Documentation

CMSE 890-402

Content

What should documentation include?

Suggestions

- Home page
- Search
- Installation instructions
- Quick start
- Feature descriptions
- Application programming interface

API: Application Programming Interface

- How modules should be loaded
- How functions should be used
- Often created using documentation written within the code
- May include diagrams of code structure
- E.g. https://docs.astropy.org/en/stable/time/index.html#reference-api

Design

How should documentation be designed?

Suggestion

“Diataxis” format:

- Tutorials
- How-to Guides
- Explanation
- Reference
- https://diataxis.fr/

Building documentation

- Documentation frameworks usually need at least one page, the index
- Also common: configuration file to set up the documentation
- In mkdocs:
  - mkdocs new (create a new mkdocs setup)
  - mkdocs serve (serve the docs locally)
  - mkdocs build (build the docs locally)
- Add the sites directory to .gitignore!

Configuring documentation

- Use the mkdocs.yml file
- YAML = YAML Ain’t Markup Language (because it is used for data)
- Structure is based on indents and colons
- https://yaml.org/ is a demonstration of the syntax
- Many, many options- read the mkdocs documentation

Automatically documenting code: docstrings

- docstrings are specific to Python, but similar principles apply to other langs
- Common styles:
  - Google
  - Numpy
  - Sphinx
- Mkdocs Python extension uses Google by default
- https://google.github.io/styleguide/pyguide.html#38-comments-and-docstrings
- Tests do not typically require docstrings

Anatomy of a (Google-style) function docstring

```
"""This first part is a short description of the function.

Then a longer one with more information.

Args:
    argument: One of the function arguments. May include type information.
    second_argument: Another function argument.

Returns:
    A description of what the function returns and its type.

Raises:
    Information about any exceptions the function creates
"""
```

R docstrings

```
#' Short title for function

#'

#' Longer description of the function

#' @param first An object of class "?". Description of parameter

#' @param second An object of class "?". Description of parameter

#' @return Returns an object of class "?". Description of what the function returns

#' @examples

#' # Add some code illustrating how to use the function

my_new_function <- function(first, second){

      # Some code here

      return(something)

}
```

A note about variable types

- “Type” means the form of a variable
- It may be a fundamental type e.g. integer, string, boolean
- It may be a custom type you wrote yourself (a class)
- Types can often be transformed to other types
- It is useful to know what type function arguments need to be

Automatically documenting code: Type annotation

- Relevant for weakly-typed languages (R, Python…)
- Built-in to Python
- For R, use the “types” package
- Shows types in docs and in VSCode/your IDE
- Helps you track the input/output of functions

Type hinting syntax (Python)

```
from typing import Union

def function(a: float, b: bool, c: Union[float, int]) -> float:
    return a*c
```

Note that Union is not needed in Python 3.10+

```
def function(a: float, b: bool, c: float | int) -> float:
    return a*c
```

Type hinting syntax (R)

```
f <- function(x = 1 ? numeric) {
  x %% 2 == 0 ? boolean
} ? boolean


f <- function(x = ? numeric) {
  x + 1
} ? numeric
```

Hosting

- Documentation must be converted to html for internet hosting
  - Can use many documentation generators to handle automatically
    - Python: Mkdocs, Sphinx
    - R: roxygen
    - Java: javadoc
- GitHub provides free hosting via github.io

Using mkdocs

`mkdocs new .`

- Creates a new documentation directory and configuration file in the current directory

`mkdocs serve`

- Builds and serves the docs locally

`mkdocs deploy`

- Builds and deploys the docs to a remote server (e.g. github pages)

Using pkgdown

- Standard for R packages
- Limited capability for documentation beyond the package itself
- Need roxygen to get an API listing
- Vignettes provide examples

Local build:

```
pkgdown::build_site()
```

Set up github pages site:

```
usethis::use_pkgdown_github_pages()
```

Write documentation for all of the functions from last time!

- Clone the assignment repo https://classroom.github.com/a/oV-M3emh
- Copy in my and your functions from last week. Don’t bother with tests.
- **Add** and **style** docstrings to all the functions.
- Add type annotations
- If you wrote R functions I will provide the Python alternatives.

Building your docs

- Install the mkdocstrings plugin.
- Generate documentation HTML
- Include:
  - Home page
  - One page per function describing what they do as separate, hand-written markdown files
  - The automatically generated API for each function

Push your documentation *and scripts* to the assignment repo

https://classroom.github.com/a/oV-M3emh

Homework

- Organise your docs using the Diataxis format
- Write **at least one page** for each Diataxis category.
- Host the documentation on a github.io site!
- Make sure you check the Github Pages settings in your repository

https://www.mkdocs.org/user-guide/deploying-your-docs/

Pre-class 5: Advanced git

Complete as much of https://learngitbranching.js.org/ as possible

Upload screenshot(s) of your progress to D2L