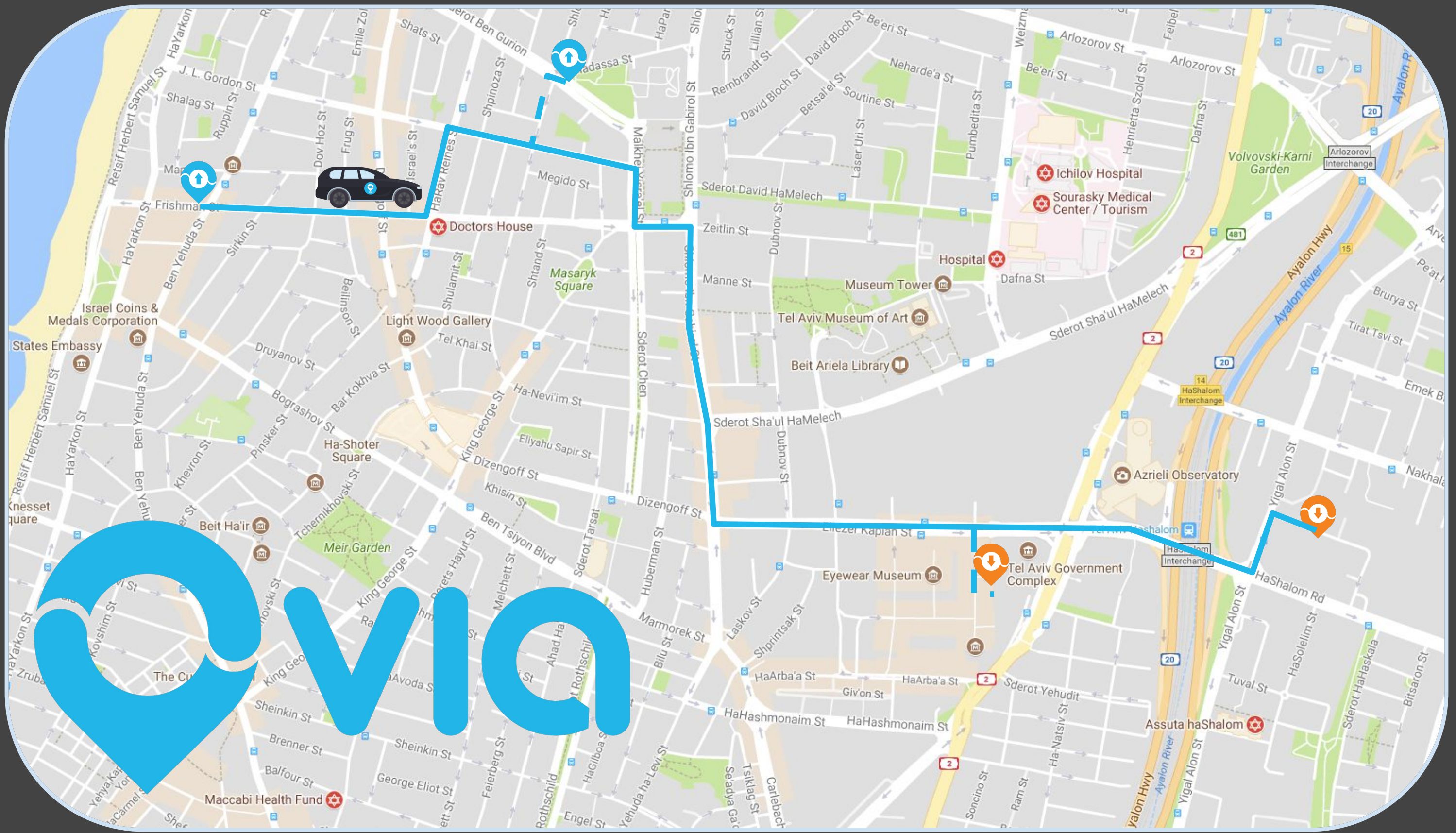
Help your colleagues help themselves - a Sphinx tutorial

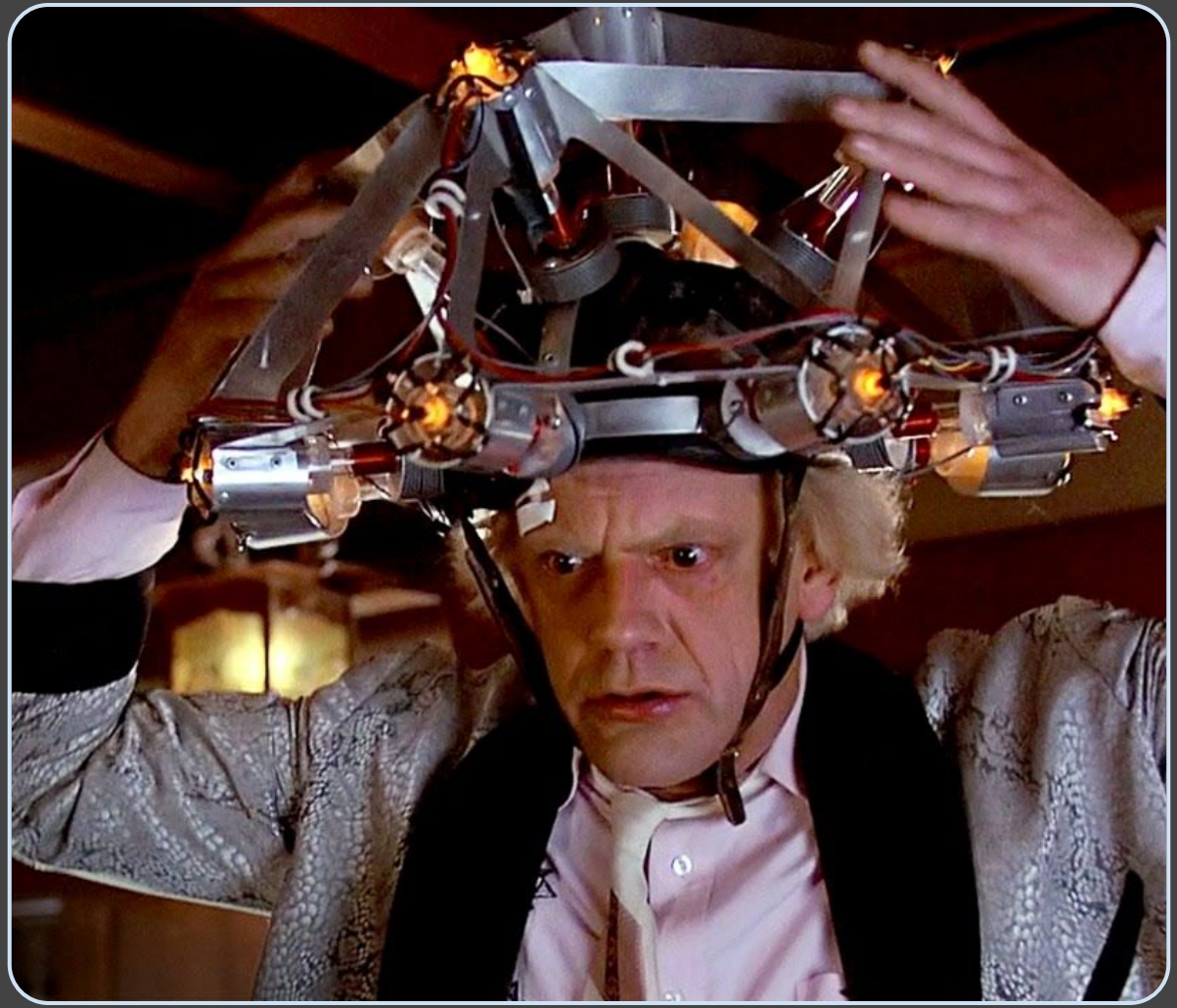
Dalya Gartzman

github.com/DalyaG/Sphinx185



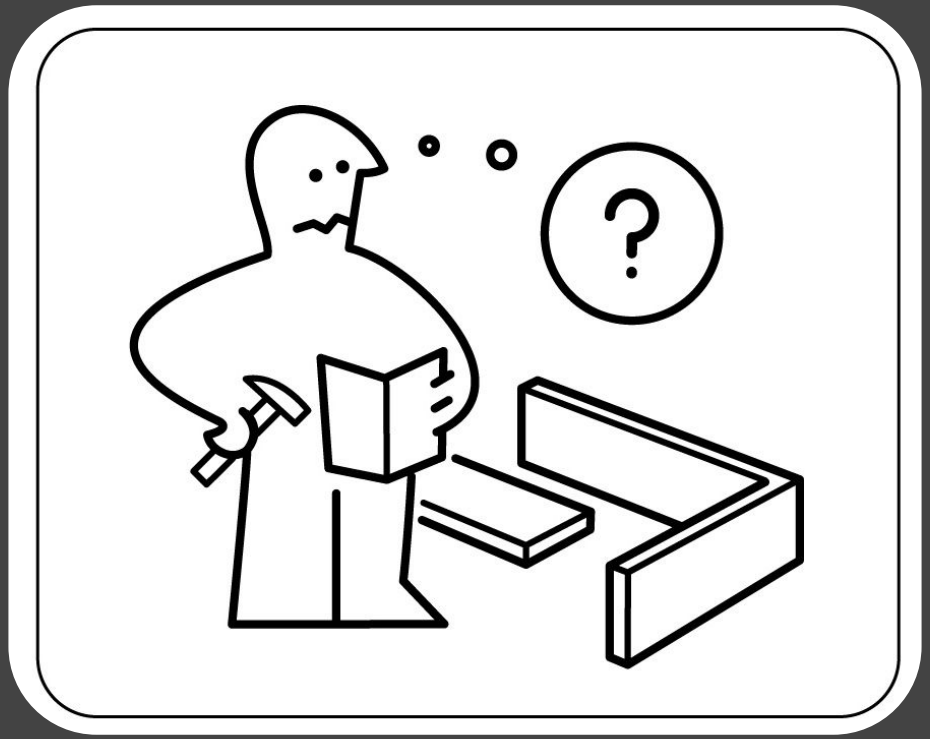
Arlozorov St
Be'eri St
Nehardea St
Soutine St
Rembrandt St
David Bloch St
Betsal'el St
Shlomo Ibn Gabirol St
Sderot David HaMelech
Zeitlin St
Manne St
Museum Tower
Tel Aviv Museum of Art
Beit Ariela Library
Sderot Sha'ul HaMelech
Ichilov Hospital
Sourasky Medical Center / Tourism
Hospital
Dafna St
Henrietta Szold St
Volvovski-Karni Garden
Arlozorov Interchange
Ayalon Hwy
Ayalon River
HaShalom Interchange
Azrieli Observatory
Yigal Alon St
HaShalom Rd
Tel Aviv Government Complex
Eyewear Museum
Dizengoff St
Ben Tsiyon Blvd
Marmorek St
HaArba'a St
Giv'on St
Sderot Yehudit
HaHashmonaim St
Assuta haShalom
Tuval St
J. L. Gordon St
Shalag St
Frishman St
Doctors House
Megido St
Masaryk Square
Light Wood Gallery
Tel Khai St
Israel Coins & Medals Corporation
States Embassy
Druyanov St
Bograshov St
Bar Kokhva St
Ha-Shoter Square
Ha-Nevi'im St
Elyahu Sapir St
King George St
Meir Garden
Beit Ha'ir
Sheinkin St
Brenner St
Balfour St
George Eliot St
Maccabi Health Fund
Sderot Chen
Sderot Tarsat
Huberman St
Shprintsak St
Rothschild
Engel St
Ben Yehuda St
HaYarkon St
Retsif Herbert Samuel St
Emek B
Nakhalat
Tirat Tsvi St
Brurya St
via


MT0A
CR1A
MT1A
MT2A
CR1A
CP1A
CPJA
TY1A
CR1A
LP1A
MT2A

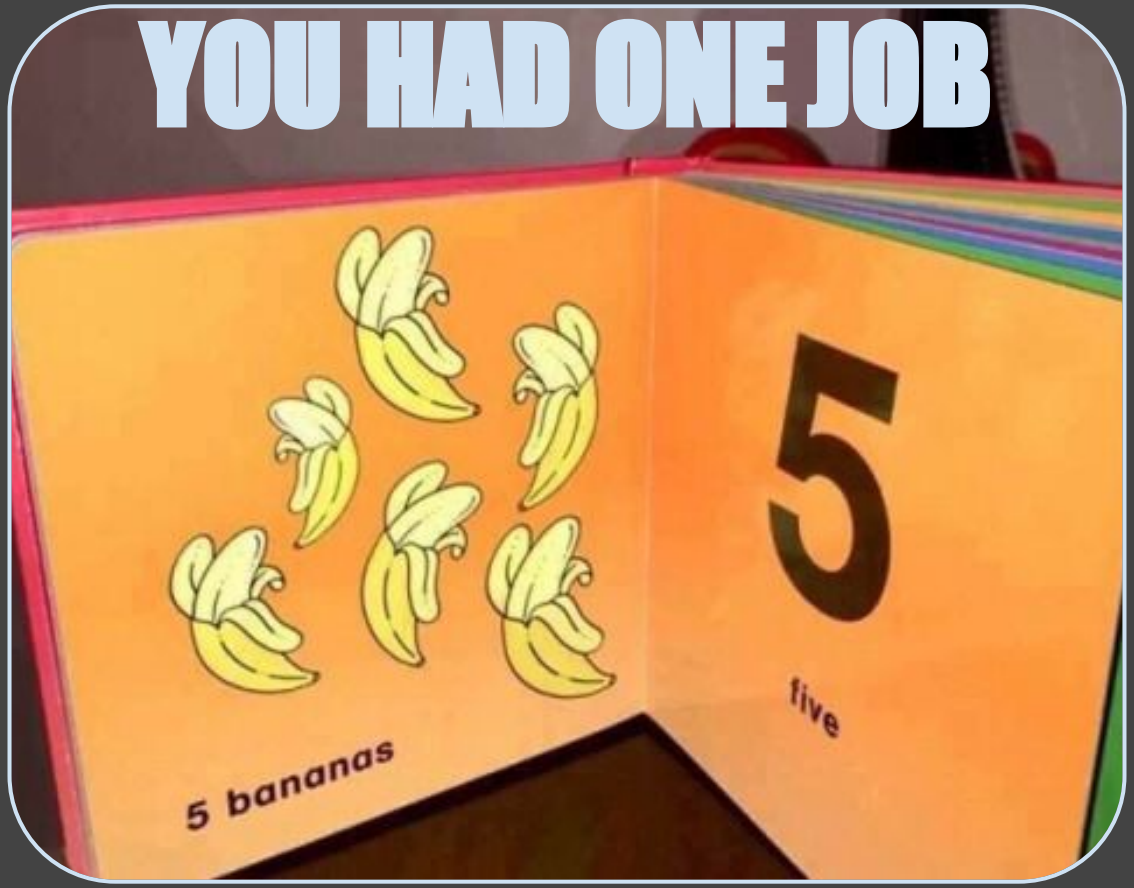



SPHINX
PYTHON


PUBLISH
PUBLISH EVERYTHING




NEWBIES


YOU HAD ONE JOB
5 bananas
5
five

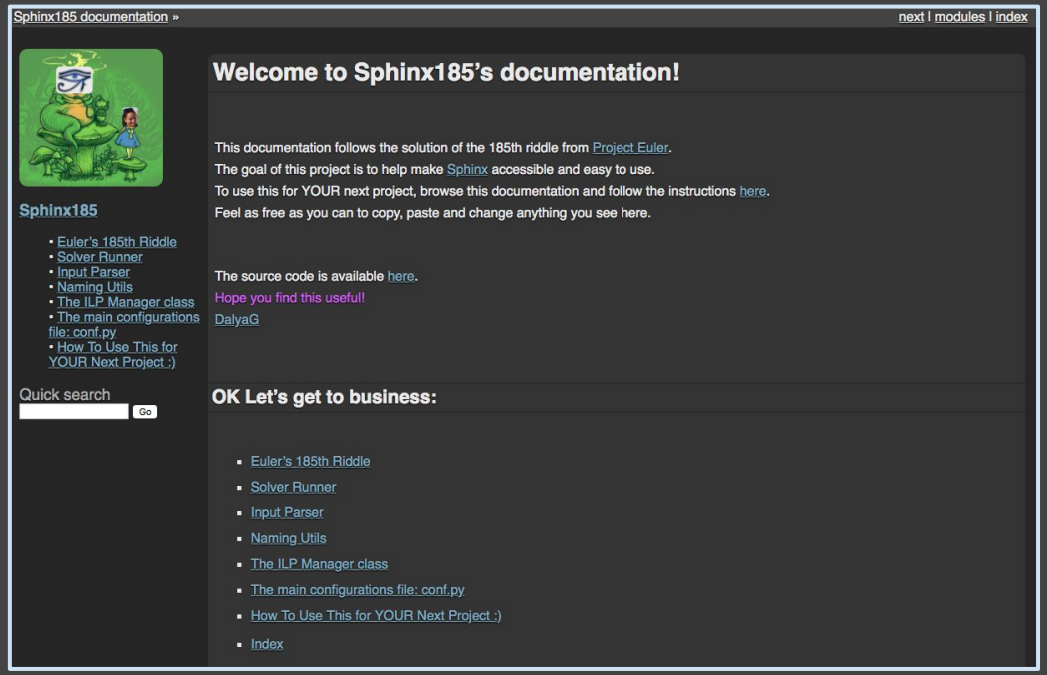
Press Start

READY PLAYER ONE

Press Start

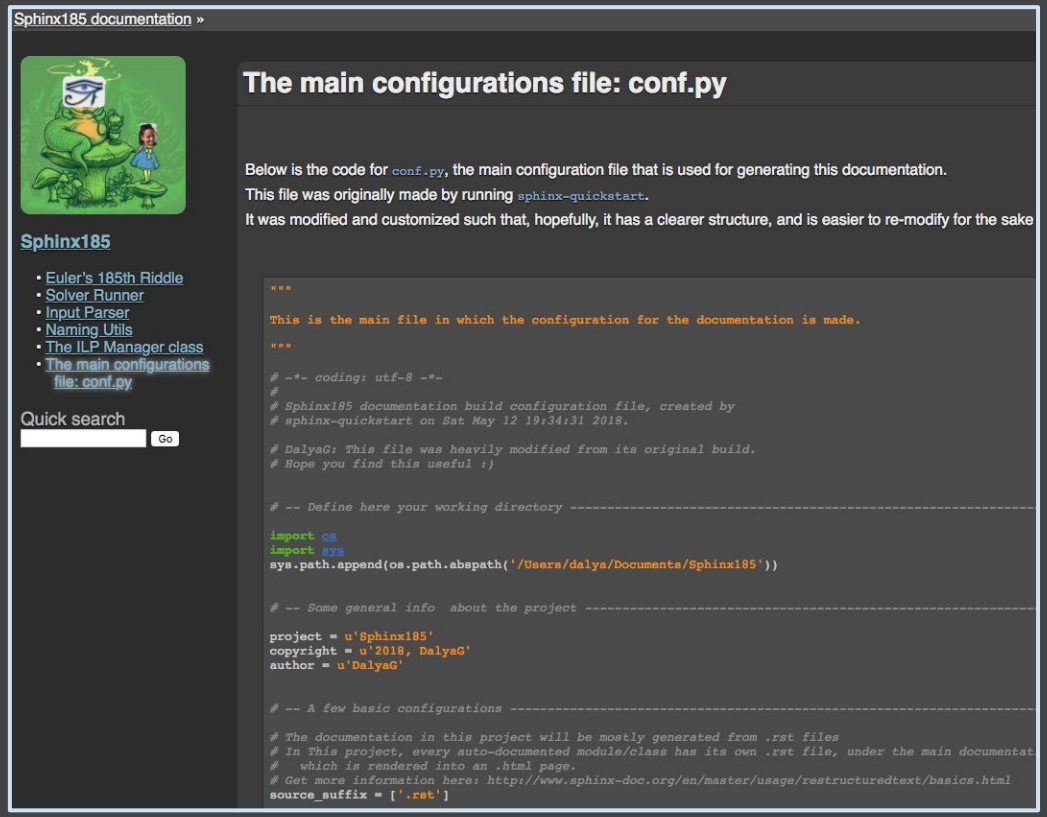
Intro - What will we accomplish today?

Intro - What will we accomplish today?



Part I - Overview

Intro - What will we accomplish today?



Part II - Main configuration file - conf.py

Intro - What will we accomplish today?

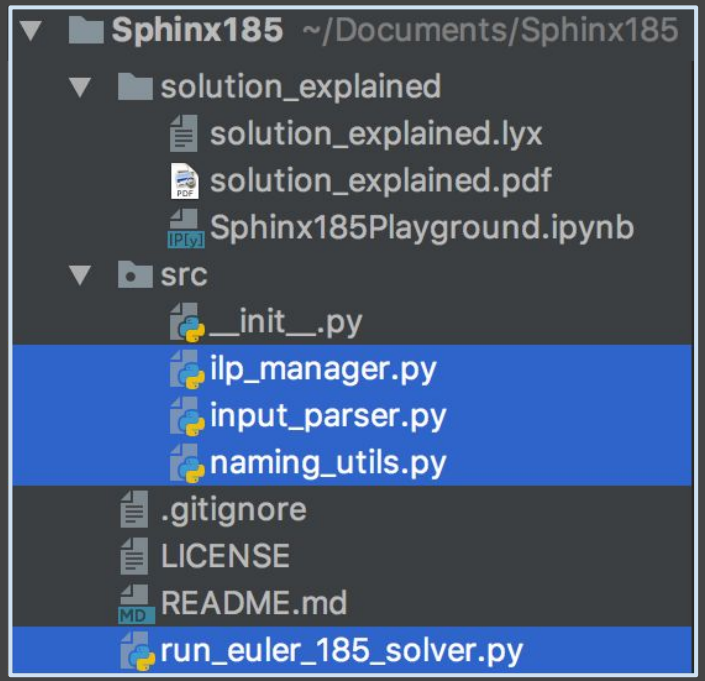
How To Use This for YOUR Next Project :)

1. `pip install sphinx`
2. Copy the `documentation_template_for_your_next_project` folder in this repository,
   make it a subdirectory in your local project and rename it `documentation`.
3. Edit `conf.py` by searching the pattern `#CHNAGEME#` inside the file and follow the instructions.
4. Edit `index.rst` by following the inline instructions.
5. To add a page to your documentation:
   a. Create a `.rst` file for the function/module you wish to document
      (you can use the templates supplied here), and
   b. Add the name of the `.rst` file to the `toctree` in `index.rst`.
6. In terminal, inside the `documentation` folder, run `make html`.
   (**TIP OF THE WEEK**: actually, always run `make clean html` to clear sphinx cache and build from scratch)
7. To view locally - open the file `documentation/_build/html/index.html` in your browser, and enjoy the read :)
8. To share - you can use GitHub Pages to host your documentation:
   a. Copy the content of `documentation/_build/html/` into a new `docs` folder, under the root of the project.
   b. Create an empty file `.nojekyll` inside `docs` folder
      (this tells GitHub Pages to bypass the default `jekyll` themes and use the `html` and `css` in your project)
   c. Push your changes to master branch.
   d. In your repository on GitHub, go to “Settings” -> “GitHub Pages” -> “Source”
      and select “master branch /docs folder”.
   e. Share your beautiful documentation site at `https://<your_git_usrname>.github.io/<project_name>/`

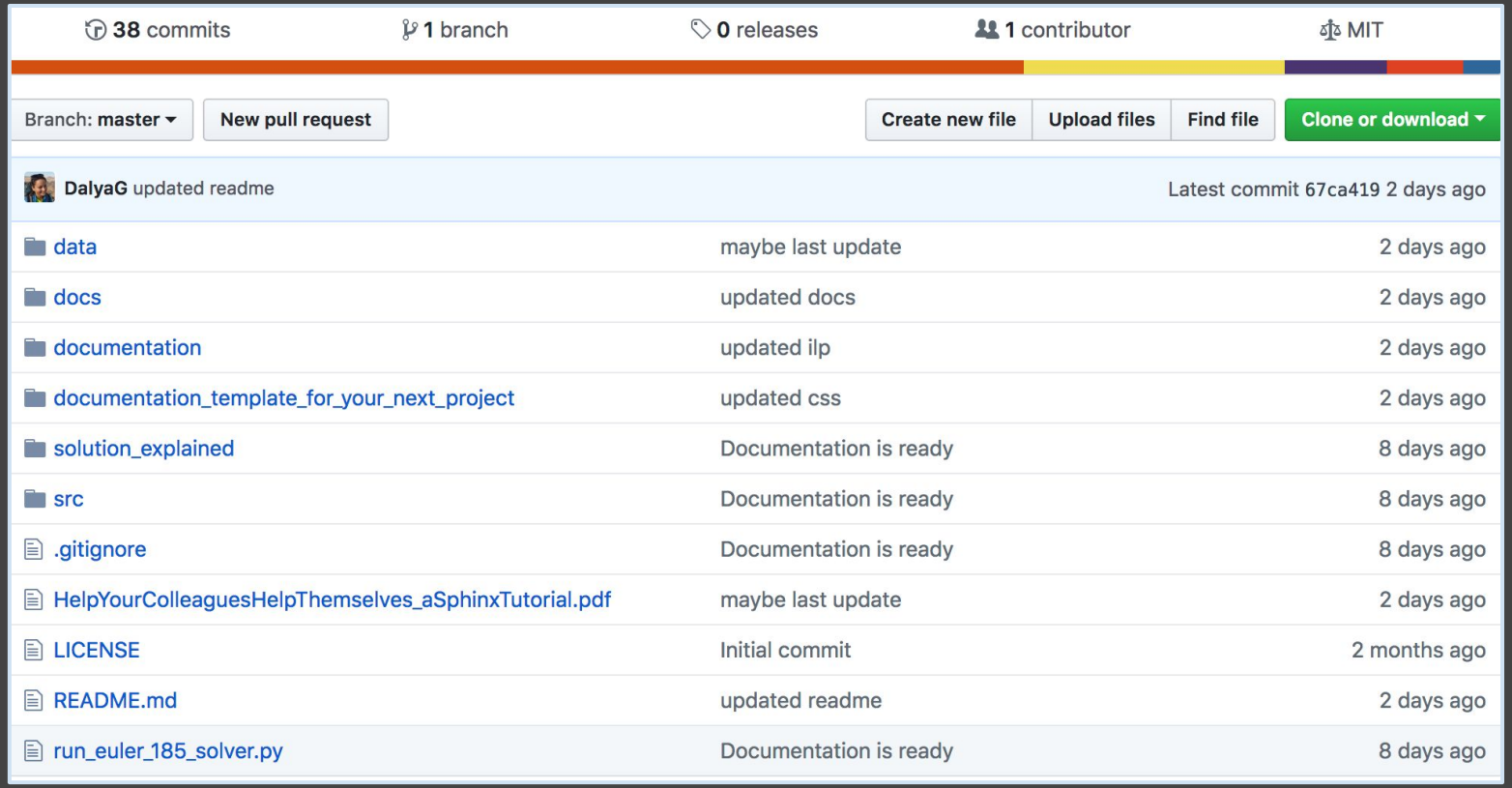
Part III - Document YOUR next project

Part I - Overview

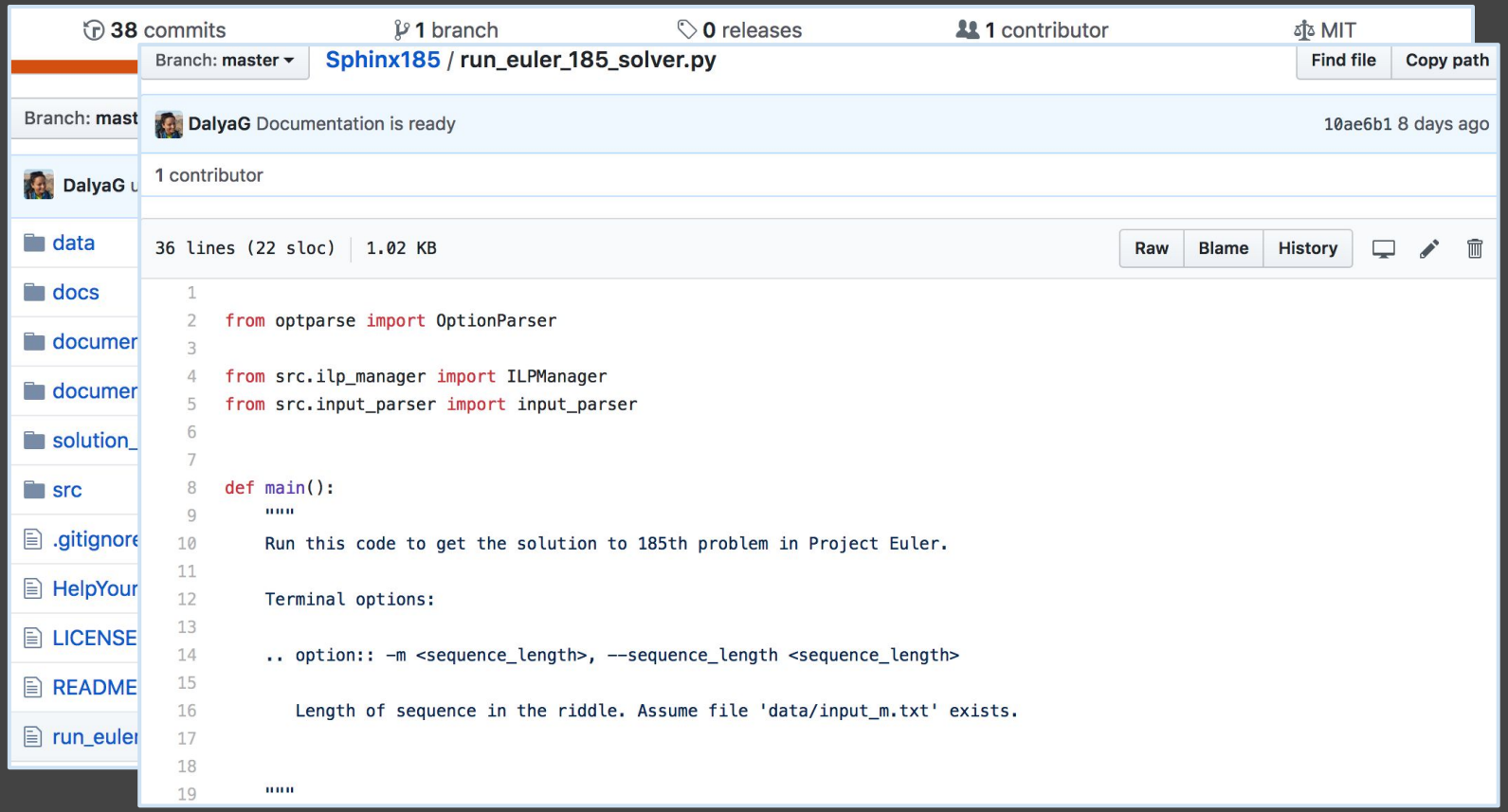
Part I - Overview



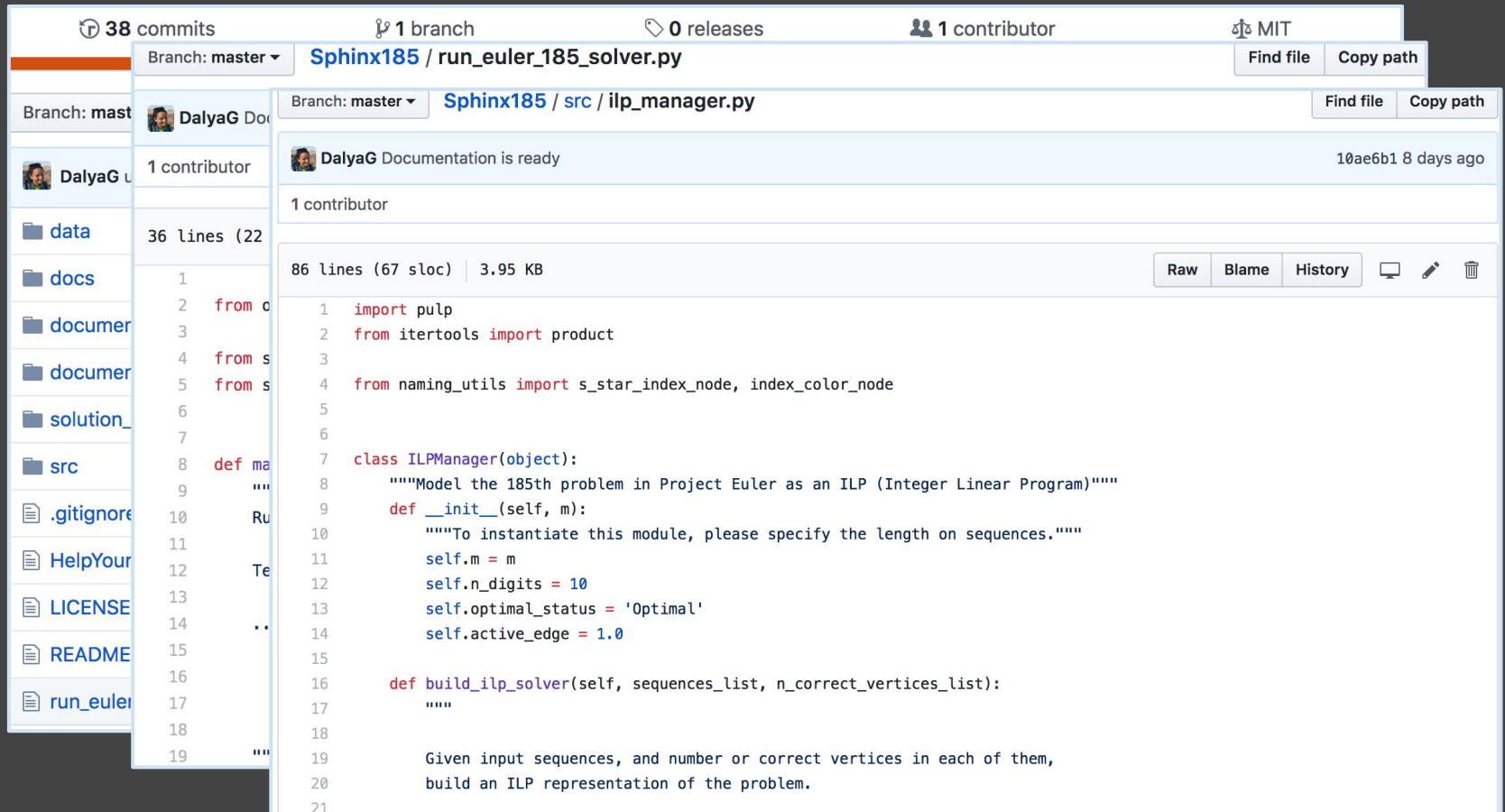
Part I - Overview



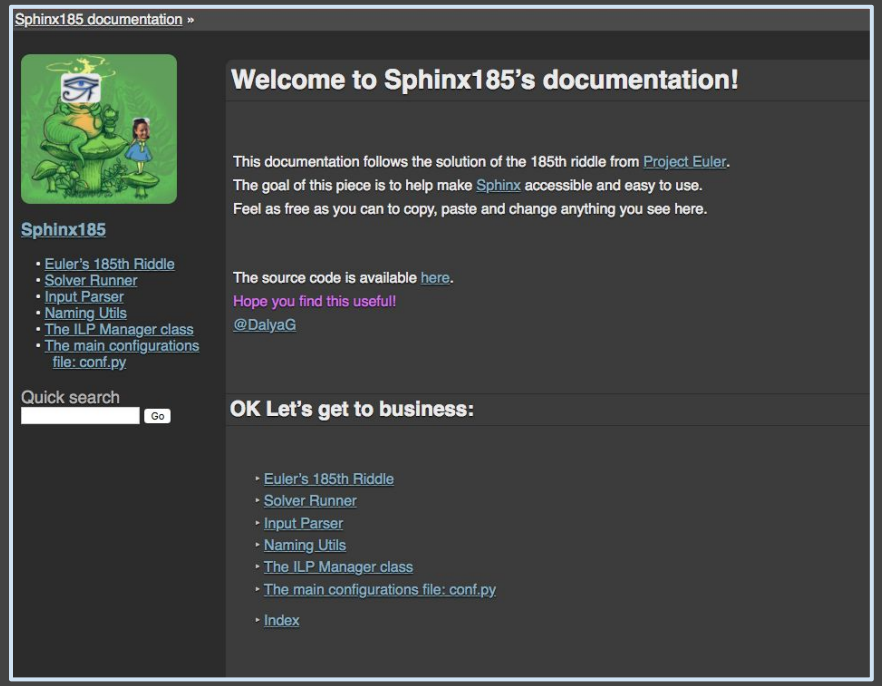
Part I - Overview



Part I - Overview

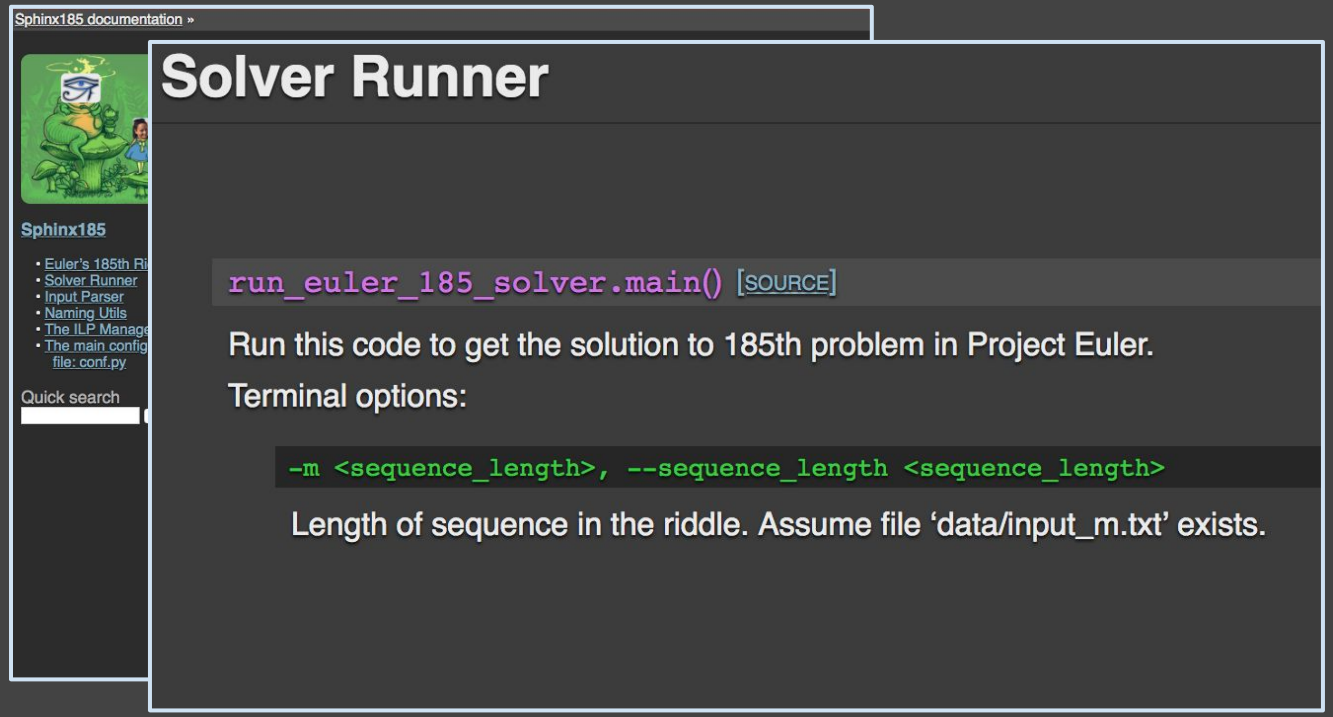


Part I - Overview



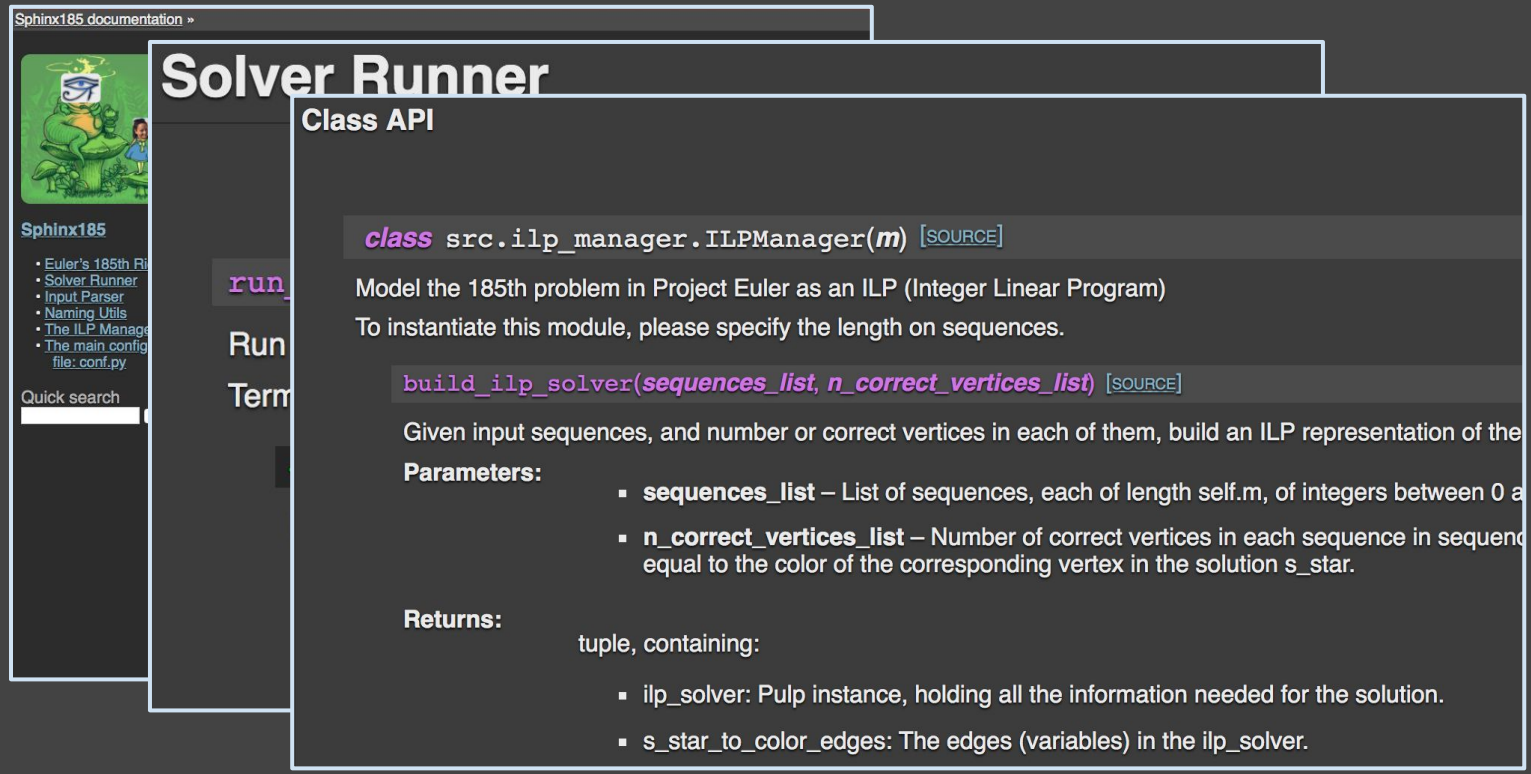
dalyag.github.io/Sphinx185

Part I - Overview



dalyag.github.io/Sphinx185

Part I - Overview



dalyag.github.io/Sphinx185

Part I - Overview

Euler's 185th Riddle

In case you are interested, the riddle goes something like this:

> Let s^* be a sequence of m ***unknown*** digits (colors from 0 to 9).
> We are given n sequences of m ***known*** colors, $x_{i,j}$ marks the color of the j^{th} index in sequence i.
> For each sequence we are given the number of ***correct colors***, that is, indices in the sequence for which $x_{i,j} = s^*_j$.
> Our goal is to discover the colors of s^*, which we are promised are unique, given the input.

Part I - Overview

Euler's 185th Riddle

In case you are interested, the riddle goes something like this:

> Let s^* be a sequence of m ***unknown*** digits (colors from 0 to 9).
> We are given n sequences of m ***known*** colors, $x_{i,j}$ marks the
> For each sequence we are given the number of ***correct colors***,
> Our goal is to discover the colors of s^*, which we are promised a

Solver Runner

`run_euler_185_solver.main()` [SOURCE]

Run this code to get the solution to 185th problem in Project Euler.

Terminal options:

```
-m <sequence_length>, --sequence_length <sequence_length>
```

Length of sequence in the riddle. Assume file 'data/input_m.txt' exists.

Part I - Overview

Euler's 185th Riddle

In case you are inter

Let s^* be a se
We are given
For each sequ
Our goal is to

Source code for run_euler_185_solver

```
from optparse import OptionParser

from src.ilp_manager import ILPManager
from src.input_parser import input_parser


[docs]def main():
    """
    Run this code to get the solution to 185th problem in Project Euler.

    Terminal options:

    .. option:: -m <sequence_length>, --sequence_length <sequence_length>

       Length of sequence in the riddle. Assume file 'data/input_m.txt' exists.

    """
    parser = OptionParser()
    parser.add_option("-m", "--sequence_length",
                      help="Length of sequence in the riddle. Assume file 'data/input_m.txt' exists.")
    options, _ = parser.parse_args()

    m = int(options.sequence_length)
    sequences_list, n_correct_vertices_list = input_parser(m)

    ilp_manager = ILPManager(m)
    ilp_solver, s_star_to_color_edges = ilp_manager.build_ilp_solver(sequences_list, n_correct_vertices_list)
    s_star = ilp_manager.solve_ilp(ilp_solver, s_star_to_color_edges)

    print "Solution is: {}".format(''.join([str(i) for i in s_star]))


if __name__ == '__main__':
    main()
```

Solver Runner

_solver.main() [SOURCE]

et the solution to 185th problem in Project Euler.

_length>, --sequence_length <sequence_length>

uence in the riddle. Assume file 'data/input_m.txt' exists.

Part I - Overview

Euler's 185th Riddle

In case you are inter

Let s^* be a se
We are given
For each sequ
Our goal is to

Source code for run_euler_185_solver

```
from optparse import OptionParser

from src.ilp_manager import ILPManager
from src.input_parser import input_parser


[docs]def main():
    """
    Run this code to get the solution to 185th problem in Project Euler.

    Terminal options:

    .. option:: -m <sequence_length>, --sequence_length <sequence_length>

       Length of sequence in the riddle. Assume file 'data/input_m.txt' exis

    """
    parser = OptionParser()
    parser.add_option("-m", "--sequence_length",
                      help="Length of sequence in the riddle. Assume file '
    options, _ = parser.parse_args()

    m = int(options.sequence_length)
    sequences_list, n_correct_vertices_list = input_parser(m)

    ilp_manager = ILPManager(m)
    ilp_solver, s_star_to_color_edges = ilp_manager.build_ilp_solver(sequenc
    s_star = ilp_manager.solve_ilp(ilp_solver, s_star_to_color_edges)

    print "Solution is: {}".format(''.join([str(i) for i in s_star]))


if __name__ == '__main__':
    main()
```

Solver Runner

class src.ilp_manager.ILPManager(*m*) [SOURCE]

Model the 185th problem in Project Euler as an ILP (Integer Linear Program)
To instantiate this module, please specify the length on sequences.

build_ilp_solver(*sequences_list, n_correct_vertices_list*) [SOURCE]

Given input sequences, and number or correct vertices in each of them, build an ILP representation of the problem.

Parameters:

- **sequences_list** – List of sequences, each of length self.m, of integers between 0 and 9.
- **n_correct_vertices_list** – Number of correct vertices in each sequence in sequences_list. A vertex is correct if its color is equal to the color of the corresponding vertex in the solution s_star.

Returns:

tuple, containing:

- ilp_solver: Pulp instance, holding all the information needed for the solution.
- s_star_to_color_edges: The edges (variables) in the ilp_solver.

solve_ilp(*ilp_solver, s_star_to_color_edges*) [SOURCE]

Given a solver with the needed information, solve the ILP and extract the solution to problem 185.

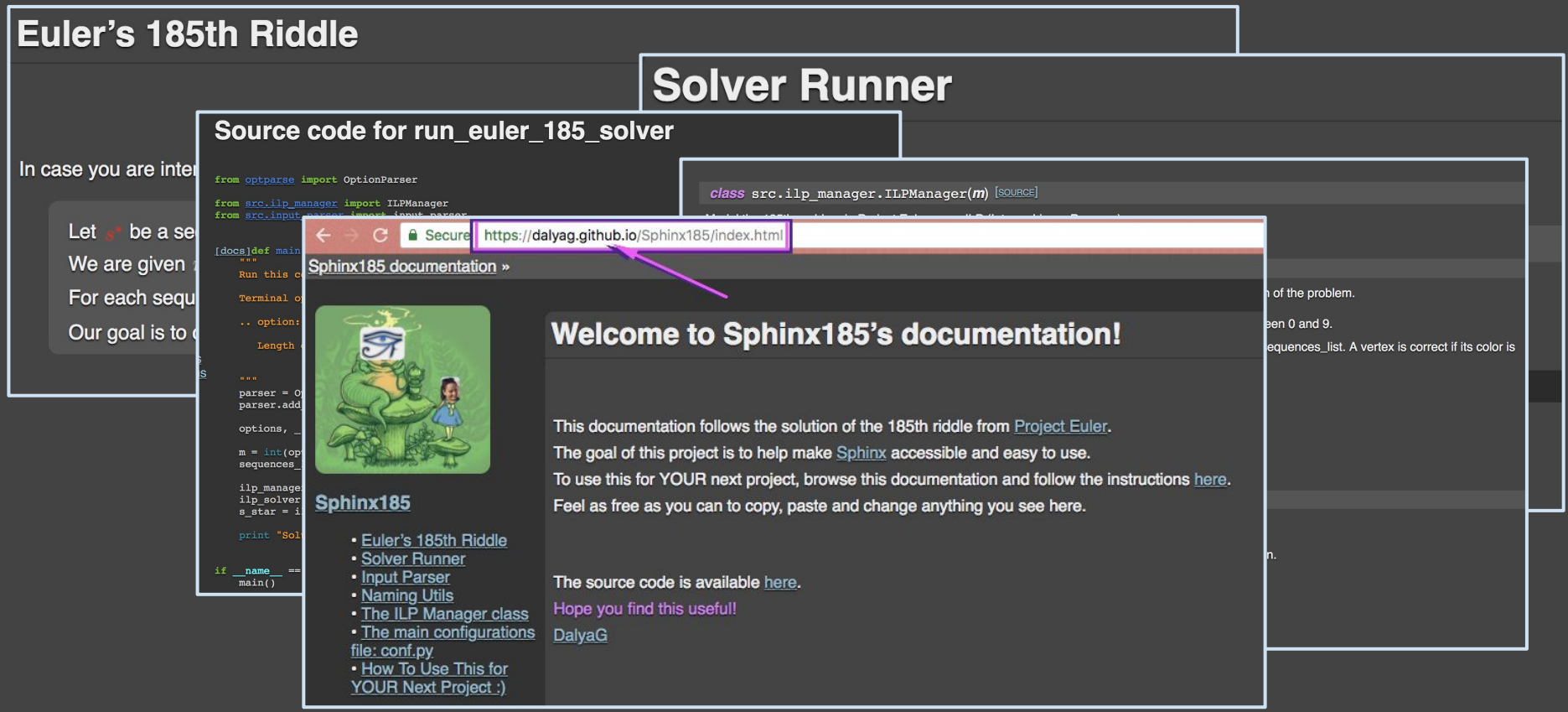
Parameters:

- **ilp_solver** – Pulp instance, holding all the information needed for the solution.
- **s_star_to_color_edges** – The edges (variables) in the ilp_solver.

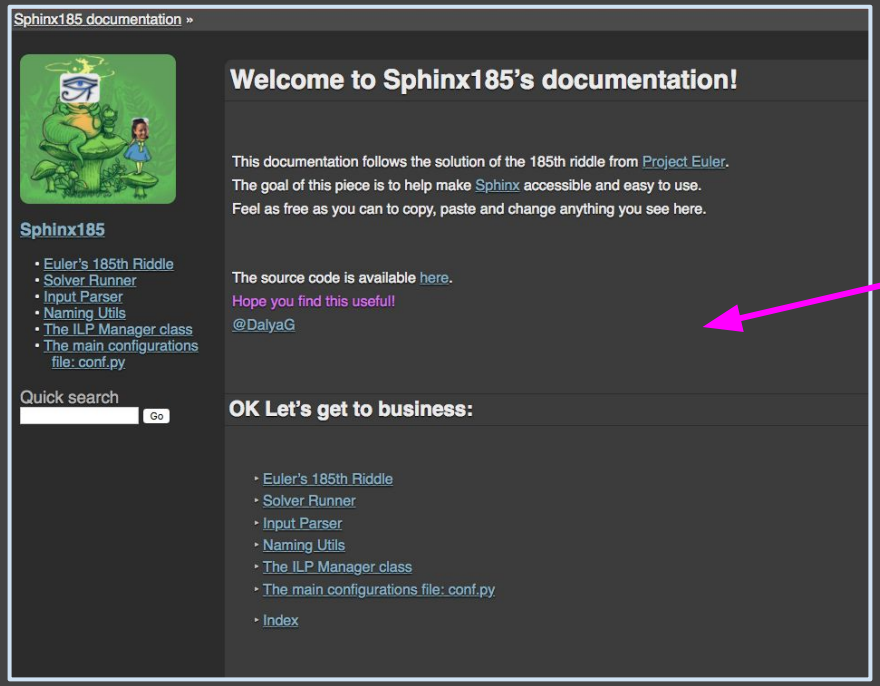
Returns:

s_star: List of integers that is the solution to problem 185.

Part I - Overview



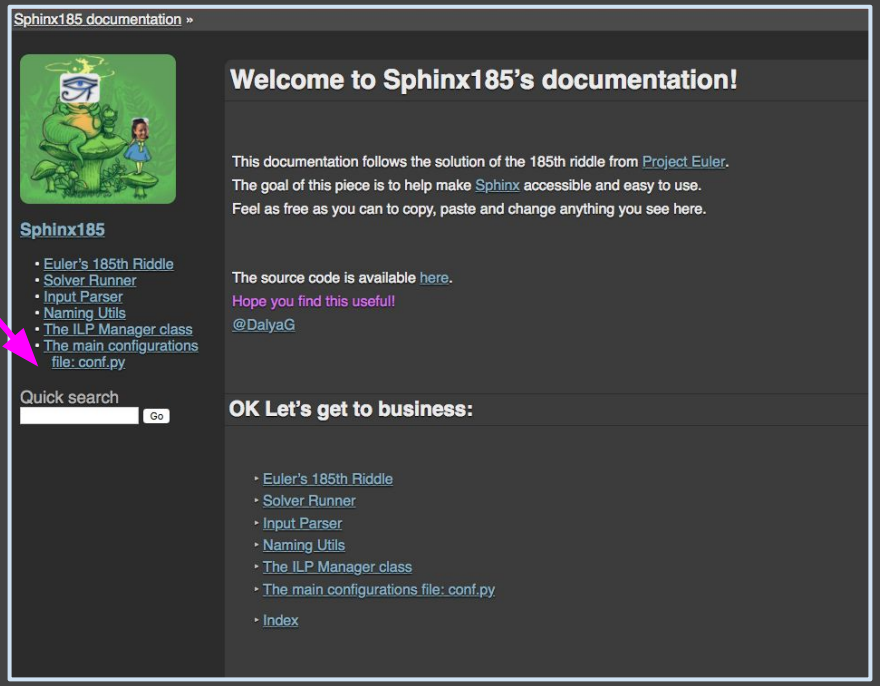
Part I - Overview



index.rst - the content of the home page

Part I - Overview

conf.py - settings for the entire project



Part II - Main configuration file - conf.py

Part II - Main configuration file - conf.py

```
"""

This is the main file in which the configuration for the documentation is made.

"""

# -*- coding: utf-8 -*-
#
# Sphinx185 documentation build configuration file, created by
# sphinx-quickstart on Sat May 12 19:34:31 2018.

# DalyaG: This file was heavily modified from its original build.
# Hope you find this useful :)


# -- Define here your working directory ------------------------------------------------------------------------------

import os
import sys
sys.path.append(os.path.abspath('/Users/dalya/Documents/Sphinx185'))


# -- Some general info  about the project ----------------------------------------------------------------------------

project = u'Sphinx185'
copyright = u'2018, DalyaG'
author = u'DalyaG'


# -- A few basic configurations --------------------------------------------------------------------------------------

# The documentation in this project will be mostly generated from .rst files
# In This project, every auto-documented module/class has its own .rst file, under the main documentation dir,
#   which is rendered into an .html page.
# Get more information here: http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html
source_suffix = ['.rst']

# This is the name of the main page of the project.
# It means that you need to have an `index.rst` file where you will design the landing page of your project.
# It will be rendered into an .html page that you can find at: `_build/html/index.html`
# (this is a standard name. change it only if you know what you are doing)
master_doc = 'index'

# List of patterns, relative to source directory, that match files and
# directories to ignore when looking for source files.
# This patterns also effect to html_static_path and html_extra_path
exclude_patterns = ['_build']

# List here any paths that contain templates, relative to this directory.
# You can find some not-so-intuitive information here: http://www.sphinx-doc.org/en/master/templating.html
# But the best way to learn is by example, right? :)
# So, for example, in this project, I wanted to change the title of the Table Of Contents in the sidebar.
#   So I copied `<Sphinx install dir>/themes/basic/globaltoc.html` into the `_templates` folder,
#      and replaced 'Table of Content' with 'Sphinx185'.
templates_path = ['_templates']
```

```
# -- Define and configure non-default extensions ---------------------------------------------------------------------

# You can find a list of available extension here: http://www.sphinx-doc.org/en/master/extensions.html
extensions = ['sphinx.ext.todo', 'sphinx.ext.viewcode', 'sphinx.ext.autodoc', 'sphinx.ext.imgmath']

# Above extensions explanation and configurations:

# todo: When you use the syntax ".. todo:: some todo comment" in your docstring,
#         it will appear in a highlighted box in the documentation.
#       In order for this extension to work, make sure you include the following:
todo_include_todos = True

# viewcode: Next to each function/module in the documentation, you will have an internal link to the source code.
#           The source code will have colors defined by the Pygments (syntax highlighting) style.
#           You can checkout the available pygments here: https://help.farbox.com/pygments.html
pygments_style = 'native'

# autodoc: The best thing about Sphinx IMHO is autodoc.
#          It allows Sphinx to automatically generate documentation for the docstrings in your code.
#          Get more info here: http://www.sphinx-doc.org/en/master/ext/autodoc.html
# Some useful configurations:
autoclass_content = "both"  # Include both the class's and the init's docstrings.
autodoc_member_order = 'bysource'  # In the documentation, keep the same order of members as in the code.
autodoc_default_flags = ['members']  # Default: include the docstrings of all the class/module members.

# imgmath: Sphinx allows use of LaTeX in the html documentation, but not directly. It is first rendered to an image.
# You can add here whatever preamble you are used to adding to your LaTeX document.
imgmath_latex_preamble = r'''
\usepackage{xcolor}
\definecolor{offwhite}{rgb}{238,238,238}
\everymath{\color{offwhite}}
\everydisplay{\color{offwhite}}
'''


# -- Options for HTML output -----------------------------------------------------------------------------------------

# The theme to use for HTML and HTML Help pages.
# You can find available themes here: http://www.sphinx-doc.org/en/master/theming.html
# In this project, I wanted to use a non-default theme, and so I downloaded the `graphite` template from here:
#    https://github.com/Cartroo/sphinx-theme-graphite
#    Some adjustments I made to graphite:
#    - I did not use the pygment configuration, and so removed "pygments_style = graphite.GraphiteStyle" from theme.conf
#                                                              and deleted graphite.py
#    - In the static folder, I configured several classes both in graphite.css and in html4css1.css,
#          you can download the original and compare to find those changes.
html_theme = 'graphite'
# When using a non-built-in theme, define the path to your template code.
html_theme_path = ['.']

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']

# Defining the static path allows me to add my own logo for the project:
#   (make sure the theme of your choice support the use of logo.
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictionary that maps document names to template names.
# In This project I chose to include in the sidebar:
#   - Table of Contents: I chose globaltoc as it is less refined,
#     and configured the title by editing the globaltoc template (see explanation above, in the templates_path comment)
#   - Search box: appears below the TOc, and can be configured by editing css attributes.
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

Part] nf.py

Part II - Main configuration file - conf.py

Full details - EXTRA SLIDES

Part II - Main configuration file - conf.py

'sphinx.ext.autodoc'

```
#        It will appear in a highlighted box in the documentation.
#       In order for this extension to work, make sure you include the following:
todo_include_todos = True

# viewcode: Next to each function/module in the documentation, you will have an internal link to the source code.
#           The source code will have colors defined by the Pygments (syntax highlighting) style.
#           You can checkout the available pygments here: https://help.farbox.com/pygments.html
pygments_style = 'native'

# autodoc: The best thing about Sphinx IMHO is autodoc.
#          It allows Sphinx to automatically generate documentation for the docstrings in your code.
#         Get more info here: http://www.sphinx-doc.org/en/master/ext/autodoc.html
# Some useful configurations:
autoclass_content = "both"  # Include both the class's and the init's docstrings.
autodoc_member_order = 'bysource'  # In the documentation, keep the same order of members as in the code.
autodoc_default_flags = ['members']  # Default: include the docstrings of all the class/module members.

# imgmath: Sphinx allows use of LaTeX in the html documentation, but not directly. It is first rendered to an image.
# You can add here whatever preamble you are used to adding to your LaTeX document.
imgmath_latex_preamble = r'''
\usepackage{xcolor}
\definecolor{offwhite}{rgb}{238,238,238}
\everymath{\color{offwhite}}
\everydisplay{\color{offwhite}}
'''


# -- Options for HTML output ----------------------------------------------------------------------------------

# The theme to use for HTML and HTML Help pages.
# You can find available themes here: http://www.sphinx-doc.org/en/master/theming.html
# In this project, I wanted to use a non-default theme, and so I downloaded the `graphite` template from here:
#    https://github.com/Cartroo/sphinx-theme-graphite
#    Some adjustments I made to graphite:
#    - I did not use the pygment configuration, and so removed "pygments_style = graphite.GraphiteStyle" from theme.conf
#                                                              and deleted graphite.py
#    - In the static folder, I configured several classes both in graphite.css and in html4css1.css,
#          you can download the original and compare to find those changes.
html_theme = 'graphite'
# When using a non-built-in theme, define the path to your template code.
html_theme_path = ['.']

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']

# Defining the static path allows me to add my own logo for the project:
#   (make sure the theme of your choice support the use of logo.
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictionary that maps document names to template names.
# In This project I chose to include in the sidebar:
#   - Table of Contents: I chose globaltoc as it is less refined,
#     and configured the title by editing the globaltoc template (see explanation above, in the templates_path comment)
#   - Search box: appears below the TOc, and can be configured by editing css attributes.
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

Part II - Main configuration file - conf.py

```
'sphinx.ext.autodoc'
```

```
.. autoclass:: src.ilp_manager.ILPManager
```

```
#       It will appear in a highlighted box in the documentation.
#       In order for this extension to work, make sure you include the following:
todo_include_todos = True

                                    module in the documentation, you will have an internal link to the source code.
                                    ave colors defined by the Pygments (syntax highlighting) style.
                                    vailable pygments here: https://help.farbox.com/pygments.html


                                    hinx IMHO is autodoc.
                                    omatically generate documentation for the docstrings in your code.
                                    p://www.sphinx-doc.org/en/master/ext/autodoc.html

                                    ude both the class's and the init's docstrings.
                                    # In the documentation, keep the same order of members as in the code.
                                    ] # Default: include the docstrings of all the class/module members.

# imgmath: Sphinx allows use of LaTeX in the html documentation, but not directly. It is first rendered to an image.
# You can add here whatever preamble you are used to adding to your LaTeX document.
imgmath_latex_preamble = r'''
\usepackage{xcolor}
\definecolor{offwhite}{rgb}{238,238,238}
\everymath{\color{offwhite}}
\everydisplay{\color{offwhite}}
'''


# -- Options for HTML output ----------------------------------------------------------------------------------------

# The theme to use for HTML and HTML Help pages.
# You can find available themes here: http://www.sphinx-doc.org/en/master/theming.html
# In this project, I wanted to use a non-default theme, and so I downloaded the `graphite` template from here:
#    https://github.com/Cartroo/sphinx-theme-graphite
#    Some adjustments I made to graphite:
#    - I did not use the pygment configuration, and so removed "pygments_style = graphite.GraphiteStyle" from theme.conf
#                                                              and deleted graphite.py
#    - In the static folder, I configured several classes both in graphite.css and in html4css1.css,
#          you can download the original and compare to find those changes.
html_theme = 'graphite'
# When using a non-built-in theme, define the path to your template code.
html_theme_path = ['.']

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']

# Defining the static path allows me to add my own logo for the project:
#   (make sure the theme of your choice support the use of logo.
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictionary that maps document names to template names.
# In This project I chose to include in the sidebar:
#   - Table of Contents: I chose globaltoc as it is less refined,
#     and configured the title by editing the globaltoc template (see explanation above, in the templates_path comment)
#   - Search box: appears below the TOc, and can be configured by editing css attributes.
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

Part II - Main configuration file - conf.py

```
'sphinx.ext.autodoc'
```

```
.. autoclass:: src.ilp_manager.ILPManager
```

class `src.ilp_manager.ILPManager`(***m***) [SOURCE]

Model the 185th problem in Project Euler as an ILP (Integer Linear Program)

To instantiate this module, please specify the length on sequences.

`build_ilp_solver`(***sequences_list, n_correct_vertices_list***) [SOURCE]

Given input sequences, and number or correct vertices in each of them, build an ILP representation of the problem.

Parameters:

- **sequences_list** – List of sequences, each of length self.m, of integers between 0 and 9.
- **n_correct_vertices_list** – Number of correct vertices in each sequence in sequences_list. A vertex is correct if its color is equal to the color of the corresponding vertex in the solution s_star.

Returns:

tuple, containing:

- ilp_solver: Pulp instance, holding all the information needed for the solution.
- s_star_to_color_edges: The edges (variables) in the ilp_solver.

`solve_ilp`(***ilp_solver, s_star_to_color_edges***) [SOURCE]

Given a solver with the needed information, solve the ILP and extract the solution to problem 185.

Parameters:

- **ilp_solver** – Pulp instance, holding all the information needed for the solution.
- **s_star_to_color_edges** – The edges (variables) in the ilp_solver.

Returns:

s_star: List of integers that is the solution to problem 185.

```
#        It will appear in a highlighted box in the documentation.
#       In order for this extension to work, make sure you include the following:
todo_include_todos = True

                                    module in the documentation, you will have an internal link to the source code.
                                    ave colors defined by the Pygments (syntax highlighting) style.
                                    vailable pygments here: https://help.farbox.com/pygments.html


                                    phinx IMHO is autodoc.
                                    omatically generate documentation for the docstrings in your code.
                                    p://www.sphinx-doc.org/en/master/ext/autodoc.html

                                    ude both the class's and the init's docstrings.
                                     # In the documentation, keep the same order of members as in the code.
                                    ]  # Default: include the docstrings of all the class/module members.

# imgmath: Sphinx allows use of LaTeX in the html documentation, but not directly. It is first rendered to an image.
# You can add here whatever preamble you are used to adding to your LaTeX document.
imgmath_latex_preamble = r'''
\usepackage{xcolor}
\definecolor{offwhite}{rgb}{238,238,238}
\everymath{\color{offwhite}}
\everydisplay{\color{offwhite}}
'''


# -- Options for HTML output ----------------------------------------------------------------------------------------

# The theme to use for HTML and HTML Help pages.
# You can find available themes here: http://www.sphinx-doc.org/en/master/theming.html
# In this project, I wanted to use a non-default theme, and so I downloaded the `graphite` template from here:
#    https://github.com/Cartroo/sphinx-theme-graphite
#    Some adjustments I made to graphite:
#    - I did not use the pygment configuration, and so removed "pygments_style = graphite.GraphiteStyle" from theme.conf
#                                                                                    and deleted graphite.py
#    - In the static folder, I configured several classes both in graphite.css and in html4css1.css,
#          you can download the original and compare to find those changes.
html_theme = 'graphite'
# When using a non-built-in theme, define the path to your template code.
html_theme_path = ['.']

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']

# Defining the static path allows me to add my own logo for the project:
#   (make sure the theme of your choice support the use of logo.
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictionary that maps document names to template names.
# In This project I chose to include in the sidebar:
#   - Table of Contents: I chose globaltoc as it is less refined,
#     and configured the title by editing the globaltoc template (see explanation above, in the templates_path comment)
#   - Search box: appears below the TOc, and can be configured by editing css attributes.
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

Part II - Main configuration file - conf.py

'sphinx.ext.autodoc'

Full details here:

http://www.sphinx-doc.org/en/master/ext/autodoc.html

```
#        It will appear in a highlighted box in the documentation.
#       In order for this extension to work, make sure you include the following:
todo_include_todos = True

# viewcode: Next to each function/module in the documentation, you will have an internal link to the source code.
#           The source code will have colors defined by the Pygments (syntax highlighting) style.
#           You can checkout the available pygments here: https://help.farbox.com/pygments.html
pygments_style = 'native'

# autodoc: The best thing about Sphinx IMHO is autodoc.
#          It allows Sphinx to automatically generate documentation for the docstrings in your code.
#          Get more info here: http://www.sphinx-doc.org/en/master/ext/autodoc.html
# Some useful configurations:
autoclass_content = "both"  # Include both the class's and the init's docstrings.
autodoc_member_order = 'bysource'  # In the documentation, keep the same order of members as in the code.
autodoc_default_flags = ['members']  # Default: include the docstrings of all the class/module members.

# imgmath: Sphinx allows use of LaTeX in the html documentation, but not directly. It is first rendered to an image.
# You can add here whatever preamble you are used to adding to your LaTeX document.
imgmath_latex_preamble = r'''
\usepackage{xcolor}
\definecolor{offwhite}{rgb}{238,238,238}
\everymath{\color{offwhite}}
\everydisplay{\color{offwhite}}
'''


# -- Options for HTML output ----------------------------------------------------------------------------------

# The theme to use for HTML and HTML Help pages.
# You can find available themes here: http://www.sphinx-doc.org/en/master/theming.html
# In this project, I wanted to use a non-default theme, and so I downloaded the `graphite` template from here:
#    https://github.com/Cartroo/sphinx-theme-graphite
#    Some adjustments I made to graphite:
#    - I did not use the pygment configuration, and so removed "pygments_style = graphite.GraphiteStyle" from theme.conf
#                                                              and deleted graphite.py
#    - In the static folder, I configured several classes both in graphite.css and in html4css1.css,
#          you can download the original and compare to find those changes.
html_theme = 'graphite'
# When using a non-built-in theme, define the path to your template code.
html_theme_path = ['.']

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']

# Defining the static path allows me to add my own logo for the project:
#   (make sure the theme of your choice support the use of logo.
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictionary that maps document names to template names.
# In This project I chose to include in the sidebar:
#   - Table of Contents: I chose globaltoc as it is less refined,
#     and configured the title by editing the globaltoc template (see explanation above, in the templates_path comment)
#   - Search box: appears below the TOc, and can be configured by editing css attributes.
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

Part III - Document YOUR next project

Part III - Document YOUR next project

Part III - Document YOUR next project

We will follow these instructions:

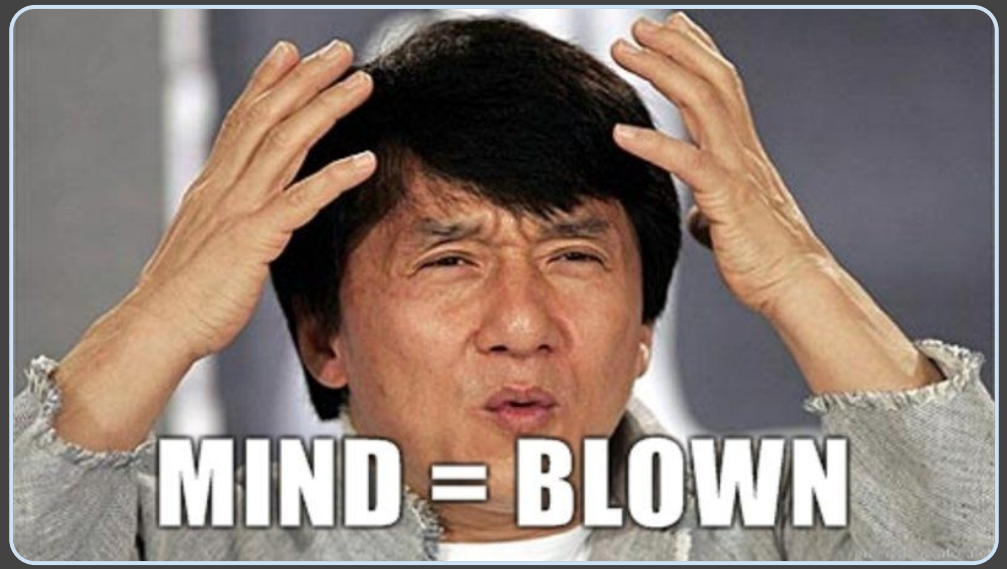
(found here https://dalyag.github.io/Sphinx185/how_to_use_this_for_your_next_project.html)

How To Use This for YOUR Next Project :)

1. `pip install sphinx`
2. Copy the `documentation_template_for_your_next_project` folder in this repository, make it a subdirectory in your local project and rename it `documentation`.
3. Edit `conf.py` by searching the pattern `#CHNAGEME#` inside the file and follow the instructions.
4. Edit `index.rst` by following the inline instructions.
5. To add a page to your documentation:
    a. Create a `.rst` file for the function/module you wish to document (you can use the templates supplied here), and
    b. Add the name of the `.rst` file to the `toctree` in `index.rst`.
6. In terminal, inside the `documentation` folder, run `make html`.
   (**TIP OF THE WEEK**: actually, always run `make clean html` to clear sphinx cache and build from scratch)
7. To view locally - open the file `documentation/_build/html/index.html` in your browser, and enjoy the read :)
8. To share - you can use GitHub Pages to host your documentation:
    a. Copy the content of `documentation/_build/html/` into a new `docs` folder, under the root of the project.
    b. Create an empty file `.nojekyll` inside `docs` folder (this tells GitHub Pages to bypass the default `jekyll` themes and use the `html` and `css` in your project)
    c. Push your changes to master branch.
    d. In your repository on GitHub, go to “Settings” -> “GitHub Pages” -> “Source” and select “master branch /docs folder”.
    e. Share your beautiful documentation site at `https://<your_git_usrname>.github.io/<project_name>/`

Part III - Document YOUR next project

1. pip install sphinx

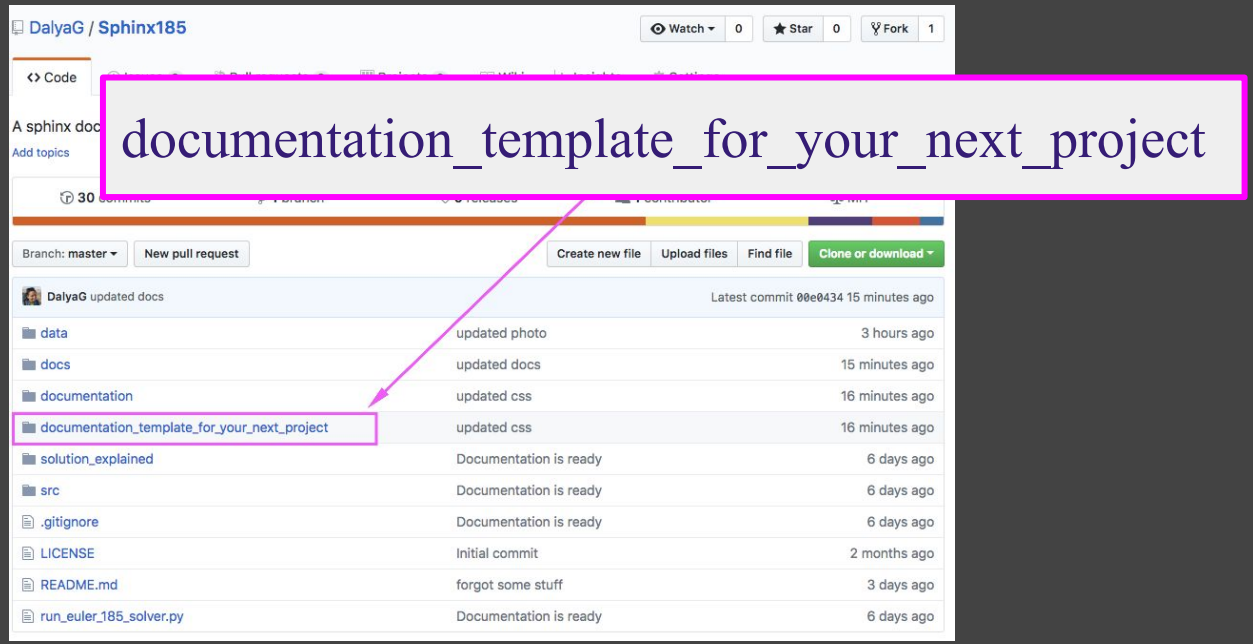


Part III - Document YOUR next project

2. Go to https://github.com/DalyaG/Sphinx185

Part III - Document YOUR next project

2. Go to https://github.com/DalyaG/Sphinx185



Part III - Document YOUR next project

2. Go to https://github.com/DalyaG/Sphinx185

* Download this folder to your local project: *documentation_template_for_your_next_project*

* Rename it *documentation*

Part III - Document YOUR next project

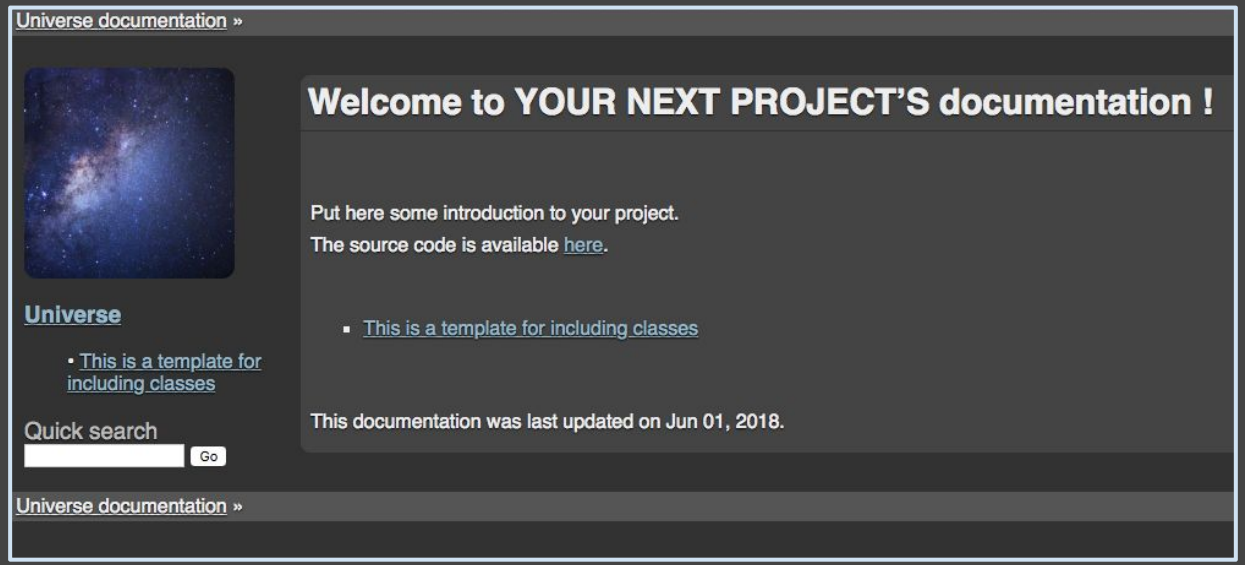
Bravo! You now have a documentation!

Part III - Document YOUR next project

Bravo! You now have a documentation!

Open this file in your browser to see:

<your_project>/documentation/_build/html/index.html



Part III - Document YOUR next project

3. Edit *conf.py*

Part III - Document YOUR next project

3. Edit *conf.py* by searching the pattern *#CHNAGEME#* inside the file and follow the instructions.

```
# -- Some general info  about the project -----

#CHNAGEME# Change this to fit your project.
project = u'Universe'
copyright = u'2018, DalyaG'
author = u'DalyaG'
```

Part III - Document YOUR next project

4. Edit *index.rst*

Part III - Document YOUR next project

4. Edit *index.rst* by following the inline instructions.

```
Welcome to YOUR NEXT PROJECT'S documentation !
==============================================

Put here some introduction to your project.

The source code is available `here <https://github.com/usrname/projectname>`_.
```

Part III - Document YOUR next project

5. To add a page to your documentation:

Part III - Document YOUR next project

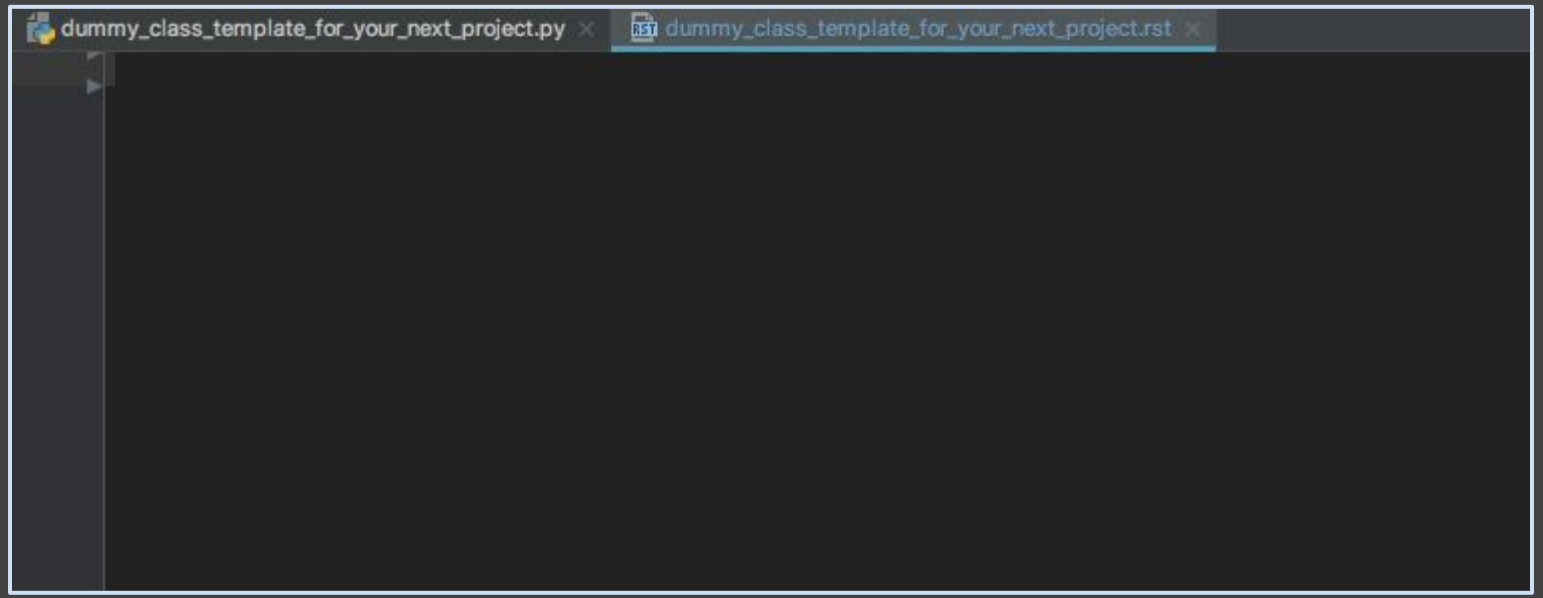
5. To add a page to your documentation:

dummy_class_template_for_your_next_project.py

```
class DummyClassTemplateForYourNextProject(object):
    """Some general info about this class."""
    def __init__(self):
        """Some info about the init method"""
        self.some_att = 42

    def answer_to_the_universe(self):
        """

        To make a "real" docstring leave blank lines before and after the text.

        :return: Prints the answer to the universe.
        """
        print "The answer to the universe is: {}".format(self.some_att)
```

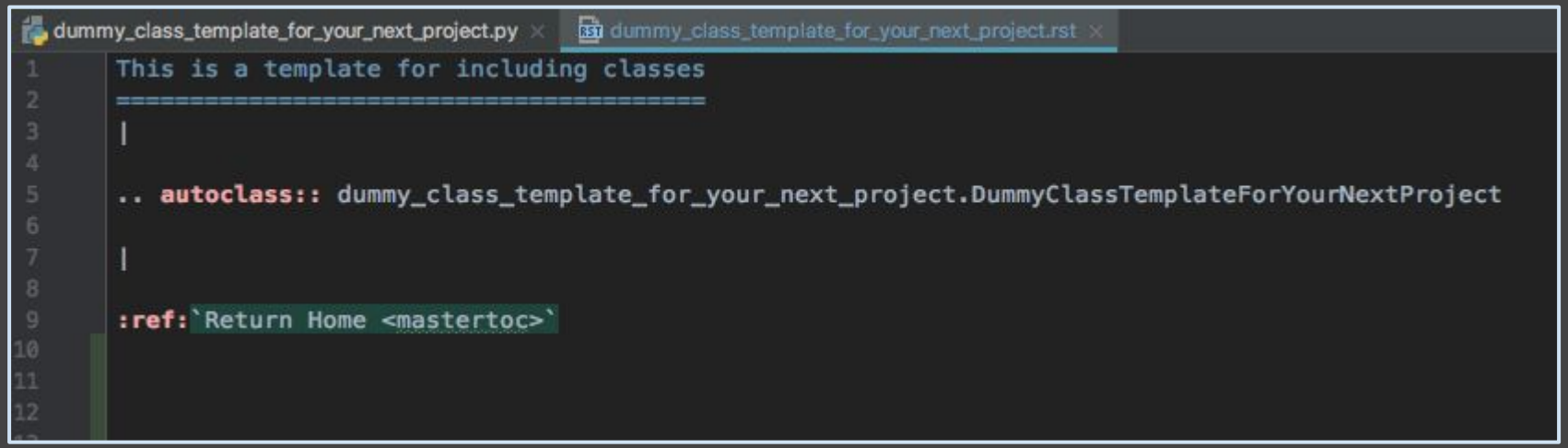
Part III - Document YOUR next project

5. To add a page to your documentation: Create a *.rst* file



Part III - Document YOUR next project

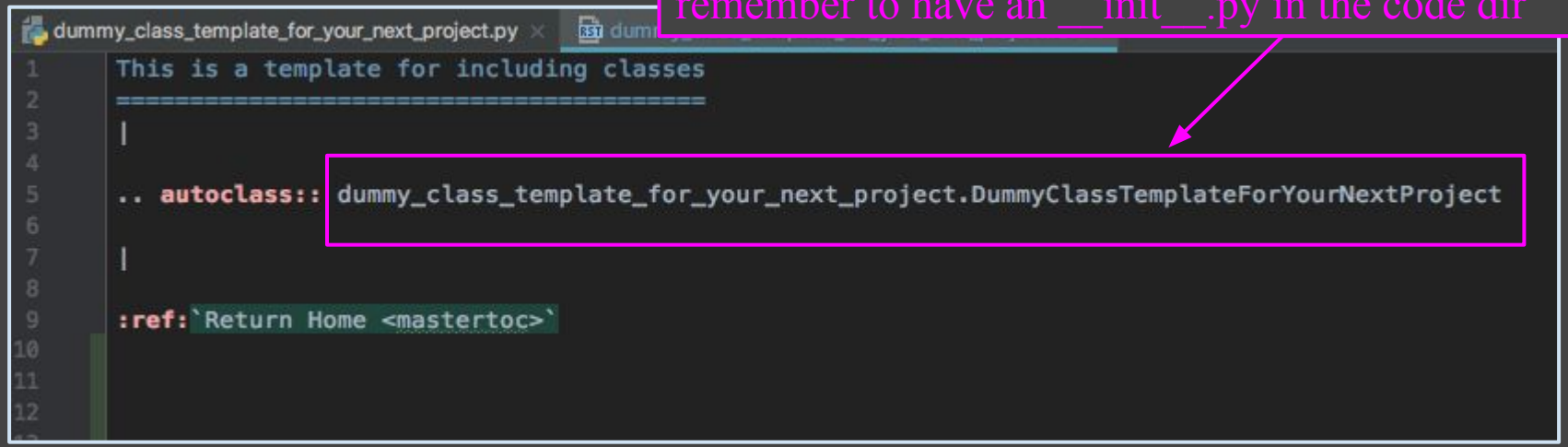
5. To add a page to your documentation:
Create a *.rst* file and edit is to include your module



Part III - Document YOUR next project

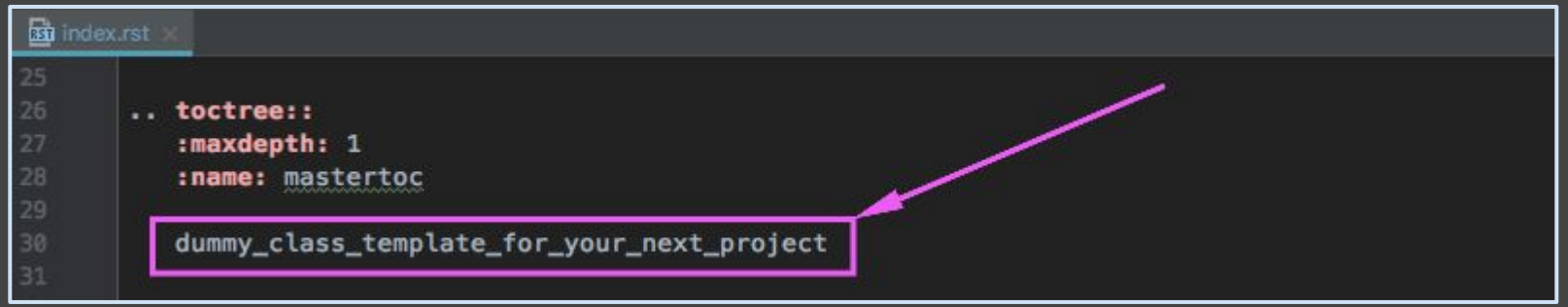
5. To add a page to your documentation:
Create a *.rst* file and edit is to include your module

Full path to the class +
remember to have an __init__.py in the code dir



Part III - Document YOUR next project

5. To add a page to your documentation:
   Create a *.rst* file and edit is to include your module
   And add its name to the *TOC* in *index.rst*

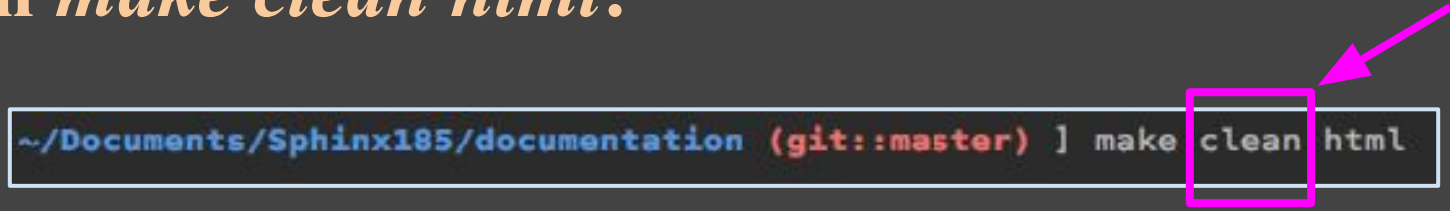


Part III - Document YOUR next project

6. In terminal, inside the *documentation* folder, run *make clean html*:

Part III - Document YOUR next project

6. In terminal, inside the *documentation* folder, run *make clean html*:



Part III - Document YOUR next project

6. In terminal, inside the *documentation* folder, run *make clean html*:

```
[ dalya@Dalya-Gartzman-Mackbook ~/Documents/Sphinx185/documentation (git::master) ] make clean html
Removing everything under '_build'...
Running Sphinx v1.6.6
making output directory...
loading pickled environment... not yet created
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 8 source files that are out of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [100%] run_euler_185_solver
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] run_euler_185_solver
generating indices... genindex py-modindex
highlighting module code... [100%] src.naming_utils
writing additional pages... search
copying static files... done
copying extra files... done
dumping search index in English (code: en) ... done
dumping object inventory... done
build succeeded.

Build finished. The HTML pages are in _build/html.
[ dalya@Dalya-Gartzman-Mackbook ~/Documents/Sphinx185/documentation (git::master) ]
```

Part III - Document YOUR next project

6. In terminal, inside the *documentation* folder, run *make clean html*:

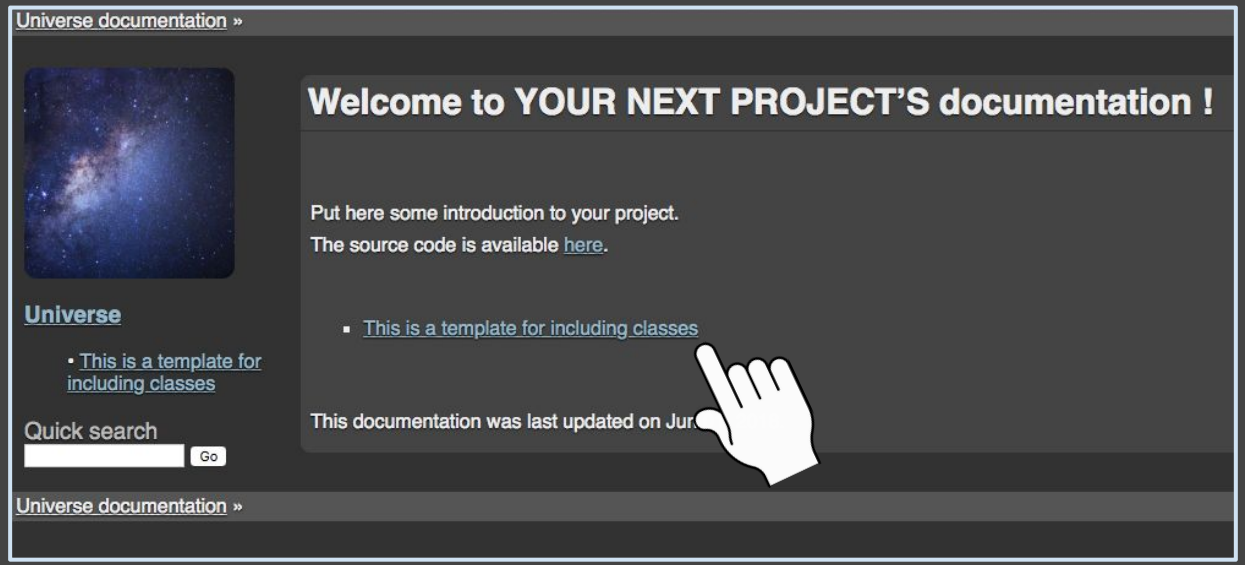
```
[ dalya@Dalya-Gartzman-Mackbook ~/Documents/Sphinx185/documentation (git::master) ] make clean html
Removing everything under '_build'...
Running Sphinx v1.6.6
making output directory...
loading pickled environment... not yet created
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 8 source files that are out of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [100%] run_euler_185_solver
/Users/dalya/Documents/Sphinx185/documentation/index.rst:9: WARNING: Line block ends without a blank line.
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] run_euler_185_solver
generating indices... genindex py-modindex
highlighting module code... [100%] src.naming_utils
writing additional pages... search
copying static files... done
copying extra files... done
dumping search index in English (code: en) ... done
dumping object inventory... done
build succeeded, 1 warning.

Build finished. The HTML pages are in _build/html.
[ dalya@Dalya-Gartzman-Mackbook ~/Documents/Sphinx185/documentation (git::master) ]
```

Part III - Document YOUR next project

7. To view locally

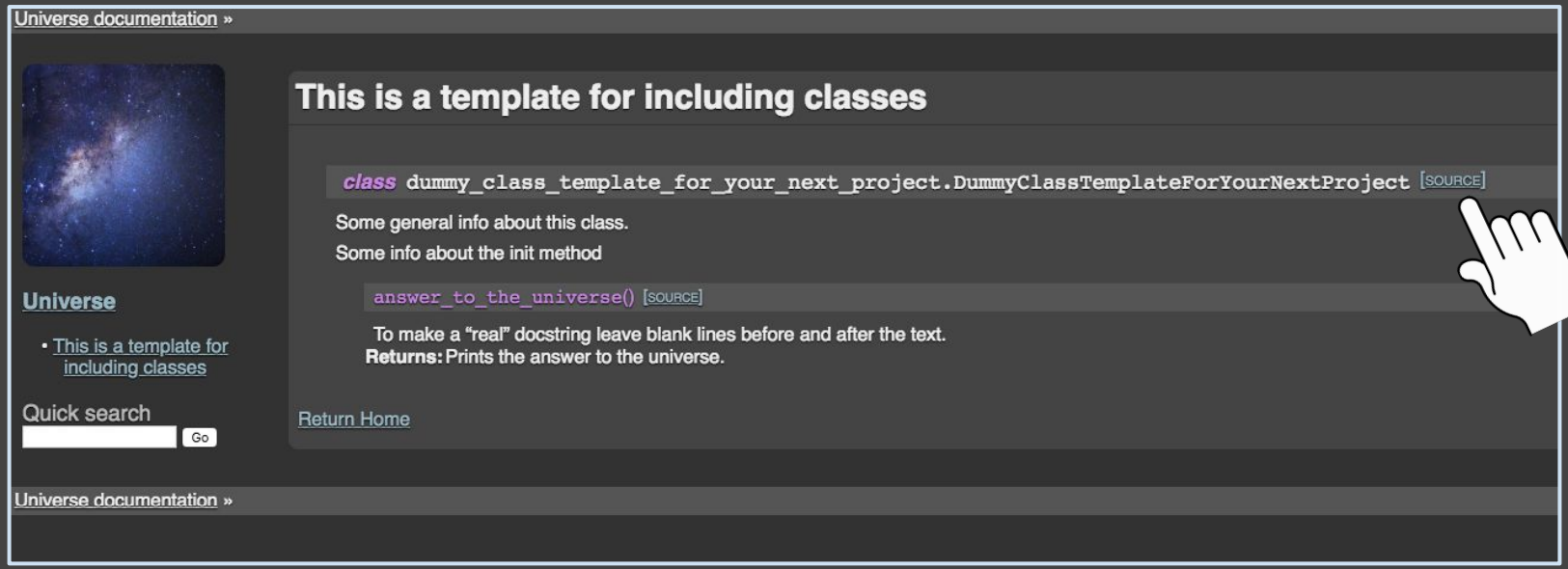
Part III - Document YOUR next project

7. To view locally, open this file in your browser *documentation/_build/html/index.html*



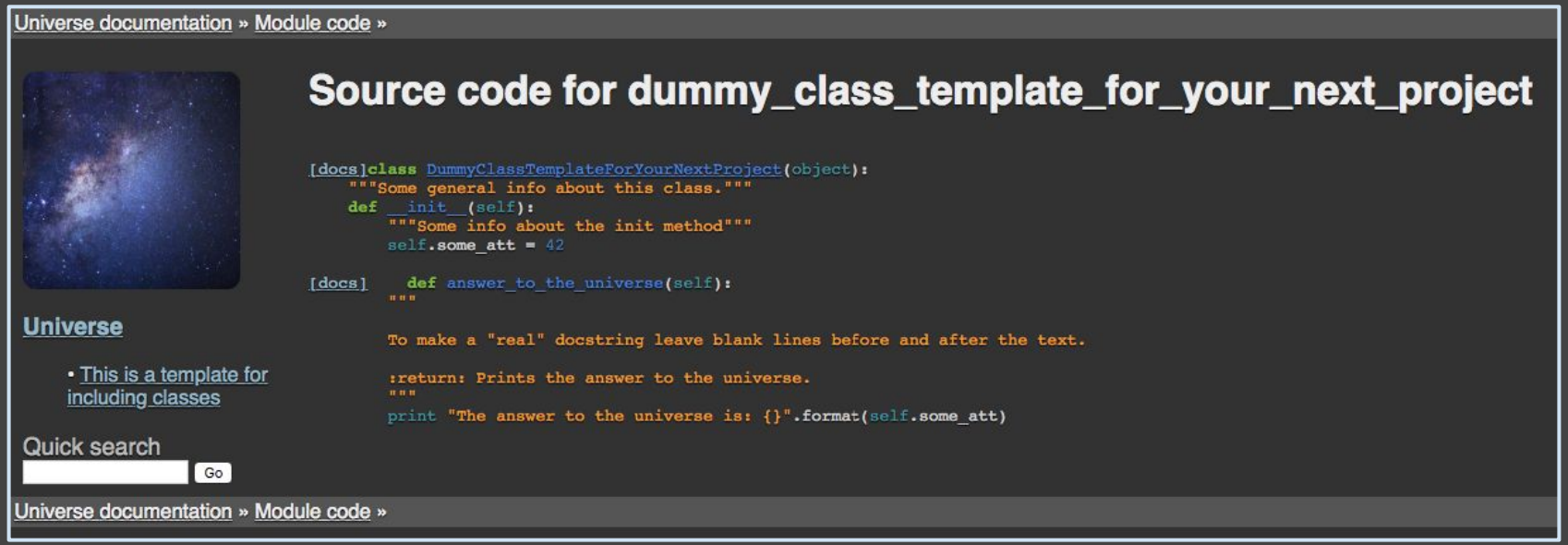
Part III - Document YOUR next project

7. To view locally, open this file in your browser *documentation/_build/html/index.html*



Part III - Document YOUR next project

7. To view locally, open this file in your browser *documentation/_build/html/index.html*



Part III - Document YOUR next project

7. To view loc browser
   documenta

Part III - Document YOUR next project

8. SHARE your documentation!

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

Part III - Document YOUR next project

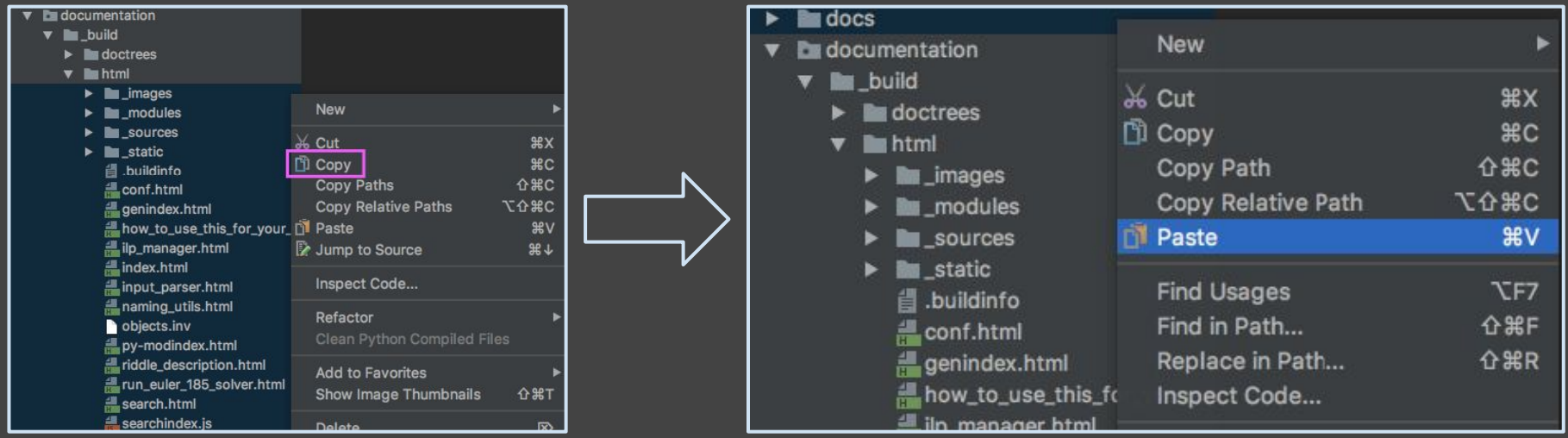
8. Hosting on GitHub Pages:

a. Create a folder *docs* under the root of the project

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

a. And copy inside it the content of *documentation/_build/html/*

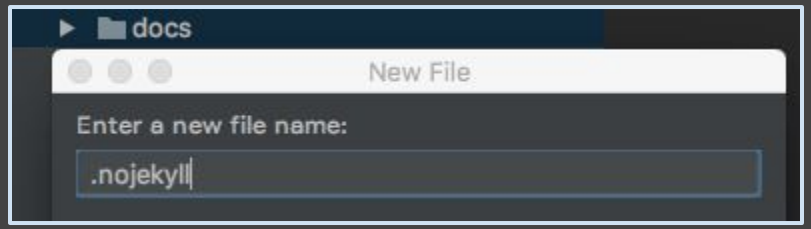


Part III - Document YOUR next project

8. Hosting on GitHub Pages:

b. Create an empty file *.nojekyll* inside *docs* folder

(this tells GitHub Pages to bypass the default jekyll themes and use the html and css in your project)



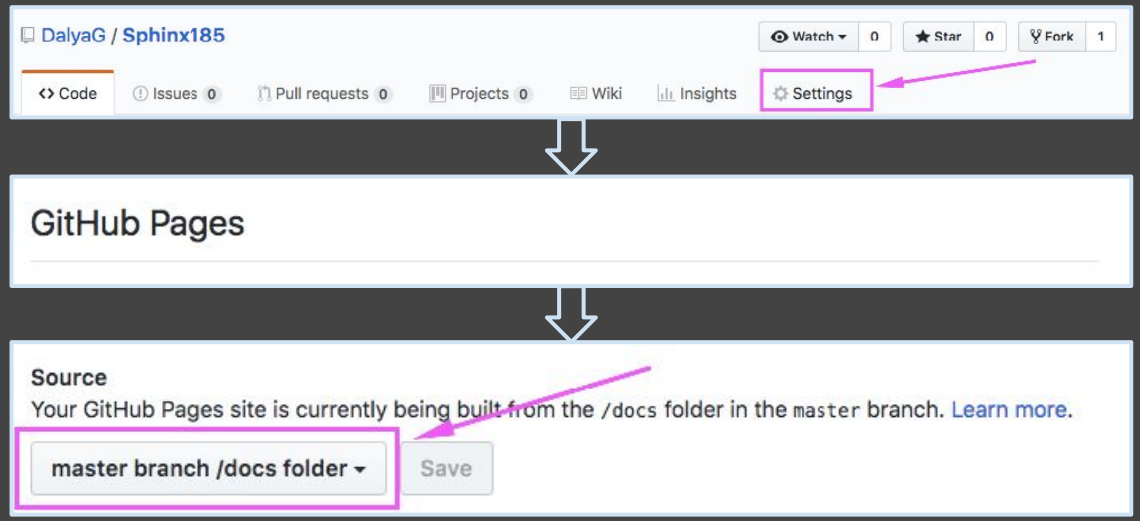
Part III - Document YOUR next project

8. Hosting on GitHub Pages:
    c. Push your changes to master branch.

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

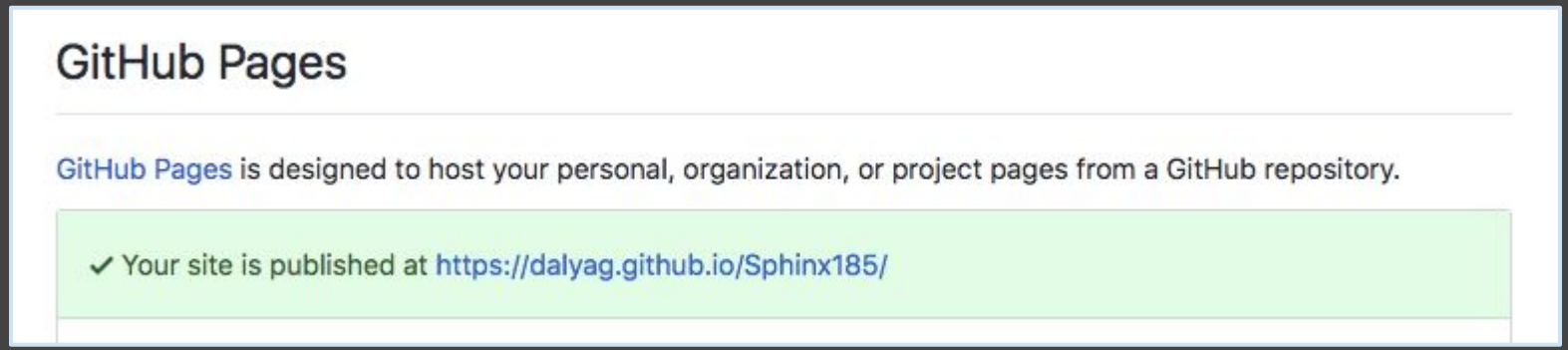
d. In your repository on GitHub, go to “Settings” -> “GitHub Pages” -> “Source”



Part III - Document YOUR next project

8. Hosting on GitHub Pages:

e. Share your beautiful documentation site at https://<your_git_usrname>.github.io/<project_name>/

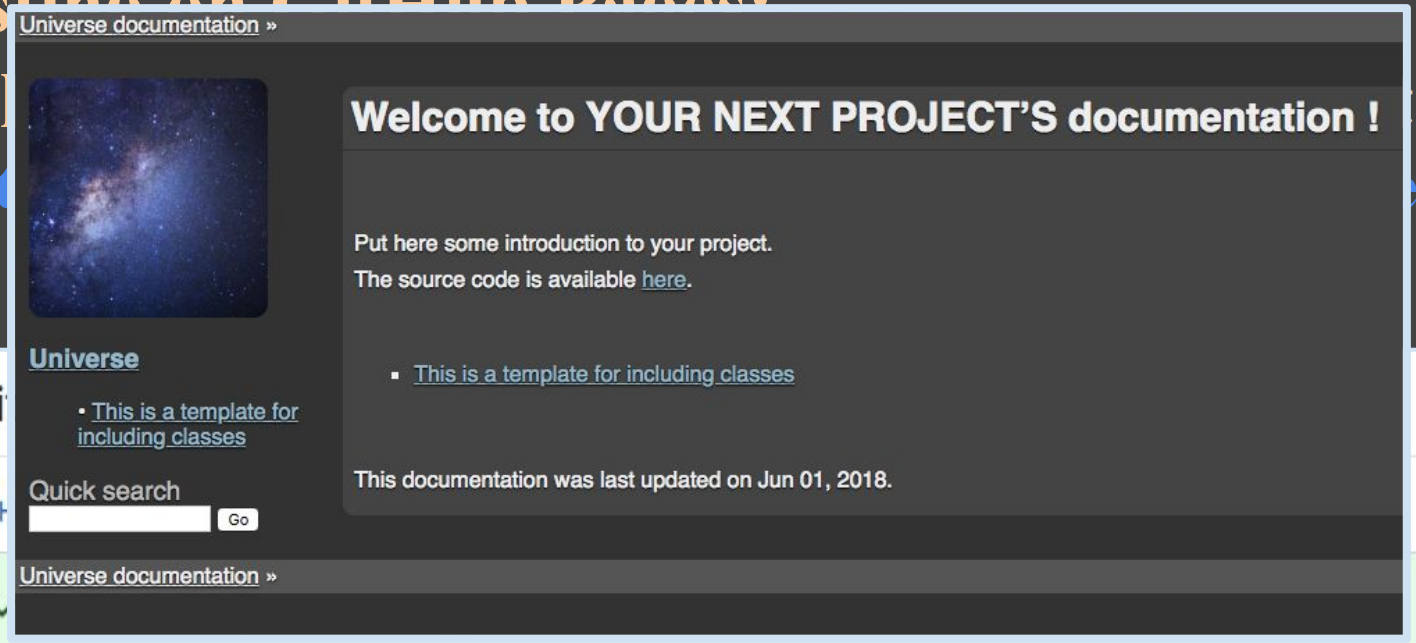


Part III - Document YOUR next project

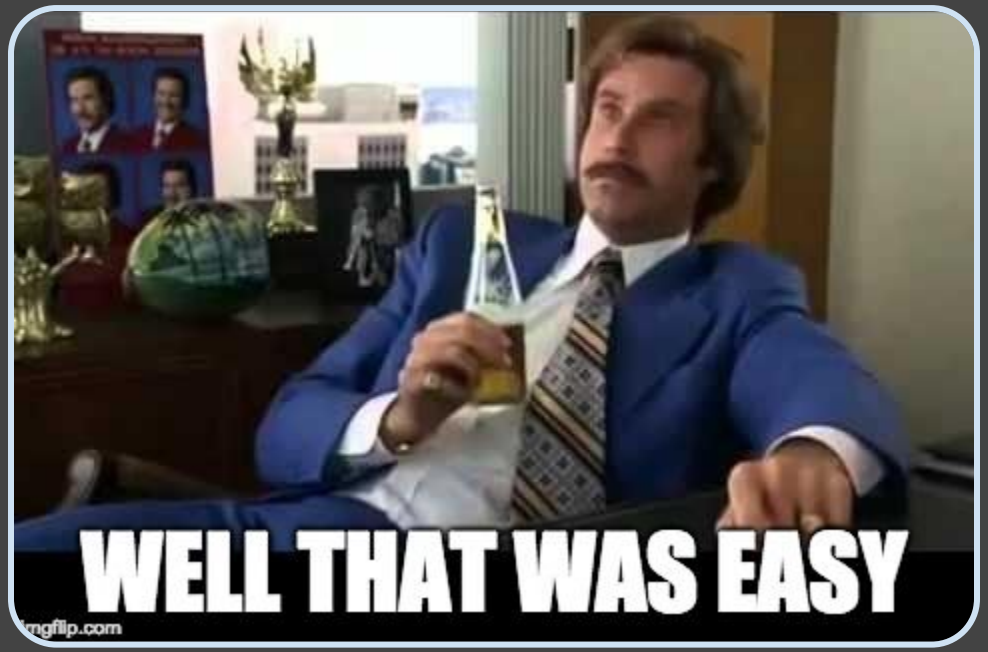
8. Hosting on GitHub Pages:

e. Sl

ht



Part III - Document YOUR next project

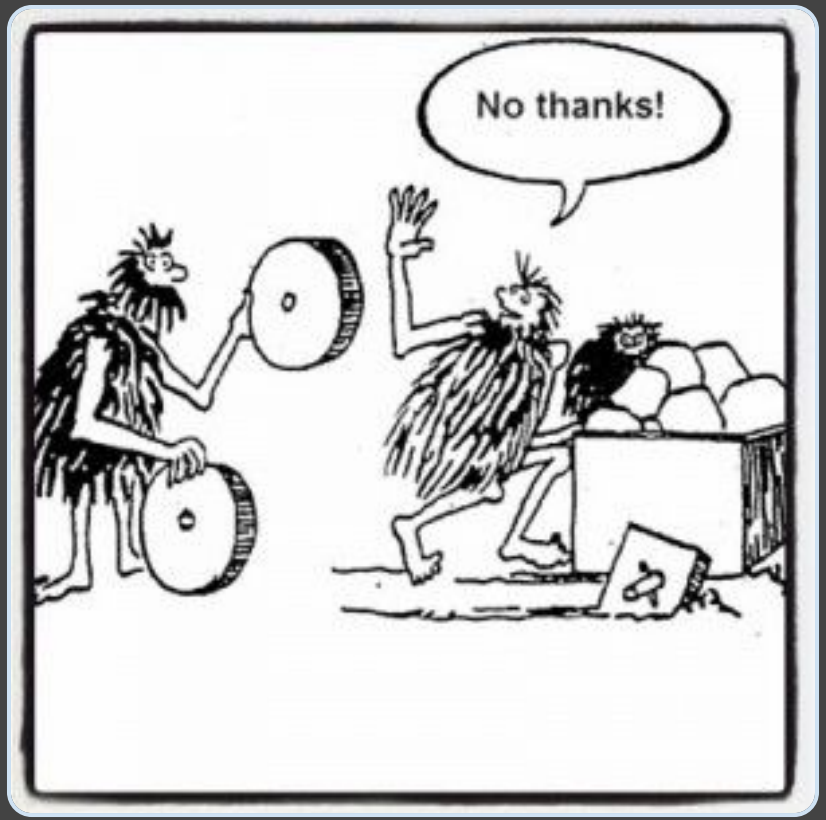


Part III - Recap

Part IV - Surprise! Live Demo!

github.com/DalyaG/Sphinx185

Take Home Message



Thank you :)

Questions?

DalyaG@gmail.com