1. **Run pytest using command line**.
   Go to python project -> Type cmd -> venv\scripts\activate -> pytest

2. pytest : - Run all the test cases in all the package

3. pytest -v: Verbose mode (detailed view)

4. pytest automation\test_module01.py:--> Run a single test file

5. pytest automation\py_assertions -> Run whole directory.

6. pytest -v automation\test_module02.py::TestMyStuff ->Run Test Class

7. pytest -v automation\test_module02.py::TestMyStuff::test_strs -> Run Test Method of a Test Class

8. pytest -v automation\test_module01.py::test_a1 -> Run Test Function of a TestModule

9. Test can have multiple markers. A marker can be on multiple tests

10. pytest -m <markname> -> Run only marked test

11. pytest -v -m sanity -> Run only sanity test.

12. pytest -v -m "sanity and not str" -> Run only sanity test not str

13. pytest -v -m "sanity and not str" -> Run only sanity test

14. pytest -v -m "not sanity" automation\test_markers.py -> Run all the test except sanity

15. pytest -v -m "sanity and str" -> Run sanity and str

16. pytest -v -m "sanity or str" -> either sanity or str or both

17. pytest -v -m "smoke" ->Marking whole class or modules level

18. Define markers in config file [pytest.ini file in root folder] to avoid warning

19. xPASS: test passed despite being expected to fail (marked with pytest.mark.fail)

20. xfail -> It should be failed only when the condition is true

21. pytest -v -k "module" -> Run test case using substring.

22. pytest -v -k "module" --tb=no-> option "--tb=no" for short output

23. pytest -v -k "case" --tb=no-> -k filter out module or test name

24. pytest -v -k "case or str" --tb=no -> select both case or str

25. pytest -v -k "case or xfail" --tb=no -> select both case or xfail

26. pytest -v -k "module and not case" --tb=no -> do not run case test

27. pytest -v -k "module and not case" --tb=no -x -> option: -x stop after first failure(Existed on first test failed)

28. pytest -v -k "module and not case" --tb=no --maxfail = 2 -> option: 'maxfail' stop after specified ' failing

29. pytest -v -k "module and not case" --tb=no -q -> option: -q, --quiet (decrease verbosity) [useful actual testing]

30. pytest -v -k "module and not case" --tb=no --collect-only -> option: --collect-only, --co (only collect the tests, do not execute them, no run, just see the tests pytest discovered)

31. Testing Ordering: - Pytest run tests in the order they appear in a module

32. pytest -v -k "module and not case" --tb=no --lf -> option: --lf, --last-failed run only the tests that failed at the last run (or all if none failed)

33. pytest -v -k "module and not case" --tb=no --ff -> option: --ff, --failed-first run all the tests but run the last failure first

34. pytest --help -> find out all the options

35. pytest -v --tb=no -> All the executed test cases (passed or failed

    - PASSED(.): The test ran successfully.
    - FAILED(F): the test did not run successfully.
    - SKIPPED(s): the test was skipped.
    - XFAIL(x): the test was not supposed to pass, ran, and failed.
    - XPASS(X): the test was not supposed to pass, rand and passed.
    - ERROR(E): An exception happened outside of the test function.

36. Skip all tests in a module unconditionally with: pytestmark = pytest.mark.skip("skipping all tests")

37. With the -k option, you can use and, or not and parentheses.

38. Fixtures: Function that are run by pytest before (and sometimes after) the actual test functions Eg.Setup DB Connection, or initialize webdriver

39. Can put fixturs in individual test files or in confest.py for making fixtues available in multiple test files.

40. Fixture Basics
    conftest.py: -> Share fixtures across multiple tests. Can have single confest.py in centralized directory for all test to access the fixtues. Also, can have other conftest.py files in subdirectories. It should not be imported by test files.

41. conftest.py is python module. no need imported by test files.
    pytest -v -k test_fixtures03 --setup-show
    M setup01 -> Module run first
    F setup02 -> Run After Module

42. 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 be then called multiple times in test.

43. Single Fixture parametrized works in most cases

    - Fixtures Scopes: Fixtures are created when first request by a test and are destroyed based on their scope.
    - function: the default scope, the fixture is destroyed at the end of the test.
    - class: the the default scope, the fixture is destroyed during teardown of the last test in the class
    - module: the the default scope, the fixture is destroyed during teardown of the last test in the module
    - package: the the default scope, the fixture is destroyed during teardown of the last test in the package
    - session: the the default scope, the fixture is destroyed during teardown of the last test in the session

    PASS custom arg to test , run on multiple environments*

44. pytest -v automation\test_argstest.py --cmdopt=Prod -s #pass Argument from command line

45. Configuring Pytest Explantion
    Almost all projects need some kind of configuration parameters. For e.g. in test automation project, we need to pass different URL for QA and production testing Configuration should not be hardcoded config files in diff format - json, yaml or ini files