CMPSC 104 – Document Engineering

Prof. Hang Zhao

ALLEGHENY COLLEGE

Halloween

Overview

What is reStructuredText (*.rst)?

- A lightweight markup language (like Markdown and HTML).
- Easy-to-read, easy-to-write plain text format
- Commonly used in Python documentation and other technical documents

Overview

Examples of reStructuredText:

- The official Python documentation and Python Enhancements Proposals (PEPs) are all written using reStructuredText.

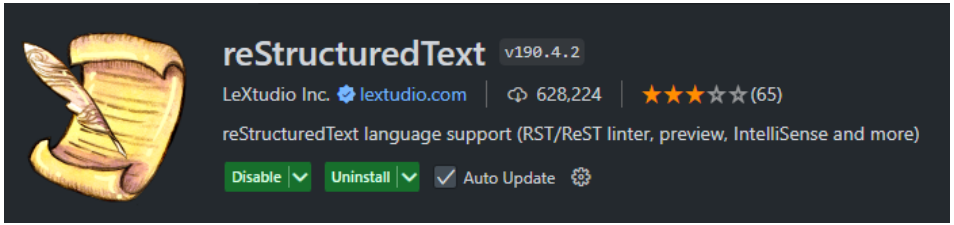
- Python open-source libraries are also using reStructuredText to author their documentation, for example: PyPA/packaging, requests, black.

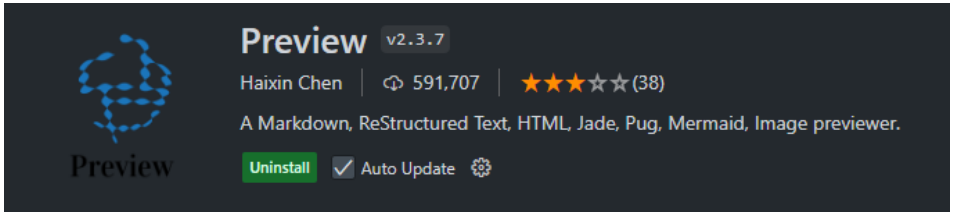
- GitHub also supports rendering reStructuredText on their web UI. For example, you can create readme.rst on your project, instead of the default readme.md.

Why Use reStructuredText

Advantages:

- Simple and readable syntax





Example:

```
This is a *simple* reST document.

-  This is a bullet point
-  Another bullet point

1. Numbered list item
2. Another numbered list
```

Why Use reStructuredText

- Highly extensible
- Converts to various output formats (HTML, LaTeX, PDF, etc.)

```
.. code-block:: python

   def greet():
       print("Hello, World!")

You can also write math:

.. math::

   E = mc^2
```

reStructuredText, Docutils and Sphinx.

Docutils:

- Docutils is the utility tool that can convert plain text documents (including reStructuredText) into other useful formats, like HTML, XML or LaTeX.
- Docutils is written in Python and is open source. The only dependency you need in order to use it, is Python itself.

Sphinx:

- Sphinx is like the engine that can take your various documentation files into an intelligent and complete Documentation project.
- reStructuredText and Sphinx becomes an indispensable skill for any Python open-source contributors.

reStructuredText

- reStructuredText works the same way as Markdown
- The file extension is .rst.
- With the acceptance of PEP 287, reStructuredText was adopted for the use of Python docstrings.
    - PEP 287: *"reStructuredText Docstring Format,"* proposes the use of reStructuredText as the standard format for writing Python docstrings.

reStructuredText Syntax: Text Emphasis

- **Paragraphs**: Just type your text
- Wrap text in single asterisks for **italics** text**:** *italic*
- Double asterisks to make it **bold**: **bold**
- **Double-backticks for inline “code” formatting:** ``code``
- **Line Breaks**: Leave a blank line

reStructuredText Syntax: Titles/Headings

Section 1

==========

Section 2

Subsection 1.1

---

reStructuredText: Links

- `` `Python Docs <https://docs.python.org>`_ ``

> **Note**
>
> Spacing is important here. There needs to be an empty space between the displayed text and the `<` sign.

- `` `Python Docs<https://docs.python.org>`_ ``

reStructuredText: Links

- If you would be referencing the same urls multiple times in the same page, you don't have to re-write the same urls again and again. You can declare a name for it.

```
.. _Python Docs: https://docs.python.org
```

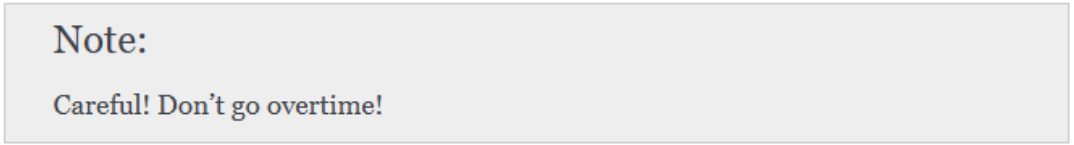
- Add that to the bottom of your .rst file. Then, anywhere in that file, you can simply type `` `Python Docs`_ `` , and at render time, it will automatically be linked.

reStructuredText: Admonition

- The “admonition” directives in reStructuredText, kind of like a callout.
- If we want to have some text in a special callout box, you can use the **.. note::** directive.

```
.. note::
   Careful! Don't go overtime!
```

Will be rendered as:



reStructuredText: Admonition

There are different kinds of admonitions, like **.. attention::, .. caution::,** and **.. tip::**

```
.. danger::
    This step is dangerous! Be real sure about it!
```

Will be rendered as:

https://docutils.sourceforge.io/docs/ref/rst/directives.html#specific-admonitions

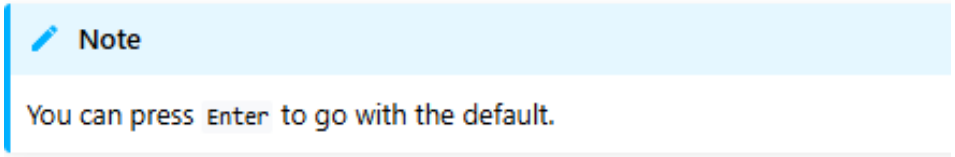
Sphinx: Installation

- `mkdir docs`
- `cd docs`
- virtual environment: `python -m venv myenv`
- Windows: `myenv\Scripts\activate`
- MacOS/Linus: `source myenv/bin/activate`

  You should see this: ((env) C:\ docs >)
- `python -m pip install sphinx`
- `sphinx-quickstart`

Sphinx

```
> Separate source and build directories (y/n) [n]: y
> Project name: Your project's name
> Author name(s): Your name or your team's name
> Project release []: Enter
> Project language [en]: Enter
```



Sphinx

```
/_build/
/_static/
/_templates
conf.py
index.rst
make.bat
Makefile
```

1. Drag the `demo.rst` file into the `/docs/source` directory.
2. Add demo to the toctree in index.rst.

Building documentation with Sphinx

Once you have Sphinx set up, you can build the documentation by typing `make html` command on the terminal, within the docs directory, with the virtual environment activated.

- `.\make.bat html or make html`

Open the index.html (optional):

- `start build\html\index.html` `# windows`
- `open build/html/index.html` `# macOS`
- `xdg-open build/html/index.html` `# Linux`

If you have a virtual environment, you can "exit" or `deactivate` it by typing deactivate from the command line: `(.env)[../docs]$ deactivate`

reStructuredText: Literal

- Literal block is a block of text where line breaks and whitespaces are significant and preserved.
- **Two colons :: followed by a blank line.**
- **Then indent the text block after it**. The following text block will become literal block.

```
::

    Lorem ipsum dolor sit amet, consectetur adipiscing elit. In viverra convallis
    feugiat. Mauris facilisis dolor placerat massa porta vehicula. Vivamus quis
    posuere quam. Sed semper, libero vel tristique tincidunt, metus diam
    pellentesque dolor, congue dignissim nibh ipsum sed dui.
```

reStructuredText: Literal

- The **::** can be in a line of its own, or added at the end of the previous paragraph's sentence.

```
This text will be formatted as usual, meanwhile::

   This passage will become literal block. Lorem ipsum dolor sit amet,
   consectetur adipiscing elit. In viverra convallis
   feugiat. Mauris facilisis dolor placerat massa porta vehicula. Vivamus quis
   posuere quam. Sed semper, libero vel tristique tincidunt, metus diam
   pellentesque dolor, congue dignissim nibh ipsum sed dui.
```

reStructuredText: Code Blocks

- Code blocks works somewhat similar to a literal block. Use the **.. code::** directive, followed by an empty line.

```
.. code:: python

    from datetime import datetime
    print(datetime.now())
```

reStructuredText: Lists

- Add **asterisks** or **numbers** in front of each list item.
- **Use #.** to auto-number the list items.

```
* Bullet list item 1
* Bullet list item 2
  If the list item text is too long, it can go to the next line, like this.

1. Numbered list item 1
2. Numbered list item 2

   1. Nested list item 2
   2. Nested list item 2

#. Autonumbered list item
#. Autonumbered list item
```

reStructuredText: Documentation Comments

- We can write comment in reStructuredText.
- The comment will appear hidden and not rendered.
- **Add two dots (..) followed by a space**, and the text will become “comments”.

```
.. This line will not be rendered.

..
   You can have multiline comments, by adding indented text blocks.
   This line will not be rendered.

   This is still a comment.
```

reStructuredText: Simple Table

```
========  ========  ========
Column A  Column B  Column C
========  ========  ========
row 1A    row 1B    row 1C
row 2A    row 2B    row 2C
========  ========  ========
```

Will be rendered as:

Column A	Column B	Column C
row 1A	row 1B	row 1C
row 2A	row 2B	row 2C

reStructuredText: Grid Tables

```
+----------+----------+----------+
| Column A | Column B | Column C |
+==========+==========+==========+
| row 1A   | row 1B   | row 1C   |
+----------+----------+----------+
| row 2A   | row 2B   | row 2C   |
+----------+----------+----------+
```

Will be rendered as:

Column A	Column B	Column C
row 1A	row 1B	row 1C
row 2A	row 2B	row 2C

reStructuredText: CSV Tables

```
.. csv-table::
   :header: "Column A", "Column B", "Column C"

   "row 1A", "row 1B", "row 1C"
   "row 2A", "row 2B", "row 2C"
```

Will be rendered as:

Column A	Column B	Column C
row 1A	row 1B	row 1C
row 2A	row 2B	row 2C

reStructuredText: Table of Contents

Use the `.. contents::` directive to automatically generate a Table of Contents based on the section headers on your documentation.

```
.. contents::
```

reStructuredText: Rendering Image

- There is an **.. image::** directive for this:

```
.. image:: path_to_image.jpg
```

Reference and Resources

Intro to Documentation with Sphinx and reStructedText (https://sphinx-intro-tutorial.readthedocs.io/)