pytest_cheat_sheet.md

pytest basic usage reference card

testing you have pytest installed and working

- at the command line issue type command:

```
pytest --version
```

  this should produce a response starting **This is pytest version....**

writing pytest tests

- The Python file for a test must be called `test_`*something*`.py` or *something*`_test.py`
- Use `import` to access the function to be tested:

```
from somewhere import something
```

  this will import the function `something` from the Python file `somewhere.py`
- pytest tests are functions whose name starts with `test_` this should be followed by a human readable description of what is tested. Within the function use `assert` to perform the test:

```
def test_description_of_what_is_tested():
    assert something(some_param=some_value) == known_return_value
```

  where `known_return_value` could be a single value or a list or something else depending on the test.
- to compare floating values use `pytest.approx` to avoid rounding errors:

```
assert 0.1 + 0.2 == pytest.approx(0.3)
```

  will always be O.K.. You will need to `import pytest` at the top of the test file.
- To test that a particular exception is raised then use the following:

```
def test_something_X_ raises_exception_Y():
    with pytest.raises(ValueError):
         something(some_param=illegal_value)
```

  this will only pass if a ValueError is raised in the call.

running pytest tests.

- To run a particular test file:

```
pytest test_something.py
```

  this will run all the tests in `test_something.py`
- To run a subset of tests then use the `-k EXPRESSION` option:

```
pytest -k input test_something.py
```

  this will run just the tests in `test_something.py` that have a substring input.
- to run all tests with a directory (and subdirectories) then cd into and:

```
pytest
```

- to see a shorter output from pytest use the `-q` option
- to see a longer output from pytest use the `-v` option.