Technology
Arts Sciences
TH Köln

How To Sphinx

Python Dokumentationen mit Sphinx erstellen

Agenda

1. **Sphinx**
    1. Docstrings in a nutshell
    2. Startguide
2. **Docstrings**
    1. Docstrings 101
    2. Docstring Formate
    3. Datentypen Style Konvention
    4. Docstrings und Sphinx
3. **Docstrings in PyCharm**
    1. Template wählen
    2. Docstring schreiben

01 —

Sphinx

1.1 Sphinx in a nutshell

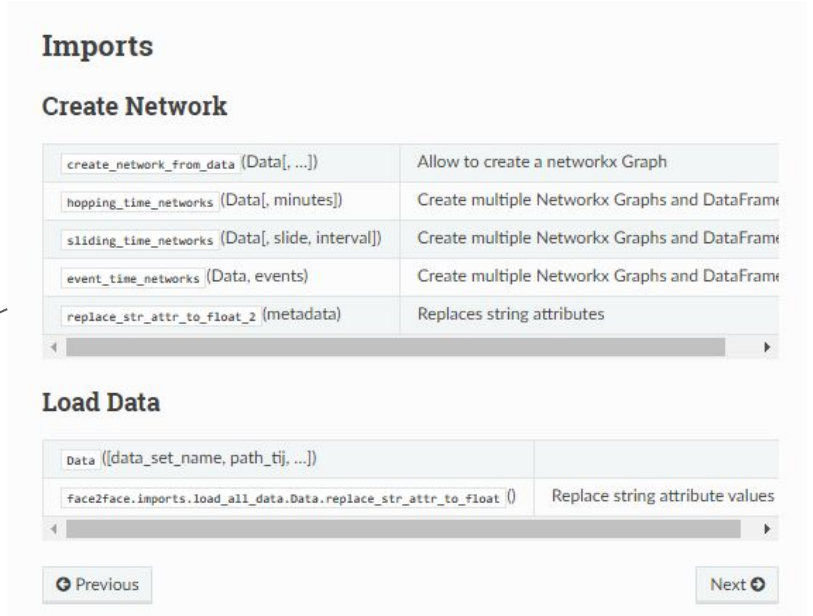
- Library zur automatisierten Generierung der Dokumentationen von packages
- Unterstützt verschiedene Dateiformate (HTML, PDF (via Latex) etc.)
- wird mit hilfe von reStructuredText Dateien aufgebaut und strukturiert
- beinhaltet verschiedene Extensions zur automatischen Generierung der Dokumentationen auf Basis des gewählten Docstring Formats

https://www.sphinx-doc.org/en/master/usage/quickstart.html

1.2 Startguide

Technology
Arts Sciences
TH Köln

1. Anlegen einer Documentation Directory
2. Command `$ sphinx-quickstart` erstellt die notwendigen Files in der Documentation Directory
3. wichtige Files:
   a. conf.py
   b. makefile
4. conf.py beinhaltet:
   a. Pfadsetup zum zu dokumentierenden Package
   b. Projektinformationen
   c. Verwendete Extension
      i. ‘sphinx.ext.autodoc',
      ii. 'sphinx.ext.autosummary',
      iii. 'sphinx.ext.napoleon’
   d. Scripte die ausgeschlossen werden sollen
5. Generierung des HTML Files mit
   a. `$ sphinx-build -b html sourcedir builddir`



02

Docstrings

2.1 Docstrings 101

- Docstrings beschreiben die Funktion von Funktionen und Klassen.
- Sie werden als mehrzeiliger String direkt nach der Deklaration der Funktion oder Klasse angegeben.
- Docstrings können beinhalten:
  - Beschreibung
  - Parameter
  - Returns
  - Types
  - Examples
  - Links

```python
def create_connection(self):
    """
    Create connection and cursor object to connect with the database.

    Environment variables:
    USER -- mariadb username
    PASSWORD -- mariadb password
    HOST -- mariadb host address
    PORT -- mariadb port number
    DATABASE -- mariadb database name
    """
    try:
        self.conn = mariadb.connect(
            user=os.environ.get('USER'),
            password=os.environ.get('PASSWORD'),
            host=os.environ.get('HOST'),
            port=int(os.environ.get('PORT')),
            database=os.environ.get('DATABASE')
        )

        self.cur = self.conn.cursor()

    except mariadb.Error as e:
        print(f"Error connecting to MariaDB Platform: {e}")
        sys.exit(1)
```

2.2 Docstring Formate

Technology
Arts Sciences
TH Köln

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)
    """
    if imag == 0.0 and real == 0.0:
        return complex_zero
```

PEP 257

Google

```
def function_with_types_in_docstring(param1, param2):
    """Example function with types documented in the docstring.

    `PEP 484`_ type annotations are supported. If attribute, parameter, and
    return types are annotated according to `PEP 484`_, they do not need to be
    included in the docstring:

    Args:
        param1 (int): The first parameter.
        param2 (str): The second parameter.

    Returns:
        bool: The return value. True for success, False otherwise.

    .. _PEP 484:
        https://www.python.org/dev/peps/pep-0484/

    """
```

NumPy

```
def function_with_types_in_docstring(param1, param2):
    """Example function with types documented in the docstring.

    `PEP 484`_ type annotations are supported. If attribute, parameter, and
    return types are annotated according to `PEP 484`_, they do not need to be
    included in the docstring:

    Parameters
    ----------
    param1 : int
        The first parameter.
    param2 : str
        The second parameter.

    Returns
    -------
    bool
        True if successful, False otherwise.

    .. _PEP 484:
        https://www.python.org/dev/peps/pep-0484/

    """
```

https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html#example-numpy

2.3 Datentypen Style Konvention

Gängige Datentypen in Kurzform deklarieren:

- `int`
- `bool`
- `str`
- `float`
- `list`
- `dict`

Defaultparameter, wenn angegeben, wie folgt hinter die Parameter:

- `para1 : bool, default True`

Weitere Datentypen in Kurzform (wenn üblich) deklarieren:

- `Pandas DataFrame -> df`

2.4 Docstrings und Sphinx

Technology
Arts Sciences
TH Köln

```python
def mapping_function(Data, label="type"):

    r"""Creates a mapping

        Creates a mapping dictionary to create the attribute mixing matrix dynamic,
        independent of the amount of attributes.

        Parameters
        ----------
        Data: Data
            Data Object that contains Tij- and Metadata for a data set.
        label: str
            A string that tells the function which attribute should be mapped.

        Returns
        -------
        mapping: dict
            A dictionary that contains the dimensions of the given attribute.

        Examples
        ---------
        Based on the amount of attribute values the used attribute has, you have to create a mapping for the
        null model functions.

        >>> mapping = mapping_function(test_df, label="Age")
        >>> print(mapping)
        {0: 0, 1: 1, 2: 2}

        >>> mapping = mapping_function(test_df, label="Sex")
        >>> print(mapping)
        {0: 0, 1: 1}

        See Also
        ---------
        face2face.statistics.null_modell.configuration_model_label_z_score_mixing_matrix
        face2face.statistics.null_modell.shuffle_label_z_score_mixing_matrix

        """
```

face2face.statistics.null_modell.mapping_function

face2face.statistics.null_modell.mapping_function(*Data*, *label='type'*)

Creates a mapping

Creates a mapping dictionary to create the attribute mixing matrix dynamic, independent of the amount of attributes.

Parameters

- **Data** (*Data*) – Data Object that contains Tij- and Metadata for a data set.
- **label** (*str*) – A string that tells the function which attribute should be mapped.

Returns

mapping – A dictionary that contains the dimensions of the given attribute.

Return type

dict

Examples

Based on the amount of attribute values the used attribute has, you have to create a mapping for the null model functions.

```
>>> mapping = mapping_function(test_df, label="Age")
>>> print(mapping)
{0: 0, 1: 1, 2: 2}
```

```
>>> mapping = mapping_function(test_df, label="Sex")
>>> print(mapping)
{0: 0, 1: 1}
```

See also

face2face.statistics.null_modell.configuration_model_label_z_score_mixing_matrix() ,
face2face.statistics.null_modell.shuffle_label_z_score_mixing_matrix()

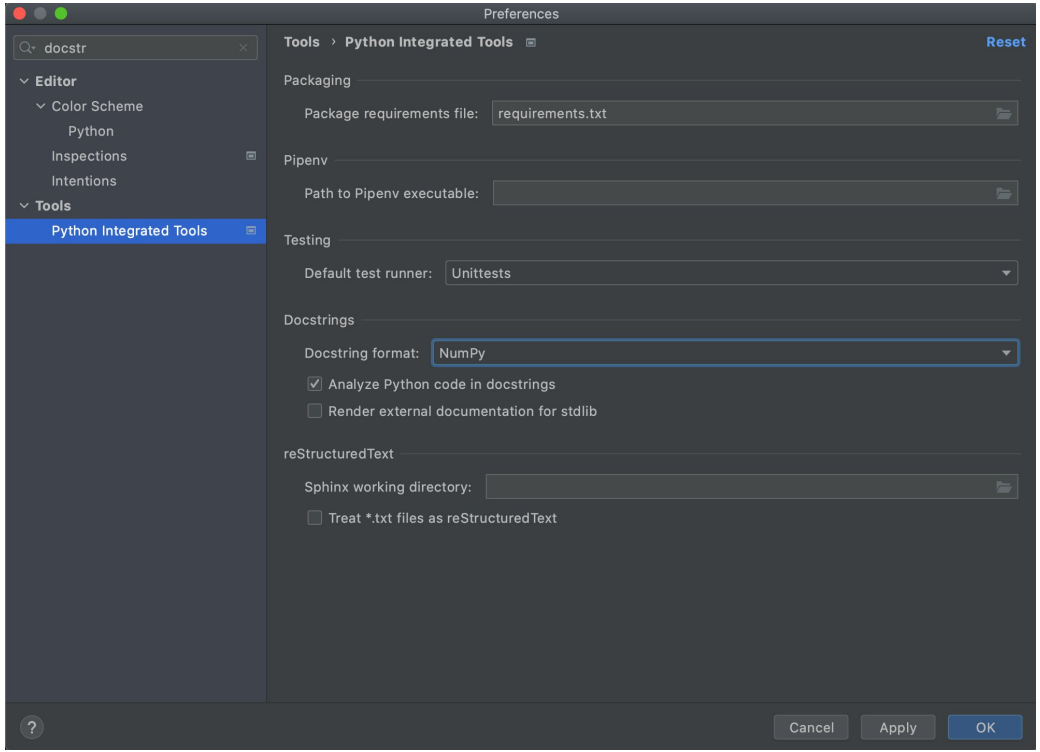
Previous Next

https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html#example-numpy

03

Docstrings in PyCharm

3.1 Template wählen

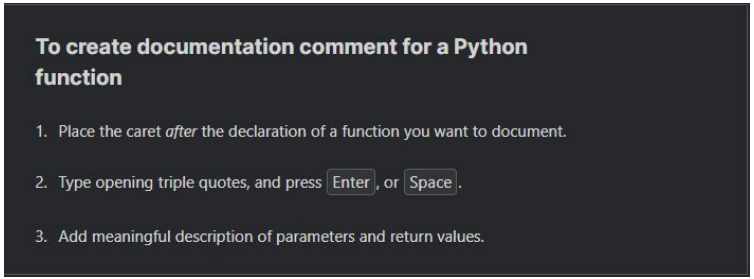


1. Einstellungen öffnen
2. nach “docstring” suchen
3. Docstring format in NumPy ändern

https://www.jetbrains.com/help/pycharm/creating-documentation-comments.html

3.2 Docstring schreiben

Technology
Arts Sciences
TH Köln



To create documentation comment for a Python function

1. Place the caret *after* the declaration of a function you want to document.
2. Type opening triple quotes, and press `Enter`, or `Space`.
3. Add meaningful description of parameters and return values.

https://www.jetbrains.com/help/pycharm/creating-documentation-comments.html

BONUS

How To Commit

A properly formed Git commit subject line should always be able to complete the following sentence:

- If applied, this commit will *your subject line here*

For example:

- If applied, this commit will *refactor subsystem X for readability*
- If applied, this commit will *update getting started documentation*
- If applied, this commit will *remove deprecated methods*
- If applied, this commit will *release version 1.0.0*
- If applied, this commit will *merge pull request #123 from user/branch*