Sphinx:

Sphinx is by far the best and most used documentation generator for Python.As it supports reStructuredText in docstrings and produces a HTML output with a clean visual style. In a range of languages it gives facilities for documentation of software projects.

The highlighted features of Sphinx:

Output formats: HTML (including Windows HTML Help), LaTeX (for printable PDF versions), ePub, Texinfo, manual pages, plain text

- **Extensive cross-references:** semantic markup and automatic links for functions, classes, citations, glossary terms and similar pieces of information
- **Hierarchical structure:** easy definition of a document tree, with automatic links to siblings, parents and children
- **Automatic indices:** general index as well as a language-specific module indices
- **Code handling:** automatic highlighting using the Pygments highlighter
- **Extensions:** automatic testing of code snippets, inclusion of docstrings from Python modules (API docs), and more
- **Contributed extensions:** more than 50 extensions contributed by users in a second repository; most of them installable from PyPI

Cons of Sphinx:

The only con one can find is that setting it up requires a bit of configuration (using Makefiles) and the documentation for getting started assumes the user is working with a fresh repo. One can also run it with a quickstart script that uses default configurations but it still requires multiple steps. Works for Python 2 and 3 and loads docstrings dynamically through introspection.

Getting Started:

Sphinx is a documentation generator or a tool that translates a set of plain text source files into various output formats, automatically producing cross-references, indices, etc. That is, if you have a directory containing a bunch of reStructuredText or Markdown documents, Sphinx can generate a series of HTML files, a PDF file (via LaTeX), man pages and much more.

Sphinx focuses on documentation, in particular handwritten documentation, however, Sphinx can also be used to generate blogs, homepages and even books. Much of Sphinx's power comes from the richness of its default plain-text markup format, reStructuredText, along with it's significant extensibility capabilities.

The goal of this document is to give you a quick taste of what Sphinx it is and how you might use it. When you're done here, you can check out the installation guide followed by the intro to the default markup format used by Sphinx, reStucturedText.

For a great "introduction" to writing docs in general – the whys and hows, see also Write the docs, written by Eric Holscher.

Setting up the documentation sources
The root directory of a Sphinx collection of plain-text document sources is called the source directory. This directory also contains the Sphinx configuration file conf.py, where you can configure all aspects of how Sphinx reads your sources and builds your documentation. 1

Sphinx comes with a script called sphinx-quickstart that sets up a source directory and creates a default conf.py with the most useful configuration values from a few questions it asks you. To use this, run:

$ sphinx-quickstart

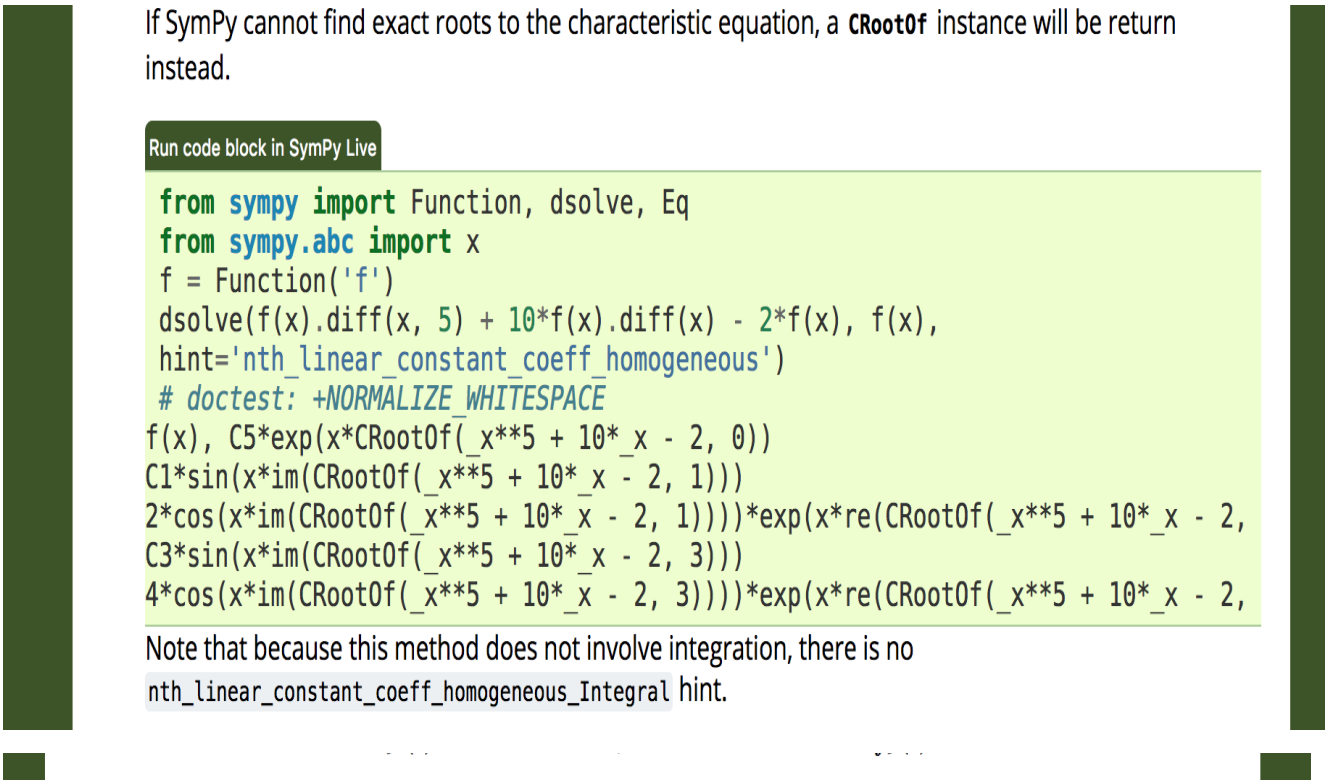
Basic configuration:

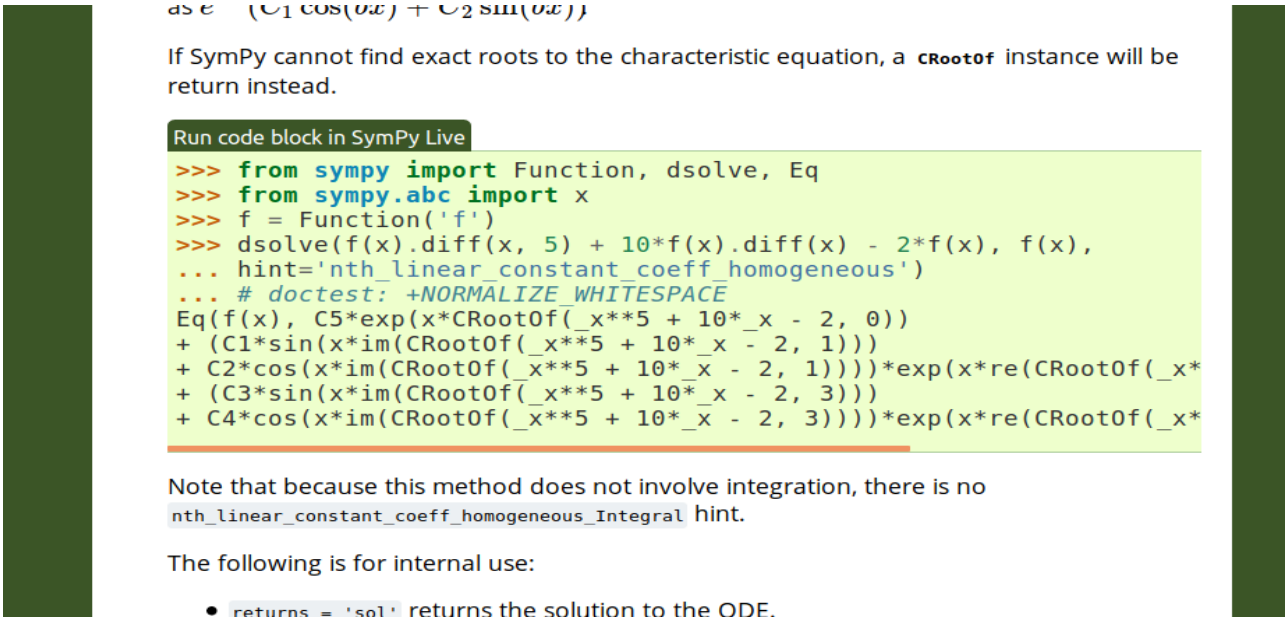
Earlier we mentioned that the conf.py file controls how Sphinx processes your documents. In that file, which is executed as a Python source file, you assign

configuration values. For advanced users: since it is executed by Sphinx, you can do non-trivial tasks in it, like extending **sys.path** or importing a module to find out the version you are documenting.

The config values that you probably want to change are already put into the conf.py by **sphinx-quickstart** and initially commented out (with standard Python syntax: a # comments the rest of the line). To change the default value, remove the hash sign and modify the value. To customize a config value that is not automatically added by **sphinx-quickstart**, just add an additional assignment.

Keep in mind that the file uses Python syntax for strings, numbers, lists and so on. The file is saved in UTF-8 by default, as indicated by the encoding declaration in the first line.





as $e^{x}\left(C_1\cos(bx)+C_2\sin(bx)\right)$

If SymPy cannot find exact roots to the characteristic equation, a `CRootOf` instance will be return instead.

Run code block in SymPy Live

```
>>> from sympy import Function, dsolve, Eq
>>> from sympy.abc import x
>>> f = Function('f')
>>> dsolve(f(x).diff(x, 5) + 10*f(x).diff(x) - 2*f(x), f(x),
... hint='nth_linear_constant_coeff_homogeneous')
... # doctest: +NORMALIZE_WHITESPACE
Eq(f(x), C5*exp(x*CRootOf(_x**5 + 10*_x - 2, 0))
+ (C1*sin(x*im(CRootOf(_x**5 + 10*_x - 2, 1)))
+ C2*cos(x*im(CRootOf(_x**5 + 10*_x - 2, 1))))*exp(x*re(CRootOf(_x*
+ (C3*sin(x*im(CRootOf(_x**5 + 10*_x - 2, 3)))
+ C4*cos(x*im(CRootOf(_x**5 + 10*_x - 2, 3))))*exp(x*re(CRootOf(_x*
```

Note that because this method does not involve integration, there is no `nth_linear_constant_coeff_homogeneous_Integral` hint.

The following is for internal use:

- `returns = 'sol'` returns the solution to the ODE.

Sites that used Sphnix(Example):

https://docs.aiohttp.org/en/stable/

https://alabaster.readthedocs.io/en/latest/

Reference:https://www.sphinx-doc.org/en/master/