Programming: PyLint, C2x, Librem 5, Portable Document Format (PDF) in Python

By *Roy Schestowitz*
Created *14/05/2019 - 6:39pm*
Submitted by Roy Schestowitz on Tuesday 14th of May 2019 06:39:07 PM Filed under Development [1]

- **Writing Cleaner Python Code With PyLint** **[2]**

  PyLint is a well-known static analysis tool for Python 2 and 3. It has a number of useful features, like checking your code for compliance with the PEP 8 Python style guide. It makes sure that your code follows the code style guide and it can also automatically identify common bugs and errors in your Python code.

  In this video series you?ll see how to install and set up the PyLint code linter tool. You?ll learn why you should use code linters like PyLint, Flake8, PyFlakes, or other static analysis tools?and how they can help you write cleaner and more Pythonic code.

  You can get this setup up and running in a few minutes and it?ll quickly help you write better and cleaner Python code.

- **LLVM Clang 9.0 Picks Up Initial C2x Language Mode** **[3]**

  Merged today to the mainline Clang compiler front-end is the initial C2x language mode support as what will eventually be the successor to the C18 programming language.

  C2x is still quite a ways out from release and its changes still under determination. At this stage the C2x language support for LLVM Clang is just enabling support by default for the [[attribute]] (double square brackets attribute; similar to C++) support.

- **Librem 5 App Design Tutorial ? Part II** **[4]**

Hello and welcome to the second of my series of blog posts on how to design your own, brand new app for the Librem 5.

In my last post we went over the philosophy and process, goals and relevant art of building a read-it-later app; today we?ll be discussing sketches and mockups ? specifically in what concerns navigation, article and article list screens, and desktops.

- **Working with PDFs in Python: Adding Images and Watermarks [5]**

  Today, a world without the Portable Document Format (PDF) seems to be unthinkable. It has become one of the most commonly used data formats ever. Up to PDF version 1.4, displaying a PDF document in an according PDF viewer works fine. Unfortunately, the features from the newer PDF revisions, such as forms, are tricky to implement, and still require further work to be fully functional in the tools. Using various Python libraries you can create your own application in an comparable easy way.

  This article is part two of a little series on PDFs with Python. In part one we already gave you an introduction into reading PDF documents using Python, and started with a summary of the various Python libraries. An introduction followed that showed how to manipulate existing PDFs, and how to read and extract the content - both the text and images. Furthermore, we showed you how to split documents into its single pages.

  In this article you will learn how add images to your PDF in the form of watermarks, stamps, and barcodes. For example this is quite helpful in order to stamp or mark documents that are intended to be read by a specific audience, only, or have a draft quality, or to simply add a barcode for identification purposes.

Development

Source URL: http://www.tuxmachines.org/node/123894

Links:
[1] http://www.tuxmachines.org/taxonomy/term/145
[2] https://realpython.com/courses/writing-cleaner-python-code-pylint/
[3] https://www.phoronix.com/scan.php?page=news_item&px=LLVM-Clang-9-C2x
[4] https://puri.sm/posts/librem-5-app-design-tutorial-part-ii/
[5] https://stackabuse.com/working-with-pdfs-in-python-adding-images-and-watermarks/