1.5 Testing Your Work

- Last step of function design recipe
- Super important ;)

DOCTESTS:

- Manually copying examples and comparing outputs = time consuming & error prone!!!!!
- The Python library doctests :
  1. examples in docstrings
  2. converts them automatically to runnable tests
- Con: file too long for multiple tests
- Add this to use it :

```
if __name__ == '__main__':
    import doctest      # import the doctest library
    doctest.testmod()   # run the tests
```

CREATING TEST SUITES/ UNITTESTS:

- Using pytest
- Mainly used in the course!
- Tests in separate file
- Important:
  1. Function that starts with "test" is a separate test (run independently & random order)
  2. Assert statement: action that verifies correctness of the code
     - assert <expression>
     - The expression is boolean (True = test passes , False = test fails)
     - **Pro**: tests usually straight-forward, **Con** : choosing and implementing test cases is time-consuming.

CHOOSING TEST CASES:

PROPERTIES OF INPUTS:

- Intergers: 0, 1, positive, negative, "small", "large"
- Lists: empty, length 1, no duplicates, duplicates, sorted, unsorted
- Strings: empty, length 1, alphanumeric characters only , special characters like punctuation marks.

- When functions have more than one input... tests also based on RELATIONSHIPS

PROPERTY-BASED TESTING:

- Large set of possible inputs generated in a programmic way

- Pro: short about of code -> lots of inputs ———> with hypothesis library
- Con: not always easy to know what the corresponding output should be
- *Knowing the types of parameters & and what function does, what should the OUTPUTS be based on this?*