with `pyflakes` or `pylint`.
- Highlights possible errors in your code as you type.

Syntax checkers



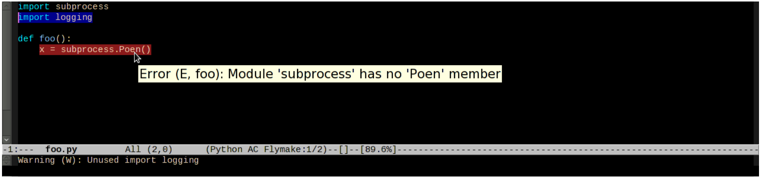
```
import subprocess
import logging

def foo():
    x = subprocess.Poen()
```

Error (E, foo): Module 'subprocess' has no 'Poen' member

```
-1:---  foo.py        All (2,0)      (Python AC Flymake:1/2)--[]--[89.6%]------------------------------------------------------------------------
Warning (W): Unused import logging
```

- Example with `pylint`.
- Uses (ugly) tooltips by default.
- Uses heuristics so not totally accurate.
- Not virtualenv aware.

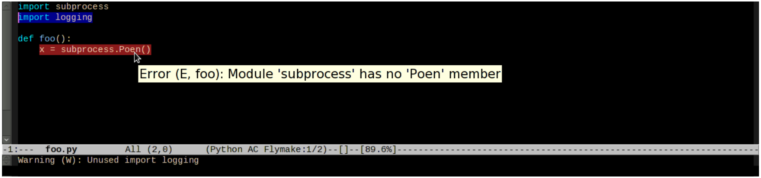
Syntax checkers



```
import subprocess
import logging

def foo():
    x = subprocess.Poen()
                             Error (E, foo): Module 'subprocess' has no 'Poen' member

-1:---  foo.py        All (2,0)      (Python AC Flymake:1/2)--[]--[89.6%]----------------------------
Warning (W): Unused import logging
```

- Example with `pylint`.
- Uses (ugly) tooltips by default.
- Uses heuristics so not totally accurate.
- Not virtualenv aware.

Syntax checkers



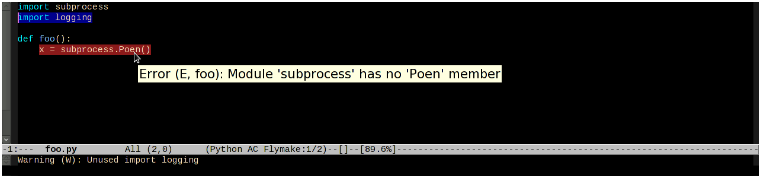
```
import subprocess
import logging

def foo():
    x = subprocess.Poen()
```

Error (E, foo): Module 'subprocess' has no 'Poen' member

```
-1:---  foo.py        All (2,0)      (Python AC Flymake:1/2)--[]--[89.6%]-------------------------------------------------------------------------------
Warning (W): Unused import logging
```

- Example with `pylint`.
- Uses (ugly) tooltips by default.
- Uses heuristics so not totally accurate.
- Not virtualenv aware.

Syntax checkers



```
import subprocess
import logging

def foo():
    x = subprocess.Poen()
```

Error (E, foo): Module 'subprocess' has no 'Poen' member

```
-1:---  foo.py        All (2,0)      (Python AC Flymake:1/2)--[]--[89.6%]------------------------------------------------------------------------
Warning (W): Unused import logging
```

- Example with `pylint`.
- Uses (ugly) tooltips by default.
- Uses heuristics so not totally accurate.
- Not virtualenv aware.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

Rope

- Rope is a Python refactoring library.
- You need PyMacs.
- Makes refactoring tools available.
  - Boilerplate for classes, functions etc.
  - Extraction, inlining.
  - Completion and assistance.
  - Finding occurrences.
- Undo is *outside* the regular emacs flow.
- I don't like things that generate code.

CEDET

- Heavy duty IDE. Part of Emacs now.
- Project support
- Mostly geared towards static languages.
- Speedbar (file/class/function tree)

CEDET

- Heavy duty IDE. Part of Emacs now.
- Project support
- Mostly geared towards static languages.
- Speedbar (file/class/function tree)

CEDET

- Heavy duty IDE. Part of Emacs now.
- Project support
- Mostly geared towards static languages.
- Speedbar (file/class/function tree)

CEDET

- Heavy duty IDE. Part of Emacs now.
- Project support
- Mostly geared towards static languages.
- Speedbar (file/class/function tree)

CEDET

- Heavy duty IDE. Part of Emacs now.
- Project support
- Mostly geared towards static languages.
- Speedbar (file/class/function tree)

Org mode

- Rule #6: Always mention `org-mode` in an Emacs talk.
- Org mode is an outline mode that can also be used as a PIM and to keep notes.
- Very powerful and worth exploring.
- Hard to describe without a demo.

Org mode

- Rule #6: Always mention `org-mode` in an Emacs talk.
- Org mode is an outline mode that can also be used as a PIM and to keep notes.
- Very powerful and worth exploring.
- Hard to describe without a demo.

Org mode

- Rule #6: Always mention `org-mode` in an Emacs talk.
- Org mode is an outline mode that can also be used as a PIM and to keep notes.
- Very powerful and worth exploring.
- Hard to describe without a demo.

Org mode

- Rule #6: Always mention `org-mode` in an Emacs talk.
- Org mode is an outline mode that can also be used as a PIM and to keep notes.
- Very powerful and worth exploring.
- Hard to describe without a demo.

Basics

- Create tasks.
- Set schedules and deadlines.
- Clock time spent.
- Create agendas.
- And finish them off.

Basics

- Create tasks.
- Set schedules and deadlines.
- Clock time spent.
- Create agendas.
- And finish them off.

Basics

- Create tasks.
- Set schedules and deadlines.
- Clock time spent.
- Create agendas.
- And finish them off.

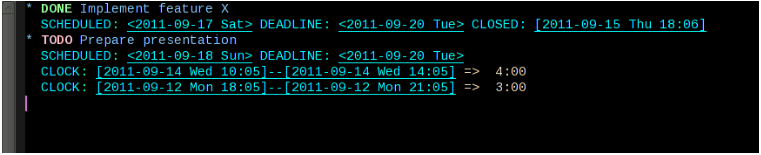
Basics

- Create tasks.
- Set schedules and deadlines.
- Clock time spent.
- Create agendas.
- And finish them off.

Basics

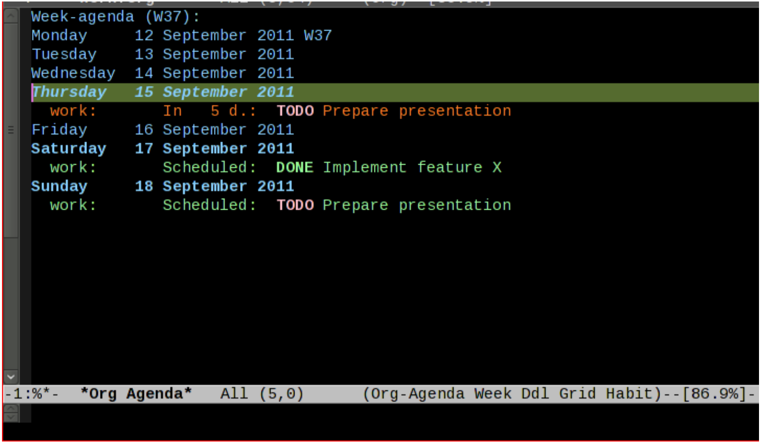
- Create tasks.
- Set schedules and deadlines.
- Clock time spent.
- Create agendas.
- And finish them off.

Sample tasks



```
* DONE Implement feature X
  SCHEDULED: <2011-09-17 Sat> DEADLINE: <2011-09-20 Tue> CLOSED: [2011-09-15 Thu 18:06]
* TODO Prepare presentation
  SCHEDULED: <2011-09-18 Sun> DEADLINE: <2011-09-20 Tue>
  CLOCK: [2011-09-14 Wed 10:05]--[2011-09-14 Wed 14:05] =>  4:00
  CLOCK: [2011-09-12 Mon 18:05]--[2011-09-12 Mon 21:05] =>  3:00
```

Agenda



```
Week-agenda (W37):
Monday      12 September 2011 W37
Tuesday     13 September 2011
Wednesday   14 September 2011
Thursday    15 September 2011
  work:        In   5 d.:  TODO Prepare presentation
Friday      16 September 2011
Saturday    17 September 2011
  work:        Scheduled:  DONE Implement feature X
Sunday      18 September 2011
  work:        Scheduled:  TODO Prepare presentation
-1:%*-  *Org Agenda*   All (5,0)     (Org-Agenda Week Ddl Grid Habit)--[86.9%]-
```

Sources of tasks

- #TBD while coding.
- Via. Email ("Can you do this?").
- Via. Chat message ("Can you do this?").
- Via browser ("Nice article. I need to read this.").
- Via. real life ("Need to buy textbooks.").
- Repetitive tasks ("Need to pay rents").

Sources of tasks

- `#TBD` while coding.
- Via. Email ("Can you do this?").
- Via. Chat message ("Can you do this?").
- Via browser ("Nice article. I need to read this.").
- Via. real life ("Need to buy textbooks.").
- Repetitive tasks ("Need to pay rents").

Sources of tasks

- `#TBD` while coding.
- Via. Email ("Can you do this?").
- Via. Chat message ("Can you do this?").
- Via browser ("Nice article. I need to read this.").
- Via. real life ("Need to buy textbooks.").
- Repetitive tasks ("Need to pay rents").

Sources of tasks

- `#TBD` while coding.
- Via. Email ("Can you do this?").
- Via. Chat message ("Can you do this?").
- Via browser ("Nice article. I need to read this.").
- Via. real life ("Need to buy textbooks.").
- Repetitive tasks ("Need to pay rents").

Sources of tasks

- `#TBD` while coding.
- Via. Email ("Can you do this?").
- Via. Chat message ("Can you do this?").
- Via browser ("Nice article. I need to read this.").
- Via. real life ("Need to buy textbooks.").
- Repetitive tasks ("Need to pay rents").

Sources of tasks

- `#TBD` while coding.
- Via. Email (“Can you do this?”).
- Via. Chat message (“Can you do this?”).
- Via browser (“Nice article. I need to read this.”).
- Via. real life (“Need to buy textbooks.”).
- Repetitive tasks (“Need to pay rents”).

Org-capture

- Single keystroke (`C-c r`) to *capture* something.
- Captures current “context” as an org-mode task.
- Works with email, code, chat buffers.
- Hipster PDA to capture real life tasks.
- Org can natively handle repetitive tasks.
- Once in org, you can schedule etc. it.

Org-capture

- Single keystroke (`C-c r`) to *capture* something.
- Captures current "context" as an org-mode task.
- Works with email, code, chat buffers.
- Hipster PDA to capture real life tasks.
- Org can natively handle repetitive tasks.
- Once in org, you can schedule etc. it.

Org-capture

- Single keystroke (`C-c r`) to *capture* something.
- Captures current “context” as an org-mode task.
- Works with email, code, chat buffers.
- Hipster PDA to capture real life tasks.
- Org can natively handle repetitive tasks.
- Once in org, you can schedule etc. it.

Org-capture

- Single keystroke (`C-c r`) to *capture* something.
- Captures current “context” as an org-mode task.
- Works with email, code, chat buffers.
- Hipster PDA to capture real life tasks.
- Org can natively handle repetitive tasks.
- Once in org, you can schedule etc. it.

Org-capture

- Single keystroke (`C-c r`) to *capture* something.
- Captures current “context” as an org-mode task.
- Works with email, code, chat buffers.
- Hipster PDA to capture real life tasks.
- Org can natively handle repetitive tasks.
- Once in org, you can schedule etc. it.

Org-capture

- Single keystroke (`C-c r`) to *capture* something.
- Captures current “context” as an org-mode task.
- Works with email, code, chat buffers.
- Hipster PDA to capture real life tasks.
- Org can natively handle repetitive tasks.
- Once in org, you can schedule etc. it.

Resources

- `http://www.emacswiki.org/emacs/PythonProgrammingInEmacs`
- `http://orgmode.org/` and `http://members.optusnet.com.au/ charles57/GTD/`
- `http://nibrahim.net.in/2011/07/17/my_org_mode_setup.html`
- `https://github.com/nibrahim/Config-files`

Questions