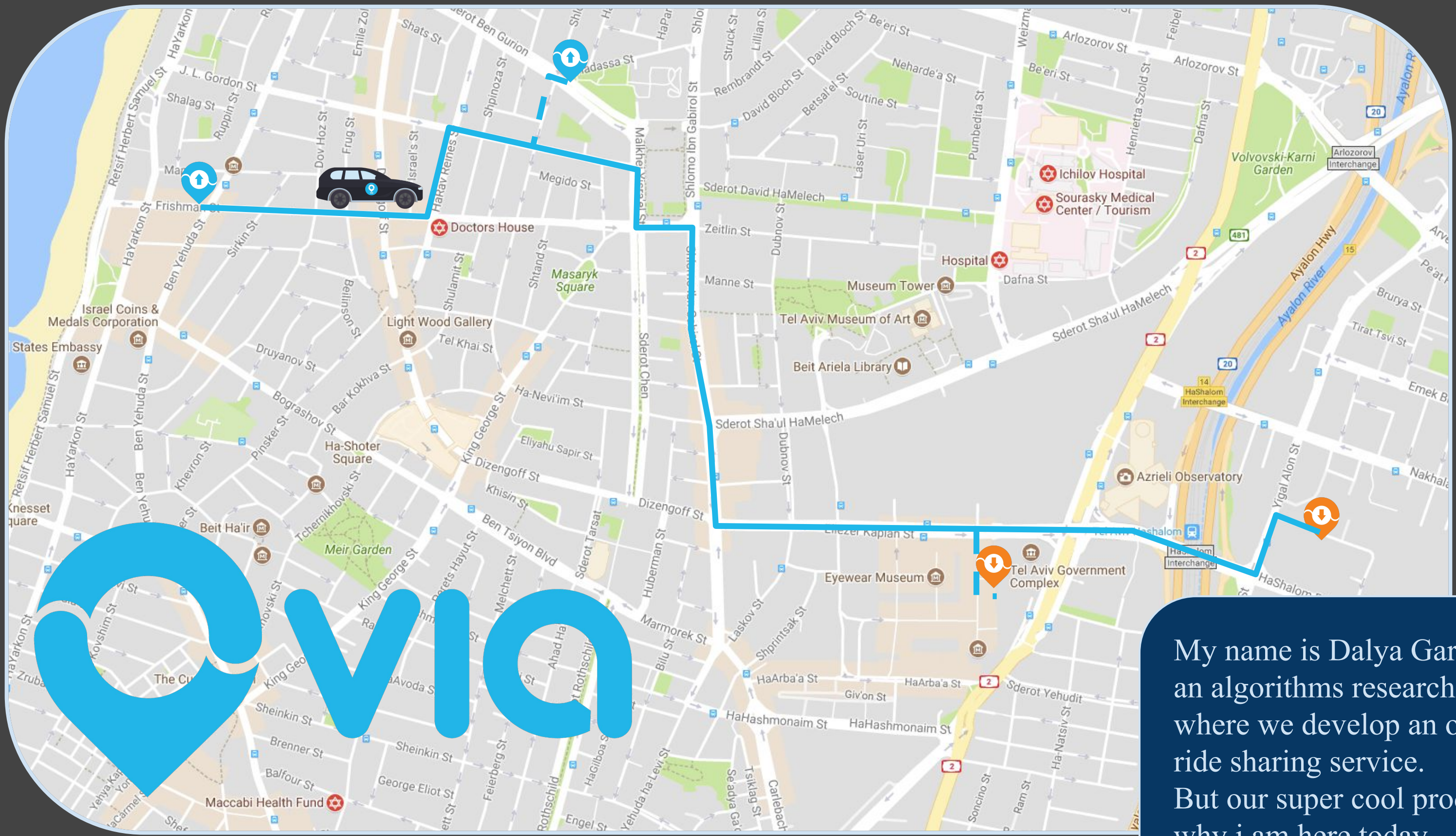
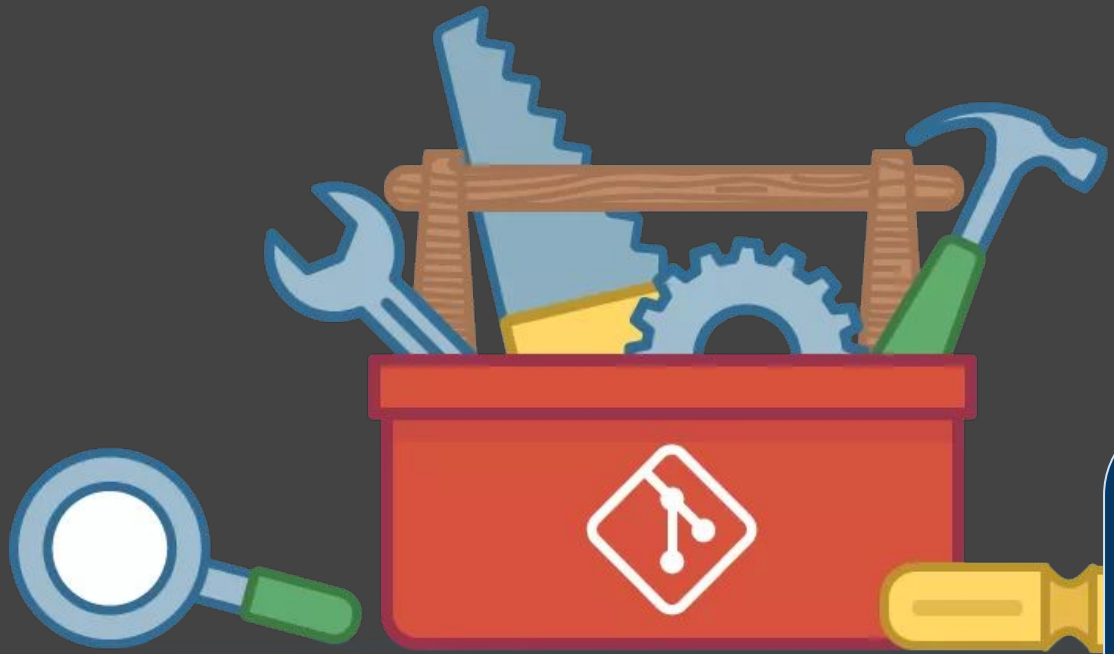
Help your colleagues help themselves - a Sphinx tutorial

Dalya Gartzman

github.com/DalyaG/Sphinx185

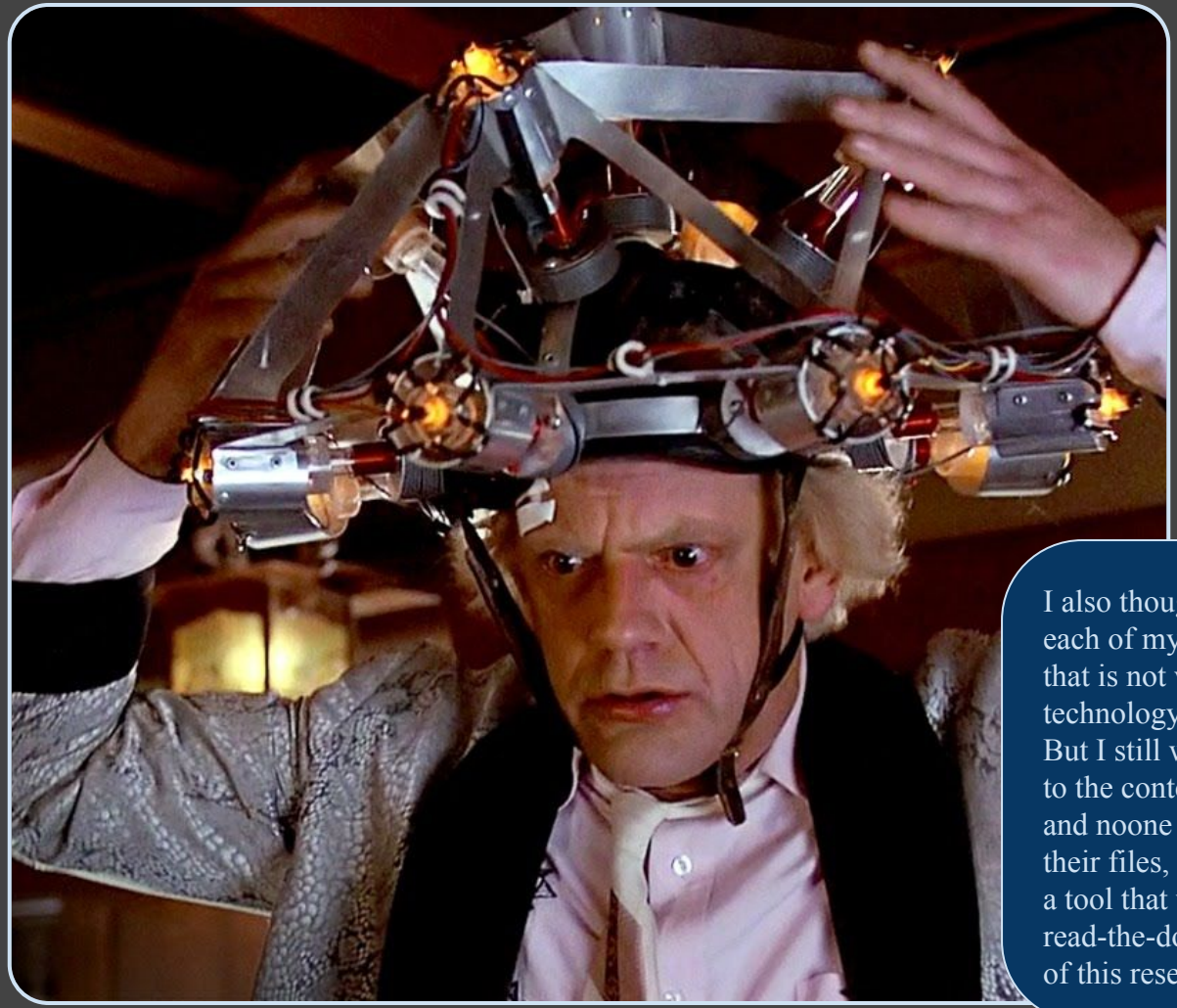


My name is Dalya Gartzman, I am an algorithms researcher in Via, where we develop an on demand ride sharing service.
But our super cool product is not why i am here today.



Im here because we have a really cool research repository that holds, well, our research, but also lots of scripts and modules that we write for ourselves but are for the common use.
And the thing is, that inorder to discover all the gems in this repo,

I would need to go over every file and read the docstring to understand whats in there, and i wasnt going to do that

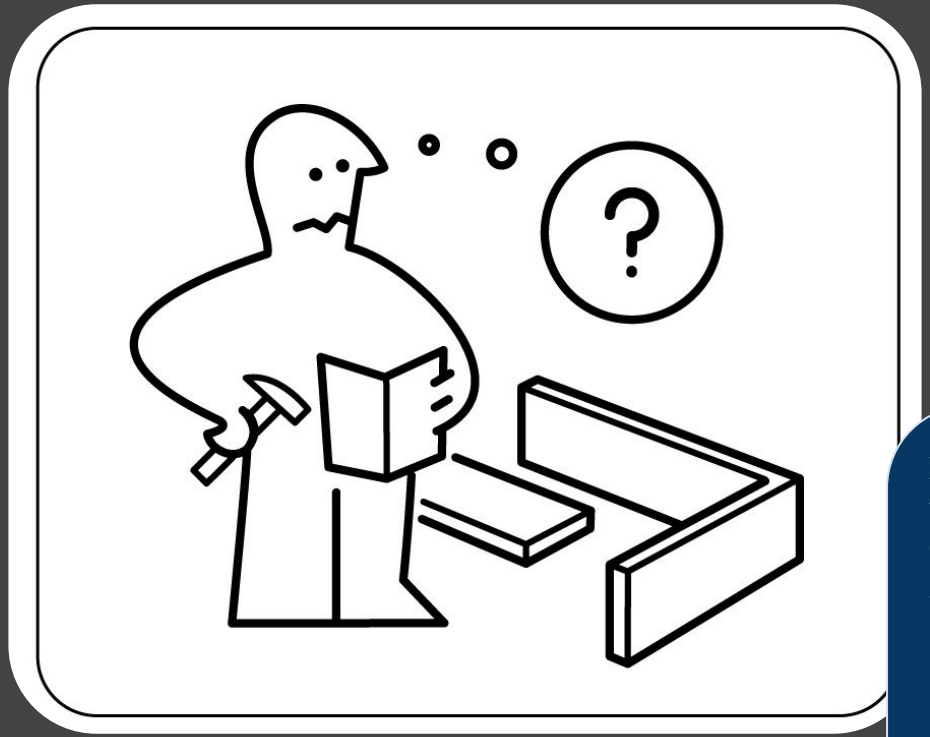


I also thought about browsing each of my colleagues brains, but that is not where the world is, technology-wise.
But I still wanted an easy access to the content of the repository, and noone was going to wiki all their files, so my goal was to find a tool that will help me make a read-the-docs style documentation of this research repository

And that’s how I stumbled upon Sphinx - that The Internet said will automagically do all the work for me, and it seemed really promissing

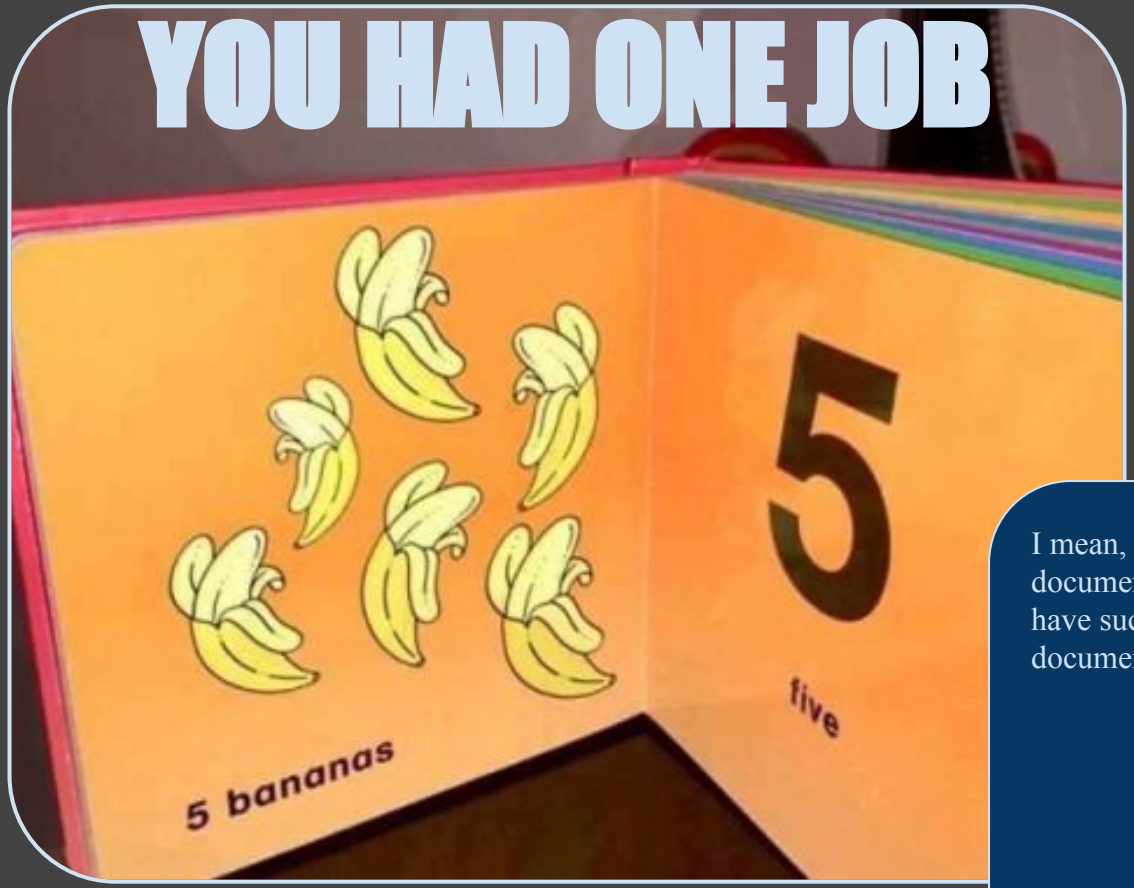
So i thought , hey, thats great, this package could be used also for documentation of inner colebarative projects, like the research repository i mentioned,

and also for a “read the docs”, or “api guide” for publishing your work publically.



But when i tried using it i found thats sooo not how it goes, and i couldnt figure out how it DOES go!!

It seems like one of those many cases i come across, where once i would already know the basics, then it will be a smooth ride, but the onboarding stage is so damn frustrating i nearly abandoned my mission



I mean, who would have thought a documentation package would have such terrible documentation??

So don’t worry, I didnt, and I really like sphinx now, and my goal today is to make this beautiful tool accessible, and to help you sail through this onboarding stage.

Press Start

Ready?

READY PLAYER ONE

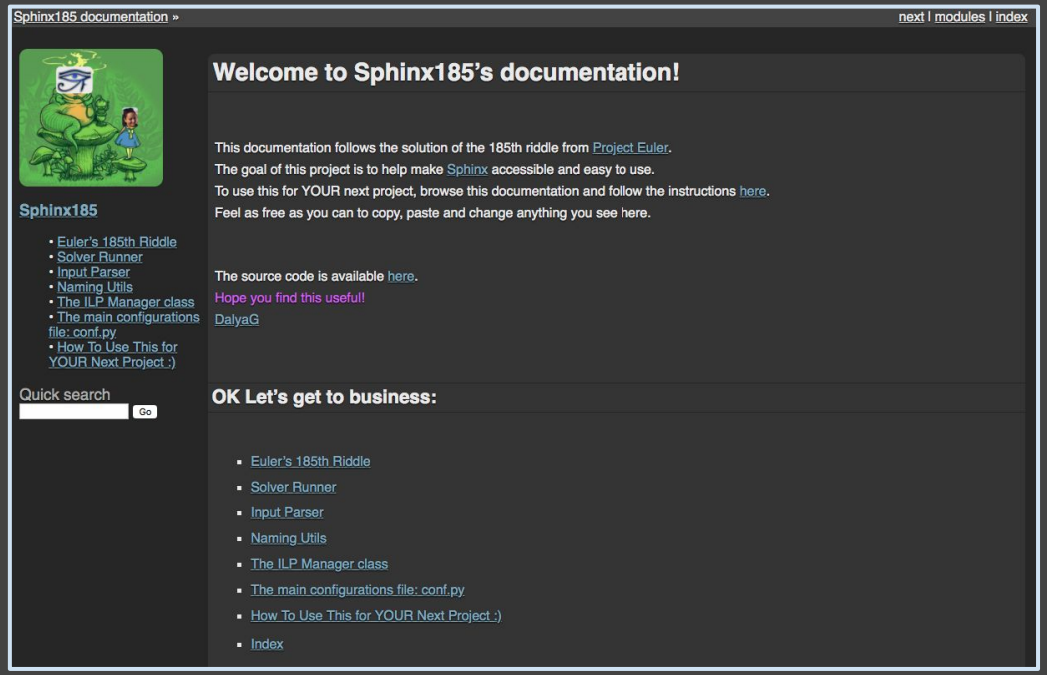
Press Start

Let’s start :)

Intro - What will we accomplish today?

So - what are we going to acoomplish today?

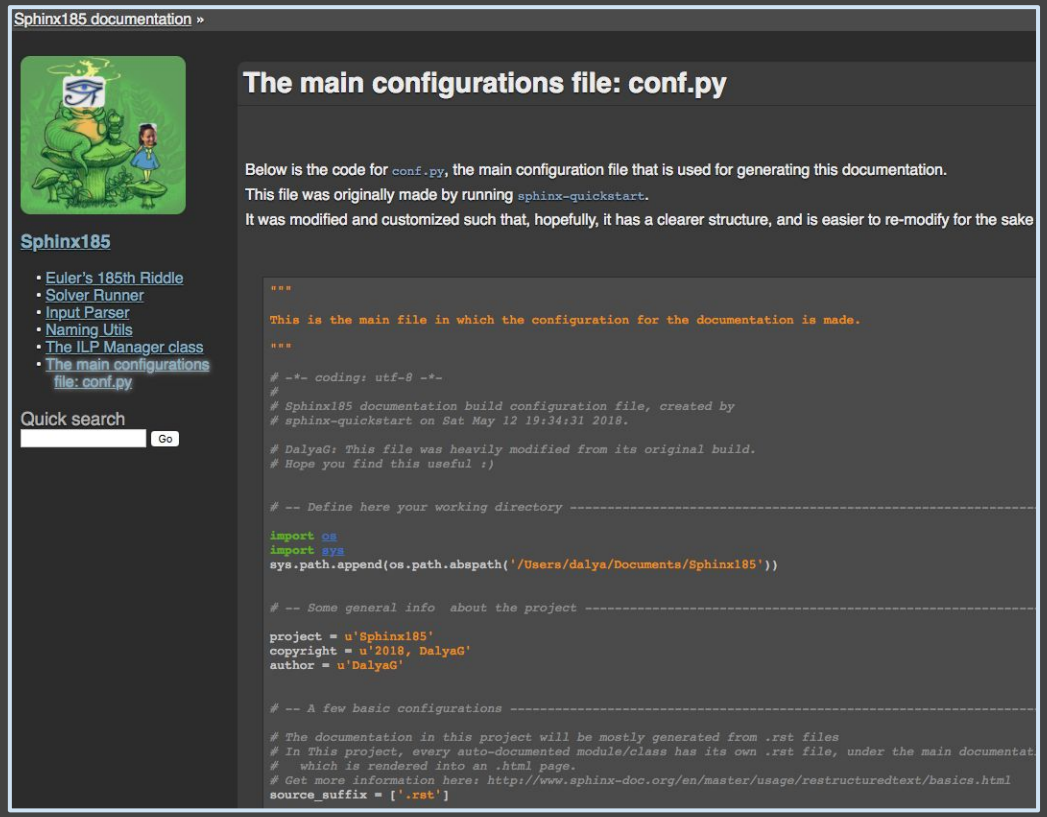
Intro - What will we accomplish today?



Part I - Overview

In the first part I will give an overview about what is sphinx and how it can be useful for you

Intro - What will we accomplish today?



Part II - Main configuration file - conf.py

Then I will give a breif note regarding an important configurations file, that will give us a background to the main part -

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>/`

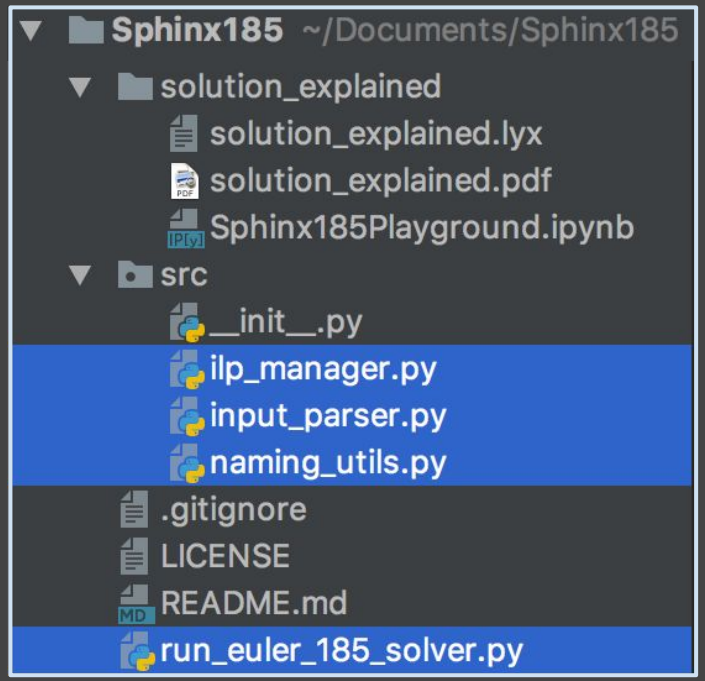
Part III - Document YOUR next project

In which I am going to walk you step-by-step through the process of creating a read-the-docs style documentation for YOUR next project. and we are going to do that using the repository I prepared for this talk.

Part I - Overview

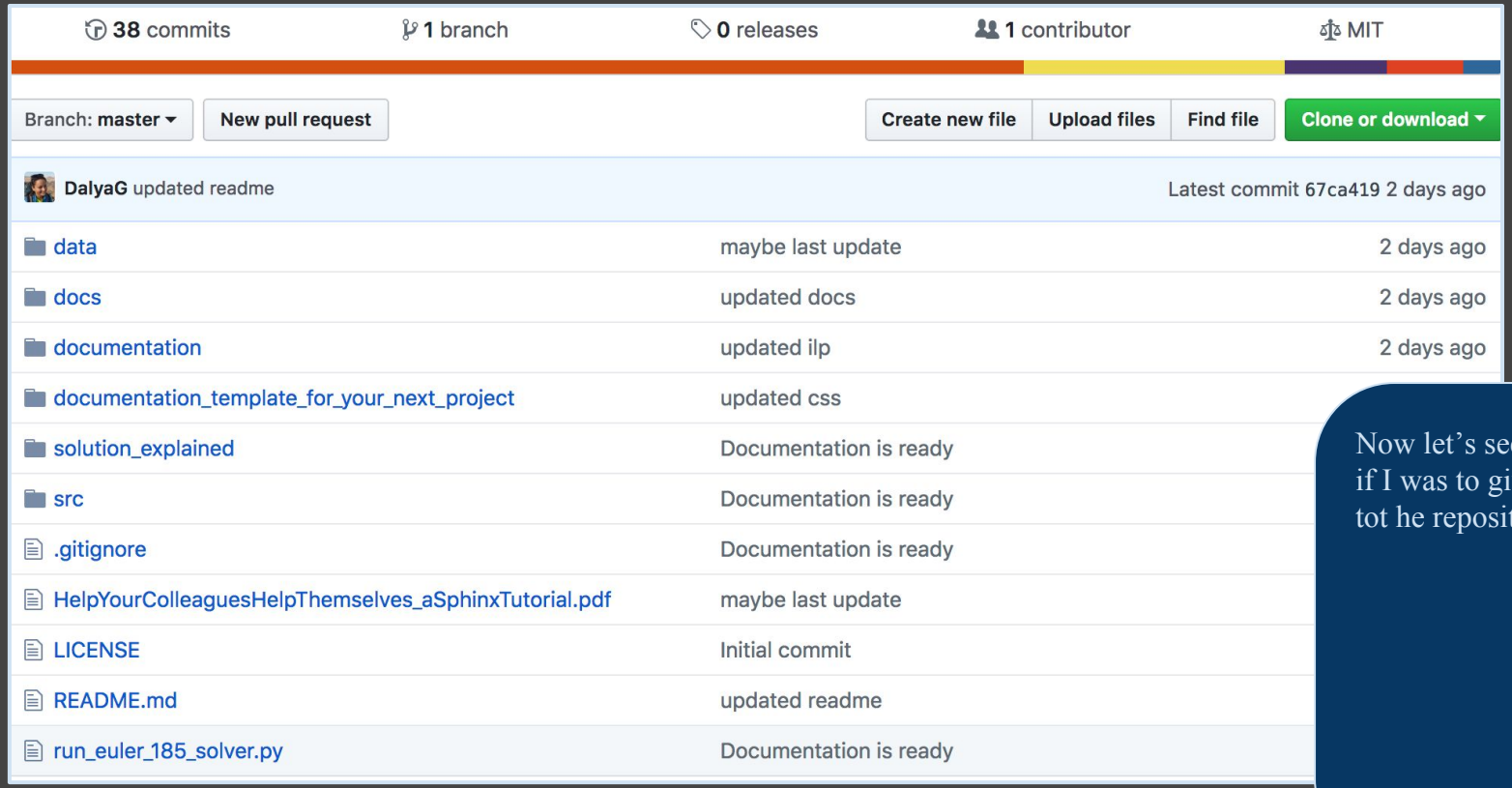
I will start with some background:

Part I - Overview



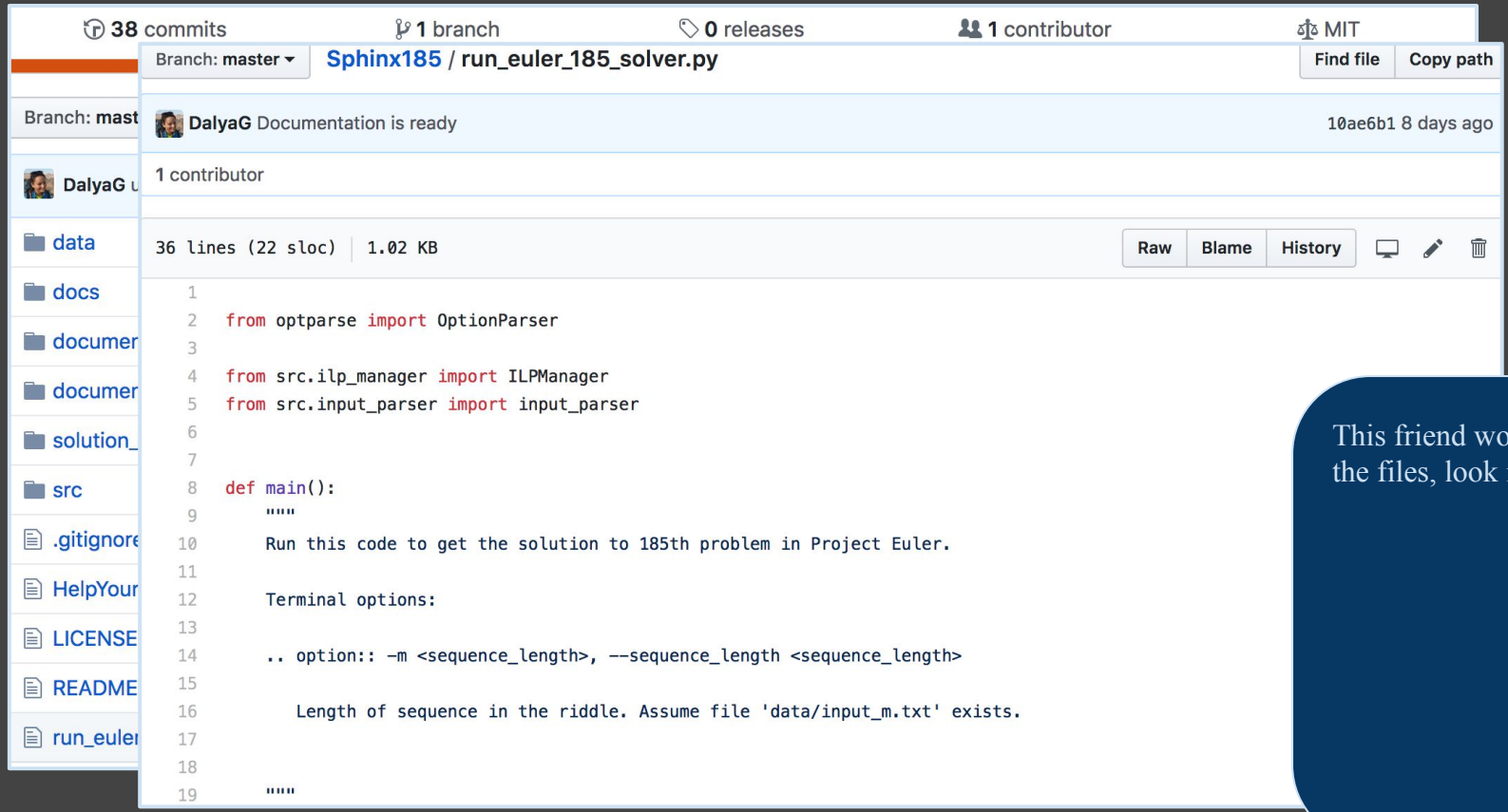
The setting is this: I took one riddle from the Euler's project website, (which is a website with mathematical riddles that you solve using code) and I want to show my friend my solution - there is some background material such as the playground jupyter notebook, which are less important, and I want to share the three components of the solution, and the runner function.

Part I - Overview



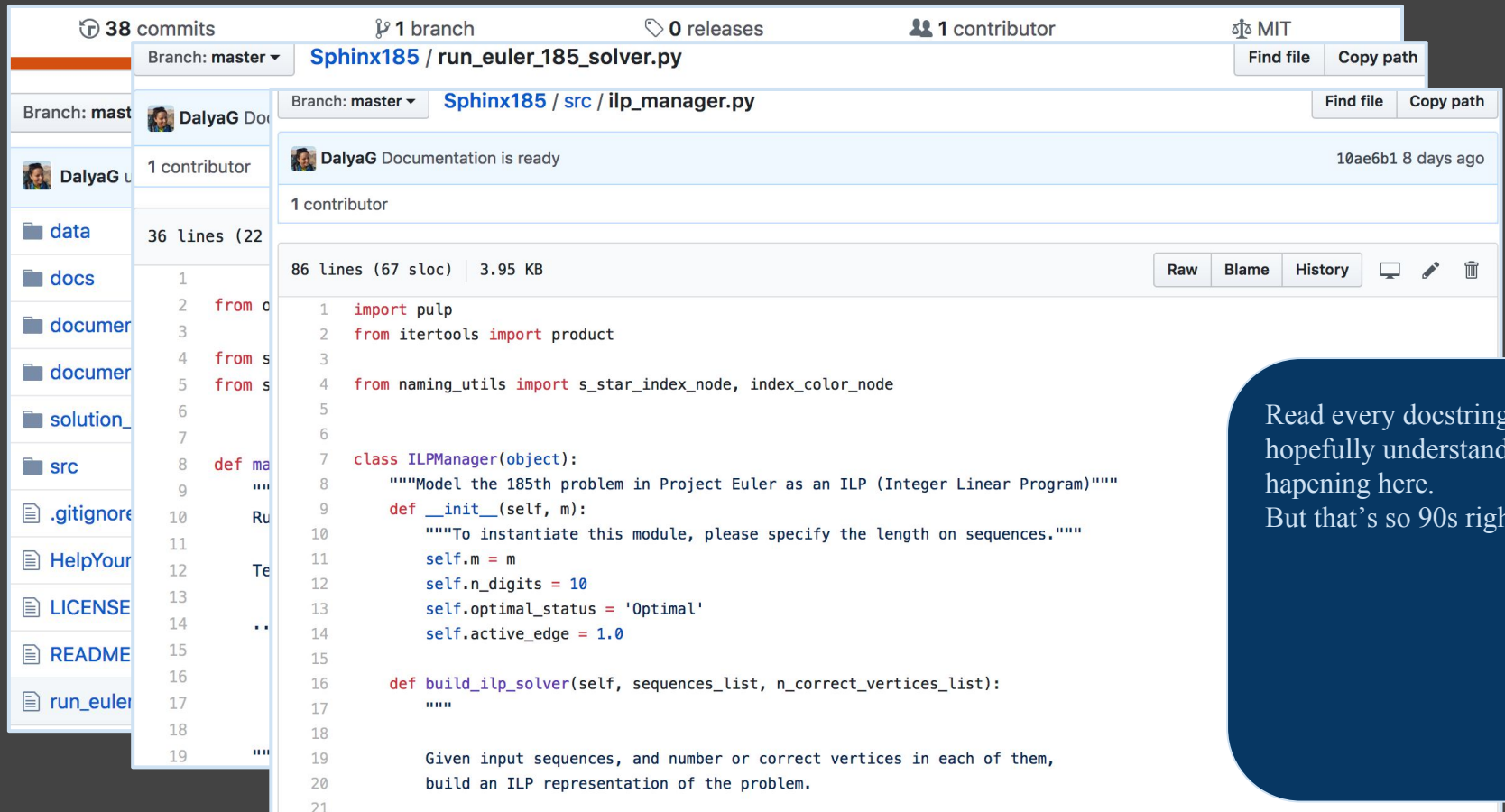
Now let's see how it would work if I was to give my firend a link tot he repository on GitHub -

Part I - Overview



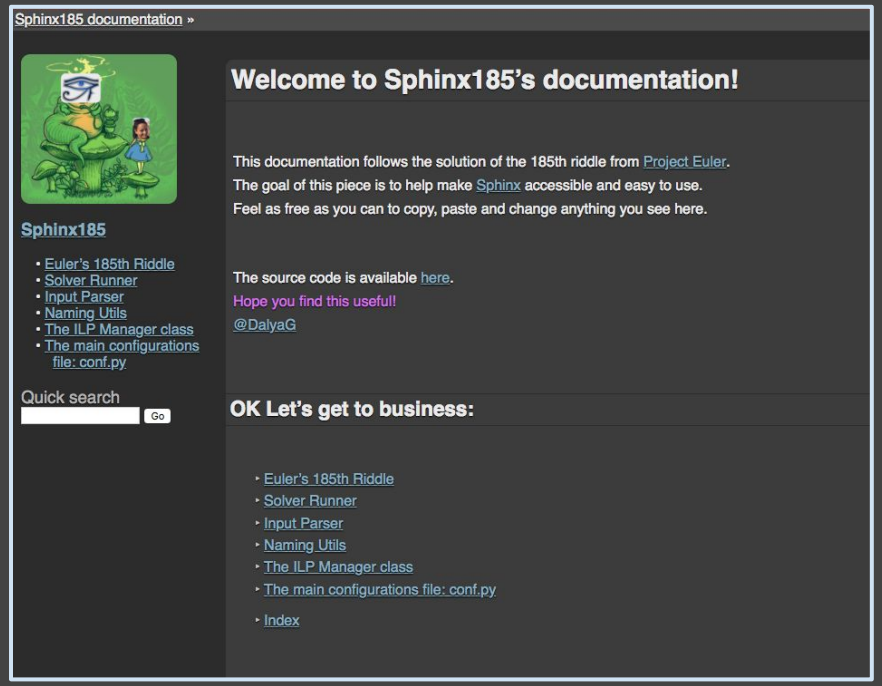
This friend would need to browse the files, look inside every file,

Part I - Overview



Read every docstring, and hopefully understand what is hapening here.
But that's so 90s right?!

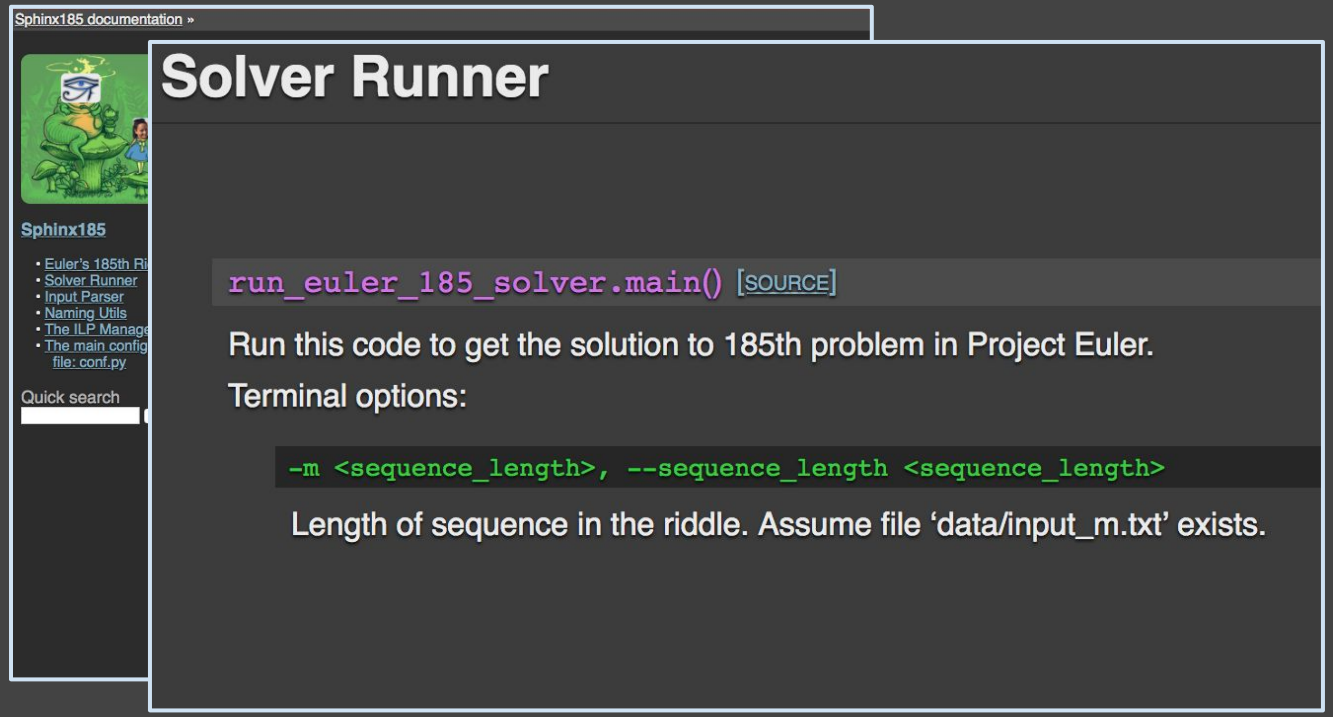
Part I - Overview



So what I am going to show you today is how to make a website that hosts the documentation to your project

dalyag.github.io/Sphinx185

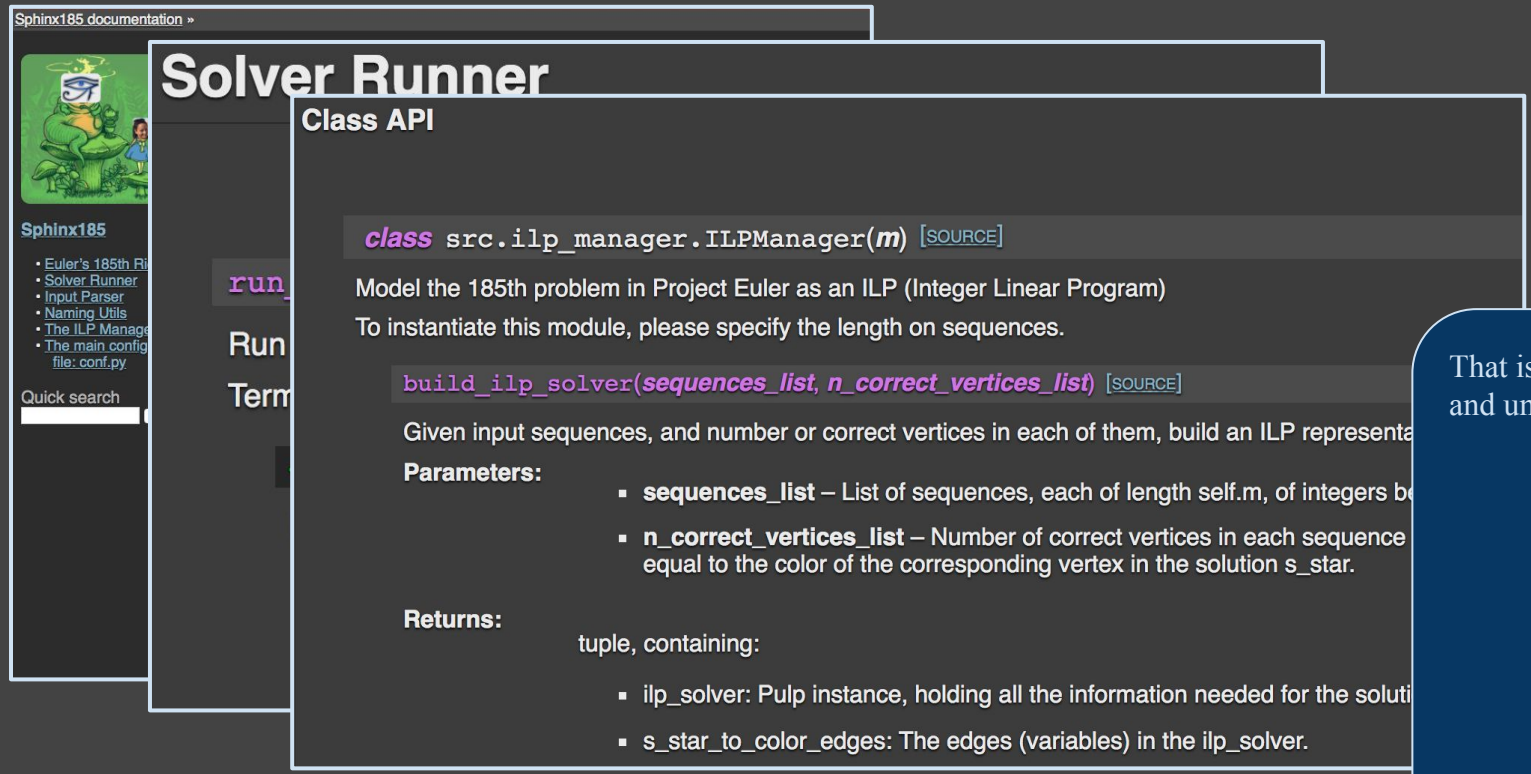
Part I - Overview



dalyag.github.io/Sphinx185

And present your docstring nicely in a readable way

Part I - Overview



That is easy to follow and understand

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.

So what does this sphinx give you?

First, i can use latex! Anything that lets me use latex is immediately my friend

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 of sequence in the riddle. As

Second - it lets me take the docstring of my functions, and automatically make it into an organized and designed output.

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

Solver Runner

_solver.main() [SOURCE]

et the solution to 185th problem in Project Euler

length>, --sequence

uence in the riddle. As

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()
```

It lets me present the source code, if I want, as part of the documentation

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 se
- **n_correct_vertices_list** – Number of correct vertice
  equal to the color of the corresponding vertex in the

Returns:

tuple, containing:

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

solve_ilp(*ilp_solver, s_star_to_color_edges*) [SOURCE]

Given a solver with the needed information, solve the ILP and extract the s

Parameters:

- **ilp_solver** – Pulp instance, holding all the informatio
- **s_star_to_color_edges** – The edges (variables) in

Returns:

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

It allows me to take long complex modules and with one line turn them into a clear and structured API

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

Solver Runner

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 c

    Terminal o

    .. option:

        Length

    """
    parser = O
    parser.add

    options, _

    m = int(op
    sequences_

    ilp_manage
    ilp_solver
    s_star = i

    print "Sol

if __name__ ==
    main()
```

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

Secure https://dalyag.github.io/Sphinx185/index.html

Sphinx185 documentation »

Sphinx185

- Euler's 185th Riddle
- Solver Runner
- Input Parser
- Naming Utils
- The ILP Manager class
- The main configurations file: conf.py
- How To Use This for YOUR Next Project :)

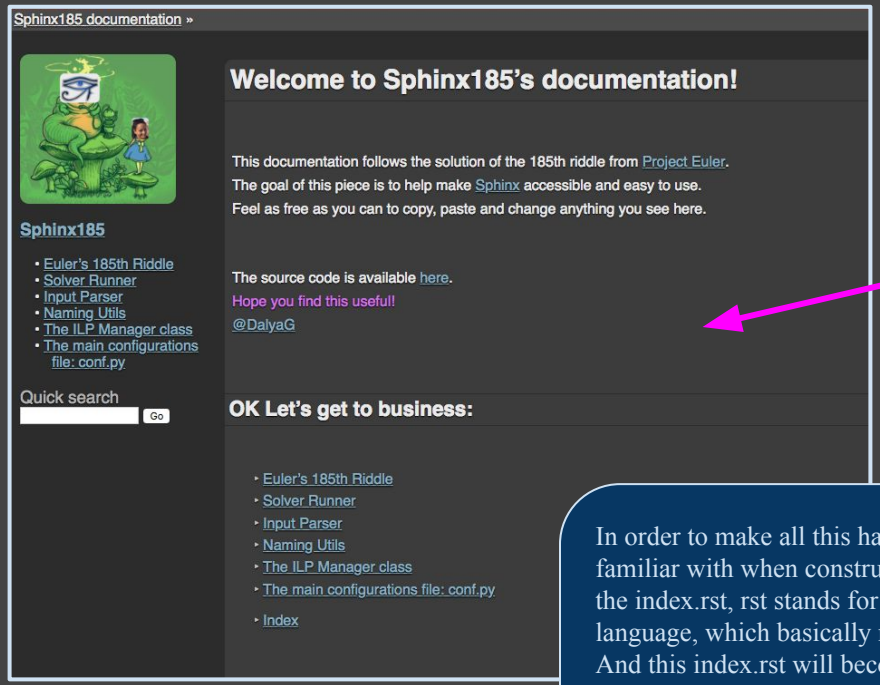
Welcome to Sphinx185's documentation!

This documentation follows the solution of the 185th riddle from Project Euler.
The goal of this project is to help make Sphinx accessible and easy to use.
To use this for YOUR next project, browse this documentation and follow the instr
Feel as free as you can to copy, paste and change anything you see here.

The source code is available here.
Hope you find this useful!
DalyaG

And finally, its really easy to make a website out of this. Because thats people do, right? If someone wants to publish a pckage they made, and they want other people to know how to use it, so you make a website for your read-the-docs.
So we'll also gonna learn how to do this today - how to host our beautiful documentation on github pages.

Part I - Overview

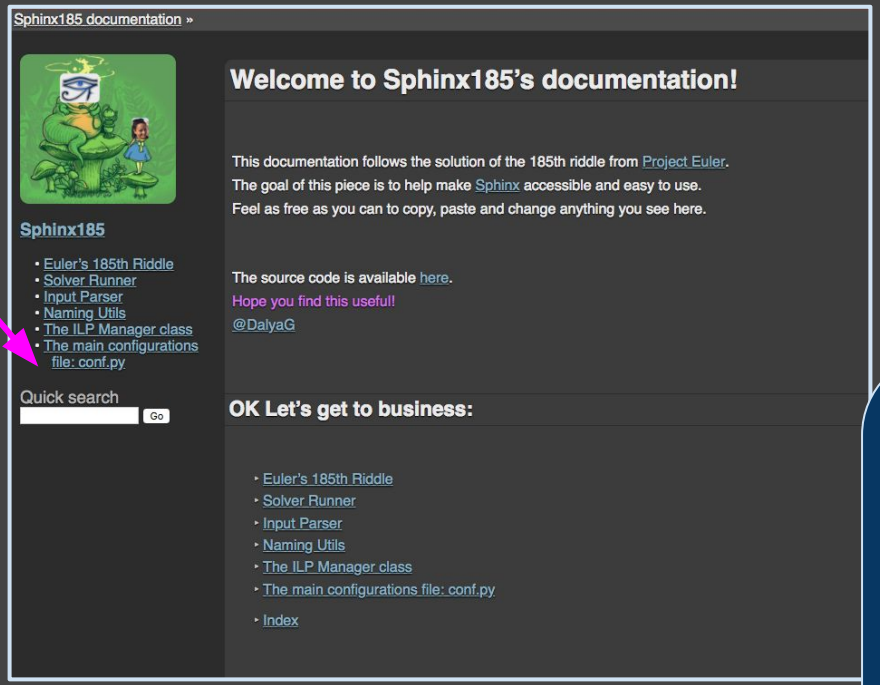


index.rst - the content of the home page

In order to make all this happen, there are two main files I need to be familiar with when constructing a documentation with sphinx - one is the index.rst, rst stands for ReStructuredText, which is a markdown language, which basically means its text with some cool features.
And this index.rst will become the homepage of your documentation - index.html
And also every page in your documentation is a something.rst file that you write, and is then transformed into something.html in the documentation.

Part I - Overview

conf.py - settings for the entire project



The second file you need to know is conf.py, which holds the configurations and settings for the entire project.
This file could be quite cryptic, so i made some editing that i hope made it more easy to use, and if you go to the repository of this project you could find slides that explain this file line-by-line.

Part II - Main configuration file - conf.py

And now for the dreadful 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 pa
# You can find available themes here: http:/
# In this project, I wanted to use a non-def
#     https://github.com/Cartroo/sphinx-theme
#     Some adjustments I made to graphite:
#     - I did not use the pygment configurati
#
#     - In the static folder, I configured se
#         you can download the original and
html_theme = 'graphite'
# When using a non-built-in theme, define th
html_theme_path = ['.']

# Add any paths that contain custom static f
# relative to this directory. They are copie
# so a file named "default.css" will overwri
html_static_path = ['_static']

# Defining the static path allows me to add
#   (make sure the theme of your choice supp
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictio
# In This project I chose to include in the
#   - Table of Contents: I chose globaltoc a
#     and configured the title by editing th
#   - Search box: appears below the TOc, and
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

At first when i planned this talk i thought i would go over all this file with you, since there are several non-trivial stuff here, but it was soooo exhausting,

Part] nf.py

even with this cute panda to cheer us up

Part II - Main configuration file - conf.py

Full details - EXTRA SLIDE

So i decided to edit almost all of it out, and the full details are under extra slides, and here i just want to talk about one property that is pretty much the essence of Sphinx -

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 pa
# You can find available themes here: http://
# In this project, I wanted to use a non-def
#    https://github.com/Cartroo/sphinx-theme
#    Some adjustments I made to graphite:
#    - I did not use the pygment configurati
#
#    - In the static folder, I configured se
#          you can download the original and
html_theme = 'graphite'
# When using a non-built-in theme, define th
html_theme_path = ['.']

# Add any paths that contain custom static f
# relative to this directory. They are copie
# so a file named "default.css" will overwri
html_static_path = ['_static']

# Defining the static path allows me to add
#   (make sure the theme of your choice supp
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictio
# In This project I chose to include in the
#   - Table of Contents: I chose globaltoc a
#     and configured the title by editing th
#   - Search box: appears below the TOc, and
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

The autodoc extenssion -
This thing is the magic behind sphinx,

Part II - Main configuration file - conf.py

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

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

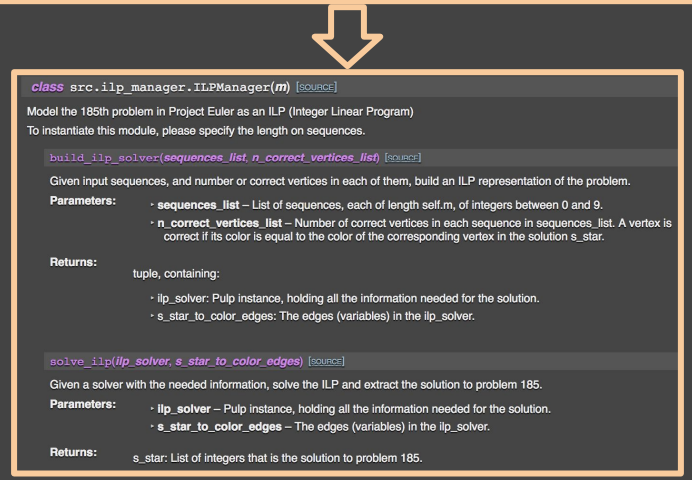
```
#       It will appear in a highlighted box in the documentation.
#       In order for this extension to work, make sure you include the following:
todo_include_todos = True

                                                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


                                                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 pa
# You can find available themes here: http:/
# In this project, I wanted to use a non-def
#     https://github.com/Cartroo/sphinx-theme
#     Some adjustments I made to graphite:
#     - I did not use the pygment configurati
#
#     - In the static folder, I configured se
#           you can download the original and
html_theme = 'graphite'
# When using a non-built-in theme, define th
html_theme_path = ['.']

# Add any paths that contain custom static f
# relative to this directory. They are copie
# so a file named "default.css" will overwri
html_static_path = ['_static']

# Defining the static path allows me to add
#   (make sure the theme of your choice supp
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictio
# In This project I chose to include in the
#   - Table of Contents: I chose globaltoc a
#     and configured the title by editing th
#   - Search box: appears below the TOc, and
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

and it lets you, with one simple line

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.

```
# 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 pa
# You can find available themes here: https:/
# In this project, I wanted to use a non-def
#    https://github.com/Cartroo/sphinx-theme
#    Some adjustments I made to graphite:
#    - I did not use the pygment configurati
#
#    - In the static folder, I configured se
#          you can download the original and
html_theme = 'graphite'
# When using a non-built-in theme, define th
html_theme_path = ['.']

# Add any paths that contain custom static f
# relative to this directory. They are copie
# so a file named "default.css" will overwri
html_static_path = ['_static']

# Defining the static path allows me to add
#   (make sure the theme of your choice supp
html_logo = '_static/sphinx_and_dalya.png'

# Custom sidebar templates, must be a dictio
# In This project I chose to include in the
#   - Table of Contents: I chose globaltoc a
#     and configured the title by editing th
#   - Search box: appears below the TOc, and
html_sidebars = {
    '**': [
        'globaltoc.html',
        'searchbox.html'
    ]
}
```

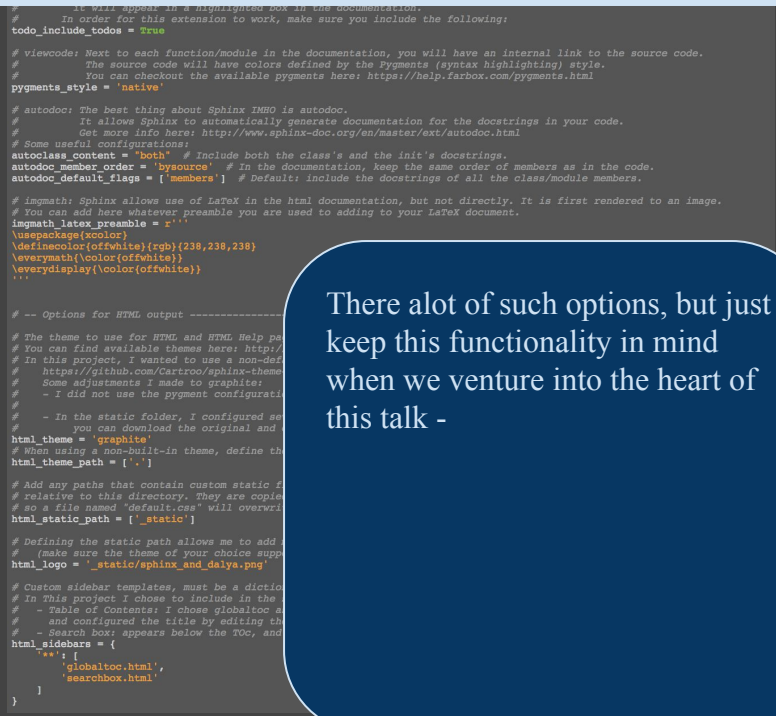
Create a documentation for the entire module

Part II - Main configuration file - conf.py

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

Full details here:

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



There alot of such options, but just keep this functionality in mind when we venture into the heart of this talk -

Part III - Document YOUR next project

Where we will finally start crating a documentation for YOUR project

Part III - Document YOUR next project

Before we start, I would like to point out that this part is going to be rather technical.
My goal here is not to dictate actions that you will remember by heart, and go home to apply, but rather give an overview of how this process goes, so that later, if you want to make a documentation for yourself, you could use the project I published on GitHub, and these slides that are available there, to guide you in the process.

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)

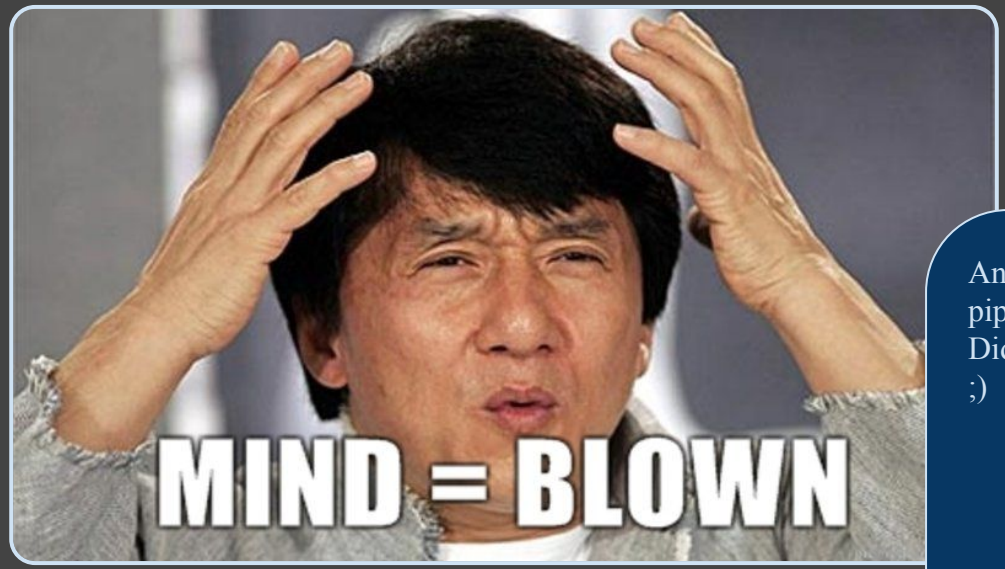
How To Use This for YOUR Next Project :)

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>/`

In this part we will follow the instructions that you can find in the same place where you can find all the other things in this talk

Part III - Document YOUR next project

1. pip install sphinx



And we will start with
pip-install-sphinx
Did not see this one coming, ha?
;)

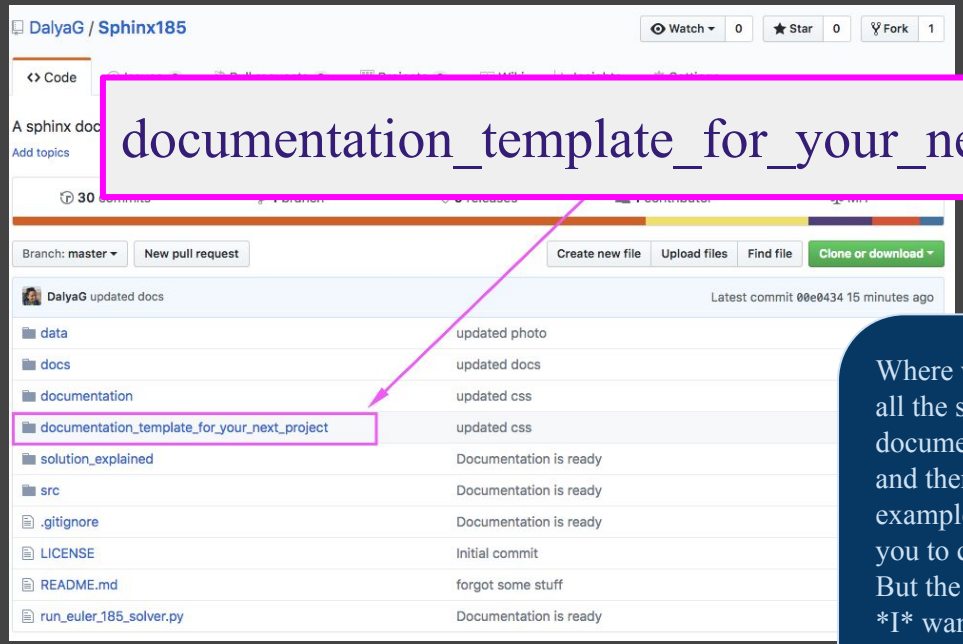
Part III - Document YOUR next project

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

Ok now we can really start, by going to the github of this project

Part III - Document YOUR next project

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



documentation_template_for_your_next_project

Where we can find a “documentation” folder, that holds all the source code for the *objectively* beautiful documentation i made for this project,
and there you can get lots of tips and ideas and usage examples for sphinx and .rst features that i curated for you to copy-paste at your convenience.
But the thing is - that this folder is busy with things that *I* wanted to do, and you may not want to use them all,
So for this purpose I created a lean and simple version, under

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***

and in my opinion the best way to start with sphinx is by downloading this folder into your local project, and rename it “documentation”,
and from now on i am going to assume that this is what you did

Part III - Document YOUR next project

Bravo! You now have a documentation!

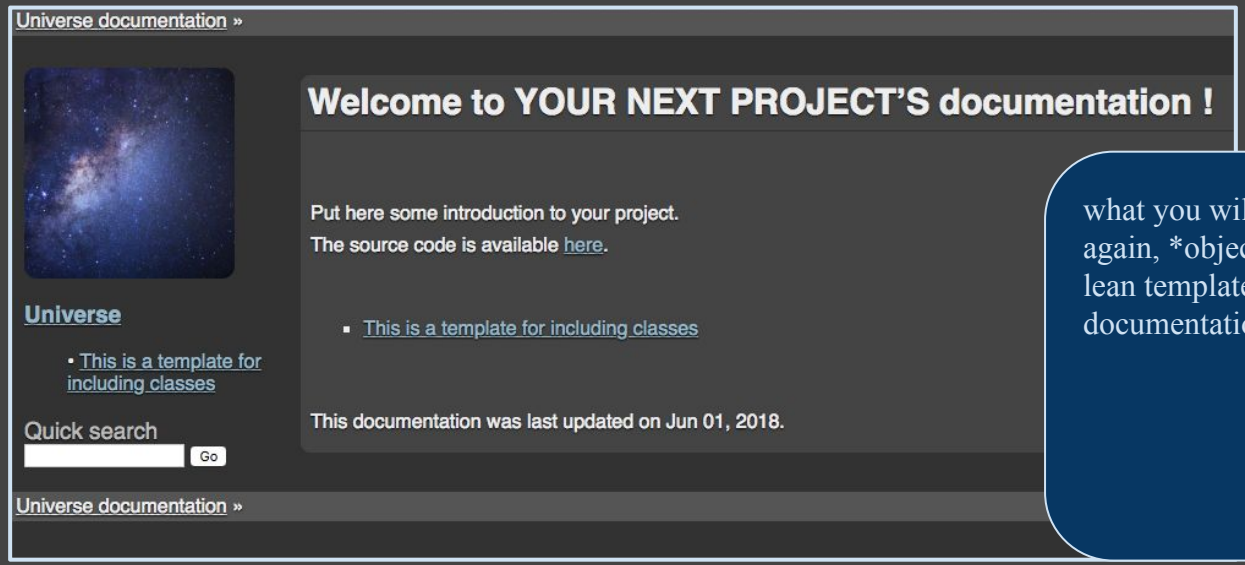
And i promise i didnt make you dowload some virus or something,

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



what you will download is this, again, *objectively* beautiful and lean template to start building your documentation upon.

Part III - Document YOUR next project

3. Edit *conf.py*

So - the first step in the personalization of this template is to edit the dreaded 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 proj

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

I made this process as easy as i could by including a #CHANGEME# pattern with helper notes on what you need to edit

Part III - Document YOUR next project

4. Edit *index.rst*

The next step will be to edit the home page of your project's documentation

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/usrn
```

This index.rst that you downloaded also has helper notes inside, and this file you will keep editing as you go along with your documentation, so you can start with a simple introduction, just to get things going.

Part III - Document YOUR next project

5. To add a page to your documentation:

Now, if you want to add a page to your documentation -

Part III - Document YOUR next project

5. To add a page to your documentation:

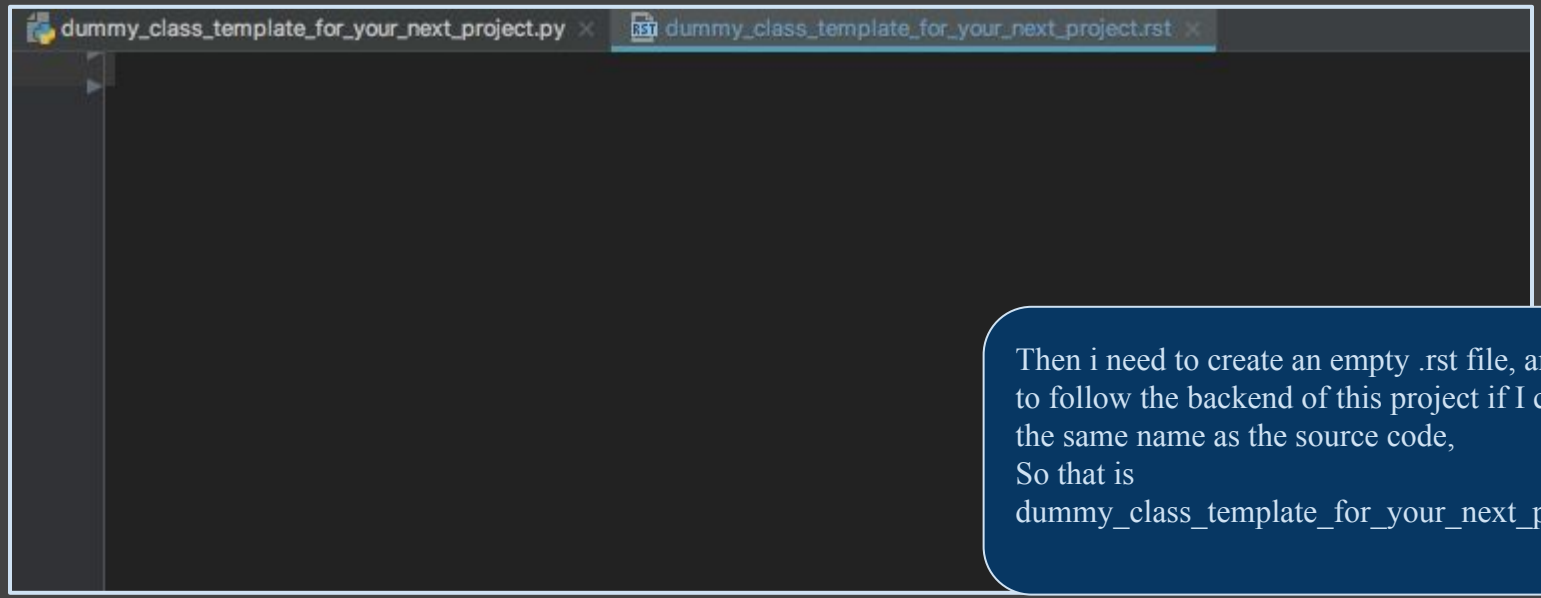
dummy_class_template_for_your_next_project.py

```
1
2   class DummyClassTemplateForYourNextProject(object):
3       """Some general info about this class."""
4       def __init__(self):
5           """Some info about the init method"""
6           self.some_att = 42
7
8       def answer_to_the_universe(self):
9           """
10
11          To make a "real" docstring leave blank lin
12
13          :return: Prints the answer to the universe
14          """
15          print "The answer to the universe is: {}".
```

So lets say i have this class called DummyClassTemplateForYourNextProject sitting inside a file called dummy_class_template_for_your_next_project.py, And i want to add this file to my documentation

Part III - Document YOUR next project

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



Then i need to create an empty .rst file, and its easy to follow the backend of this project if I call it by the same name as the source code,
So that is
dummy_class_template_for_your_next_project.rst

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

dummy_class_template_for_your_next_project.py | dummy_class_template_for_your_next_project.rst

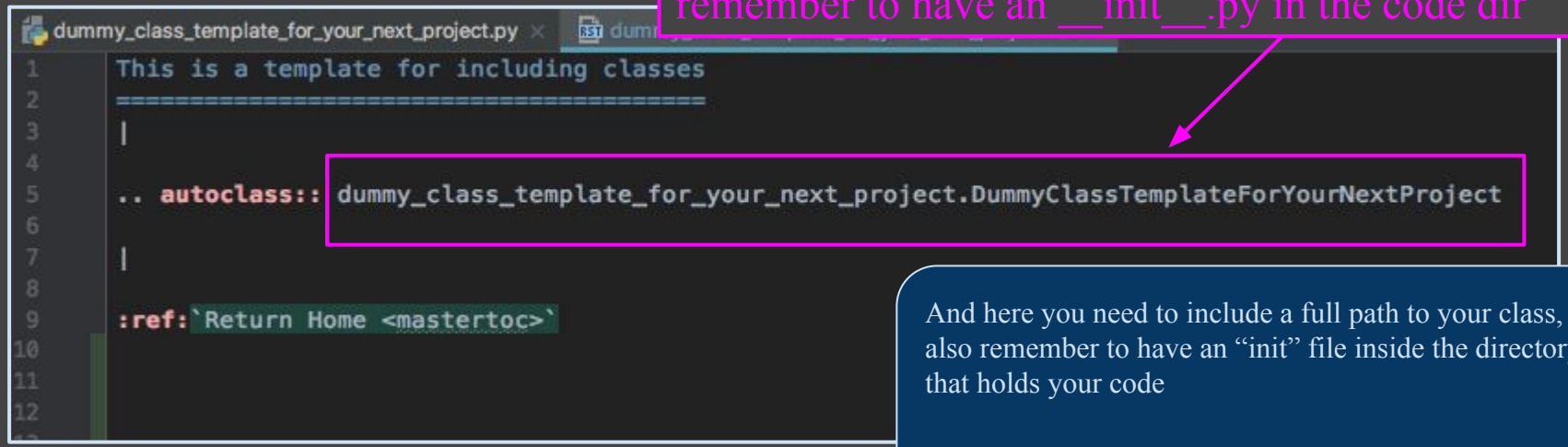
```
This is a template for including classes
========================================
|

.. autoclass:: dummy_class_template_for_your_next_project.DummyClassTemplateForYourNextProject

|

:ref:`Return Home <mastertoc>`
```

And then i copy-pase inside it some template that i found under the original "documentation" folder of the "sphinx185" project -
In this occussion i looked for a template for a class documentation and copied this call for "autoclass"

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

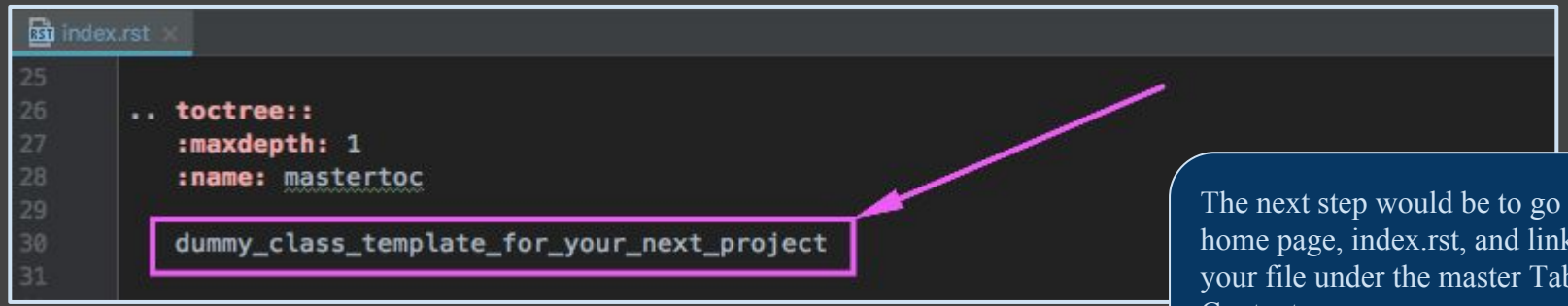
Full path to the class +
remember to have an __init__.py in the code dir



And here you need to include a full path to your class, and also remember to have an “init” file inside the directory that holds your code

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*



The next step would be to go to the home page, index.rst, and link to your file under the master Table Of Content

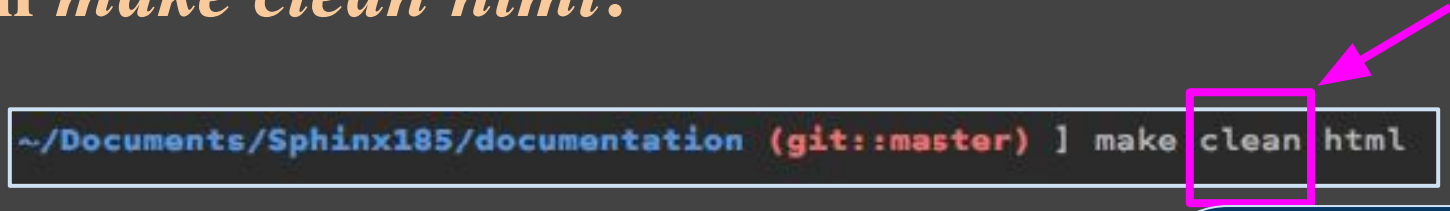
Part III - Document YOUR next project

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

Now in order to apply these changes

Part III - Document YOUR next project

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



Run “make clean html”
You can also run just “make html” but then sphinx holds some cache of your previouse builds, and in order for this build to catch all the changes you made, i recommand always running with “clean”

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) ]
```

And then hopefully you will see this clean output

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 b
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) ]
```

If you do happen to see some warning or errors, they are sometimes rather indicative, and sometimes they also have available solutions online, so just look for the error, and, you know, copy-paste from somewhere the solution

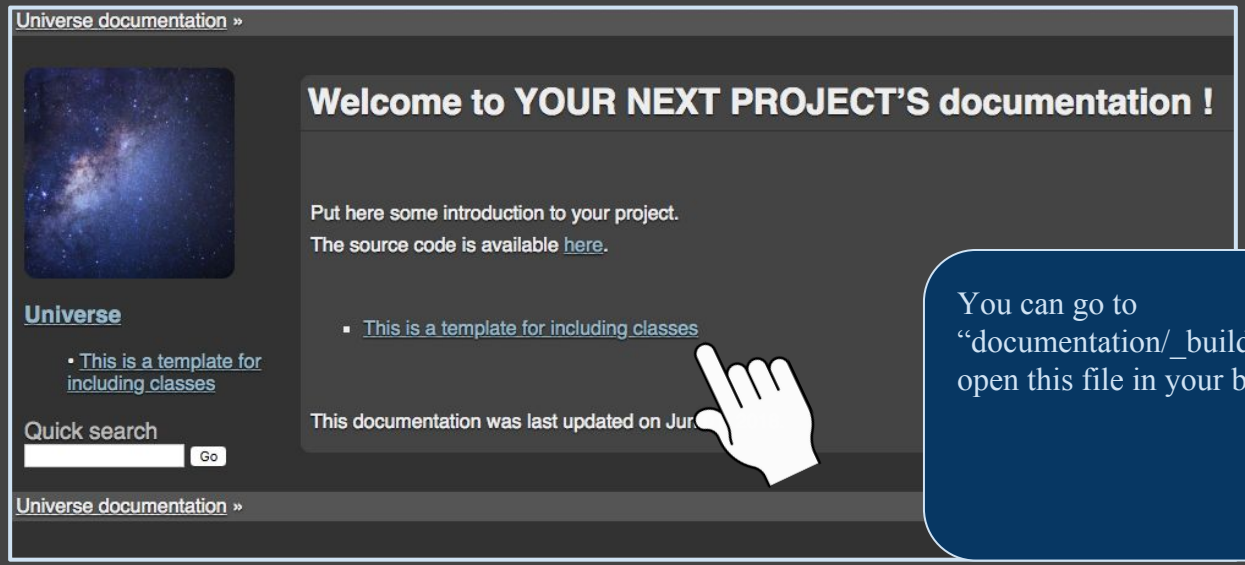
Part III - Document YOUR next project

7. To view locally

And thats it, its ready

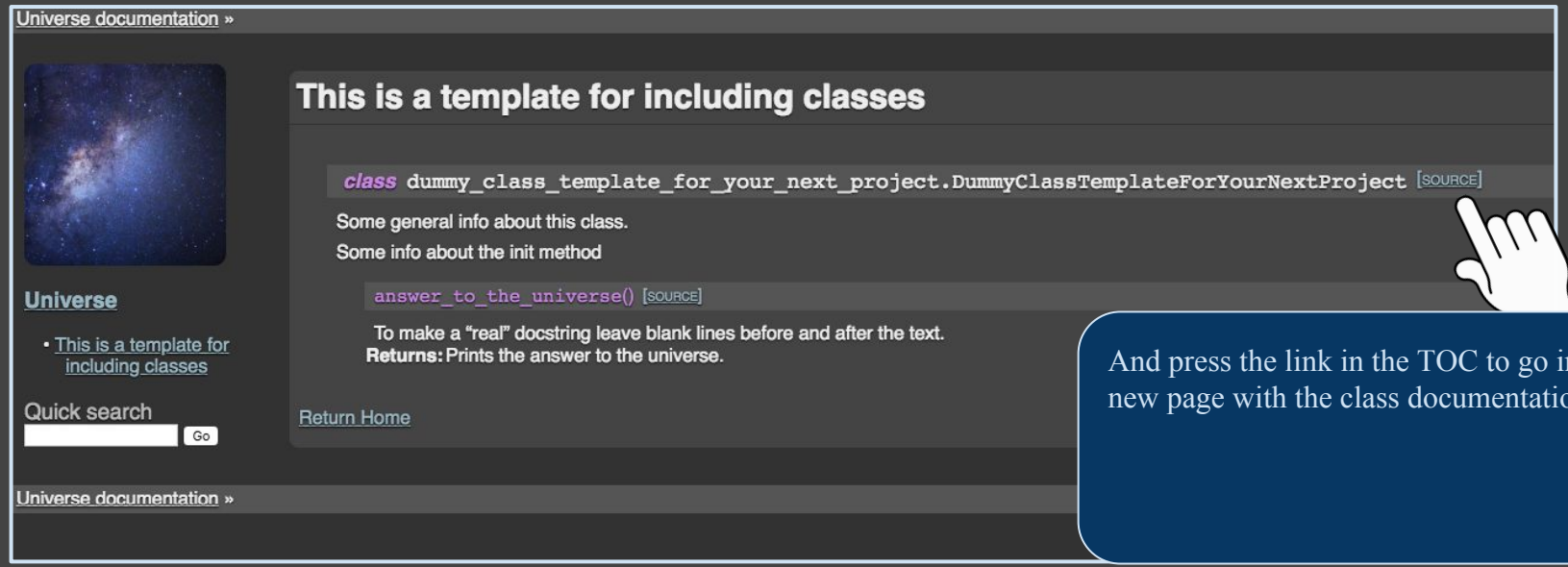
Part III - Document YOUR next project

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



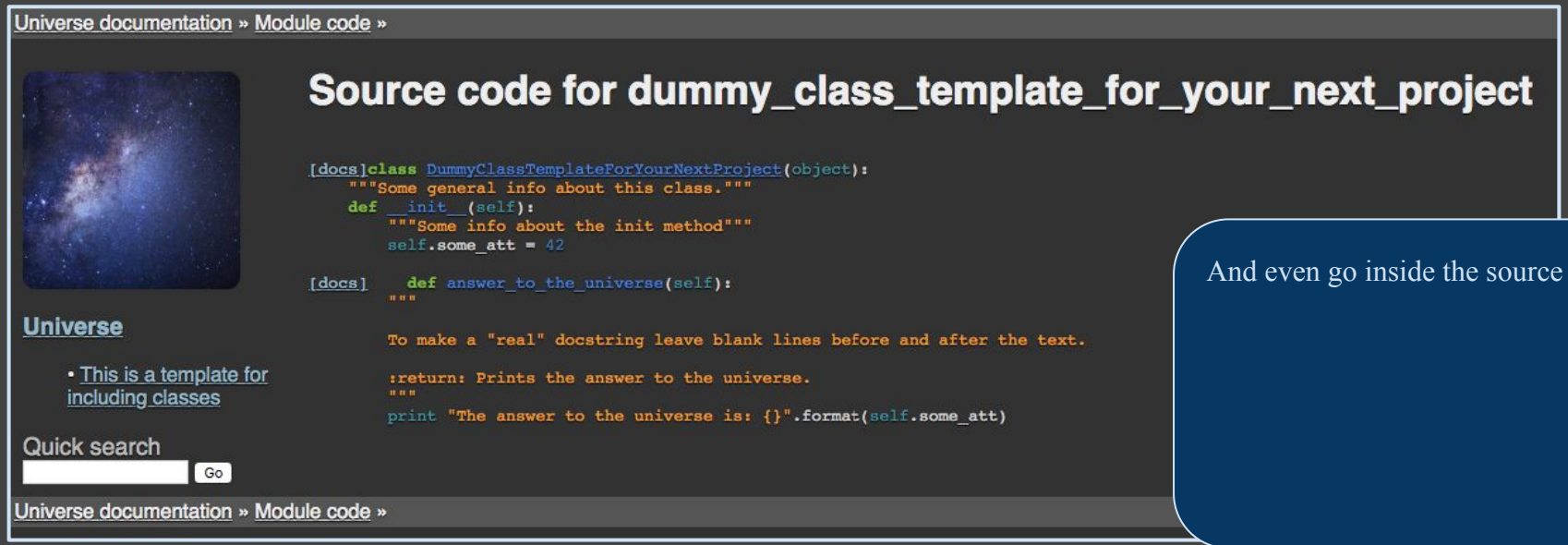
Part III - Document YOUR next project

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 lo… browser *documenta…*

But the thing is, that having your documentation on your own device is nice and all, but we are not here to share the documentation with ourselves, *we* already know what this project does.

Part III - Document YOUR next project

8. SHARE your documentation!

Our goal is to share this documentation with others!

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

So, last streatch, lets build a website for our documentation, using github pages -

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

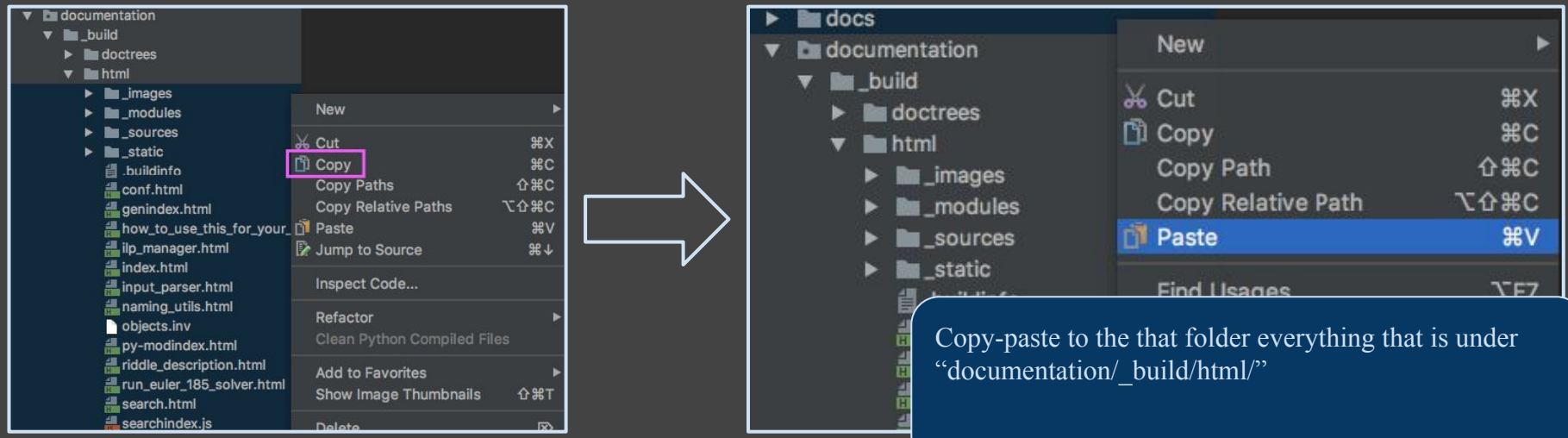
a. Create a folder *docs* under the root of the project

First step - inside you local project, create a directory called "docs"

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

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



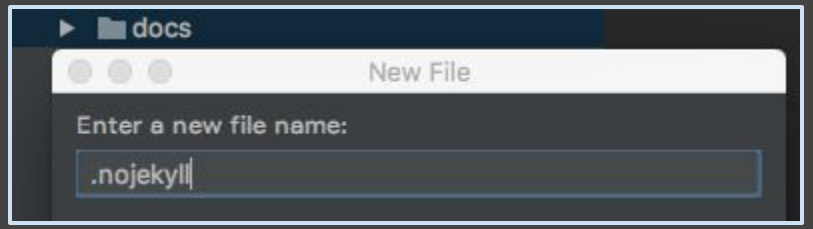
Copy-paste to the that folder everything that is under "documentation/_build/html/"

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)



Inside this directory create an empty file called ".nojekyll" - this is a sign for github pages to bypass the "jekyll" themes, and use the html and css in your project

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

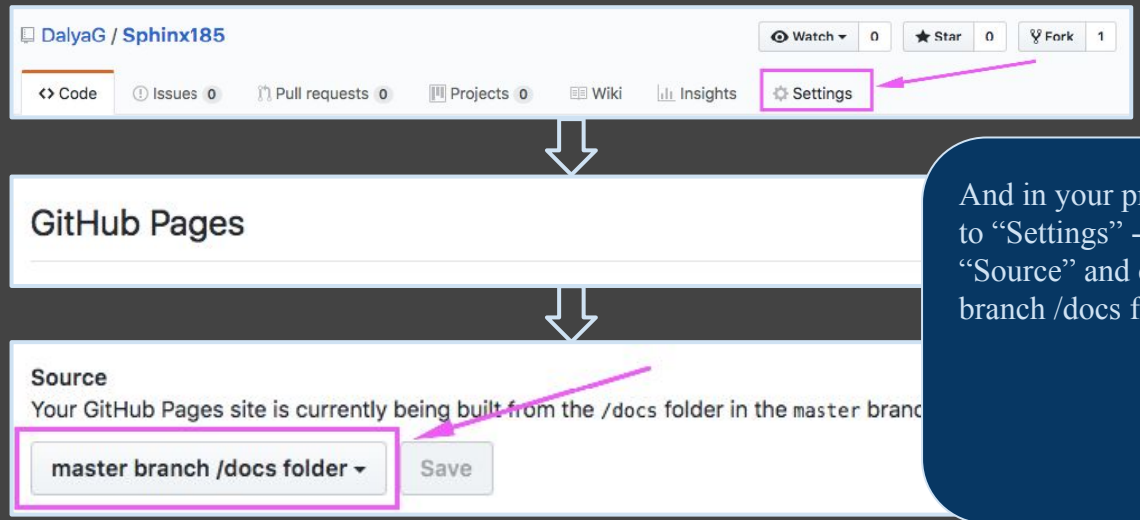
c. Push your changes to master branch.

Now push your changes to your 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”

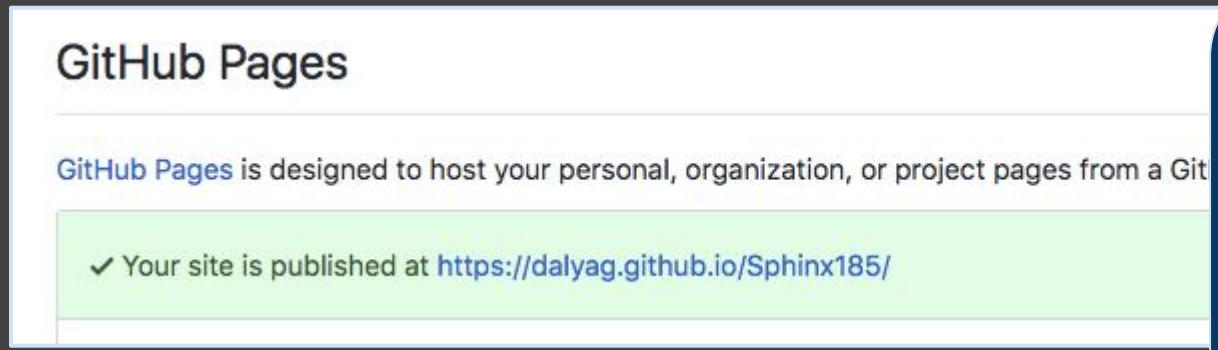


And in your project on github - go to “Settings” -> “GitHub Pages” -> “Source” and choose “master branch /docs folder”

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>/



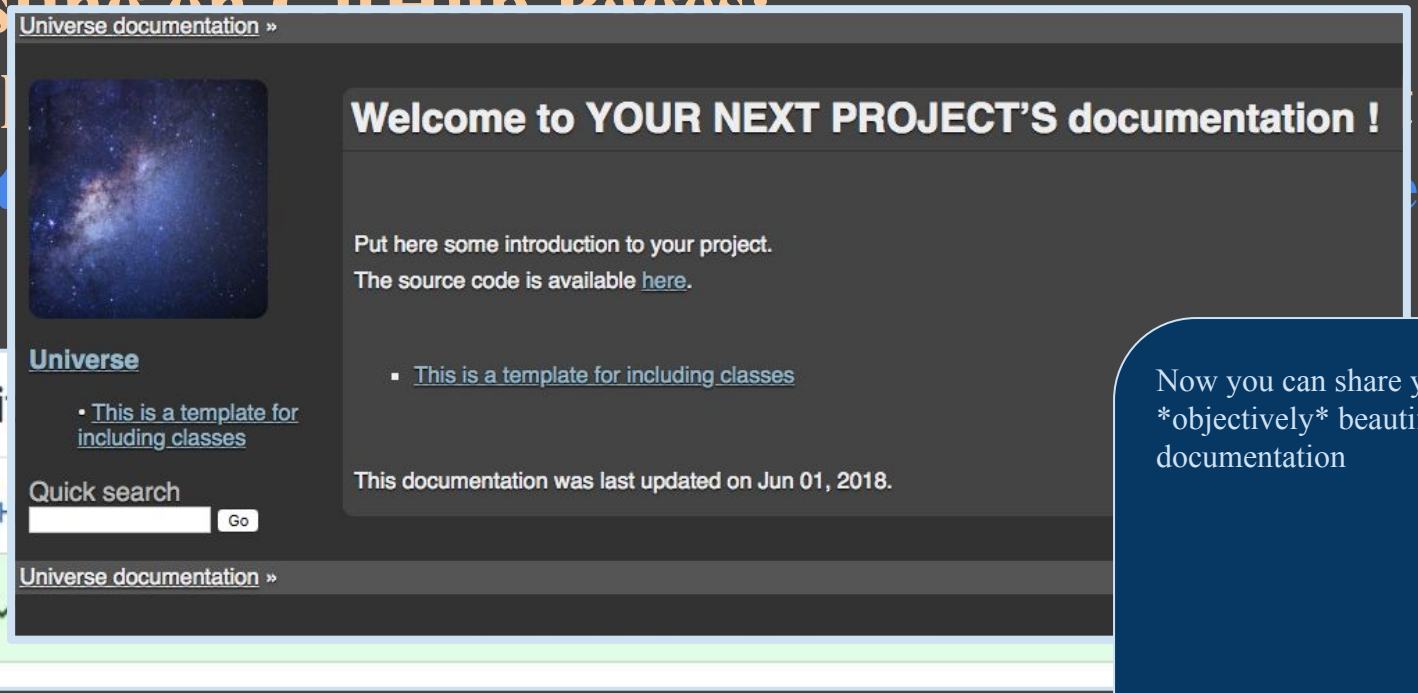
And its ready!

Part III - Document YOUR next project

8. Hosting on GitHub Pages:

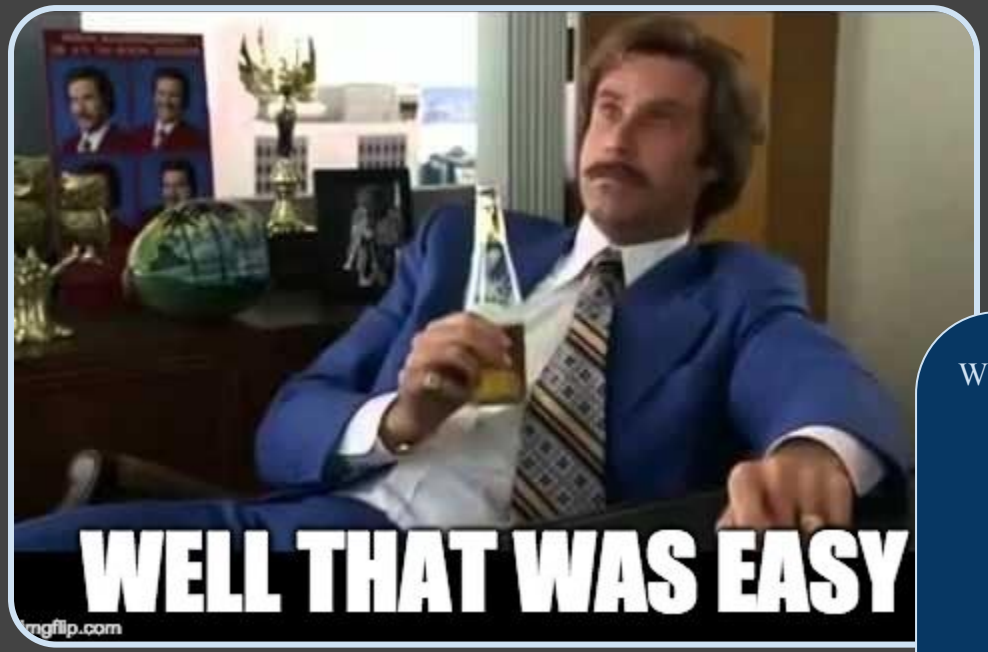
e. Sl

ht /



Now you can share your *objectively* beautiful documentation

Part III - Document YOUR next project



Wasnt so hard right?

Part III - Recap

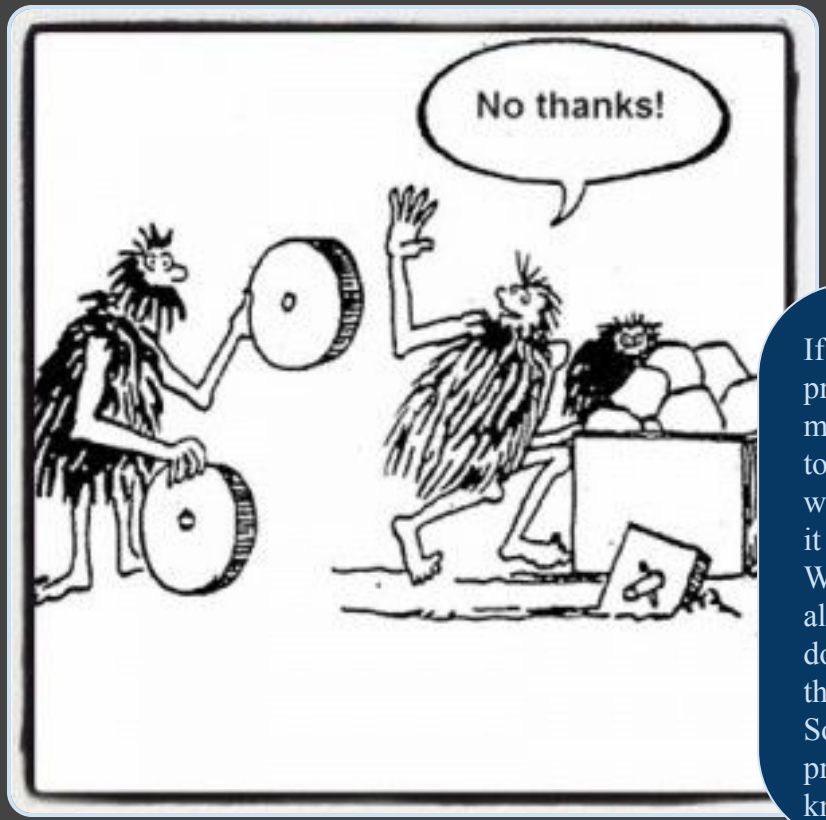
As I am a true believer in knowledge sharing, I did my best in this tutorial to make this tool accesible and easy to use, so that next time you want to share something, you will have an easy time doing so.

Part IV - Surprise! Live Demo!

github.com/DalyaG/Sphinx185

So I am going to take a leap of faith in the universe, and try to do a modest live demo, of browsing the repository i kept talking about -

Take Home Message



If you are working on a collaborative project, like the research repository i mentioned at the begining, or you want to publish an open source tool, the best way to make people use it, is by making it accesible and easy to use.
What Sphinx allowes you to do is skip all the overhead of managing your documentation system, and simply use the docstrings you have in your code.
So i wish you all good luck with your projects, and keep sharing the knowledge :D

Thank you :)

Questions?

DalyaG@gmail.com