PySci Documentation

Release 0.1

Eric Pierce

Mar 22, 2017

Contents

This module attempts to provide a more Pythonic interface to the QsciScintilla editor widget. Although QsciScintilla is a rich and full-featured editor widget that works nicely with PyQt4, its API is very C-like, and feels too low-level in the context of a Python application. PySci was developed to bring a more idiomatic Python approach to QsciScintilla.

Until PySci is officially released, you can install it from GitHub like so:

```
$ git clone git://github.com/wapcaplet/pysci.git
$ cd pysci
$ pip install .
```

PySci uses the MIT License.

CHAPTER 1

Contents

Usage

Here's a very simple PyQt4 application using the PySci editor widget:

```
from PyQt4 import QtGui
from pysci import PySci

if __name__ == '__main__':
    app = QtGui.QApplication([])
    editor = PySci()
    editor.show()
    app.exec_()
```

The `PySci` widget is a subclass of QsciScintilla, so you can use it in all the same ways you would normally use that widget. For some of the additional features it provides, read on.

Configuration

QsciScintilla makes heavy use of enumerated types with individual getters and setters for configuration settings. `PySci` borrows a more Pythonic approach from Tkinter, with the following features:

- Arbitrary configuration settings are accepted by the widget constructor
- Multiple settings can be configured with a single method call
- Plain strings can be used instead of enumerated types

For example, with QSciScintilla alone you might need six lines of code to create an editor widget and configure it the way you want:

```
editor = QsciScintilla()
editor.setWhitespaceVisibility(QsciScintilla.WsInvisible)
editor.setBraceMatching(QsciScintilla.SloppyBraceMatch)
editor.setWrapMode(QsciScintilla.WrapWord)
```

```
editor.setTabIndents(True)
editor.setTabWidth(4)
```

With `PySci`, this can be condensed into a single constructor call:

```
editor = PySci(
    whitespaceVisibility = 'WsInvisible',
    braceMatching = 'SloppyBraceMatch',
    wrapMode = 'WrapWord',
    tabIndents = True,
    tabWidth = 4)
```

Additional configuration changes can be made via the `configure` method:

```
editor.configure(
    indentationGuides = True,
    eolVisibility = True,
    edgeColumn = 72)
```

You can also get or set individual configurations by passing their name as a string to the `get_config` or `set_config` methods:

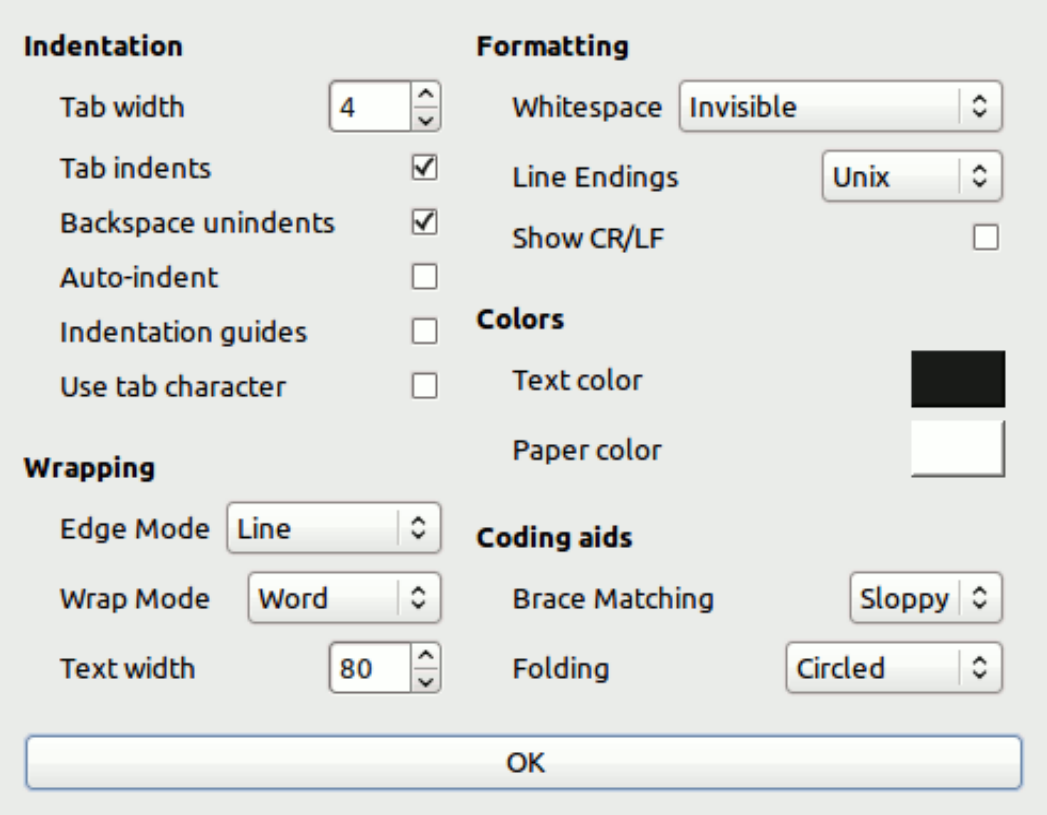
```
eol_mode = editor.get_config('eolMode')
editor.set_config('eolMode', 'EolMac')
```

This can be useful if you have setting names in variables.

Settings Widget

Considering the enormous number of methods in QsciScintilla for dealing with configuration settings, it's a little surprising that a dedicated configuration widget is not provided. If you've worked with QsciScintilla with any seriousness, you've probably built an ad-hoc configuration widget yourself (and so has everyone else). That's a shameful duplication of programming effort. PySci provides a ready-made one called `PySciSettings` to save you the trouble.

Here's an example of what it looks like:



To use it, connect an event handler to a button or menu entry, then instantiate `PySciSettings`, passing it your `PySci` editor instance:

```
from pysci import PySciSettings
settings = PySciSettings(editor)
```

All changes to configuration settings in this widget take effect in realtime in the associated `PySci` widget.

Future Plans

As of this writing, PySci is a fairly minimal wrapper around the existing QsciScintilla widget. It provides a few convenience features, but nothing earth-shattering. Here are some features that may appear in the future:

- Support for choosing syntax highlighting for any of the language lexers that QsciScintilla supports
- Direct attribute-based wrappers around the getter/setter methods
- Loading and saving of files, with auto-detection of syntax highlighting based on filename extension
- Loading and saving of PySci configuration preferences
- Built-in docstrings for the standard QsciScintilla methods

Missing methods

QsciScintilla provides several "setter" methods with no "getter" counterpart. For example, many of the methods for specifying colors:

- `setMarginsBackgroundColor`

- `setMarginsForegroundColor`
- `setCaretLineBackgroundColor`
- etc.

Once a color is set, there's no way to retrieve it later. In other words, these are **write-only** attributes.

Another feature that is lacking from QsciScintilla is the ability to easily retrieve geometry information. For instance, what if you need to know the Y-coordinate in screen pixels of a given line in your editor widget? Or a bounding rectangle around the text in a given line? Many of the other PyQt4 widgets provide such methods, but QsciScintilla doesn't. PySci aims to remedy this, making the widget fit more closely with its Qt counterparts.

API

This is the PySci API documentation, autogenerated from the source code.

editor

enums

settings

util

CHAPTER 2

Indices and tables

- genindex
- modindex
- search