5. Skip/Mark and Pytest Options:

- **Skip test function:**
  - Skip simple: @pytest.mark.skip()
  - skip with condition @pytest.mark.skipif(condition)
  - Skip whole module/file:
    - pytestmark=pytest.mark.skipif(condition)

- **Markers:**
  - Use markers for grouping our tests.
  - A test can have more than one marker, and a marker can be on multiple tests.
  - E.g. marking a test:
    - @pytest.mark.sanity, @pytest.mark.uitest, etc
  - While running tests, use pytest -m sanity, to run only test marked with "sanity"
  - Markers running supports: and, or, not, parentheses operators.
    - E.g. pytest -m "sanity and uitest"
  - Can define markers in module/file level:
    - pytestmark=[pytest.mark.smoke, pytest.mark.sanity]
  - For stopping the warnings, need add/define the markers in pytest.ini in root folder

5. Skip/Mark and Pytest Options:

- **Markers-xfail: Marking Tests as Expecting to Fail:**
  - We can use the xfail marker to indicate that you expect a test to fail.
  - An xfail means that you expect a test to fail for some reason. A common example is a test for a feature not yet implemented, or a bug not yet fixed. The output will be XFAIL
  - When a test passes despite being expected to fail (marked with pytest.mark.xfail), it's an XPASS output.
  - Defining at test level: @pytest.mark.xfail
  - The outputs will be XFAIL or XPASS, based on the tests. This test will run but no traceback will be reported when it fails. Instead, terminal reporting will list it in the "expected to fail" (XFAIL) or "unexpectedly passing" (XPASS) sections.
  - Supports condition also, e.g. @pytest.mark.xfail(condition)
  - Also supported specifying a "raises" argument e.g. @pytest.mark.xfail(raises=IndexError)
  - **Note:** pytest.raises() vs xfail: Using pytest.raises() is likely to be better for cases where you are testing exceptions your own code is deliberately raising, whereas using @pytest.mark.xfail with a check function is probably better for something like documenting unfixed bugs (where the test describes what "should" happen) or bugs in dependencies.

- **Running tests using test-name(-k option):**
  - The -k option enables you to pass in an expression to run tests that have certain names specified by the expression as a substring of the test name.
  - You can use: and, or, and not in your expression to create complex expressions.
  - Works for both module/file name and test-name.

5. Skip/Mark and Pytest Options:

- **Cmd line options:**
  - See all options with pytest --help.
  - Few of them below:
    - -m MARKEXPR only run tests matching given mark expression.
    - -x, --exitfirst, exit instantly on first error or failed test.
    - --tb=no, no traceback
    - --maxfail=num exit after first num failures or errors.
    - -q, --quiet decrease verbosity.
    - --collect-only only collect tests, don't execute them.
    - --version display pytest lib version and import information.

- **Test Outcomes:**
  - PASSED (.): The test ran successfully.
  - FAILED (F): The test did not run successfully (or XPASS + strict).
  - SKIPPED (s): The test was skipped. You can tell pytest to skip a test by using either the @pytest.mark.skip() or pytest.mark.skipif() decorators, discussed in Skipping Tests .
  - xfail (x): The test was expected to fail, so ran and failed. You can tell pytest that a test is expected to fail by using the @pytest.mark.xfail() decorator.
  - XPASS (X): The test was not supposed to pass but ran and passed.
  - ERROR (E): An exception happened outside of the test function, in either a fixture.