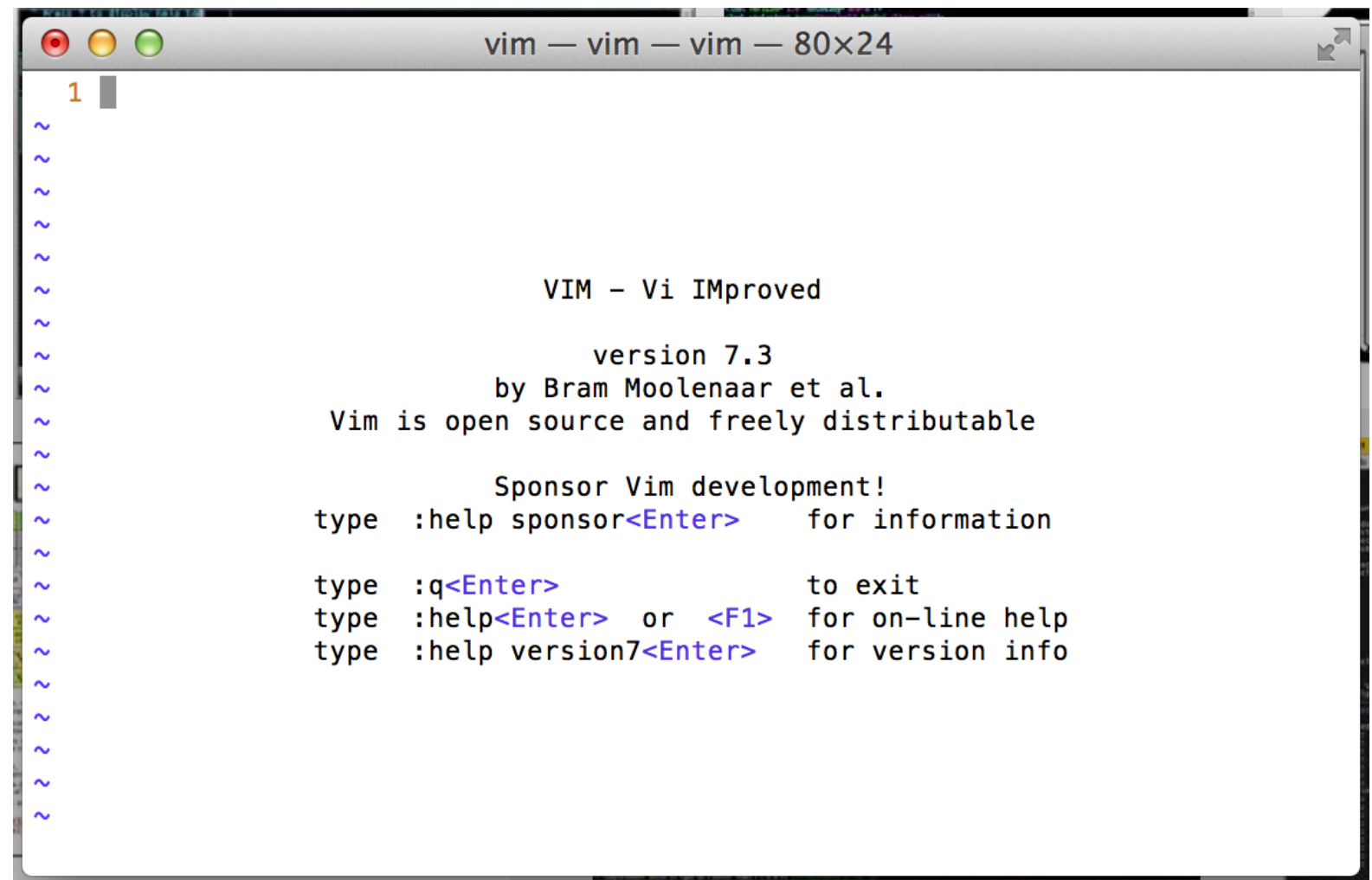


Vim

@ibotdotout

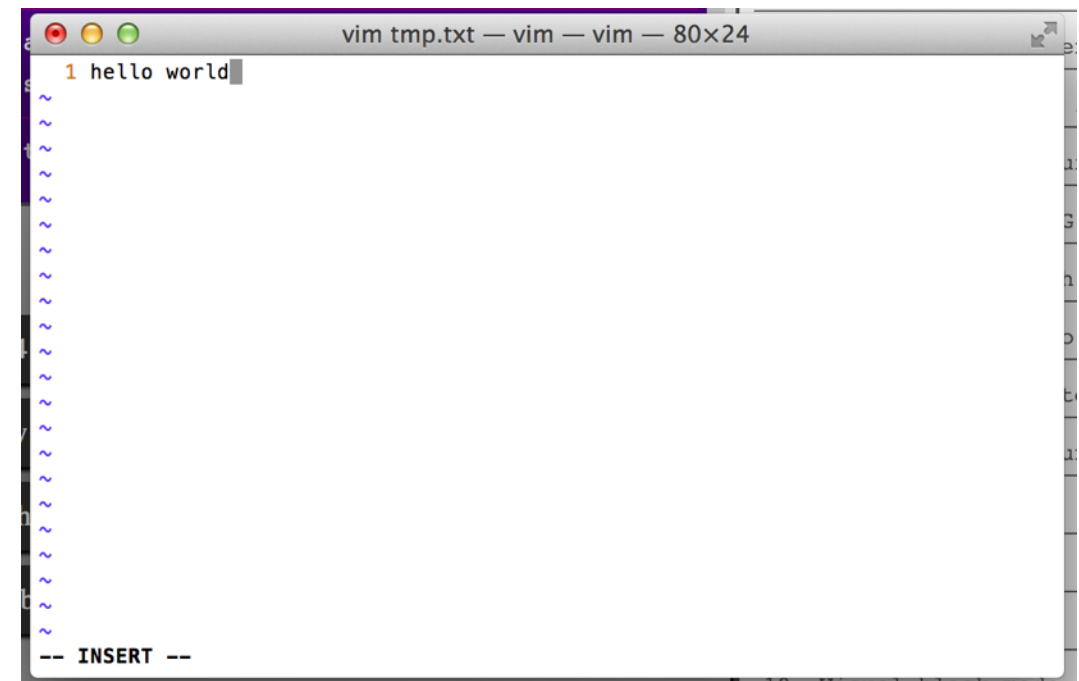
Why is Vim ?

- faster
- simple
- scriptable
- a lot plugin

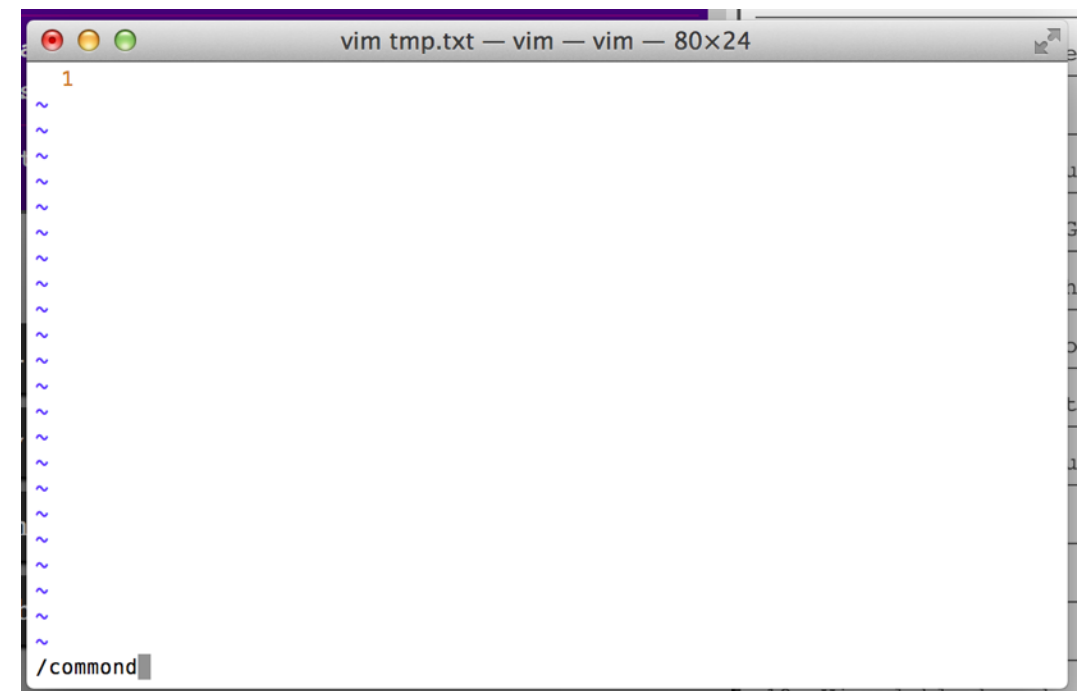
A screenshot of a Vim terminal window. The window title bar shows 'vim — vim — vim — 80x24'. The terminal content displays the Vim startup screen, which includes the text 'VIM - Vi IMproved', 'version 7.3', 'by Bram Moolenaar et al.', and 'Vim is open source and freely distributable'. It also lists several help commands: ':help sponsor' for information, ':q' to exit, ':help' or '<F1>' for on-line help, and ':help version7' for version info. The left margin of the terminal shows line numbers from 1 to 24, with the first line being the start of the screen.

Two Mode

- Insert (Texting)
- Command
 - movement
 - search
 - replace
 - insert/delete



A screenshot of a Vim editor window titled "vim tmp.txt — vim — vim — 80x24". The window shows a single line of text "1 hello world" on line 1. The status bar at the bottom indicates "-- INSERT --".



A screenshot of a Vim editor window titled "vim tmp.txt — vim — vim — 80x24". The window shows a single line of text "1" on line 1. The status bar at the bottom indicates "/command".

vi / vim graphical cheat sheet

Esc
normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence) end sentence	"soft" bol down	+ next line
\ goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto ³ format
Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank ^{1,3}	u undo	i insert mode	o open below	p paste after ¹	[misc]	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	. ex cmd line	" reg. ¹ spec	bol/ goto col	
a append	s subst char	d delete ^{1,3}	f find char	g extra ⁶ cmds	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	' goto mk. bol	\ not used!	
Z quit ⁴	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- ³ indent	> indent ³	? find (rev.)			
Z extra ⁵ cmds	X delete char	c change ^{1,3}	v visual mode	b prev word	n next (find)	m set mark	, reverse t/T/f/F	. repeat cmd	/ find			

motion	moves the cursor, or defines the range for an operator
command	direct action command, if red , it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input
q.	commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: quux(foobaz);
WORDS: quux(foobaz);

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

Open

- `vim` - open with blank
- `vim <file>` - new file / edit file
- `:open <file>` - open file
- `:edit <file>` - open file

Quite

- `:q` - quite
- `:q!` - quite without save
- `:wq` - save and quite
- `:w <file>` - save

Movement

- Basic (Arrow) : h, j, k, j
- Word movement: w, e, b
- Number powered movement: 5w

- Move to head/end of line: ^ , \$
- Move to begin/end of file: gg, G
- Go to line: :<line number> ex. :3

Insert

- i - insert at cursor
- a - insert after cursor
- I - insert at first of line
- A - insert at end of line
- o - insert line after cursor
- O - insert line before cursor

Delete

- x,X - delete char
- dw, 2dw - delete word
- dd,4dd - delete line
- . repeat command

Search

- /<text> - to search
- n,N - move target

Replace

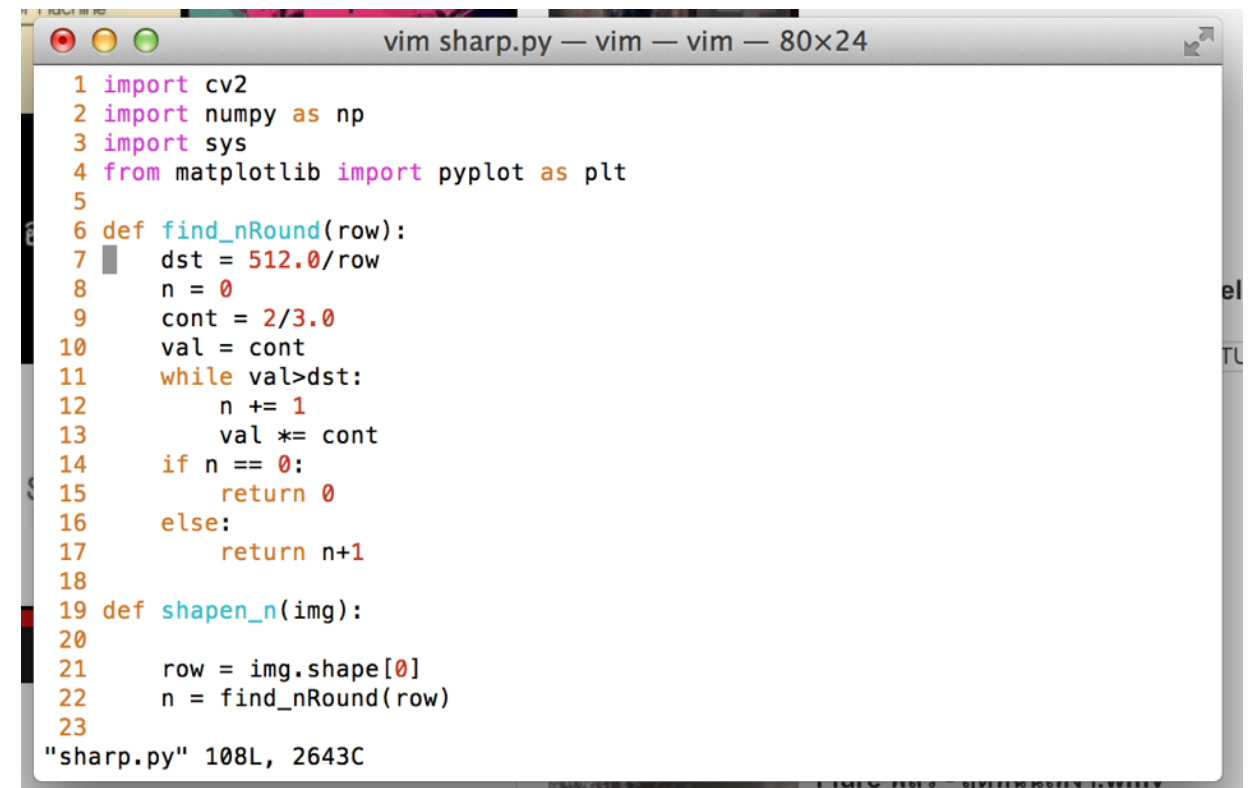
- `:<area>s/<text>/<replace>/g<flag>`
- — `<area>` { (blank), %, start,end }
- — `<flag>` { i - case insensitive , c - confirm }
- ex. `:%s/hello/HELLO/g`

Yank (Copy)

- yy,3yy - copy line
- yw,2yw - copy word
- p,P - paste
- ctrl+v - visual mode

Another

- :syntax on
 - enable syntax highlight
- :nu
 - show line number



The screenshot shows a vim editor window titled "vim sharp.py — vim — vim — 80x24". The window displays a Python script with syntax highlighting and line numbers. The code is as follows:

```
1 import cv2
2 import numpy as np
3 import sys
4 from matplotlib import pyplot as plt
5
6 def find_nRound(row):
7     dst = 512.0/row
8     n = 0
9     cont = 2/3.0
10    val = cont
11    while val>dst:
12        n += 1
13        val *= cont
14    if n == 0:
15        return 0
16    else:
17        return n+1
18
19 def shapen_n(img):
20
21     row = img.shape[0]
22     n = find_nRound(row)
23
```

The status bar at the bottom of the window shows "sharp.py" 108L, 2643C.

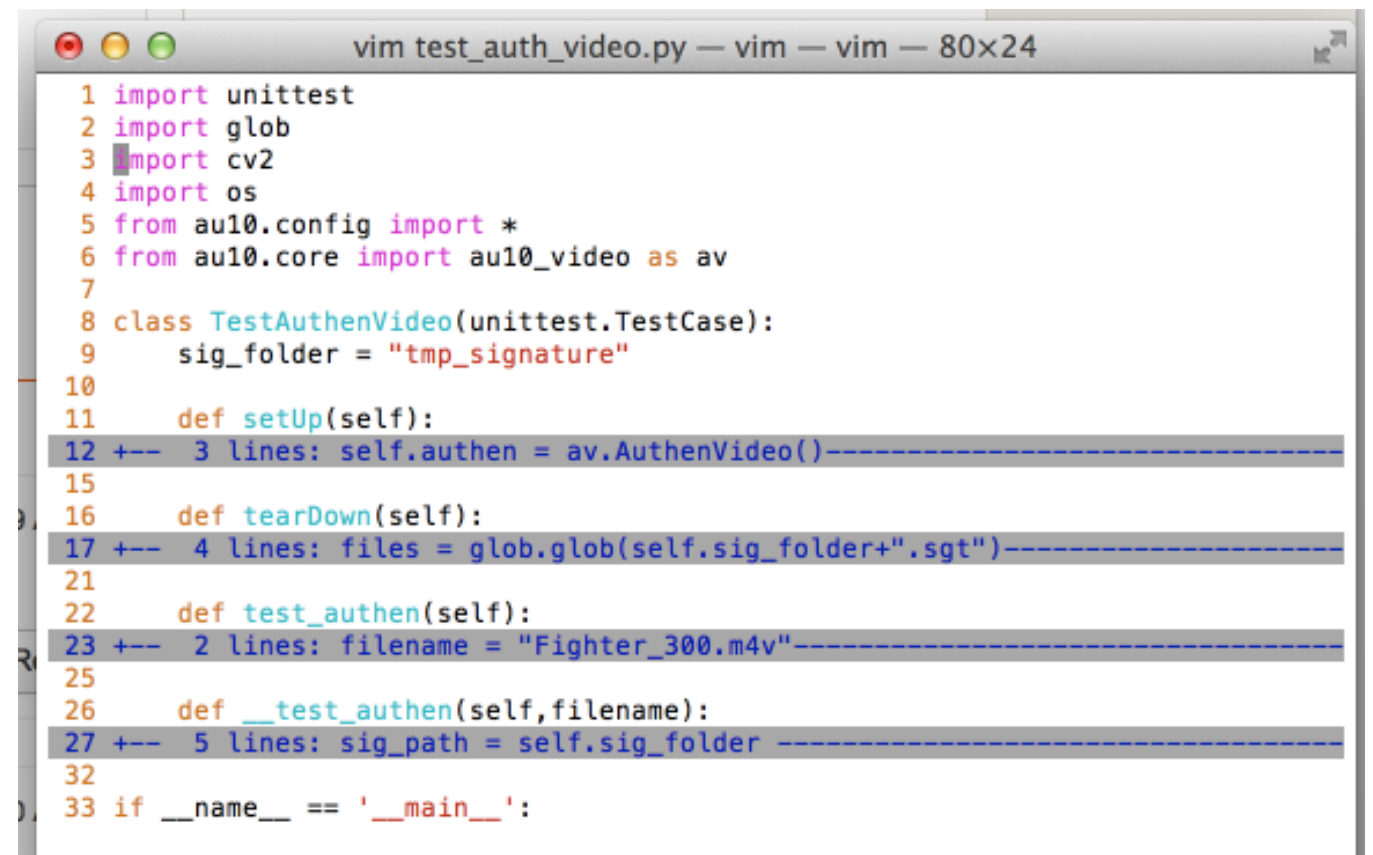
Any word completion

- ctrl + n , ctrl + p

[illegible]

Folding

- :<line,line>fold - fold
- zM - closes all fold
- zO - open all fold
- zd - delete fold



The screenshot shows a Vim editor window titled "vim test_auth_video.py — vim — vim — 80x24". The code is a Python file named "test_auth_video.py" containing imports, a class definition, and several methods. The code is color-coded, and several sections are folded, indicated by blue dashed lines and fold markers. The folded sections are:

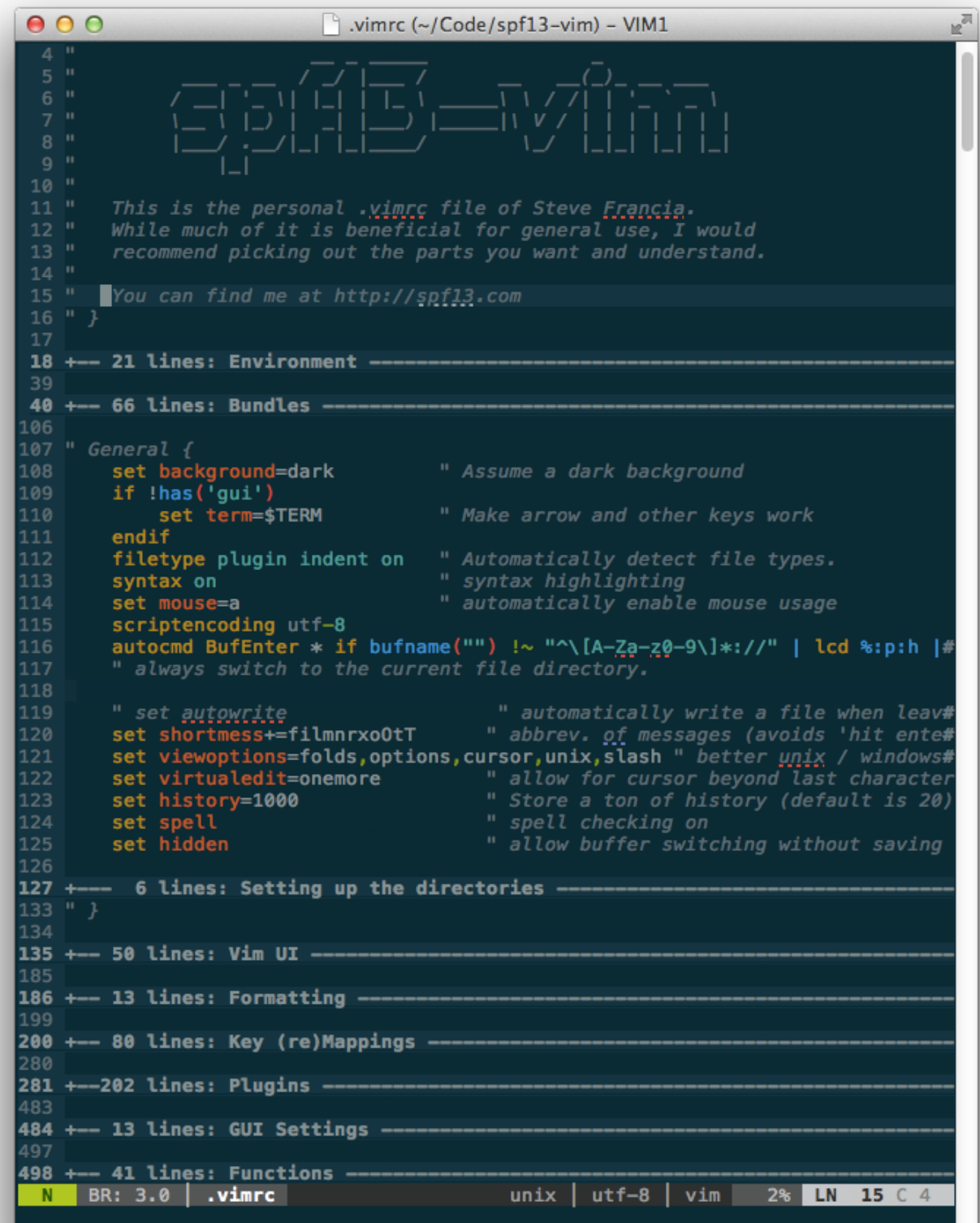
```
1 import unittest
2 import glob
3 import cv2
4 import os
5 from au10.config import *
6 from au10.core import au10_video as av
7
8 class TestAuthenVideo(unittest.TestCase):
9     sig_folder = "tmp_signature"
10
11     def setUp(self):
12 +-- 3 lines: self.authen = av.AuthenVideo()-----
15
16     def tearDown(self):
17 +-- 4 lines: files = glob.glob(self.sig_folder+".sgt")-----
21
22     def test_authen(self):
23 +-- 2 lines: filename = "Fighter_300.m4v"-----
25
26     def __test_authen(self,filename):
27 +-- 5 lines: sig_path = self.sig_folder -----
32
33 if __name__ == '__main__':
```


~/vimrc

- syntax on
- set nu
- set smartindent
- set tabstop=4
- au BufWinLeave * mkview
- au BufWinEnter * silent loadview

Another Vim

- jedi-vim
- neovim
- spf13-vim



The screenshot shows a Vim editor window titled ".vimrc (~/.Code/spf13-vim) - VIM1". The editor displays the contents of the .vimrc file, which includes a large ASCII art logo for "spf13-vim" at the top. Below the logo, there is a comment block explaining that this is the personal .vimrc file of Steve Francia, and it is recommended to pick out the parts one wants and understand. The file is organized into sections separated by dashed lines, including "Environment", "Bundles", "General", "Setting up the directories", "Vim UI", "Formatting", "Key (re)Mappings", "Plugins", "GUI Settings", and "Functions". The "General" section contains various Vim settings such as background, term, filetype, syntax, mouse, scriptencoding, autocmd, and more. The status bar at the bottom shows "N BR: 3.0 | .vimrc unix | utf-8 | vim 2% LN 15 C 4".

```
4 "  
5 "  
6 "  
7 "  
8 "  
9 "  
10 "  
11 " This is the personal .vimrc file of Steve Francia.  
12 " While much of it is beneficial for general use, I would  
13 " recommend picking out the parts you want and understand.  
14 "  
15 " You can find me at http://spf13.com  
16 " }  
17  
18 +--- 21 lines: Environment ---  
39  
40 +--- 66 lines: Bundles ---  
106  
107 " General {  
108     set background=dark           " Assume a dark background  
109     if !has('gui')  
110         set term=$TERM           " Make arrow and other keys work  
111     endif  
112     filetype plugin indent on     " Automatically detect file types.  
113     syntax on                     " syntax highlighting  
114     set mouse=a                   " automatically enable mouse usage  
115     scriptencoding utf-8  
116     autocmd BufEnter * if bufname("") !~ "^\\[A-Za-z0-9\\]*:/" | lcd %:~:~ |  
117     " always switch to the current file directory.  
118  
119     " set autowrite                 " automatically write a file when leav#  
120     set shortmess+=filnrxo0tT     " abbrev. of messages (avoids 'hit ente#  
121     set viewoptions=folds,options,cursor,unix,slash " better unix / windows#  
122     set virtualedit=onemore       " allow for cursor beyond last character  
123     set history=1000              " Store a ton of history (default is 20)  
124     set spell                     " spell checking on  
125     set hidden                    " allow buffer switching without saving  
126  
127 +--- 6 lines: Setting up the directories ---  
133 " }  
134  
135 +--- 50 lines: Vim UI ---  
185  
186 +--- 13 lines: Formatting ---  
199  
200 +--- 80 lines: Key (re)Mappings ---  
280  
281 +---202 lines: Plugins ---  
483  
484 +--- 13 lines: GUI Settings ---  
497  
498 +--- 41 lines: Functions ---  
N BR: 3.0 | .vimrc unix | utf-8 | vim 2% LN 15 C 4
```

Resource

- Fb:Thai Vim User Grop - <http://on.fb.me/1IZ5DI8>
- (Interactive tutorial) - <http://www.openvim.com/tutorial.html>
- (Game tutorial) - <http://vim-adventures.com>
- (learn resource) - <https://learn.thoughtbot.com/vim>
- Byte of Vim (Book) - <http://www.swaroopch.com/notes/vim/>
- Folding - <http://www.linux.com/learn/tutorials/442438-vim-tips-folding-fun>
- Vim Cheat Sheet - <http://www.glump.net/files/2012/08/vi-vim-cheat-sheet-and-tutorial.pdf>