

sontek (John M. Anderson)

May 11, 2008

Python with a modular IDE (Vim)

Filed under: [Programming](#), [python](#), [vim](#) — Tags: [Programming](#), [python](#), [vim](#) — sontek @ 3:18 pm

On Thursday, May 9th, 2008 the Utah Python User Group decided to settle the debate that has plagued us developers since the beginning of time: If you were a programming language, what editor would you use?

I was tasked with showing Eclipse with the PyDev plugin in all its glory-but we all know-[real men](#) / [developers](#) don't use IDE's, so we are going to talk about using Python and Vim together, reaching a state of Zen that the Dalai LLama would be jealous of and establishing more Feng Shui than Martha Stewart's Kitchen.

Freely jump between your code and python class libraries

There are 2 ways to add your ability to jump between python class libraries, the first is to setup vim to know where the Python libs are so you can use 'gf' to get to them (gf is goto file). You can do this by adding this snippet to your .vimrc:

```
python << EOF
import os
import sys
import vim
for p in sys.path:
    if os.path.isdir(p):
        vim.command(r"set path+=%s" % (p.replace(" ", r"\ ")))
EOF
```

With that snippet you will be able to go to your import statements and hit 'gf' on one of them and it'll jump you to that file.

Continuing accessibility of the Python class libraries we are going to want to use [ctags](#) to generate an index of all the code for vim to reference:

```
$ ctags -R -f ~/.vim/tags/python.ctags /usr/lib/python2.5/
```

and then in your .vimrc

```
set tags+=$HOME/.vim/tags/python.ctags
```

This will give you the ability to use CTRL+] to jump to the method/property under your cursor in the system libraries and CTRL+T to jump back to your source code.

I also have 2 tweaks in my .vimrc so you can use CTRL+LeftArrow and CTRL+RightArrow to move between the files with more natural key bindings.

```
map <silent><C-Left> <C-T>
map <silent><C-Right> <C-]>
```

You can also see all the tags you've been to with ":tags"

Code Completion

To enable code completion support for Python in Vim you should be able to add the following line to your .vimrc:

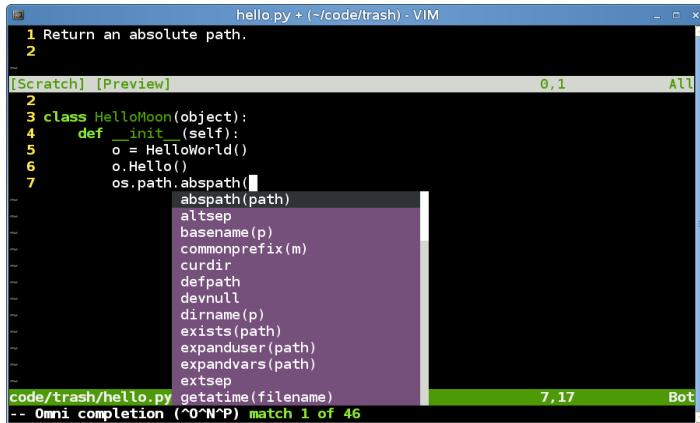
```
autocmd FileType python set omnifunc=pythoncomplete#Complete
```

but this relies on the fact that your distro compiled python support into vim (which they should!).

Then all you have to do to use your code completion is hit the unnatural, wrist breaking, keystrokes CTRL+X, CTRL+O. I've re-bound the code completion to CTRL+Space since we are making vim an IDE! Add this command to your .vimrc to get the better keybinding:

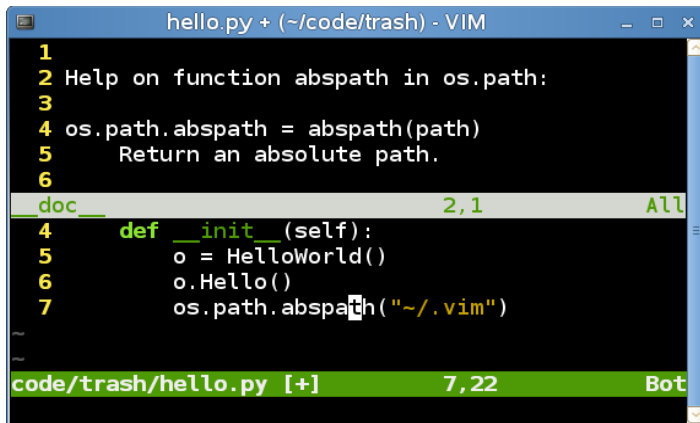
```
inoremap <Nul> <C-x><C-o>
```

Along with code completion, you will also have call tip support. Here is a screenshot:



Documentation

No IDE is complete without the ability to access the class libraries documentation! You'll need to grab [this](#) vim plugin. This gives you the ability to type :Pydoc os.path or use the keystrokes <Leader>pw and <Leader>pW to search for the item under the cursor. (Vim's default <Leader> is "\"). Here is a screenshot:



Syntax Checking

Vim already has built in syntax highlighting for python but I have a small tweak to vim to give you notifications of small syntax errors like forgetting a colon after a for loop. Create a file called ~/.vim/syntax/python.vim and add the following into it:

```
syn match pythonError "^\\s*def\\s\\+\\w\\+(\\.*)\\s*$" display
syn match pythonError "^\\s*class\\s\\+\\w\\+(\\.*)\\s*$" display
syn match pythonError "^\\s*for\\s\\.\\s*[^:]+$" display
syn match pythonError "^\\s*except\\s*$" display
syn match pythonError "^\\s*finally\\s*$" display
syn match pythonError "^\\s*try\\s*$" display
syn match pythonError "^\\s*else\\s*$" display
syn match pythonError "^\\s*else\\s*[^:]*$" display
syn match pythonError "^\\s*if\\s\\.\\s*[^:]+$" display
syn match pythonError "^\\s*except\\s\\.\\s*[^:]+$" display
syn match pythonError "[;]+$" display
syn keyword pythonError do
```

Now that you have the basics covered, lets get more complicated checking added. Add these 2 lines to your .vimrc so you can type :make and get a list of syntax errors:

```
autocmd BufRead *.py set makeprg=python\ -c\ \"import\ py_compile,sys;\ sys.stderr=sys.stdout;\
py_compile.compile(r'%')\"
autocmd BufRead *.py set efm=%C\ %.%#,%A\ \ File\ \"%f\"\\,\ line\ %l%.%#,%Z%[%^\ ]%\\@=%m
```

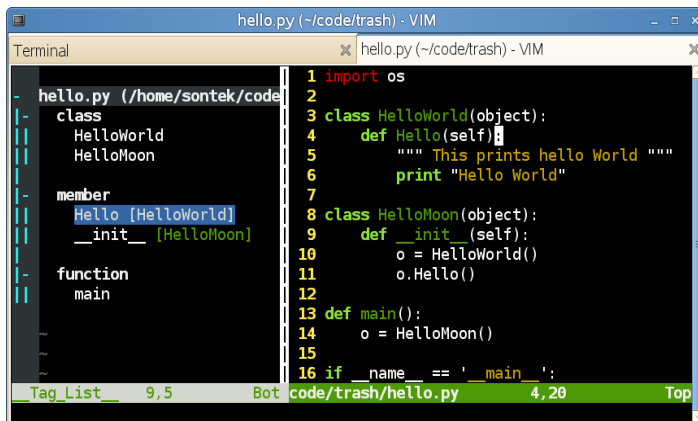
You will have the ability to type :cn and :cp to move around the error list. You can also type :clist to see all the errors, and finally, sometimes you will want to check the syntax of small chunks of code, so we'll add the ability to execute visually selected lines of code, add this snippet to your .vimrc:

```
python << EOL
import vim
def EvaluateCurrentRange():
    eval(compile('\n'.join(vim.current.range), '', 'exec'), globals())
EOL
map <C-h> :py EvaluateCurrentRange()
```

Now you will be able to visually select a method/class and execute it by hitting “Ctrl+h”.

Browsing the source

Moving around the source code is an important feature in most IDE's with their project explorers, so to get that type of functionality in vim we grab the [Tag List](#) plugin. This will give you the ability to view all opened buffers easily and jump to certain method calls in those buffers. Here is a screenshot of it in action:



The other must-have feature of an IDE when browsing code is being able to open up multiple files in tabs. To do this you type :tabnew to open up a file in a new tab and then :tabn and :tabp to move around the tabs. Add these to lines to your .vimrc to be able to move between the tabs with ALT+LeftArrow and ALT+RightArrow:

```
map <silent><A-Right> :tabnext<CR>
map <silent><A-Left> :tabprevious<CR>
```

Debugging

To add debugging support into vim, we use the pdb module. Add this to your ~/.vim/ftplugin/python.vim to have the ability to quickly add break points and clear them out when you are done debugging:

```
python << EOF
def SetBreakpoint():
    import re
    nLine = int( vim.eval( 'line(".")' ))

    strLine = vim.current.line
    strWhite = re.search( '^\s*', strLine).group(1)
```

```

vim.current.buffer.append(
    "%(space)sbdb.set_trace() %(mark)s Breakpoint %(mark)s" %
    {'space':strWhite, 'mark': '#' * 30}, nLine - 1)

for strLine in vim.current.buffer:
    if strLine == "import pdb":
        break
    else:
        vim.current.buffer.append( 'import pdb', 0)
        vim.command( 'normal j1')

vim.command( 'map <f7> :py SetBreakpoint()<cr>')

def RemoveBreakpoints():
    import re

    nCurrentLine = int( vim.eval( 'line(".")' ))

    nLines = []
    nLine = 1
    for strLine in vim.current.buffer:
        if strLine == 'import pdb' or strLine.rstrip()[:15] == 'pdb.set_trace()':
            nLines.append( nLine)
            nLine += 1

    nLines.reverse()

    for nLine in nLines:
        vim.command( 'normal %dG' % nLine)
        vim.command( 'normal dd')
        if nLine < nCurrentLine:
            nCurrentLine -= 1

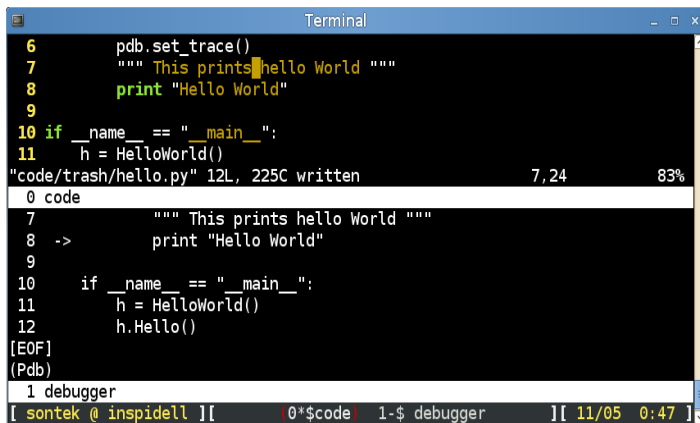
    vim.command( 'normal %dG' % nCurrentLine)

vim.command( 'map <s-f7> :py RemoveBreakpoints()<cr>')
EOF

```

With that code you can now hit F7 and Shift-F7 to add/remove breakpoints. Then you just launch your application with !python % (percent being the current file, you can declare your main file here if its different).

Another tweak I use is to have my vim inside screen with a horizontal split, that way I can see the python interpreter and debug while still having vim there so I can easily fix my code. Here is a screenshot of that in action:



```

Terminal
6      pdb.set_trace()
7      """ This prints hello World """
8      print "Hello World"
9
10 if __name__ == "__main__":
11     h = HelloWorld()
"code/trash/hello.py" 12L, 225C written          7,24      83%
0 code
7      """ This prints hello World """
8 ->     print "Hello World"
9
10     if __name__ == "__main__":
11         h = HelloWorld()
12         h.Hello()
[EOF]
(Pdb)
1 debugger
[sontek @ inspidell ][ 0*$code 1-$ debugger ][ 11/05 0:47 ]

```

Snippets

A great time saver with standard IDE's is code snippets, so you can type a few key strokes and get a lot of code out of it. An example of this would be a django model, instead of typing out the complete declaration you could type 'mmo<tab><tab>' and have a skeleton of your model done for you. To do this in vim we grab the [Snippets EMU](#) plugin.

Check out a great screencast of snippetsEmu in action [here](#)

You can get my full setup [here](#)

Emacs

[Here](#) is a great post on how to do the same with Emacs.

Share and Enjoy:



[Burmese Python Cages](#)

Unique attractive customer enclosures Request a free color catalog today!
www.CagesByDesign.com

[Python Resume](#)

Hire Qualified Tech Professionals. Post Job Listings & Connect at Dice
www.Dice.com

[Python IDE](#)

Code Intelligence and Debugger Wingware Python IDE - Free Eval
www.wingware.com

Ads by Google

71 Comments »



1.

Hey, great post - very useful.

Something happened with your code's indentation in the debugging section.. a little help there would be appreciated.

Thanks!

Comment by Johnny — May 11, 2008 @ [7:02 pm](#)



2.

Hi,

Good work and Very promising setup.

I am tempted to switch from emacs. I kind of thought I had seen the last of vi. Lets see how long I can hold out.

v.

Comment by Vj — May 11, 2008 @ [7:05 pm](#)



3.

Wow, thorough! I'll be using these tricks immediately, thanks.

Comment by Kevin — May 11, 2008 @ [7:47 pm](#)



4.

Sorry about the indentation with the debugging code, Heres the file for it: <http://sontek.net/dotfiles/vim/ftplugin/python.vim> . I also fixed the formatting in the post.

Comment by [sontek](#) — May 11, 2008 @ [7:55 pm](#)



5.

I've been looking for a good python setup with vim for quite some time. Thanks!

Comment by [Jamie](#) — May 11, 2008 @ [11:06 pm](#)



6.

Great post! I am using vim more and more lately and just embarking on a biggish python project. Your tips will really help me this time (except for debugging: real men don't use debuggers 😊)

One thing I keep wondering, is how I could get a python interactive shell inside vim. I guess that is a standard vim feature that I just haven't found?

Comment by [Daren Thomas](#) — May 11, 2008 @ [11:13 pm](#)



7.

Awesome post, What theme are you using?

Comment by [Max](#) — May 11, 2008 @ [11:16 pm](#)



8.

@Darren - You cannot get a python interpreter inside vim, The best way to do that is to use a split screen. Open up screen, and then create 2 windows, hit C-a C-S, that'll create a split. Then open up your python interpreter in one and your vim in the other. It'll look like:

http://sontek.net/blog_pictures/vim_debugging_with_screen.png

You can also look at my .screenrc and .vimrc to see tweaks I've added to make that easier.

Comment by [sontek](#) — May 11, 2008 @ [11:20 pm](#)



9.

@Max I'm using Tango theme, you can get it here: <http://sontek.net/dotfiles/vim/colors/tango.vim>

Comment by [sontek](#) — May 11, 2008 @ [11:20 pm](#)



10.

what about a ropelib integration ?

Comment by [marco](#) — May 12, 2008 @ [12:14 am](#)



11.

@marco:
There is an bicycle repair man integration for vim, works great for refactoring and the 'goto definition' works better than all other solutions shown

Comment by [steffen](#) — May 12, 2008 @ [12:55 am](#)

12. 

Hi. Great post, but I'm having trouble with your basic pythonError syn match statements (i.e. they do nothing). What are they supposed to do? I expected a highlight, but as I said, nothing happens when I forget a colon.

Also, the quotes in all but the first two lines of that section are all non-default quotes which makes them copy and paste as invalid code.

Comment by [jack](#) — May 12, 2008 @ [1:36 am](#)

13. 

[...] a wonderful tutorial explaining how to turn my text editor of choice (vim) into a full nice ide for python. [...]

Pingback by [Turning vim into a python editor | Neuronical dot Com](#) — May 12, 2008 @ [1:56 am](#)

14. 

If you want to have pydoc integration in vim, try this: http://www.vim.org/scripts/script.php?script_id=910

fs111

Comment by [fs111](#) — May 12, 2008 @ [3:43 am](#)

15. 

Thanks for the article. There is a bug with some quote signs (guessing from wordpress):

```
"^\\s*finally\\s*$"
```

Easily fixed, of course (:%s/[""]/"/g)

Comment by [Johannes](#) — May 12, 2008 @ [3:49 am](#)

16. 

Thanks for great tips! I have just switched from PyDev to Vim and I'm really pleased :). I also use NERDTree plugin which works great for browsing Python packages.

Comment by [Artur](#) — May 12, 2008 @ [4:23 am](#)

17. 

Hijohn!

A great article/howto... very nice! But in my newbie condition, anything run as is supposed...

After looking for a "vim" module for a python, I find a vim.py, but here it is the error message when I make an "import vim" in a python shell:

```
>>> import vim
Traceback (most recent call last):
File "", line 1, in
File "/usr/lib/python2.5/vim.py", line 1
```

^

SyntaxError: invalid syntax

Anybody could help me, please? I would like to start learning something about python (this type of articles encourage me).

Thank you very much!!

Regards

Comment by [TooManySecrets](#) — May 12, 2008 @ [4:38 am](#)

18. 

[...] are a couple of good resources for configuring Vim and Emacs for Python development. I know more key-bindings with Emacs than Vim, so my preference is [...]

Pingback by [Nick Carroll » Blog Archive » Old School Editors for Python Development](#) — May 12, 2008 @ [4:48 am](#)

19. 

in your ctags link you have an apostrophe after the url link. 😞
and thankies vewwy vewwy muchies for this. I've been looking for an article like this for a long time.

Comment by [imonsei](#) — May 12, 2008 @ [5:27 am](#)

20. 

sontek (John M. Anderson) » Python with a modular IDE (Vim)...

sontek (John M. Anderson) » Python with a modular IDE (Vim)...

Trackback by [roScripts - Webmaster resources and websites](#) — May 12, 2008 @ [6:14 am](#)

21. 

These are great, thanks.

I am still working through them, but found that for the 'gf' jump to a library file, your _vimrc must also contain the line 'set suffixesadd+=.py'. Maybe somewhere in filetype handling is a better place for this line?

Comment by [Kevin](#) — May 12, 2008 @ [8:46 am](#)

22. 

I have this thing where i really dont like random trailing whitespace in my code, as it can make navigation of the file annoying sometimes. So I have found the following two bits to be very helpful:

```
au BufWritePre *.py mark `|:%s/\s\+$/|e|normal "  
nmap m` :s/\s\+$/|g"
```

The first will delete trailing whitespace on save. The second will delete trailing whitespace from the current line when you hit the Tab key, in normal mode.

Also: set list listchars=tab:» ,trail: ,extends:\$,nbsp:=
will make fancy markers show whitespace at the end of a line, so it's visible for me.

Comment by [Erich](#) — May 12, 2008 @ [10:23 am](#)

23. 

Configurar Vim como IDE de Python...

Conjunto de recetas de configuración y plugins que ayudarán a los usuarios de vim a configurar este editor como IDE de Python que poco tiene que envidiar a otras opciones mucho más "pesadas"....

Trackback by [meneame.net](#) — May 12, 2008 @ [10:55 am](#)

24. **Python Scripting mit der “Vim-IDE”...**


Ein netter Artikel, um seinen vim für das Programmieren mit Python zu konfigurieren. Hab ein paar nützliche Ideen daraus gezogen, die noch nicht in meiner \$HOME/.vimrc standen....

Trackback by [/dev/linkdump](#) — May 12, 2008 @ [11:01 am](#)

25. 

[...] sontek (John M. Anderson) » Python with a modular IDE (Vim) skill vim config to turn it into a python IDE (tags: vim) [...]

Pingback by [links for 2008-05-12 | a convenient truth](#) — May 12, 2008 @ [11:34 am](#)

26. 

If you want to make additions to the Python syntax file, put them in ~/.vim/after/syntax/python.vim. Using ~/.vim/syntax/python.vim will override the system's version.

Comment by Will — May 12, 2008 @ [11:56 am](#)

27. 

[...] hay varios artículos sobre vim además del mencionado anterior, como por ejemplo uno sobre usar VIM como IDE para Python y una pregunta sobre porqué usar VIM: Can someone explain the advantages of emacs/vim/nano over [...]

Pingback by [doce cosas : vim](#) — May 12, 2008 @ [1:26 pm](#)

28. 

I seem to be missing something fundamental, since no where online can I find the solution to this.

Where is the Python VIM module? (vim.py)? It is not included in my MacPython release for OS X.

I would love to switch to VIM on OS X, but I can't seem to get the first step working...

Blaine

Comment by [Blaine](#) — May 12, 2008 @ [3:38 pm](#)

29. 

Ok, so any help with OS X would be appreciated. With the ctags command, -R is apparently an illegal option?

Thanks!

Comment by [Blaine](#) — May 12, 2008 @ [3:46 pm](#)

30. 

[...] Python with a modular IDE (Vim) (tags: python vim programming tips development vi software) [...]

Pingback by [toshism » links for 2008-05-13](#) — May 12, 2008 @ [5:43 pm](#)

31. 

[...] Python with a modular IDE (Vim) Awesome tips for attaining vim + Python nirvana. (tags: vim python programming tutorial) [...]

Pingback by [links for 2008-05-13](#) — May 12, 2008 @ [10:37 pm](#)



[...] sontek (John M. Anderson) » Python with a modular IDE (Vim) (tags: development python vim vi howto) [...]

Pingback by [links for 2008-05-13 « Breyten's Dev Blog](#) — May 13, 2008 @ [4:30 am](#)



[...] been giving Django a try. John Anderson posted a very detailed, excellent blog post about using VIM as a python IDE. If you follow that post, and use synic's colorscheme, you pretty much have my IDE [...]

Pingback by [Here we are now, entertain us » Vim as a Django IDE](#) — May 13, 2008 @ [7:43 am](#)



'Course, there's an Emacs command to do that... <http://xkcd.com/378/>

Comment by [Yoooder](#) — May 13, 2008 @ [2:30 pm](#)



[...] sontek (John M. Anderson) » Python with a modular IDE (Vim) These are now in my ~/.vimrc. Turns vim into a very nice IDE. (tags: programming python vim development editor tips) This entry was written by delicious, posted on May 13, 2008 at 6:49 pm, filed under del.icio.us. Bookmark the permalink. Follow any comments here with the RSS feed for this post. Post a comment or leave a trackback: Trackback URL. « Twitter Updates for 2008-05-10 [...]

Pingback by [Reign Drops Fall... » links for 2008-05-14](#) — May 13, 2008 @ [5:48 pm](#)



@Blaine

In no specific order:

the ctags you need is exhuberant ctags, not the manky old ctags found by default in OSX. the command 'ctags -version' should return:

Exuberant Ctags 5.7, Copyright (C) 1996-2007 Darren Hiebert

Compiled: Sep 19 2007, 17:15:34

Addresses: , <http://ctags.sourceforge.net>

Optional compiled features: +wildcards, +regex

This is the ctags you need for all vim ctags magic.

the vim python module requires that your vim be build with python. run either 'vim -version' in the the shell or inside of vim ":version"

you need to see +python in the output of those somewhere. if not you'll need to recompile

Comment by [chiggsy](#) — May 14, 2008 @ [5:45 am](#)



Why ~/.vim/ folder isn't created on my system? I've used vim to edit some files, so I'm not starting it for

the first time. Anyway the directory has not been created. What's wrong? I'm using Ubuntu 8.04.

Thanks!

Comment by [Andrea Grandi](#) — May 14, 2008 @ [8:36 am](#)



Great job. Thanks a lot !

Comment by [Martin](#) — May 14, 2008 @ [1:42 pm](#)



If you have a svn checkout of Python you can get the most up-to-date version of the syntax file in Misc/Vim/vimrc . It's auto-generated so it is as up-to-date as the Python interpreter you use to run the script that creates the file (it's in the same directory). It is also structured to support PEP 7/8 (the style guides for Python).

Comment by [Brett](#) — May 14, 2008 @ [6:11 pm](#)



@chiggsy

Thanks for the help man! I'll recompile vi and see where it gets me. 😊 -blaine

Comment by [Blaine](#) — May 15, 2008 @ [6:18 pm](#)



@Blaine

If you are looking for the vim module John uses in his .vimrc file, know that it's a Vim thing and not a module distributed by Python. It's basically the bindings between Vim the editor and the Python environment.

The only thing that I would add to all of this is the recommendation to put this stuff into .vim/after scripts instead of sticking all of it into your .vimrc and protecting it with autocommands. It reduces .vimrc bloat, keeping that global settings file simple and clean, and helps to keep your python stuff together. Plus, putting this in the after directory protects your settings from upgrades.

@Andrea Grandi

Have you upgraded Vim since you've installed Ubuntu? It has been my experience that Ubuntu ships with a *minimal* version of Vim; there's a package you can apt-get which gives you the full-blown version of Vim. I believe it's called vim-full. Anyway, it's not like Vim is a hog of a program, and in my opinion shouldn't be distributed like shareware by default... but that could explain why your .vim directory isn't there. At any rate, it's as simple as `mkdir .vim` from a terminal.

Comment by [Erik](#) — May 20, 2008 @ [7:59 am](#)



Having just found myself a Python-coding job, I'm sure the effect of this article will be a life-changer.

Just one suggestion for the debug script: On the line "if strLine == 'import pdb' or strLine.lstrip()[:15] == 'pdb.set_trace()':" in RemoveBreakpoints(), would it not be a bit clearer to use "...strLine.lstrip().startswith('pdb.set_trace()')." ? Or am I missing something?

Comment by [Walter](#) — May 22, 2008 @ [1:41 am](#)



Hi,

this article is awesome, but I have problem with folding. How to use it in python when syntax method is chosen?

I cannot set and use folding features (hitting etc.)

Please help,
kgs

Comment by kgs — May 22, 2008 @ [5:14 am](#)

44. 

[...] there is a lot of information on the web about how to set up VI to use as a decent Python editor. This article by John M. Anderson was helpful to cover a bunch of the basics. I also found a great 6 easy steps [...]

Pingback by [SuperBoB » Blog Archive » Vim options for python](#) — May 23, 2008 @ [6:40 am](#)

45. 

[...] sontek (John M. Anderson) » Python with a modular IDE (Vim) Great tips for using Vim as a Python IDE. I don't do anything nearly this fancy with mine, but I definitely want to give it a shot now. (tags: python vim) [...]

Pingback by [McGrew Security Blog » Blog Archive » links for 2008-05-23](#) — May 23, 2008 @ [3:31 pm](#)

46. 

[...] sontek (John M. Anderson) » Python with a modular IDE (Vim) (tags: vimarticle) [...]

Pingback by [links for 2008-05-27 | 甘先生blog](#) — May 26, 2008 @ [9:31 pm](#)

47. 

[...] [...]

Pingback by [ides](#) — May 29, 2008 @ [1:24 pm](#)

48. 

Using your vim collection, highlighting of some python keywords doesn't work as it should.

try: <- doesn't get highlighted while
try : does.

The same goes for else:

other than that, nice job!

Comment by Henrik — June 1, 2008 @ [6:03 am](#)

49. 


Thanks a ton, this rocks

Comment by Jafraldo — June 4, 2008 @ [11:23 am](#)

50. 

This is really a good stuff. I love it! thanks.

Comment by Shrek — June 5, 2008 @ [11:15 pm](#)

51. 

Thanks a lot for these tips on setting up Vim as a Python IDE! I have a few suggestions:

- 1) You can easily hop between tabs using gt and gT in normal mode, as well. You can even use numbers to hop that many tabs, e.g., to hop 3 tabs to the right, do g3t
- 2) Items you suggest putting in .vimrc could alternatively be placed in ~/.vim/after/ftplugin/python.vim. Just ensure that your .vimrc has filetype plugin indent on
- 3) I would suggest moving ~/.vim// paths to ~/.vim/after//, ensuring that the system files are loaded first, then tweaking them with the additions presented here. This was hinted at by Will.
- 4) Use setlocal rather than set to ensure that changes apply only to the local (Python) buffer, not any other buffers you may be editing in the same Vim session.

Many thanks for writing this up!
Chris

Comment by [Chris Lasher](#) — June 9, 2008 @ [2:35 pm](#)

52. 

[...] de forma confortável, contendo todas as funcionalidades encontradas nas grandes IDEs. O post é esse aqui. Um excelente post, dispensa qualquer adição da minha [...]

Pingback by [Vim com esteróides para python « linil](#) — June 10, 2008 @ [11:44 am](#)

53. 

Wow, awesome. I loved vim before, but now my productivity has gone through the roof. Thanks!

Comment by [Joe Smith](#) — June 11, 2008 @ [10:28 am](#)

54. 

Thank you very much, you saved my life

Comment by kakarotoBR — June 14, 2008 @ [7:36 am](#)

55. 

An IDE with no code folding? — Booooo! — Very helpful tips though 😊 - Why do you recommend for code folding because currently when I go to a function and hit zf it says:

E350: Cannot create fold with current 'foldmethod'

Where before I could at least fold the code manually.

Comment by Bob — June 17, 2008 @ [11:47 am](#)

56. 

@Bob I prefer to not use code folding, but I've tested this script: http://www.vim.org/scripts/script.php?script_id=1494 and it worked well.

Comment by [sontek](#) — June 22, 2008 @ [3:18 pm](#)

57. 

The link to your install isn't working, it's saying the file isn't found. Any chance of you correcting this, would love to have a copy of your setup.

Comment by RockDJ — June 24, 2008 @ [6:30 pm](#)

58. 


Yeah, Sorry, I deleted the tar ball last night because I was updating the scripts. I'll add it back in a few, sorry!

Comment by [sontek](#) — June 25, 2008 @ [8:16 am](#)

59. 

Can you revive the link of your setup . Would like to use it.

Comment by [Sharad](#) — July 2, 2008 @ [8:05 am](#)

60. 

I've resubmitted the download link, sorry about that guys

Comment by [sontek](#) — July 2, 2008 @ [11:47 am](#)

61. 

Nice article, thank you. Some tweaks:

“Dalai LLama” should be “Dalai Lama”.

The second invocation of “python <<” should be “python << EOF” (and the end delimiter corrected to “EOF” also), like the first.

And thanks for giving a reference to equal coverage for Emacs 😊

Comment by [bignose](#) — July 4, 2008 @ [12:49 am](#)

62. 

I actually do not have to set omnifunc for my Python setup to get CTRL-X CTRL-O to work. Only for CSS do I need to due to the defacto #Complete being named #CompleteCSS there.

Comment by [Jeroen Ruigrok van der Werven](#) — July 4, 2008 @ [2:29 am](#)

63. 

I like to control a running IPython using VIM shortcuts, but this can be used to control any other terminal:

http://vim.wikia.com/wiki/IPython_integration

Comment by [mark dufour](#) — July 4, 2008 @ [4:50 am](#)

64. 

There is a typo in your .vimrc line 115:

```
au FileType python if &ft !~ 'django' | setlocal filetype=python.django_tempate.django_model | endif
```

Anyhow I would change it to:

```
au FileType python if &ft !~ 'django' | setlocal filetype=python.django_model | endif
au FileType html if &ft !~ 'django' | setlocal filetype=html.django_template | endif
```

Comment by Me — July 13, 2008 @ [3:14 am](#)

65. 

This was really helpful, thanks!

I had a couple of issues, possibly related to Windows, that I'll list in case it helps anyone:

The binding "inoremap " didn't have an effect when I used CTRL+Space, but changing to did the trick.

Using pW from pydoc.vim works with your example of

```
print os.path.abspath("~/_vimrc")
```

but fails on

```
print os.path.abspath('~/.vimrc')
```

I managed to figure out that this was because of single/double quoting issues when the contents are passed to the PyDoc function. I'm new to vim and after banging my head for a while, I could only come up with the following workaround:

```
map pWW :call ShowPyDoc("", 1)
```

Now I use pWW when my "WORD" has single quotes. Not great, but it works. 😊

Thanks again.

Comment by Ali — July 16, 2008 @ [6:47 am](#)

66. 

Oops. The comments ate up my angle brackets. The remap is from Nul to C-space.

You need a preview feature. 😞

Comment by Ali — July 16, 2008 @ [6:48 am](#)

67. 

Also, for pydoc

```
map pWW :call ShowPyDoc('[C-R][C-A]', 1)[CR]
```

with square brackets replaced by angle brackets. (test: <foo>)

Comment by Ali — July 16, 2008 @ [6:50 am](#)

68. 

Thanks! I use this every day! Perhaps someone will want to make a setup script or install package for this?

Comment by [Ivan Ven Osdel](#) — July 21, 2008 @ [6:34 am](#)

69. 

vim ide per python...

Aquesta setmana i gràcies a l'entrada del blog de sontek he retornat al vi com a editor principal per a la programació en Python.

Periòdicament estic canviant entre vim, kate o Eclipse amb PyDev, se...

Trackback by trespams.com — July 22, 2008 @ [11:40 am](#)

70. 

Great stuff, just what I've been looking for. Thanks!

Comment by [Abesto](#) — July 24, 2008 @ [6:52 am](#)

71. 

[...] found a way of doing this for Python here and it shows how to map to a better shortcut (the same as is used in Visual Studio in fact. [...]

Pingback by [Newspeak » Blog Archive » Week 2: Vim tricks - Automatic code completion](#) — July 29, 2008 @ [9:21 am](#)

[RSS feed for comments on this post.](#) [TrackBack URL](#)

Leave a comment

Name (required)

Mail (will not be published) (required)

Website

Submit Comment



- **Social Links**

- [Linked In](#)
- [Ohloh](#)
- [Facebook](#)
- [Flickr](#)
- [Last.fm](#)
- [My Space](#)

- **Dot Files**

- [Full List](#)
- [.screenrc](#)
- [.bashrc](#)
- [.vimrc](#)
- [.vim directory](#)
- Pages:
 - [About Sontek](#)
- Blogroll
 - [Christer Edwards](#)
 - [Clint Savage](#)
 - [Utah Planet](#)
- Categories:
 - [.NET](#)
 - [Bash](#)
 - [C#](#)
 - [Databases](#)
 - [Fedora](#)
 - [Firefox](#)
 - [GNOME](#)
 - [irssi](#)
 - [Linux](#)
 - [Mono](#)
 - [openWRT](#)
 - [Perl](#)
 - [Postgresql](#)
 - [Programming](#)
 - [python](#)
 - [RPM](#)
 - [SUSE](#)
 - [Tomboy](#)
 - [Twitter](#)
 - [Uncategorized](#)
 - [vim](#)
 - [Windows](#)
 - [Xorg](#)
- Search:

Search
- Archives:
 - [July 2008](#)
 - [June 2008](#)
 - [May 2008](#)
 - [April 2008](#)
 - [March 2008](#)
 - [February 2008](#)
 - [January 2008](#)
 - [December 2007](#)
 - [November 2007](#)
 - [October 2007](#)
 - [September 2007](#)
- Meta:
 - [Log in](#)
 - [Comments RSS](#)
 - [Valid XHTML](#)

Ads by Google

Python Developer

Python programming, websites and consulting.
www.sharp-ideas.net

Powerful Python Editor

Edit, create, and navigate Python code easily. Free Download.
www.edtrock.com

Python boots

200,000+ Shoes. Search Visually. Find python shoes
Like.com/python_shoes

Free Continuous Builds

Get Mojo - a free Eclipse RCP based continuous integration build server
www.OpenMakeSoftware.com

Powered by **WordPress**