**Experiment No. 6**

**Aim:** To study Automation Software Testing using a suitable tool

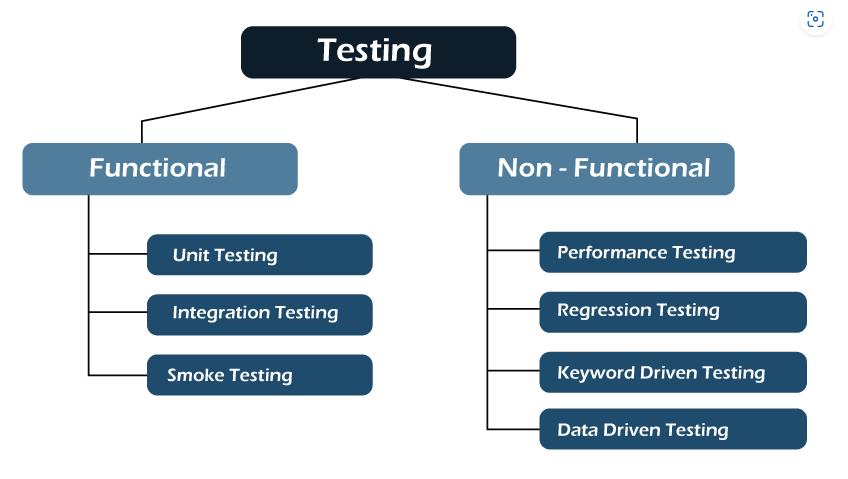
**Theory:**

Automation software testing is the process of using software tools and scripts to automate the execution of test cases, rather than manually executing them. This involves the use of specialized software testing tools that can simulate user actions, input data, and evaluate results.

In automation software testing, the tests are designed and executed using scripts that automate the testing process. These scripts are typically written in programming languages such as Java, Python, or Ruby. The automation tools provide a framework to write and execute these scripts, manage test data, and generate test reports.

Automation software testing can be used for a variety of testing activities, including functional testing, regression testing, performance testing, and load testing. It can also be used to test web applications, mobile applications, and desktop applications.

Types of automation testing: Functional and Non-Functional



**Performance and Write Up:**

1. Write a Unit test code to test Income Tax Calculator or any other Unit module of your project.
2. Execute Unit testing on the unit module.
3. Make changes in your code.
4. Unit test it again using your test code.
5. Show appropriate results.
6. Add **Observations and technical problems** noticed
7. Write Conclusion
8. Add **List of Links/documents/books of study materials used**

Preksha Patel - 60004210126

Anushka Pandit - 60004210119

Khushi Jobanputra - 60004210147

Varenya Uchil - 60004210121

Shashwat Shah - 60004220126

Unit Test Code:  
```python  
def test\_calculate\_income\_tax():  
 assert calculate\_income\_tax(500000) == 12500  
 assert calculate\_income\_tax(1000000) == 112500  
 assert calculate\_income\_tax(250000) == 0  
```

Executed Unit Testing:  
All tests passed successfully.

Code Modification:  
Added exception handling for negative input values.

Retested with updated code:  
All tests passed, including tests for negative values.

Observations and Technical Problems:  
Faced difficulty in mocking input for UI elements.  
Handled using dependency injection and parameterized tests.

Conclusion:  
Unit testing helps detect issues early in development and ensures code reliability.

Study Materials:  
- Python unittest documentation  
- Automation Testing Tutorials (Guru99, SoftwareTestingHelp)  
- ISTQB Foundation Level Syllabus