6. Parameterize and Setup/TearDown Tests

- **Parametrized Tests:**
  - Parameterized tests allow a developer to run the same test over and over again on different values.
  - Parameterized testing is when we want to do data driven testing. For example, we want to test the login page or process, with multiple input values for e.g. say 1000 different username and password conditions.
  - This lets you increase the coverage without having to add much more test code.
    - E.g. @pytest.mark.parametrize("test_input,expected", [("5+5", 10), ("5-5", 0), ("7*8", 56)])

- **Fixtures:**
  - Fixtures is used to get data ready for our tests. Fixtures are functions that are run by pytest before (and sometimes after) the actual test functions.
  - We can use fixtures to get a data set for the tests to work on. For e.g. you want to setup DB connection, or initialize webdriver and test browser if talking in terms of selenium UI tests.
  - We can put fixtures in individual test files if we want the fixture to only be used by tests in that file.
  - Proper way of use is to have fixtures in conftest.py to be used by multiple test files.
  - Fixture functions can have a any name.
  - Call fixture by using in test-function argument. e.g. def test_someTest(fixture_name)
  - Mark fixture using @pytest.fixture()
  - Its not mandatory but we can return something from fixture using return statement.

6. Parameterize and Setup/TearDown Tests

- **Fixtures:**
  - Also another way of calling fixture:
    - @pytest.mark.usefixtures("fixture_name")
    - In this case, return from fixture cannot be used. So only use it to execute some code within fixture.

- **Setup/Tear down and using multiple fixture:**
  - "yield": By using a yield statement instead of "return", all the code after the yield statement serves as the teardown code and are run.
  - We can have code to shutdown connections, etc after the "yield" stmt.
  - We can use/pass multiple fixture in same test function.

- **Sharing Fixtures:**
  - Use conftest.py - put the fixtures to share across multiple test files.
  - We can have more conftest.py files in subdirectories of the top tests directory. If we do, fixtures defined in these lower-level conftest.py files will be available to tests in that directory and subdirectories.
  - Don't import conftest from anywhere. The conftest.py file gets read by pytest, and is considered a local plugin.
  - Function- pytest_configure, vars within this are available within all test functions in that module.

6. Parameterize and Setup/TearDown Tests

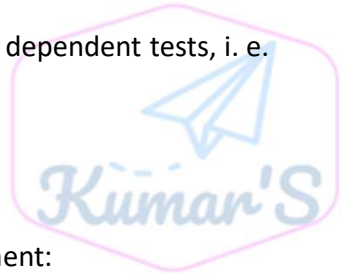
- **Tracing Fixtures:**
  - Use this option in pytest --setup-show, to trace fixture execution.

- **Introspecting The calling Test Function:**
  - Use special fixture "request“.
  - The "request" in the fixture parameter is another builtin fixture that represents the calling state of the fixture.
  - Useful in looking into the calling test function and make decision based on that.

- **Factories as Fixtures:**
  - The “factory as fixture” pattern can help in situations where the result of a fixture is needed multiple times in a single test.
  - Instead of returning data directly, the fixture instead returns a function which generates the data. This function can then be called multiple times in the test.
  - We return the function_name **without** the parenthesis. A reference to the function is sent to the calling code.

6. Parameterize and Setup/TearDown Tests

- **Parametrizing using Fixtures:**
  - Fixture functions can be parametrized in which case they will be called multiple times, each time executing the set of dependent tests, i. e. the tests that depend on this fixture.
  - We can do Data driven tests from Fixtures.
  - Pass params to fixture decorator.
  - We use the same special fixture "request" to access the params from fixture and return.
  - It is possible to customize the string used in a output testID for a certain fixture value by using the ids keyword argument:
  - You can use multiple fixtures also. In this case, the number of tests will be combination of each params fo the fixtures.



- **Fixture Scope:**
  - Fixtures also have scope and lifetime.
  - The default scope of a pytest fixture is function scope.
    - function: Runs once per test, the default value of the fixture scope. Fixture is destroyed at the end of the test.
    - class: Runs once per class of tests
    - module: Runs once per module
    - session: Runs once per session