Getting started with pytesting

Quick start

Installation

Install pytest. e.g., from the "dev" dependencies

```
pip install ".[dev]"
```

How to use

To execute the tests run e.g.

```
pytest
```

Test development tips

Folder and test naming

1. The tests for functions in `<filename>.py` should go in `tests/test_<filename>.py`

   e.g., the tests for python_package/mockup.py are in tests/test_mockup.py

2. The test names should start with `def test_<corresponding_function_name> ...`

   e.g., `def test_hello_world(): ...`

Some Pytest decorators

1. To indicate that the test function is expected to fail you can prepend

```
@pytest.mark.xfail(raises=TypeError)
def test_hello_world_str(): ...
```

2. To setup and cleanup any resources for a test you can use pytest fixtures with `yield`

```
@pytest.fixture
def temp_file():
    # set up
    < code to create a file>
    # return
    yield
        the_file
    # clean up
    < code to remove the file>
```

Doctests

You can also include tests in your docstrings using `>>>` followed by the expected result e.g.

```
def hello_world(n):
    """
    Prints 'hello world' n-times.
        ...

    Examples
    --------
    >>> hello_world(3)
    'hello world hello world hello world'
        ...
    """
```

*Needs `addopts = --doctest-modules` in pytest.ini

Skipping in doctests

If you know that the test cannot succeed but would like to include an example usage in the docstring still then you can add `# doctest: +SKIP` e.g.

```
def saved_world(filename):
    """
    Count how many times 'hello world' is in a file.
        ...

    Examples
    --------
    >>> saved_world("not-real.txt") # doctest: +SKIP
        ...
    """
```