Stani's Python Editor

Python IDE with Blender, Kiki, PyChecker, wxGlade & XRC support

User Manual

01-07-2005

Table of Contents

1 Introduction

1.1 About

Stani's Python Editor

SPE is a cross-platform python IDE with auto indentation, auto completion, call tips, syntax coloring, syntax highlighting, uml viewer, class explorer, source index, automatic todo list, sticky notes, integrated pycrust shell, python file browser, recent file browser, drag&drop, context help, ... Special is its Blender support with a Blender 3d object browser and its ability to run interactively inside Blender.

SPE runs on Windows, Linux and Mac OS X.

SPE is extensible with wxGlade.

1.2 Plugins

SPE ships with

- wxGlade (gui designer)
- PyChecker (source code doctor)
- Kiki (regular expression console)

SPE also integrates with

- XRCed (gui designer)

1.3 Internet links

- Homepage: http://SPE.pycs.net
- Website: http://projects.blender.org/projects/SPE
- Screenshots: http://SPE.pycs.net/pictures/index.html
- Forum: http://projects.blender.org/forum/?groupid=30
- RSS feed: http://SPE.pycs.net/weblog/rss.xml

1.4 Copyright

1.5 License

SPE is released under the GPL. If you need SPE under another license, contact the author.

2 Installation

2.1 Requirements

- Python 2.3+
  We recommend ActivePython distribution because of its excellent help files:
  http://www.activestate.com/Products/ActivePython/index.html
- wxPython 2.6+
  SPE follows the wxPython releases. Always use the latest wxPython release.

➢ Optional:

- Blender 2.37
  Cross-platform 3D software solution from modeling, animation, rendering and post-production to interactive creation and playback.
- Win32 extensions (Windows only)
  This module is needed to create shortcuts on Desktop and Start Menu during installation. This module is standard included in ActivePython

2.2 Windows

Installation

SPE should be installed with the windows binary. It will probably install SPE in C:\PythonXX\lib\site-packages. Afterwards, you can register 'Edit with SPE' in the context menu of Windows and create shortcuts, by running the script SPE/winInstall.py

.. image: images/SPEsetupwin.png

Removal

If you registered SPE in windows explorer context menu or created shortcuts, you have to remove them first with the script SPE/winUninstall.py

Afterwards, go to Control Panel > Add/Remove Programs and uninstall SPE (probably listed with Python).

Associate SPE with .py

SPE registers itself automatically during setup in the window explorer context menu for '.py' and '.pyw' files. This adds 'Edit with SPE' as a menu item in the context pop-up of the Windows Explorer. If this would fail, you can always do it manually...

- In the Windows Explorer choose Tools->Folder Options
- On the Dialog notebook, click File Types page.
- Select PY extension in file types list.
- Click the Advanced button to create a new action.
- Action:
  - Edit with SPE
  - Application used to perform action: absolute\path\to\python.exe absolute\path\to\SPE.py "%1"
- OK all the dialogs

2.3 Linux, FreeBSD, Mac OS X, ...

Installation

Run the 'setup.py' script:
```
>python setup.py install
```

If you have any problems with permissions:
```
>sudo python setup.py install
```

This will install SPE in the standard library directory of python:
/usr/local/lib/pythonX.X/site-packages

A wrapper script called 'SPE' will be installed to PREFIX/bin. If necessary add PREFIX/bin to your PATH environment variable. PREFIX is determined by the install location of the modules, i.e. for the above PREFIX=/usr/local.

When SPE is launched in Blender, what might be missing in the PYTHONPATH, is /usr/local/lib/python2.2/site-packages. If you add this one in your .bashrc/.tcshrc/... to the PYTHONPATH variable everything should be fine (the subdirs SPE,sm,etc. aren't needed). Though you must start Blender from a bash – e.g. desktop menus usually don't read the .bashrc/.tcshrc/... and therefore Blender does not know about your user defined environment variables. If you set the PYTHONPATH in /etc/profile instead of .bashrc/.tcshrc/... then starting SPE/SPE from Blender will work also from menus.

Removal

Just run the SPE/unixUninstall.py script (use it at your own risk!) or by removing manually the directories SPE, and sm of the standard library directory:

/usr/local/lib/pythonX.X/site-packages

One needs also to remove /usr/local/bin/SPE manually.

3 Getting Started

3.1 Startup

Normal mode

Windows

Open the SPE folder and type 'python SPE.py' at the command prompt or make a shortcut to your desktop.

Linux, FreeBSD, Mac OS X, ...

Type 'SPE' on the command line (assuming PREFIX/bin is on your PATH)

Debugging mode

If you have problems starting up SPE, type at the command prompt:

```
>python SPE.py –debug
```

and send me the error message.

Blender mode

Open SPE.blend and press Alt+P in the corresponding text window.

When SPE is active, the Blender screen will always be redrawn automatically. So the results of any command you type in the interactive shell or of any program you run within SPE, will be visible in the Blender window. Unfortunately it is not possible to interact with Blender directly when SPE is active. So it is impossible to rotate for example the view with the mouse.

3.2 Syntax-checking

Every time you save SPE does syntax checking. If there is any error, SPE will jump to the line in the source code and try to highlight the error.

3.3 Refreshing

SPE has a lot of features like explore tree, index, todo list, and so on... This gets updated every time the file is saved or every time the refresh command is given. This can be done by pressing F5 on the keyboard, the refresh toolbar button or clicking the View>Refresh menu.

3.4 Running files

➔ Warning: SPE doesn't require you to save your files before running. However it is recommended to do so not to loose source code if your program makes SPE hang.

SPE provides many ways to run files:

Run (F9)

Use this by default, unless you have specific reasons to use the other ones. It will run in the namespace of the interactive shell. So all the objects and functions of your program become available in the shell and in the locals browser (the tab next to the shell).

Run with profile (Ctrl-P)

Same as above but with a profile added. A profile is a report of the program execution which shows which processes or functions are time consuming. So if you want to speed up your code, you can define the priorities based on this report.

Run in separate namespace (Ctrl-R):

Like run, but all the objects and functions defined by the program will not become available in the namespace of the interactive shell. Instead they will be defined in the dictionary 'namespace' of the interactive shell. So if the file 'script.py' is run in this way, type namespace['script.py'] in the shell, to access this dictionary, or namespace['script.py'].keys() to get a list of all defined names, or namespace['script.py'].items() to get tuples of all the names and their values. More easy is to just browse 'namespace' in the locals browser.

Run verbose (Alt-R)

This is for very simple programs, which do not indent more than once. It will send all source lines, as if they were typed in the interactive shell. It is probably a good learning tool for beginners.

Import (F10)

Imports the source file as a module. For running files, they don't have to be saved. For importing files, it is recommended to save them first.

3.5 Separators

A separator is a label which appears in the explore tree of the sidebar to help structuring the script. An easy way to add separators is to use the 'Edit'>'Insert colored separator'wizard from the menu.

Syntax:

- normal: ``#---label``
- colored: ``#---label---#foreground color#background color``
- highlighted: ``# # # #label (4 times #, without spaces)``

Foreground and background color are in html notation, eg.:

- red on blue label: #---red on blue---#FF0000#0000FF

.. image: images/separatordialog.png

3.6 Remember option

This can be activated by checking File>Remember or by pressing the heart toolbar button. It will open automatically the scripts which were open in the last session. Useful for Blender if you have to switch continously between Blender and SPE.

3.7 Psyco

If you don't know the python psyco module, you can ignore this item, as it won't have any effect for you. Psyco programs can't run in SPE, as they disable the 'locals()' function. Of course you can edit programs using psyco in SPE, but if you want to run them, comment the psyco activation code out.

3.8 Customize

Keyboard shortcuts

If you want to change the default keyboard shortcuts, open the file SPE/framework/shortcuts.py and adapt it to your own taste. Backup this file so that when you install a new version of SPE, you can copy it back.

Menus and toolbar

You can define your own menus and toolbar buttons, which can execute any python code and also external files. Look at 'framework/menus/Extra.py' for an example.

Instructions:

1. Suppose you want to add a new menu with the name 'XXX' to the menubar. Create a new file with the name XXX.py in the 'framework/menus/' directory

2. Import or define some actions, with the following structure:

```
def action(script,app,event):
      ...
```

The arguments of the function are:

- `script`: current script window
- `app`: application window
- `event`: event

Some usefull stuff:

- `script.fileName`
- `script.source`
    - `script.source.GetText()`
    - `script.source.SetText(text)`
    - `script.source.GetSelectedText()`
    - `script.source.ReplaceSelection(text)`
- `app.run(fileName)`
  runs an external file
- `app.new()`
  creates a new file
- `app.open(fileName,lineno,col)`
  opens a file at given position
- `app.message(text)`
  shows a dialog window with the text
- `app.messageEntry(text)`
  shows a dialog prompting an entry
- `app.messageError(text)`
  shows an error dialog window with the text
- `app.SetStatus(text)`
  sets the status text

3. Define the 'main' function:

```
def main(app):
      menu(app,
            item(label=<str:label that will appear in menu>,
                  action=<function:that will be called>),
            item(...),
            SEPARATOR,
            item(...),
            item(...),
            ...
            SEPARATOR,
            item(...),
            ...)
```

If you want this menu item also to have a toolbar button, than make a 16x16pixels png image (transparency is allowed). The image has to be located in the menu folder. Pass the the fileName with the toolbar keyword:

```
item(label=<str:label that will appear in menu>,
      action=<function:that will be called>,
      toolbar=<str:fileName of the toolbar image (optional)>)
```

4 Features

4.1 Sidebar

- Class browser
- File browser
- Automatic todo list, highlighting the most important ones
- Automatic alphabetic source index of classes and methods
- Sticky notes

4.2 Tools

These tools appear as tabs down.

Shell

Interactive PyCrust shell

- Double mouse click to jump to error source code

Locals

Local object browser

- Left mouse click to open
- Right mouse click to run

Session

Separate session recorder

Find

Find recursively text in files

- Leave the 'Path' field empty to search in all open files

Browser

Quick access to python files in specified folders and their sub folders

- Left mouse click to open

Recent

Unlimited recent file list

- Left mouse click to open
- Right mouse click to run

Todo

Automatic todo list of all open files, highlighting the most important ones (jump to source)

- Left mouse click to jump to source

Index

Automatic alphabetic index of all open files (jump to source)

Notes

Sticky note for general development comments

Blender

Blender object browser. It is only working when SPE is launched in Blender mode.

Donate

If you like SPE, please consider to give a donation

4.3 Editor

As you type:

- Syntax-coloring
- Auto-indentation
- Auto-completion
- Call-tips

When you save a file:

- Syntax-checking

Uml view

- Graphical layout of class hierarchy

Special keyboard shortcuts

- Ctrl+Enter: browse source of module

4.4 Drag&Drop

Drag&drop any amount of files or folders on ...

- main frame to open them
- shell to run them
- recent files to add them
- browser to add folders

4.5 General

- Context help defined everywhere
- Add your own menus and toolbar buttons
- Exit & remember:all open files will next time automatically be loaded
    - handy for Blender sessions
    - heart icon on toolbar
- Scripts can be executed in different ways: run, run verbose and import

4.6 Blender

- Redraw the Blender screen on idle (no blackout)
- Blender object tree browser (cameras,objects,lamps,...)
- Add your favorite scripts to the menu
- 100% Blender compatible: can run within Blender, so all previous described features are available within Blender

4.7 Windows

- SPE registers itself in the windows explorer context menu
- optional creation of desktop and quick launch shortcuts

5 Tutorial

5.1 Introduction

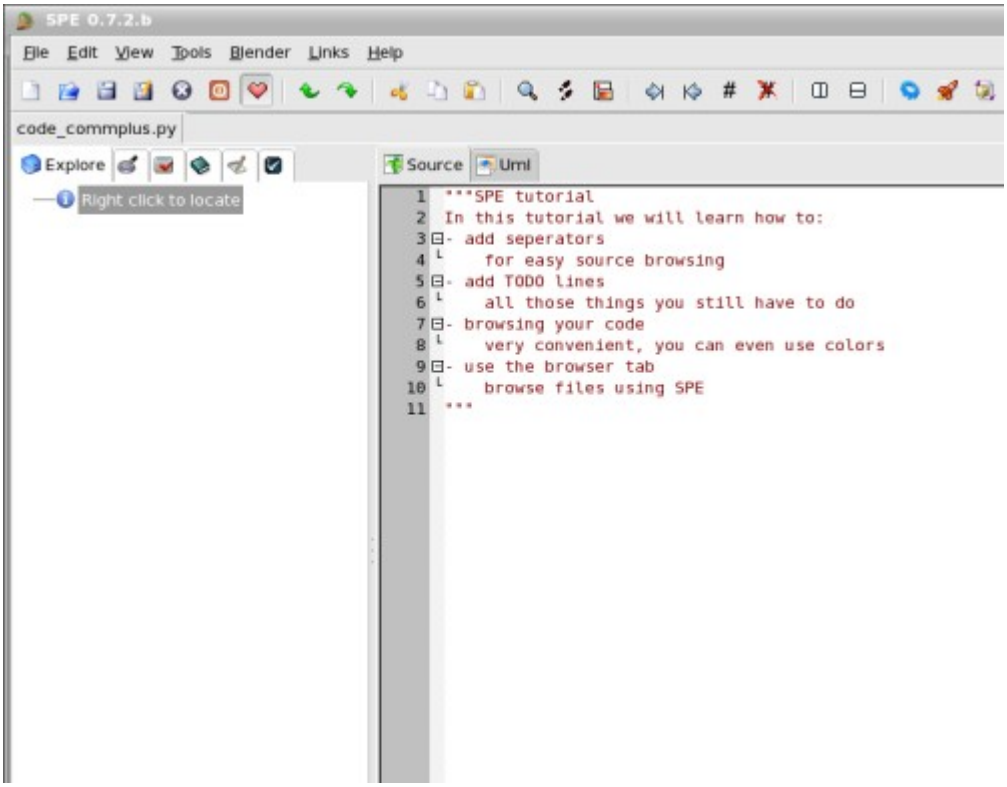
This tutorial was written by Dimitri from www.serpia.com Please visit his website for more Python tutorials.

The tutorial will focus primarily on the functionality of SPE's sidebar. SPE allows you, amongst many other things like syntax coloring, to create seperators which makes it very easy to keep your code structured. This results in a clear and fast way to maintain your code more easily. So I won't tell you anything about SPE's blender support, if there is anyone who wants to write a tutorial on that subject (or any other subject regarding SPE), I'd be more than happy to add it to this webpage.

5.2 The comments

Let's start SPE and study the screen. SPE's main window is roughly divided into four parts. The upper part is where the filemenu and toolbar resides, beneath it you will see two vertical windows, the left one is the sidebar and on the right you'll see the editor itself. On the bottom of the main window is the Python shell and clicking on one of the tabs will give you another view of you code. Some of these tabs are common to the one in the sidebar, but here you will find some extra functionality like a search function and an interface to Blender. As I said earlier, we will primarily focus on the sidebar.

First, add some comments for your source (something you should always do) starting on the first line of the editor. Something like depicted below:



code:

```
"""SPE tutorial
In this tutorial we will learn how to:
- add seperators
    for easy source browsing
- add TODO lines
    all those things you still have to do
- browsing your code
    very convenient, you can even use colors
- use the browser tab
    browse files using SPE
"""
```

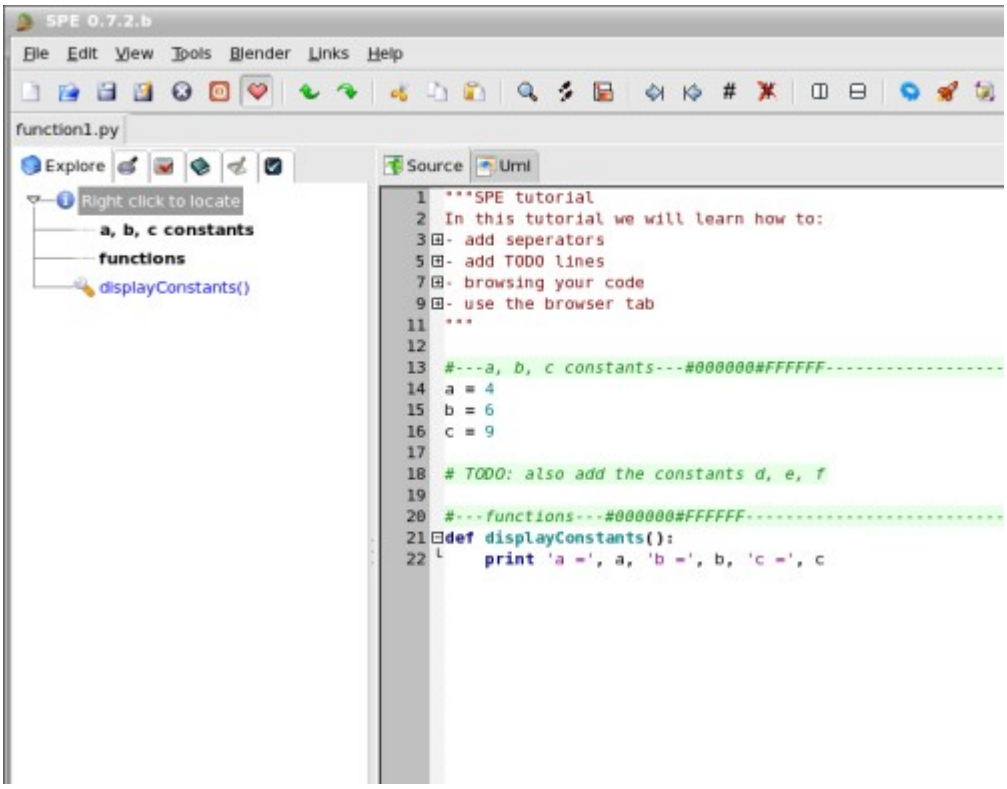
It is very convenient that the text can be placed inside a tree hierarchy, you can expand the text using '+' and vice versa. To give a line a lower hierarchy, press the <tab> key.

5.3 Adding a separator and the TODO tag

Adding a separator is a convenient way to structure your code, thus simpler to maintain. This is not only important for large files, you will find that it is also a great feature for smaller files (small files tend to grow bigger). You can add a separator using different methods: just type "#---some text" on an empty line in the editor, select it from the Edit menu or use the Alt+i shortcut. The newly created separator will appear in the explore tab of the sidebars. Rightclicking on the reference to the separator in the explore tab will locate it in your code. This allows you to quickly find chunks of code in your source without having to scroll up and down and staring at the screen. This is, from a view of usability, something that can actually increase your productivity.

Another handy feature is the auto creation of a todo list. Just add '# TODO:sometext' to your code and the todo tab of the sidebar will store the text following the '#TODO:' tag. A very easy way to keep track of the inevitable todo's! But of course, you can store all sorts of other information here for future references (e.g. 'this code fragment is from Harry's webtutorial'). A cool feature of the todo tag is that you can determine its priority by the amount of exclamation marks ("!"). The one with the most exclamation marks will be highlighted. As this you don't have to think any more about the order in which you insert your todo's. There is a special tab dedicated to the todo's on the sidebar, here you can see the priorities of your todo's.

In the next picture you can see that I also added a function definition, this will also appear in the sidebar. The reference for this function in the sidebar uses a blue font and you can use it to jump to the location in your source code. SPE's author was smart enough to add an icon also, human beings are visually orientated and icons work very well in this regard.



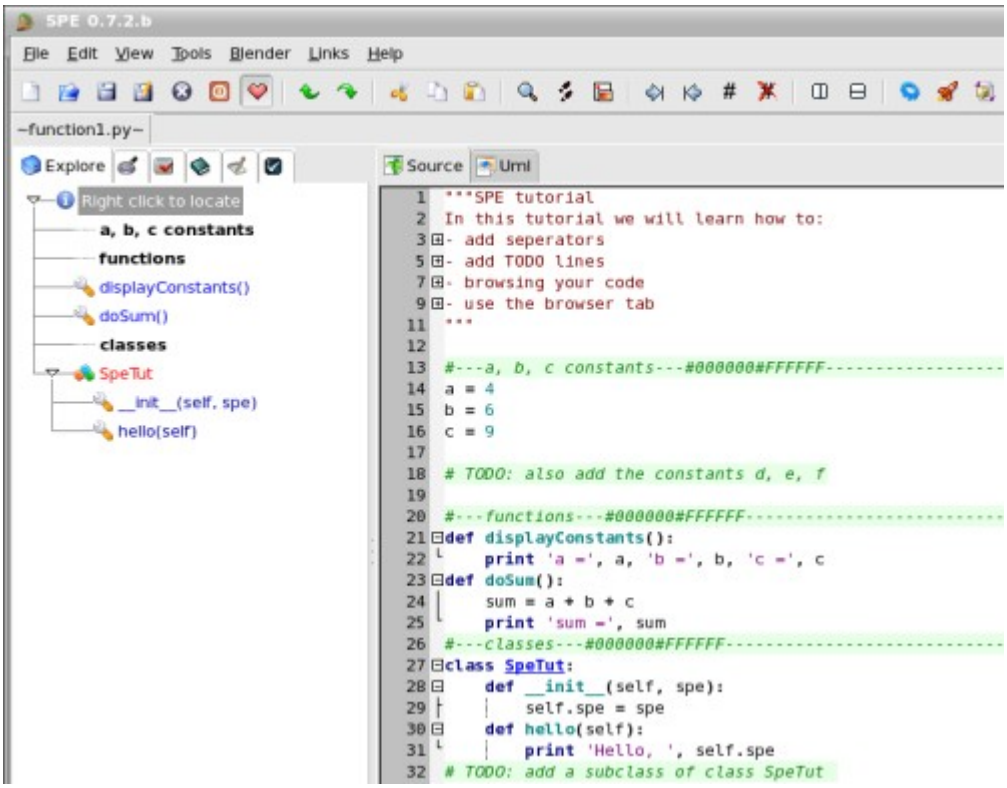
code: function1

```
"""SPE tutorial
In this tutorial we will learn how to:
- add seperators
    for easy source browsing
- add TODO lines
    all those things you still have to do
- browsing your code
    very convenient, you can even use colors
- use the browser tab
    browse files using SPE
"""

#---a, b, c constants---#000000#FFFFFF-----------------------------------
a = 4
b = 6
c = 9

# TODO: also add the constants d, e, f

#---functions---#000000#FFFFFF-------------------------------------------
def displayConstants():
    print 'a =', a, 'b =', b, 'c =', c
```

5.4 Browsing a class

The next thing we'll do is to add a class to our program, but first let's create another separator named "classes". By doing so, it will be easier to identify the location of your classes. Beneath this separator we will create a class named *SpeTut*, this class contains two methods, *__init__* (aka the constructor) and the method *hello*. As you can see in the picture below, the class browser *SpeTut* will be visible in the explore tab, including the aforementioned methods. Use the little triangle on the leftside to expand the tree and vice versa. The reference to the class in the explore tab has a red font and a unique icon (click on it to expand the tree!). You can also add a separator inside a class, a nice feature for larger classes with many methods.

One note though, after you have added a separator, a function or the like you have to refresh the tree by returning it to the highest hierarchy using the triangle.



code: 2func_class

```
"""SPE tutorial
In this tutorial we will learn how to:
- add seperators
    for easy source browsing
- add TODO lines
    all those things you still have to do
- browsing your code
    very convenient, you can even use colors
- use the browser tab
    browse files using SPE
"""

#---a, b, c constants---
#000000#FFFFFF-----------------------------------------
a = 4
b = 6
c = 9

# TODO: also add the constants d, e, f

#---functions---
#000000#FFFFFF-----------------------------------------------------
def displayConstants():
    print 'a =', a, 'b =', b, 'c =', c
def doSum():
    sum = a + b + c
    print 'sum =', sum
#---classes---
#000000#FFFFFF-----------------------------------------------------
class SpeTut:
    def __init__(self, spe):
        self.spe = spe
    def hello(self):
        print 'Hello, ', self.spe
# TODO: add a subclass of class SpeTut
```

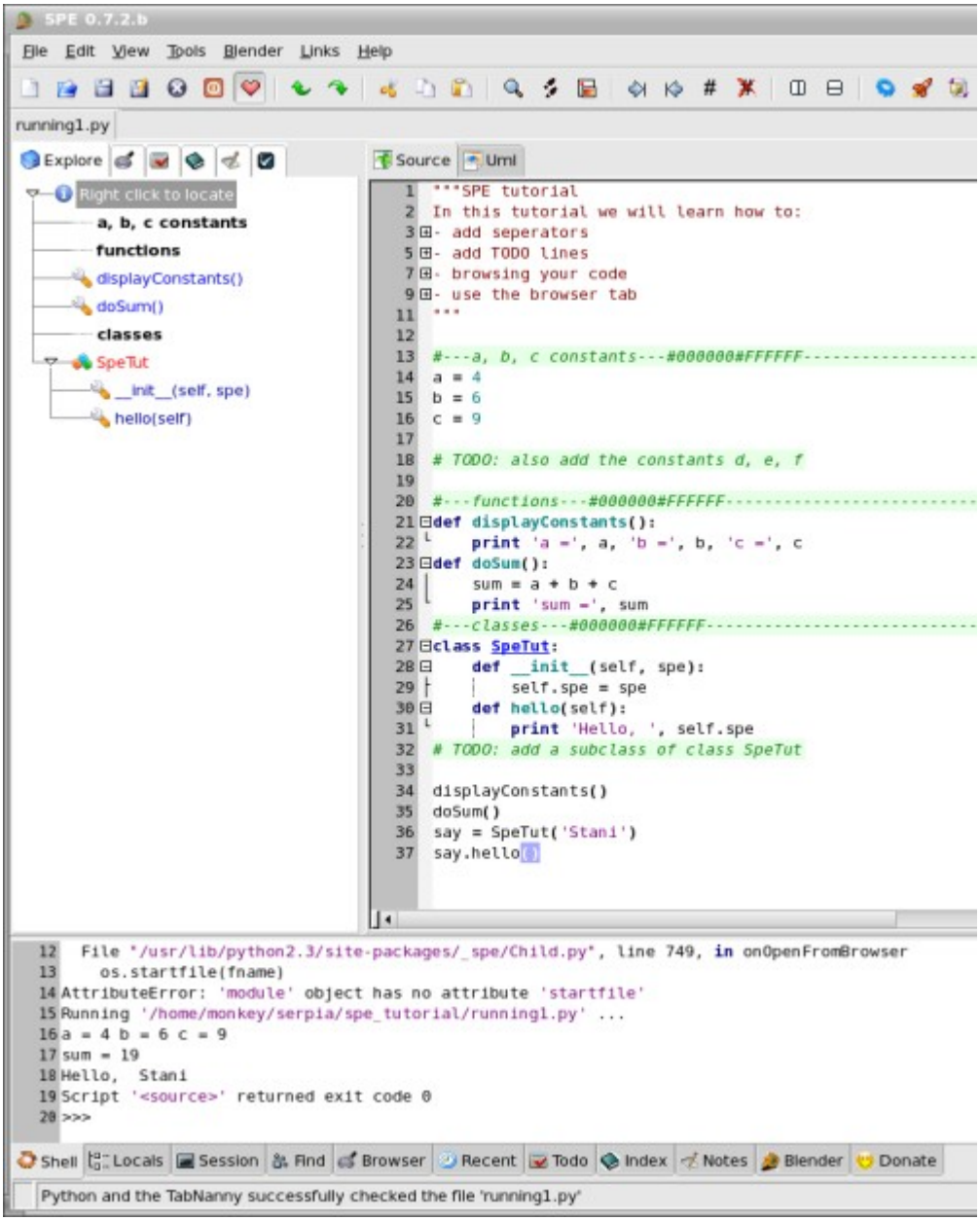
5.5 Run that py!

Now it's time to run our little program, so add:

- displayConstant() [this runs the function definition]
- doSum() [this runs the function definition]
- say = SpeTut('Stani') [this creates an *instance* with an *argument*]
- say.hello() [this runs the method from the class]

to the source code

Use F9 or use the icon on the toolbar and the code will be executed. You can see the output in the Python shell in the lower area of SPE's main window. It's a good practice to do this often as you're building your code. Another nice feature of the sidebar is the source code checker (PyChecker). You can use it by clicking on the appropriate tab located in the sidebar.

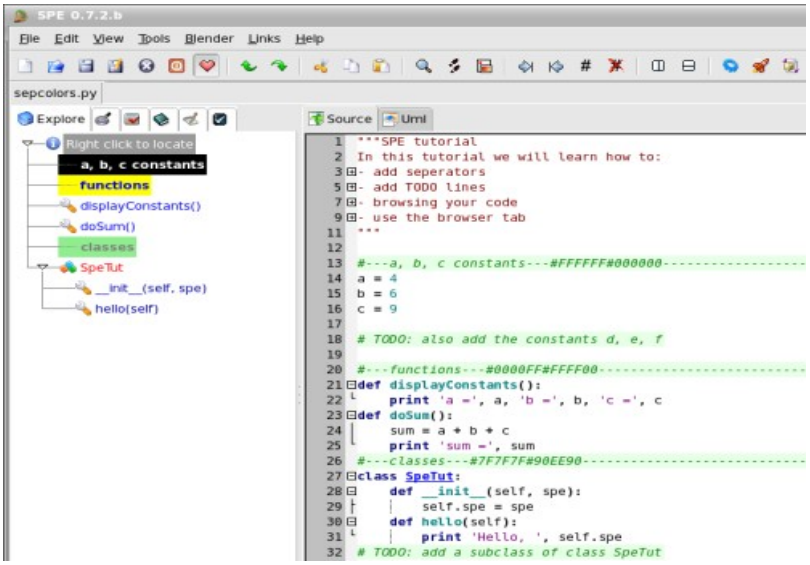

SPE 0.7.2.b
File Edit View Tools Blender Links Help
running1.py
Explore
Right click to locate
a, b, c constants
functions
displayConstants()
doSum()
classes
SpeTut
__init__(self, spe)
hello(self)
Source Uml
1 """SPE tutorial
2 In this tutorial we will learn how to:
3 - add seperators
5 - add TODO lines
7 - browsing your code
9 - use the browser tab
11 """
12
13 #---a, b, c constants---#000000#FFFFFF----------
14 a = 4
15 b = 6
16 c = 9
17
18 # TODO: also add the constants d, e, f
19
20 #---functions---#000000#FFFFFF----------
21 def displayConstants():
22 print 'a =', a, 'b =', b, 'c =', c
23 def doSum():
24 sum = a + b + c
25 print 'sum =', sum
26 #---classes---#000000#FFFFFF----------
27 class SpeTut:
28 def __init__(self, spe):
29 self.spe = spe
30 def hello(self):
31 print 'Hello, ', self.spe
32 # TODO: add a subclass of class SpeTut
33
34 displayConstants()
35 doSum()
36 say = SpeTut('Stani')
37 say.hello()
12 File "/usr/lib/python2.3/site-packages/_spe/Child.py", line 749, in onOpenFromBrowser
13 os.startfile(fname)
14 AttributeError: 'module' object has no attribute 'startfile'
15 Running '/home/monkey/serpia/spe_tutorial/running1.py' ...
16 a = 4 b = 6 c = 9
17 sum = 19
18 Hello, Stani
19 Script '<source>' returned exit code 0
20 >>>
Shell Locals Session Find Browser Recent Todo Index Notes Blender Donate
Python and the TabNanny successfully checked the file 'running1.py'

code: running1

```
"""SPE tutorial
In this tutorial we will learn how to:
- add seperators
    for easy source browsing
- add TODO lines
    all those things you still have to do
- browsing your code
    very convenient, you can even use colors
- use the browser tab
    browse files using SPE
"""

#---a, b, c constants---
#000000#FFFFFF----------------------------------------
a = 4
b = 6
c = 9

# TODO: also add the constants d, e, f

#---functions---
#000000#FFFFFF----------------------------------------------------
def displayConstants():
    print 'a =', a, 'b =', b, 'c =', c
def doSum():
    sum = a + b + c
    print 'sum =', sum
#---classes---
#000000#FFFFFF----------------------------------------------------
class SpeTut:
    def __init__(self, spe):
        self.spe = spe
    def hello(self):
        print 'Hello, ', self.spe
# TODO: add a subclass of class SpeTut

displayConstants()
doSum()
say = SpeTut('Stani')
say.hello()
```

5.6 Life is full of colors

Adding colors to the separator makes it even easier to keep track of your code (as long as you don't turn it into a Christmas tree...). There are two ways to add colors to the separator, a convenient way is to use filemenu --> edit, another way is to type the colorcode (Hex, e.g. #7F7F7F). Use whatever suits you best.

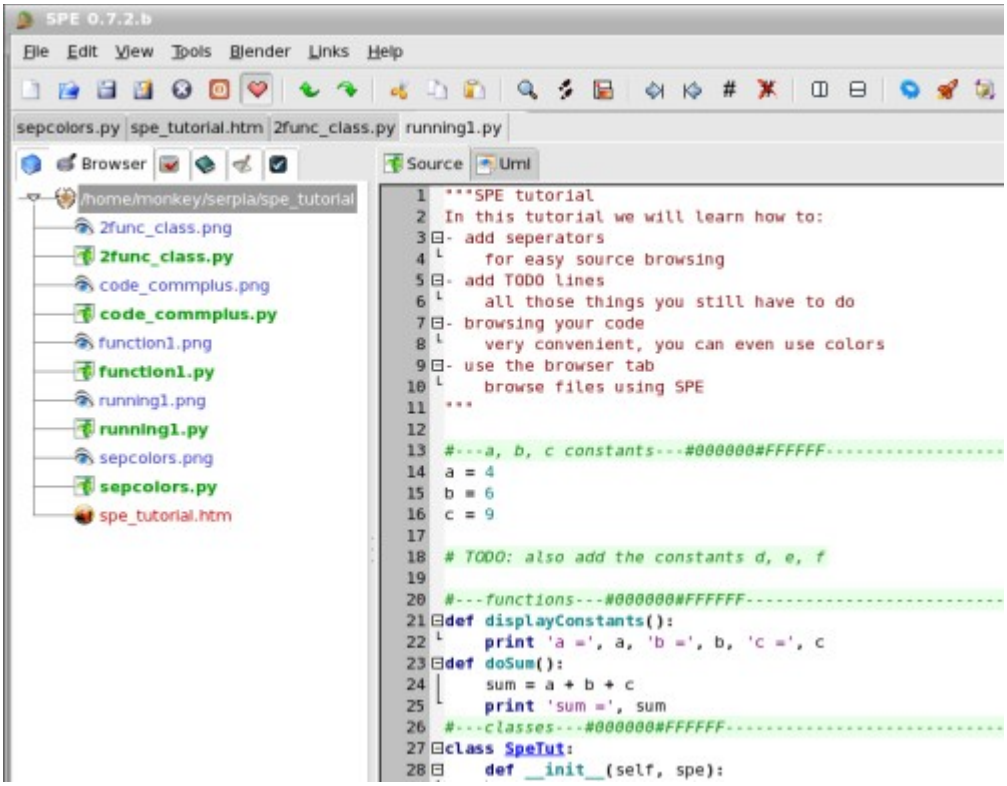


code: sepcolors

```
"""SPE tutorial
In this tutorial we will learn how to:
- add seperators
    for easy source browsing
- add TODO lines
    all those things you still have to do
- browsing your code
    very convenient, you can even use colors
- use the browser tab
    browse files using SPE
"""

#---a, b, c constants---#FFFFFF#000000-------------------------------------
a = 4
b = 6
c = 9

# TODO: also add the constants d, e, f

#---functions---#0000FF#FFFF00---------------------------------------------
def displayConstants():
    print 'a =', a, 'b =', b, 'c =', c
def doSum():
    sum = a + b + c
    print 'sum =', sum
#---classes---#7F7F7F#90EE90-----------------------------------------------
class SpeTut:
    def __init__(self, spe):
        self.spe = spe
    def hello(self):
        print 'Hello, ', self.spe
# TODO: add a subclass of class SpeTut

displayConstants()
doSum()
say = SpeTut('Stani')
say.hello()
```

5.7 Browsing your files

Working on a project often means that you have a lot of files that you have keep track of. A recent feature makes it quite easy to do this. Just click on the browser tab and the files of your current directory will be displayed, right-click on a file and it is opened in SPE and you can edit the file. This works for Python files (yeah, right...), but you can also edit html files.



One last feature I will mention in this tutorial is the ability to create sticky notes. Just click on the notes tab located in the sidebar and type some notes about your program. Making notes about your program is more important then you might think, an idea you have today may be forgotten the next day (or the next hour), it's just a small effort to make notes and SPE makes this very easy for you. The notes will be saved as an external '.txt' file and has the same name as your file. Another simple but effective way to keep track of your coding. Once you make this empty, the external file will also disappear.

5.8 The end

Here is where my little tutorial ends (the second version anyway) and I just barely scratched the surface of SPE's functionality. If you are looking for a free Python IDE, you owe it to yourself to try SPE and I think you won't be disappointed. Okay, it lacks a full blown debugger that some *commercial* IDE's have, but the dynamic nature of Python makes it quite easy to do your "own" debugging. Oh, did I tell you that SPE includes wxGlade? You can find a tutorial on wxGlade here!

If you have found any errors or want some extra stuff explained in this tutorial (this is the second version after all), please contact me.

6 Contact

6.1 Contribute

If you would like to contribute to SPE in any way, send me an email with your skills

- programming
- graphics
- icons
- 3d
- html

We are sure you can help us.

6.2 Feedback

SPE is still under development. If you use SPE, please post a message on the appropiate forum on http://projects.blender.org/forum/?groupid=30 describing the platform, the problems that occur and possible solutions if you know.

If SPE runs without any problems, I'm also interested to get a notice.

We develop SPE under Windows XP and have no access to Linux, Mac, FreeBsd or any other platform. So any help for these platforms is highly appreciated.

6.3 Contact persons

These people are contact persons for (replace $ with @):

- CVS and bugfixes: Sam Widmer (rigel$asylumwear.com)
- Documentation: Jelle Feringa (jelle.feringa$ezct.net)
- Mac OS X: Kevin Walzer (sw$wordtech-software.com)
- RPM Linux: Jason Powell (jtpowell$hotmail.com)

7 Donate

7.1 Why

Please donate if you enjoy using SPE and would like to help support it. It will definitely help. Any donation starting from 5 euro/dollar is welcome. If you know any fund which would be helpful, please let me know. Large donations can be rewarded with a link on the SPE website or name mentioning in SPE documentation.

7.2 Methods

Bank Transfer

We strongly recommended this payment for europe as no payment fees are involved. The Dutch Rabobank accepts international transfers. Using the IBAN number, this transaction is free of charge within Europe. So what you donate, is what SPE gets.

at the name of:

S. Michiels

Amsterdam

the Netherlands

Bank: Rabobank

IBAN: NL12 RABO 0393 8648 47 (for euro countries)

Swift/BIC code: RABONL2U (international code)

Account number: 3938.64.847

PayPal (Credit Card)

Visa, Mastercard and American Express are only accepted via the PayPal system.

You pay through the PayPal site (http://www.paypal.com) to s_t_a_n_i$yahoo.com (replace '$' with '@').

8 Keyboard shortcuts

Key	Action	Description
ALT '3'		Comment
ALT '4'		Uncomment
ALT 'D'	DEDENT	Dedent the lines
ALT 'I'		Insert separator
ALT BACK	UNDO	Undo one action in the undo history
ALT END	LINEENDDISPLAY	Move caret to last position on display line
ALT F4		Exit
ALT F9		Open terminal emulator
ALT HOME	HOMEDISPLAY	Move caret to first position on display line
ALT LEFT ARROW	WORDPARTLEFT	Move to the previous change in capitalisation
ALT RIGHT ARROW	WORDPARTRIGHT	Move to the next change in capitalisation
ALT+SHIFT END	LINEENDDISPLAYEXTEND	Move caret to last position on display line extending selection to new caret position
ALT+SHIFT HOME	HOMEDISPLAYEXTEND	Move caret to first position on display line extending selection to new caret position.
ALT+SHIFT LEFT ARROW	WORDPARTLEFTEXTEND	Move to the previous change in capitalisation extending selection to new caret position
ALT+SHIFT RIGHT ARROW	WORDPARTRIGHTEXTEND	Move to the next change in capitalisation extending selection to new caret position.
BACK	DELETEBACK	Dedent the selected lines
CTRL 'A'	SELECTALL	Select all the text in the document
CTRL 'B'		Load in Blender
CTRL 'C'	COPY	Copy the selection to the clipboard
CTRL 'F'		Find & replace
CTRL 'G'		Go to line
CTRL 'K'		Test regular expression with Kiki
CTRL 'L'	LINECUT	Cut the line containing the caret
CTRL 'N'		New
CTRL 'O'		Open
CTRL 'P'		Run with profile
CTRL 'R'		Run in separate namespace

Key	Action	Description
CTRL 'S'		Save
CTRL 'T'	LINETRANSPOSE	Switch the current line with the previous
CTRL 'U'	LOWERCASE	Transform the selection to lower case
CTRL 'V'	PASTE	Paste the contents of the clipboard into the document replacing the selection
CTRL 'X'	CUT	Cut the selection to the clipboard
CTRL 'Y'	REDO	Redoes the next action on the undo history
CTRL 'Z'	UNDO	Undo one action in the undo history
CTRL @		Contact author
CTRL ADD	ZOOMIN	Magnify the displayed text by increasing the sizes by 1 point
CTRL BACK	DELWORDLEFT	Delete the word to the left of the caret
CTRL DELETE	DELWORDRIGHT	Delete the word to the right of the caret
CTRL DIVIDE	SETZOOM	Set the zoom level to 0. This returns the zoom to 'normal,' i.e., no zoom.
CTRL DOWN ARROW	LINESCROLLDOWN	Scroll the document down, keeping the caret visible
CTRL END	DOCUMENTEND	Move caret to last position in document
CTRL ENTER		Browse source
CTRL F4		Close
CTRL F9		Run in terminal emulator
CTRL HOME	DOCUMENTSTART	Move caret to first position in document
CTRL INSERT	COPY	Copy the selection to the clipboard
CTRL LEFT ARROW	WORDLEFT	Move caret left one word
CTRL RIGHT ARROW	WORDRIGHT	Move caret right one word
CTRL SPACE		Auto complete
CTRL SUBTRACT	ZOOMOUT	Make the displayed text smaller by decreasing the sizes by 1 point
CTRL UP ARROW	LINESCROLLUP	Scroll the document up, keeping the caret visible
CTRL+ALT 'B'		Reference in Blender
CTRL+ALT 'C'		Check source with PyChecker
CTRL+ALT 'F'		Browse Object with PyFilling
CTRL+ALT 'G'		Design a gui with wxGlade

Key	Action	Description
CTRL+ALT 'P'		Preferences
CTRL+ALT 'R'		Run Verbose
CTRL+ALT 'X'		Design a gui with XRC
CTRL+ALT F9		Run in terminal emulator & exit
CTRL+SHIFT 'L'	LINEDELETE	Delete the line containing the caret
CTRL+SHIFT 'U'	UPPERCASE	Transform the selection to upper case
CTRL+SHIFT BACK	DELLINELEFT	Delete back from the current position to the start of the line
CTRL+SHIFT DELETE	DELLINERIGHT	Delete forwards from the current position to the end of the line
CTRL+SHIFT END	DOCUMENTENDEXTEND	Move caret to last position in document extending selection to new caret position
CTRL+SHIFT HOME	DOCUMENTSTARTEXTEND	Move caret to first position in document extending selection to new caret position
CTRL+SHIFT LEFT ARROW	WORDLEFTEXTEND	Move caret left one word extending selection to new caret position
CTRL+SHIFT RIGHT ARROW	WORDRIGHTEXTEND	Move caret right one word extending selection to new caret position
DELETE	CLEAR	Delete all text in the document
DOWN ARROW	LINEDOWN	Move caret down one line
END	LINEEND	Move caret to last position on line
ESCAPE	CANCEL	Cancel any modes such as call tip or auto-completion list display
F02		Save
F03		Find next
F05		Refresh
F09		Run
F10		Import
F11		Show/hide sidebar
F12		Show/hide shell
HOME	VCHOME	Move caret to before first visible character on line. If already there move to first character on line
INSERT	EDITTOGGLEOVERTYPE	Switch from insert to overtype mode or the reverse
LEFT ARROW	CHARLEFT	Move caret left one character
NEXT	PAGEDOWN	Move caret one page down
PRIOR	PAGEUP	Move caret one page up
RETURN	NEWLINE	Insert a new line, may use a CRLF, CR or LF depending on EOL mode
RIGHT ARROW	CHARRIGHT	Move caret right one character
SHIFT BACK	BACKTAB	Delete the selection or if no selection, the character before the caret
SHIFT DELETE	CUT	Cut the selection to the clipboard
SHIFT DOWN ARROW	LINEDOWNEXTEND	Move caret down one line extending selection to new caret position
SHIFT END	LINEENDEXTEND	Move caret to last position on line extending selection to new caret position

Key	Action	Description
SHIFT F9		Browse folder
SHIFT HOME	VCHOMEEXTEND	Like VCHome but extending selection to new caret position
SHIFT INSERT	PASTE	Paste the contents of the clipboard into the document replacing the selection
SHIFT LEFT ARROW	CHARLEFTEXTEND	Move caret left one character extending selection to new caret position
SHIFT NEXT	SCI_PAGEDOWNEXTEND	Move caret one page down extending selection to new caret position
SHIFT PRIOR	PAGEUPEXTEND	Move caret one page up extending selection to new caret position
SHIFT RETURN	NEWLINE	Insert a new line, may use a CRLF, CR or LF depending on EOL mode
SHIFT RIGHT ARROW	CHARRIGHTEXTEND	Move caret right one character extending selection to new caret position
SHIFT UP ARROW	LINEUPEXTEND	Move caret up one line extending selection to new caret position
TAB	TAB	If selection is empty or all on one line replace the selection with a tab character. If more than one line selected, indent the lines.
UP ARROW	LINEUP	Move caret up one line

9 Credits

Thanks to the following components SPE was made possible:

- Blender
    - 3D modeling, rendering, animation and game creation package
    - Copyright 2003 Blender Foundation - Ton Roosendaal
    - http://www.blender.org
- Kiki
    - free environment for regular expression testing (ferret)
    - Copyright 2003 Project 5 - Andrei
    - http://come.to/project5
- PyChecker
    - a python source code checking tool
    - Copyright (c) 2000-2001, MetaSlash Inc.
    - http://pychecker.sourceforge.net
- PyCrust
    - The flakiest python shell (Patrick K. O'Brien)
    - Sponsored by Orbtech - Your source for python programming expertise.
    - http://www.wxPython.org
- Pycs
    - Python Community Server
    - Copyright 2002 pycs
    - http://www.pycs.net
- Pyds
    - python desktop server
    - Copyright 2002 Georg Bauer
    - http://pyds.muensterland.org
- Pyframe guide to Wxpython
    - Documentation about wxStyledTextCtrl
    - Copyright 2003 Jeff Sasmor
    - http://www.pyframe.com/wxdocs/
- Pythonwin
    - python IDE and GUI Framework for Windows
    - Copyright 1994-2003 Mark Hammond
- Scintilla
    - Copyright 1998-2001 by Neil Hodgson
    - http://www.scintilla.org
- Sky icons
    - KDE icon theme made with gimp
    - Copyright 2002 David Vignoni
    - http://projects.dims.org/%7Edave/iconsky5.html

- wxGlade
    - wxGlade is a GUI designer written in Python with the popular GUI toolkit wxPython, that helps you create wxWindows/wxPython user interfaces. At the moment it can generate Python, C++ and XRC (wxWindows' XML resources) code.
    - Copyright 2003 Alberto Griggio, Marco Barisione, Marcello Semboli, Richard Lawson, D.H.
    - http://wxglade.sourceforge.net
- wxPython
    - python extension module for wxWindows GUI classes
    - Copyright 1997-2003 Robin Dunn and Total Control Software
    - http://www.wxPython.org

Special thanks to Tina Hirsch (Linux feedback).

Python is Copyright (c) 2000-2002 ActiveState Corp:

> Copyright (c) 2001, 2002 Python Software Foundation.
> All Rights Reserved.
>
> Copyright (c) 2000 BeOpen.com.
> All Rights Reserved.
>
> Copyright (c) 1995-2001 Corporation for National Research Initiatives.
> All Rights Reserved.
>
> Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
> All Rights Reserved.

This program uses IDLE extensions by Guido van Rossum, Tim Peters and others.

Alphabetical Index