Intro To Software Testing

Agenda

1. What is Software?
2. What is testing and software testing
3. Why Testing is Important
4. What is Bug and How to find Bugs
5. Type of Errors in SW
6. Testing Specification
7. Types of SW Testing in SW Engineering
8. Testing Levels
9. Testing Methodology
10. SDLC and STLC
11. V model

What is software?

Computer Programs

+

associated

^ Documentation

Documentation Types

- User Documentation
- Technical Documentation
- Marketing Documentation

What is Software Testing

- **Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free

Why Testing is Important?

- To find and correct defects.
- To Check whether the user’s needs are satisfied.
- To avoid user detecting problems.
- Also, to provide quality products.

Objectives

1. Ensure requirement matching with customer needed.
2. Define defects.
3. Prevent defects.
4. Ensure best quality.
5. Generate High quality test cases.

Why does SW have Bugs?

- Miscommunication or No Communication.
  - That we are not clear about what an application should do or shouldn’t de.
- Time Pressure.
- Programming Mistakes.
- Changing Requirements.

SW Testing Misunderstanding

- Testing is debugging.
- If programmers were more carful, testing would be unnecessary.
- Testing activities start only after the coding is complete.
- Testing never ends.
- Testing is not a creative task.
- Manual testing only.

What is Bug and How to find Bugs

A Bug is the deviation of the actual result from the expected result

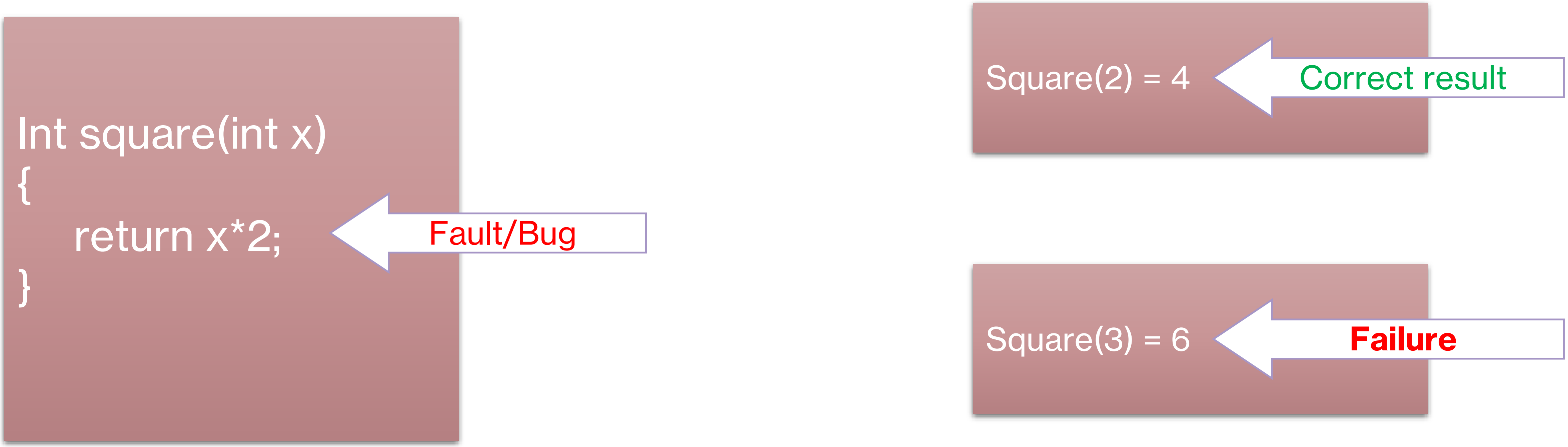
Type of Errors in SW

A person makes an Error That creates a fault in software That can cause a failure in operation

- **Error:**
  - An error is a human action that produces the incorrect result that results in a fault.
- **Bug:**
  - The presence of error at the time of execution of the software.
- **Fault:**
  - State of software caused by an error.
- **Failure:**
  - Deviation of the software from its expected result. It is an event.

Type of Errors in SW

For any integer n, square (n) = n*n



```
Int square(int x)
{
    return x*2;
}
```

Fault/Bug

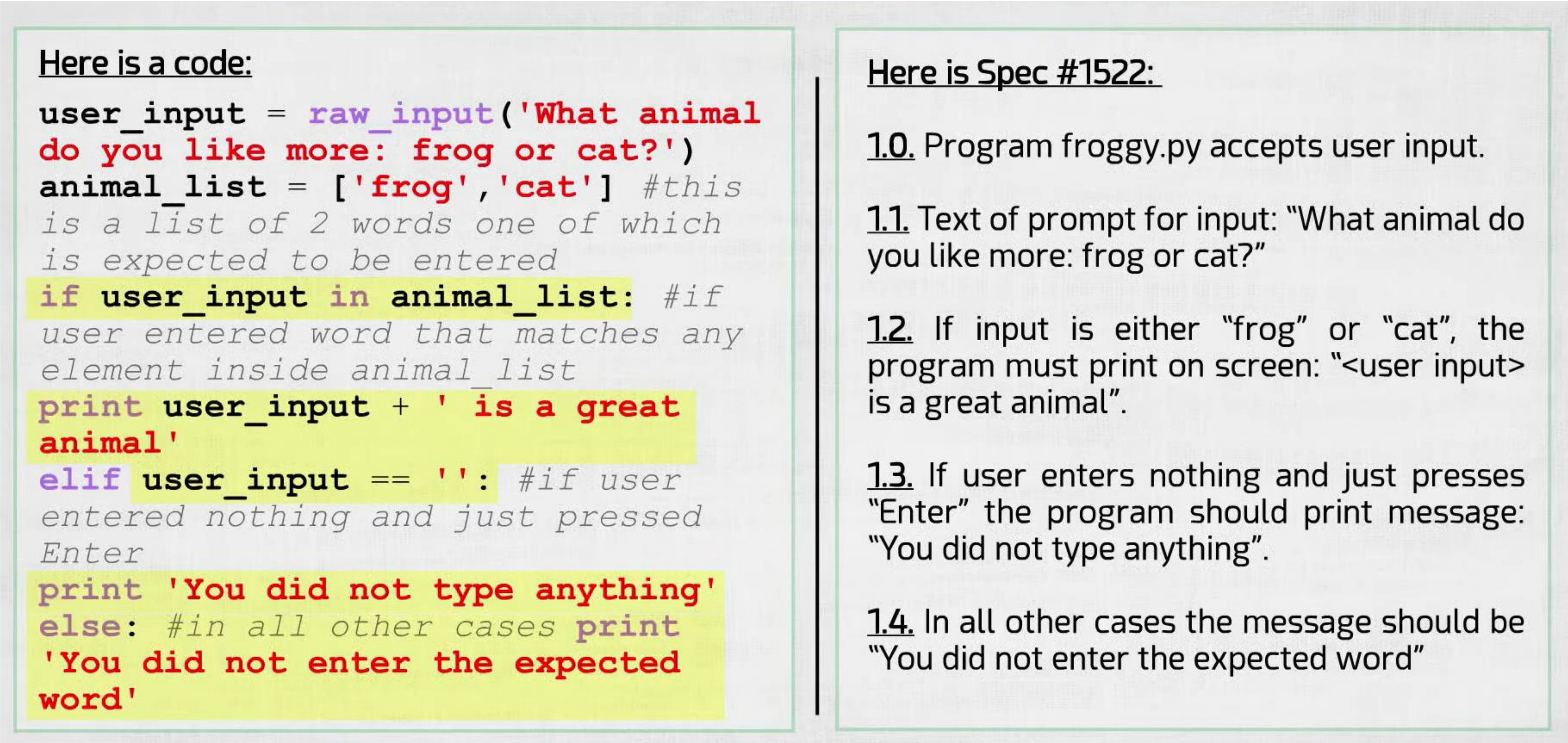
Square(2) = 4

Correct result

Square(3) = 6

Failure

Testing Specification



Here is a code:

```
user_input = raw_input('What animal
do you like more: frog or cat?')
animal_list = ['frog','cat'] #this
is a list of 2 words one of which
is expected to be entered
if user_input in animal_list: #if
user entered word that matches any
element inside animal_list
print user_input + ' is a great
animal'
elif user_input == '': #if user
entered nothing and just pressed
Enter
print 'You did not type anything'
else: #in all other cases print
'You did not enter the expected
word'
```

Here is Spec #1522:

1.0. Program froggy.py accepts user input.

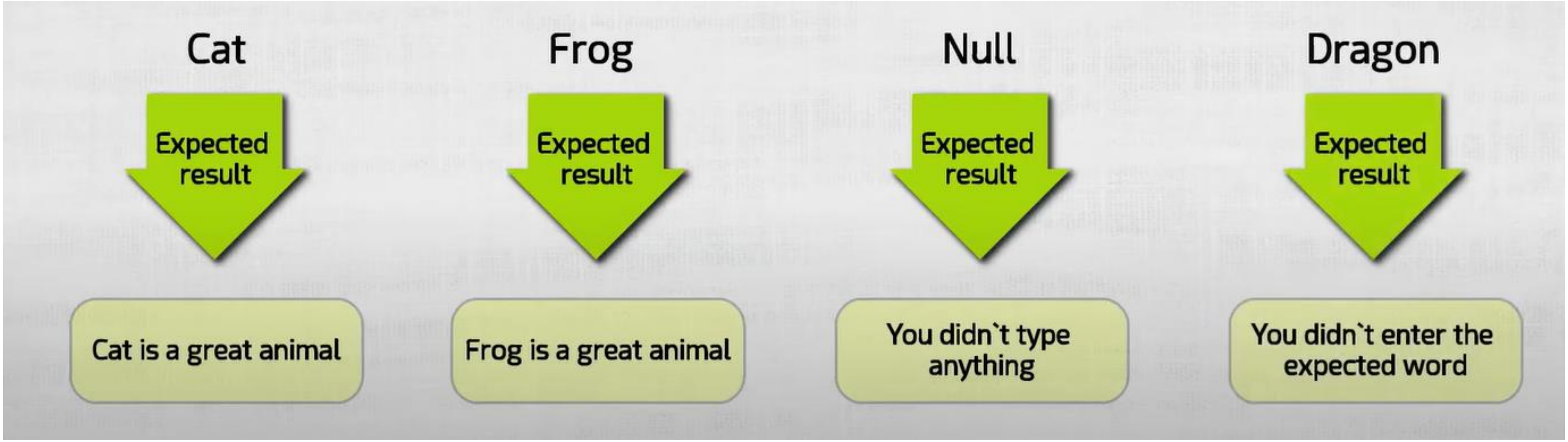
1.1. Text of prompt for input: "What animal do you like more: frog or cat?"

1.2. If input is either "frog" or "cat", the program must print on screen: "<user input> is a great animal".

1.3. If user enters nothing and just presses "Enter" the program should print message: "You did not type anything".

1.4. In all other cases the message should be "You did not enter the expected word"

Test Cases



Cat

Expected result

Cat is a great animal

Frog

Expected result

Frog is a great animal

Null

Expected result

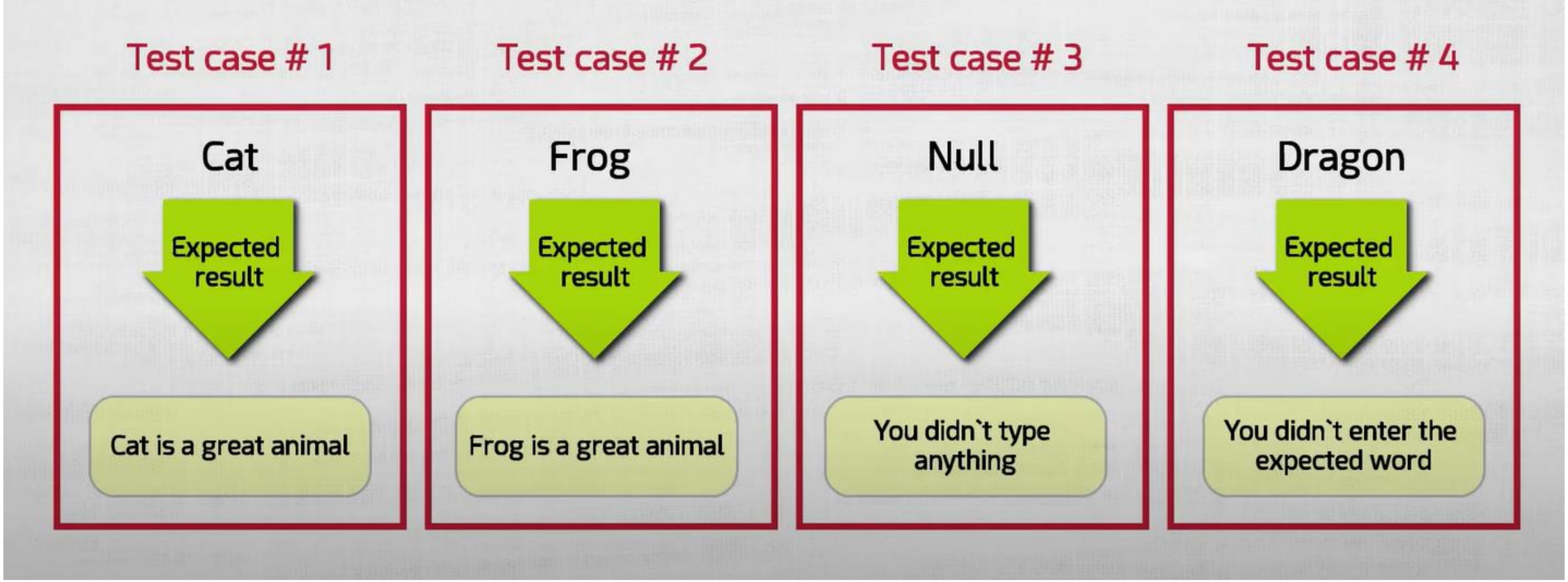
You didn`t type anything

Dragon

Expected result

You didn`t enter the expected word

Test Cases No.



Test case # 1

Cat

Expected result

Cat is a great animal

Test case # 2

Frog

Expected result

Frog is a great animal

Test case # 3

Null

Expected result

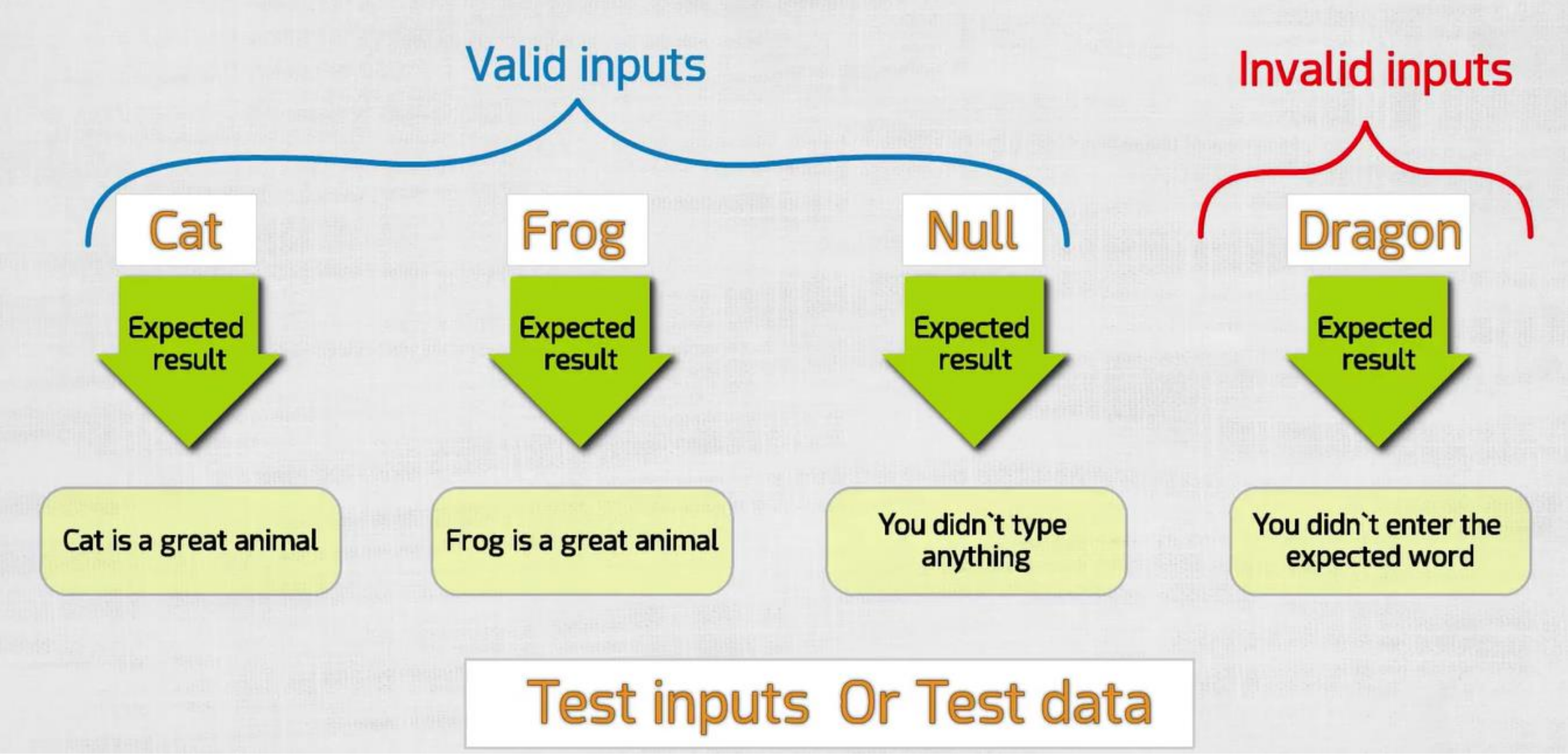
You didn`t type anything

Test case # 4

Dragon

Expected result

You didn`t enter the expected word



Valid inputs

Invalid inputs

Cat

Frog

Null

Dragon

Expected result

Expected result

Expected result

Expected result

Cat is a great animal

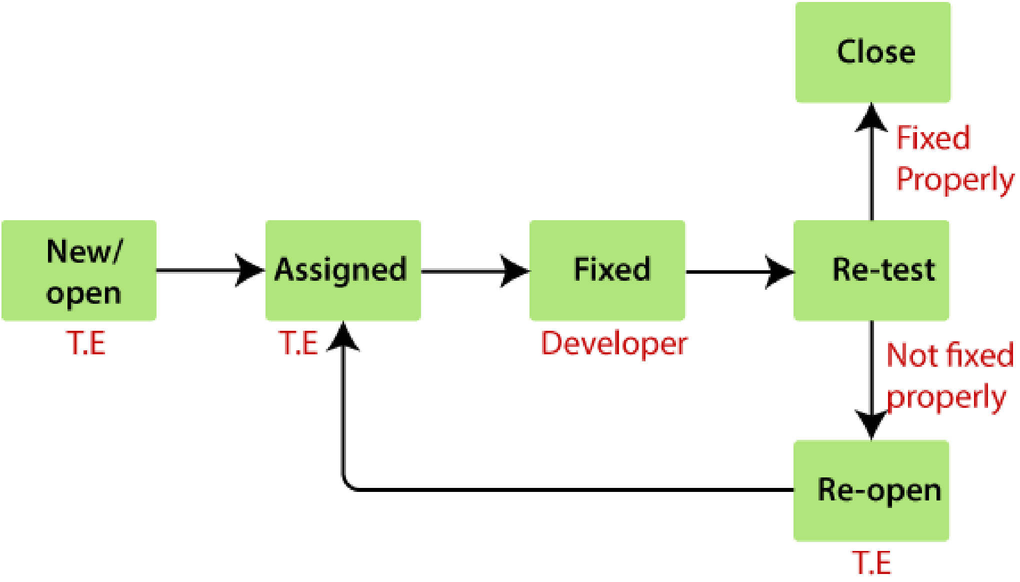
Frog is a great animal

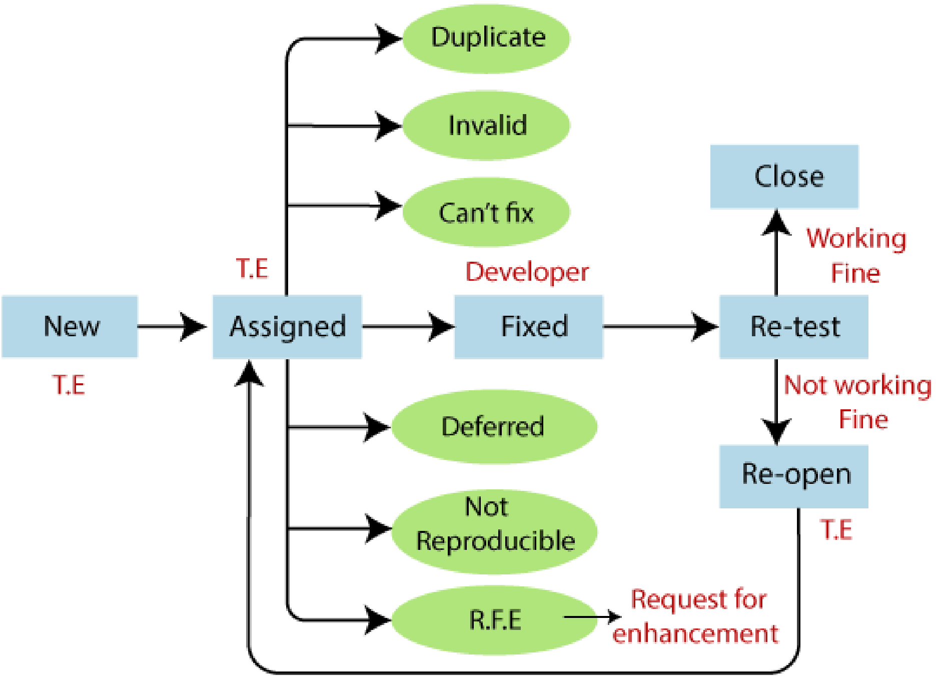
You didn`t type anything

You didn`t enter the expected word

Test inputs Or Test data

Bug Life Cycle





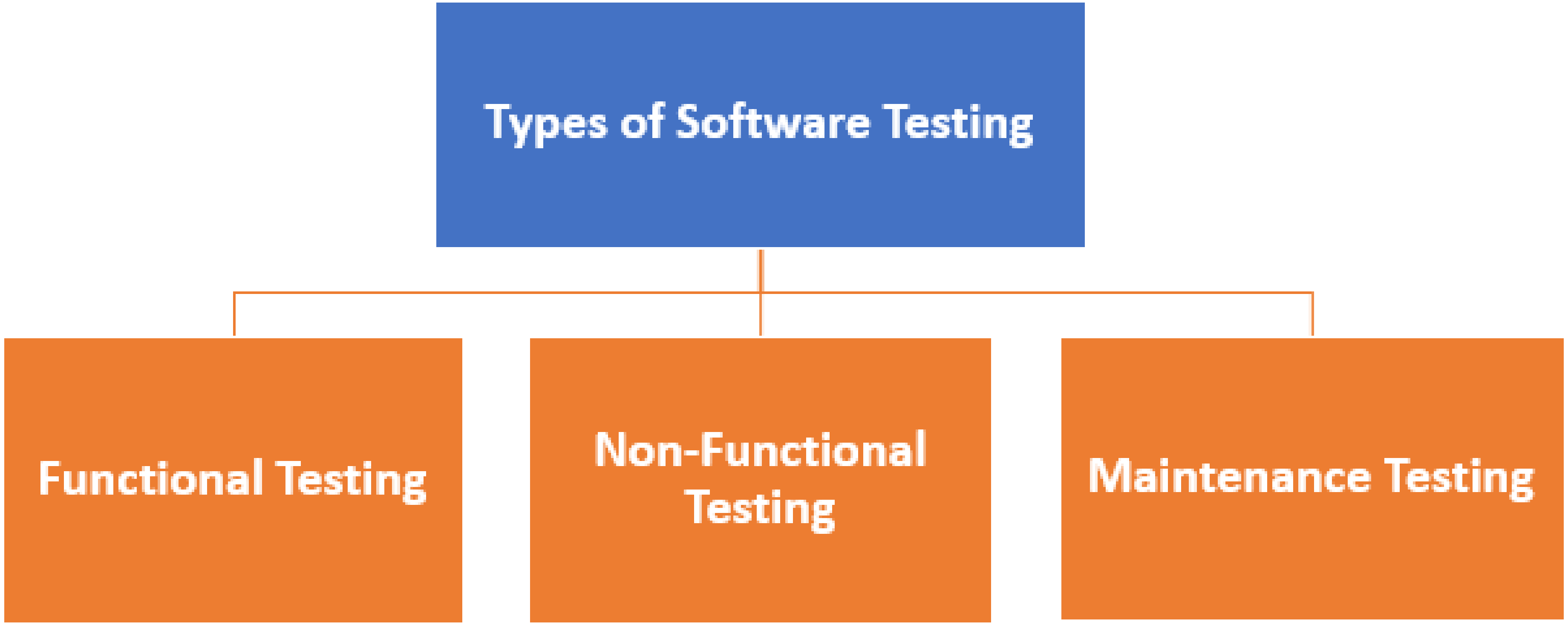
Software Free Error

Tester must make sure that 100% of the software work fine and error free

- Whether we like it or not, there is always a probability that bugs will be missed by testers.

- Testing cannot cover 100% of the possibilities of how software can operate.

Types of SW Testing



Types of Software Testing

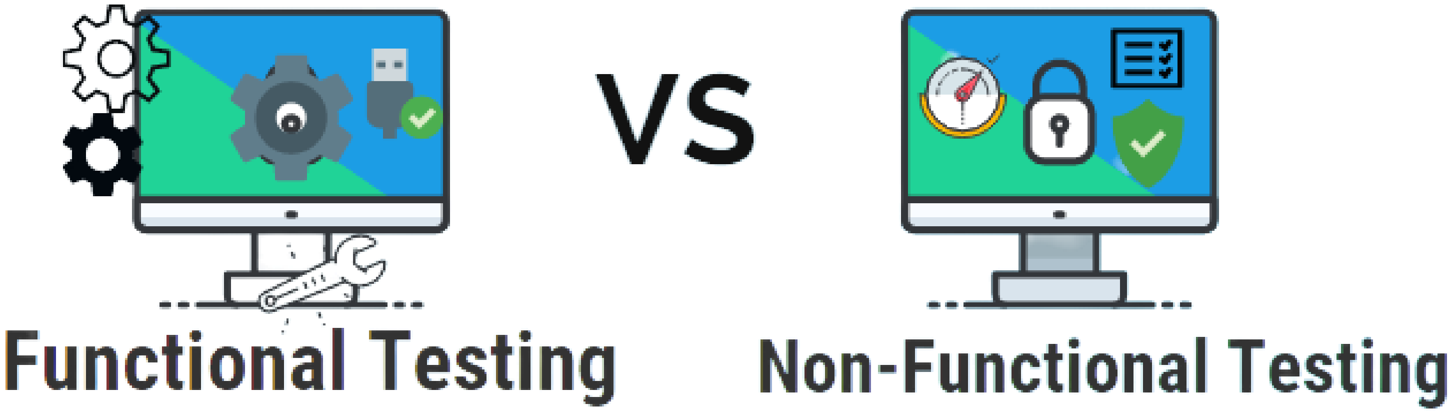
Functional Testing

Non-Functional Testing

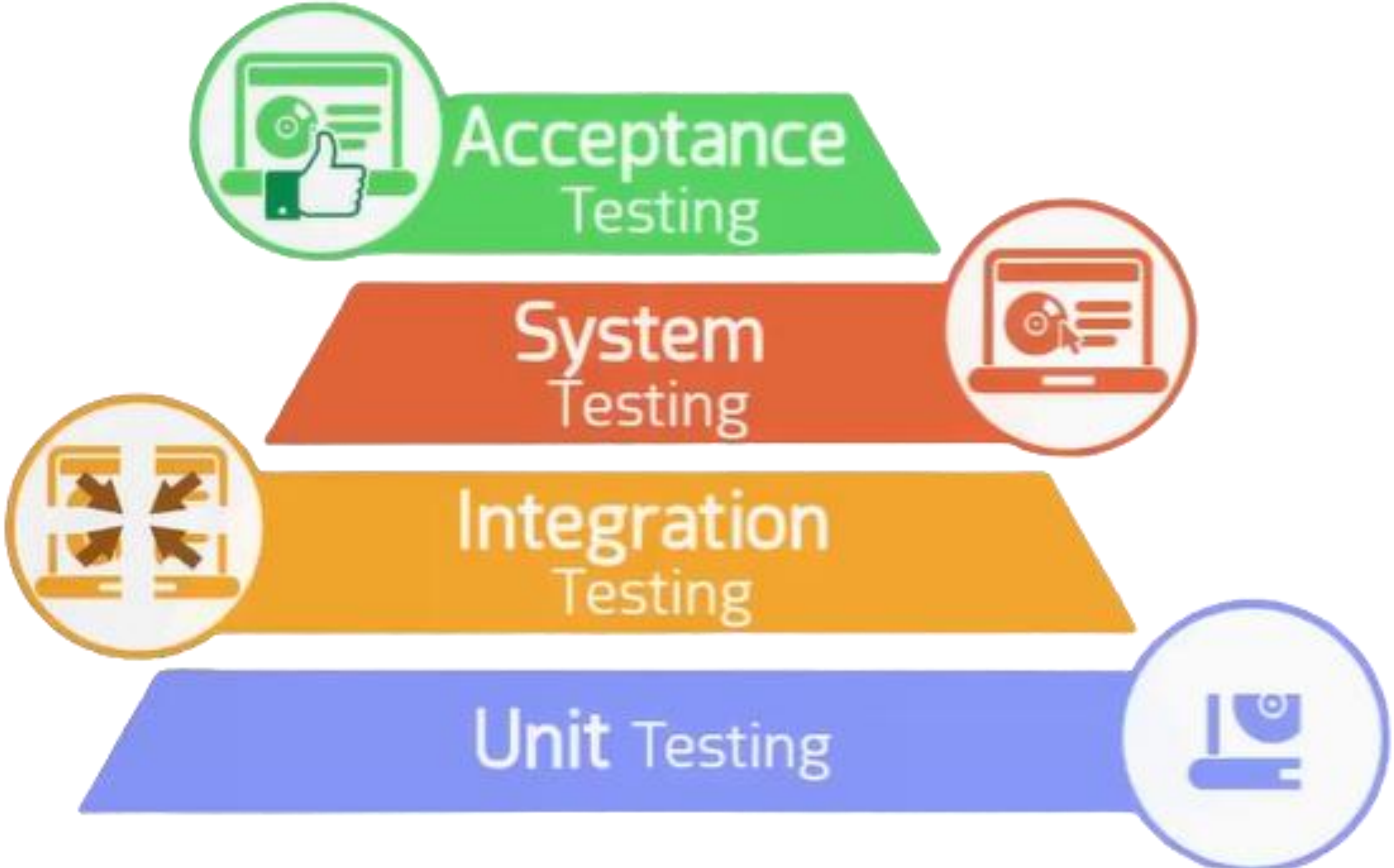
Maintenance Testing

Types of SW Testing in SW Engineering

- **Functional Testing**
  - Unit Testing
  - Integration Testing
  - UAT (user accepted testing)
- **Non-Functional Testing**
  - Performance
  - Usability
  - Scalability
- **Maintenance**
  - Regression
  - Maintenance



Level of Testing



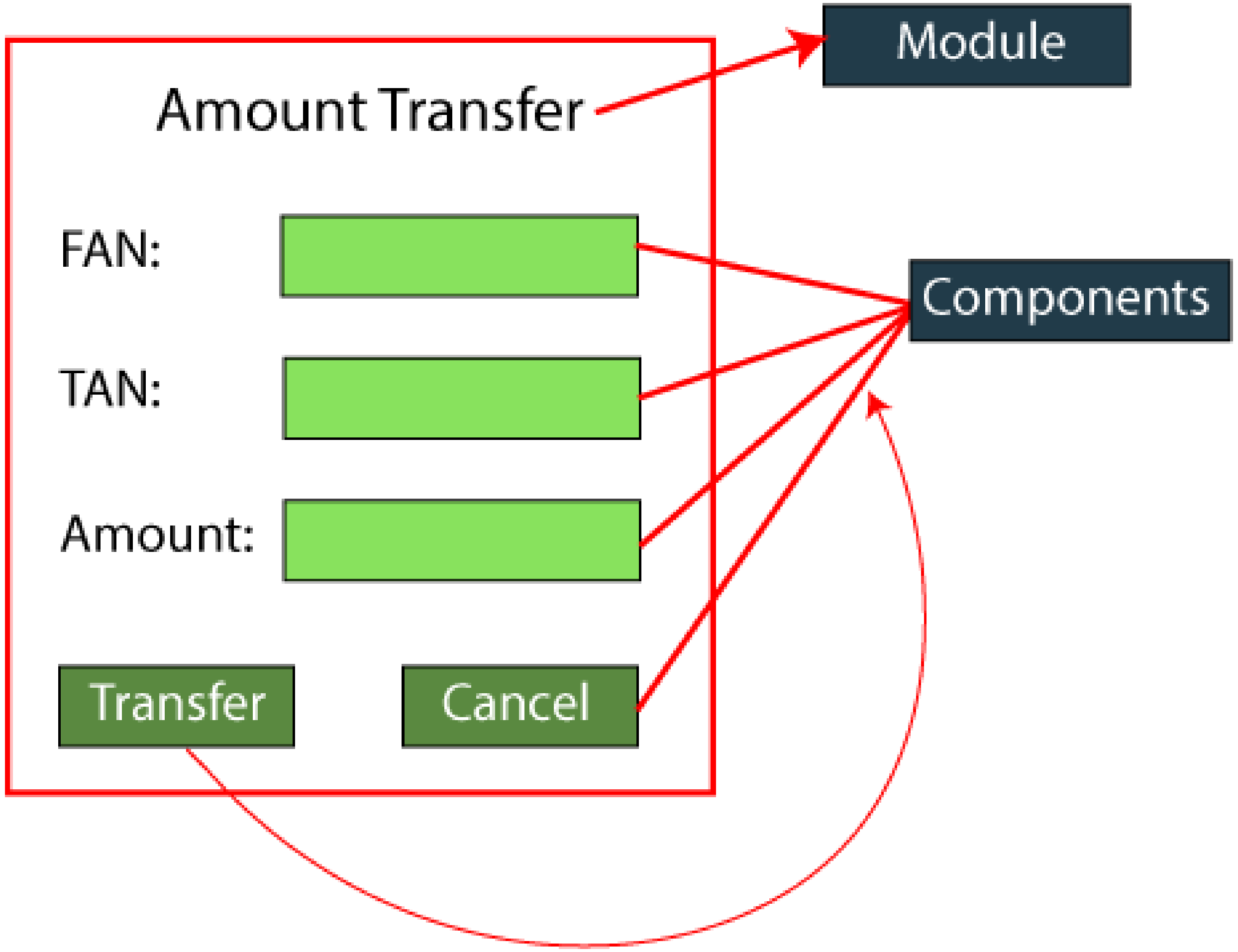
Acceptance Testing

System Testing

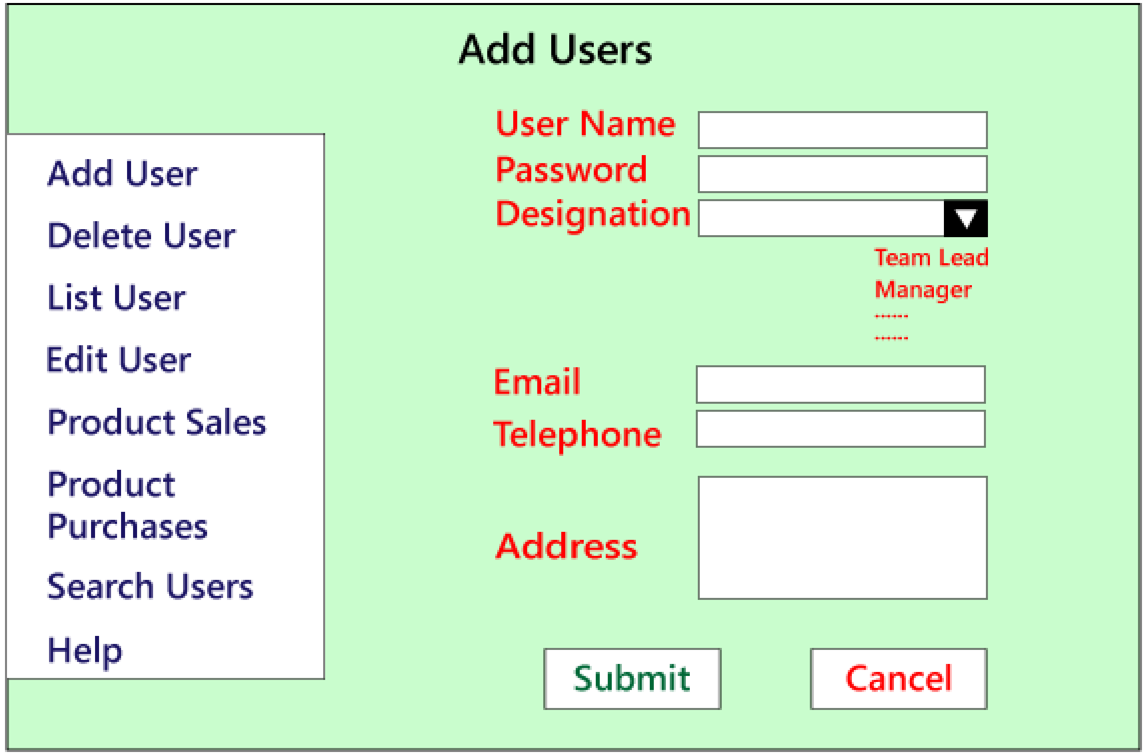
Integration Testing

Unit Testing

Unit testing example

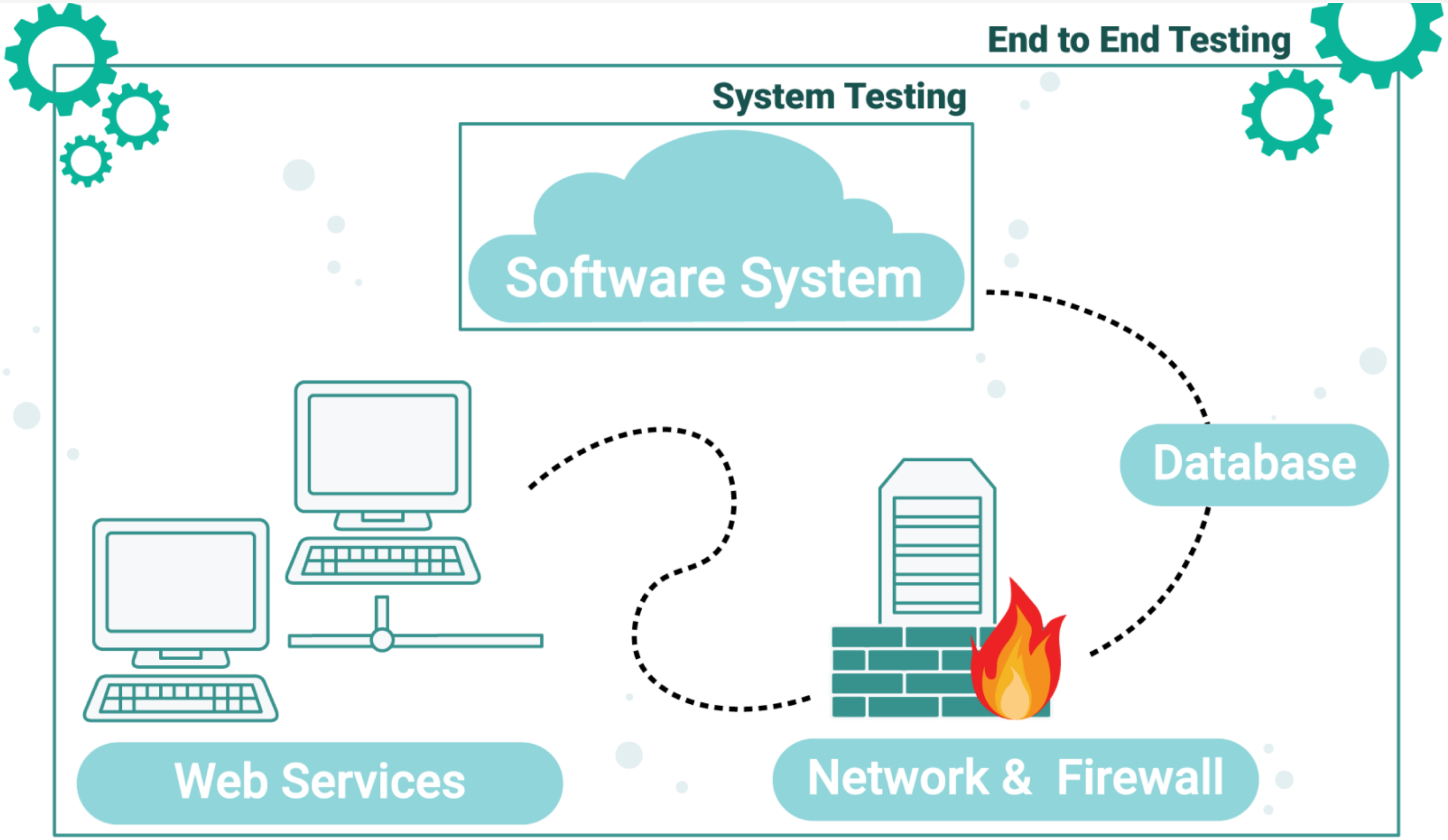


Integration testing example

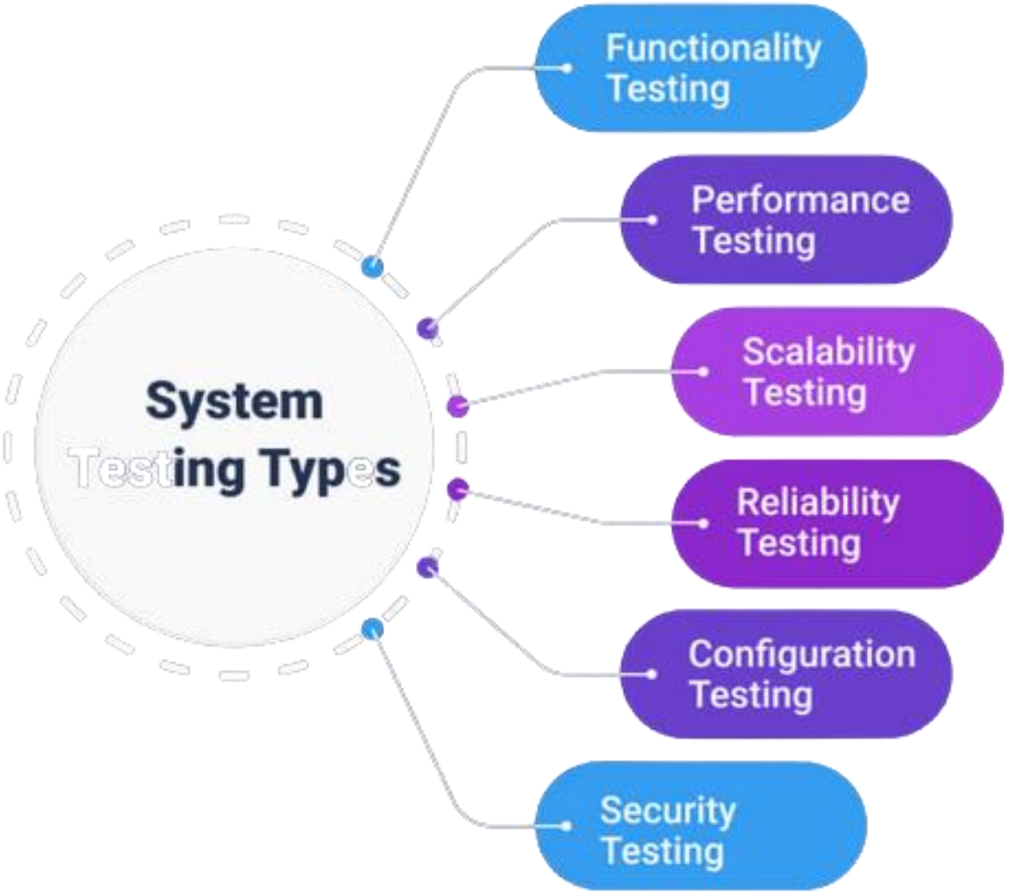


	Unit Testing	Integration Testing
	✓	✗
	✓	✓

System Testing



System Testing types

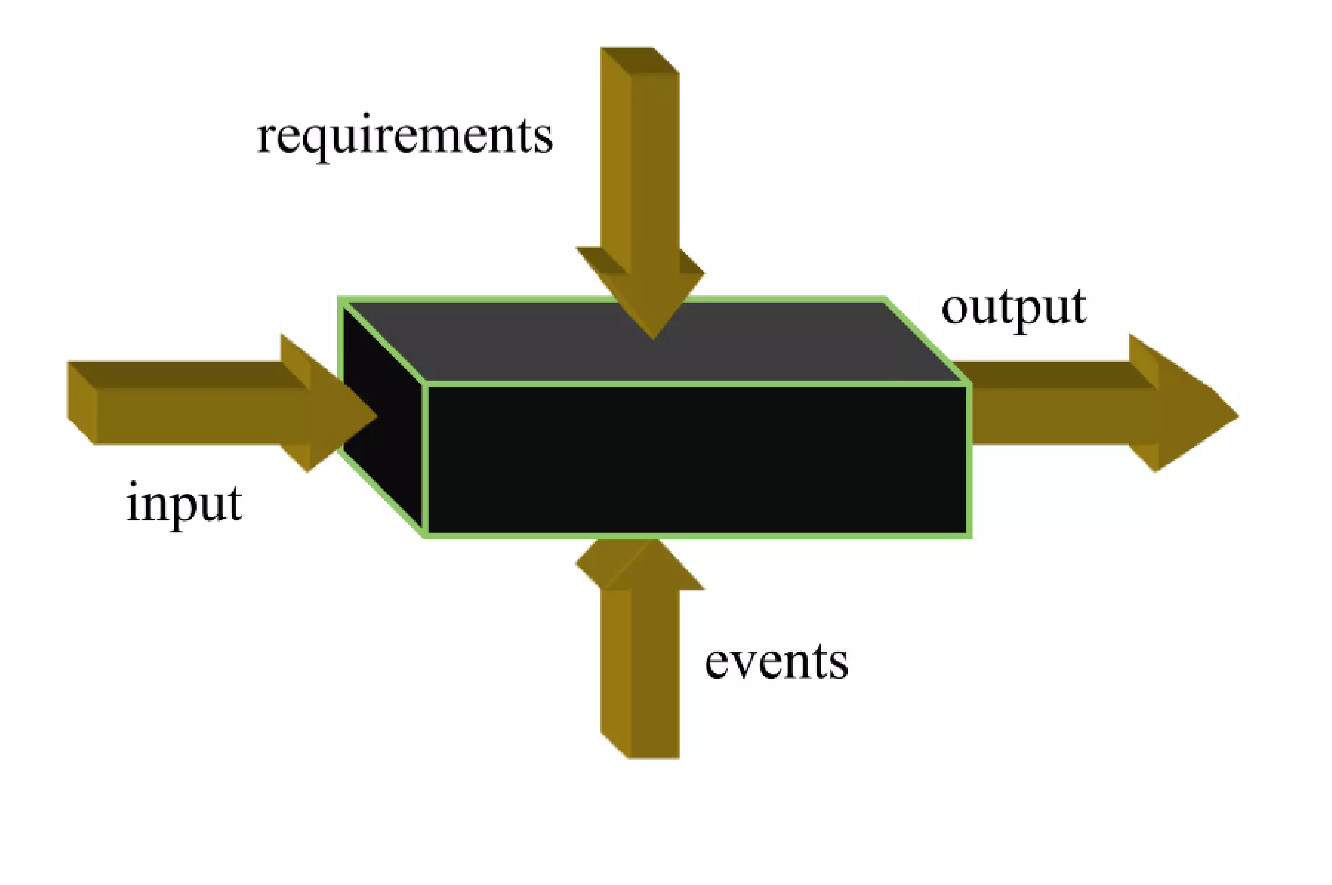


Testing Methodology

- **Black box testing**
  - No knowledge of internal program design required.
  - Tests are based on requirements and functionality.
- **White box testing**
  - Knowledge of the internal program design and code required.
  - Tests are based on coverage of code statements, branches, paths, conditions.

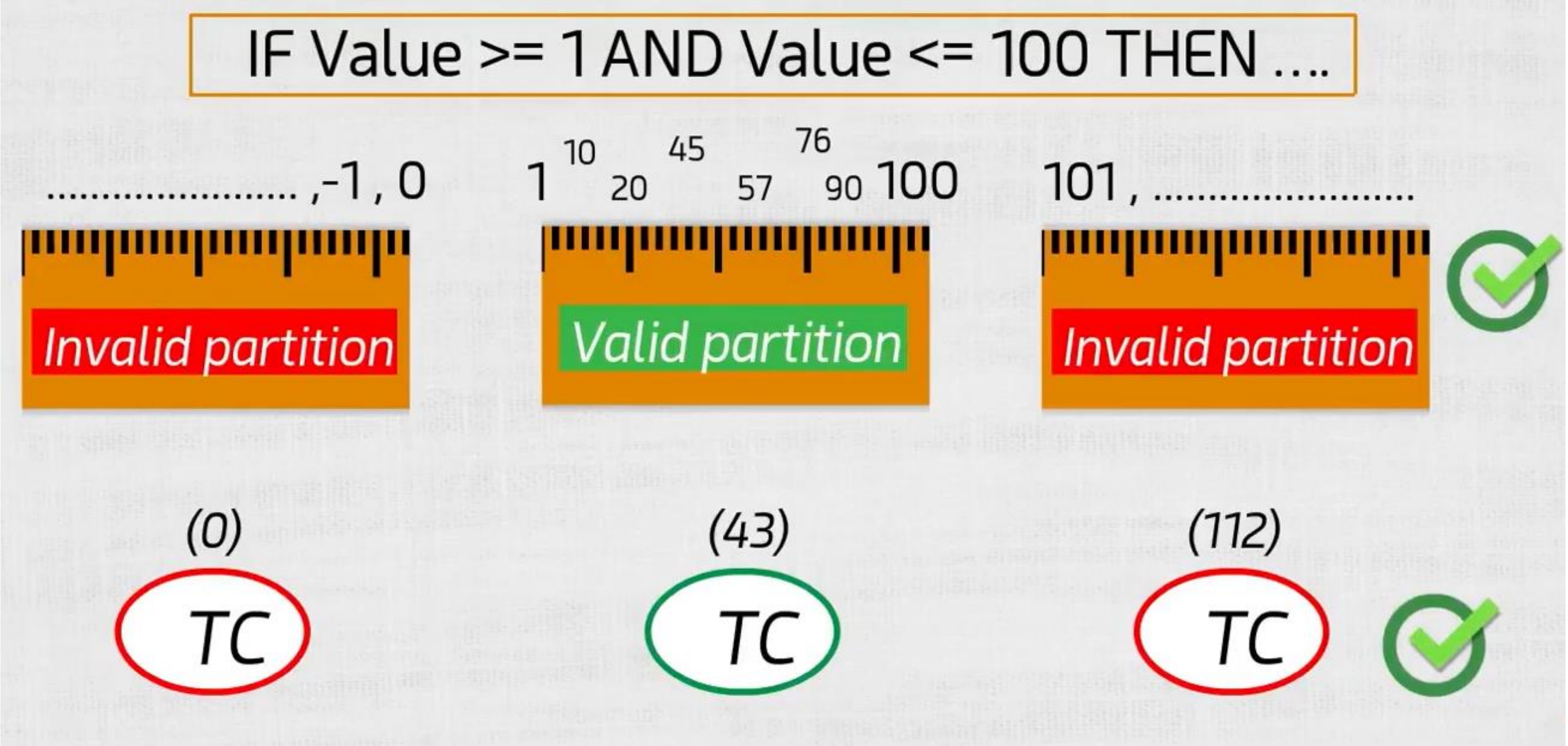
Black box testing

Applied for functional and non-functional testing, without reference to the internal structure of the system



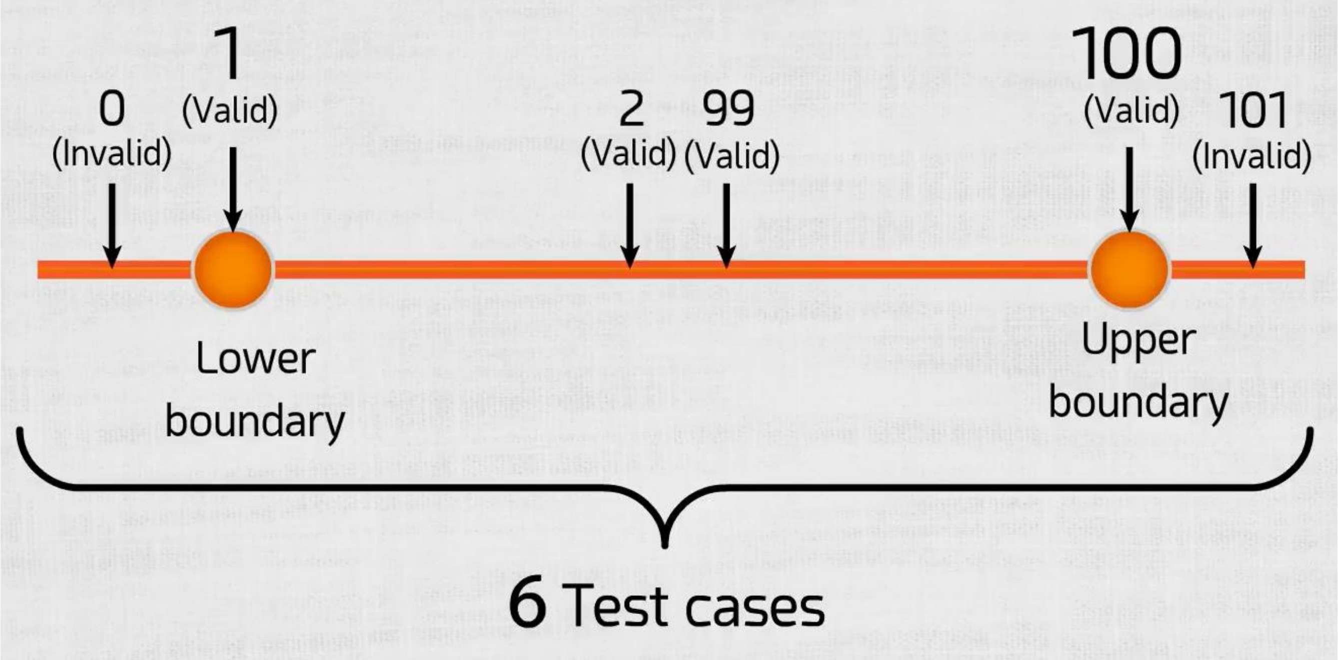
Black box testing techniques

1. Equivalence Partitioning (EP)
   - EX: range of data -> if VALUE is between 1 and 100 then print ‘Pass’



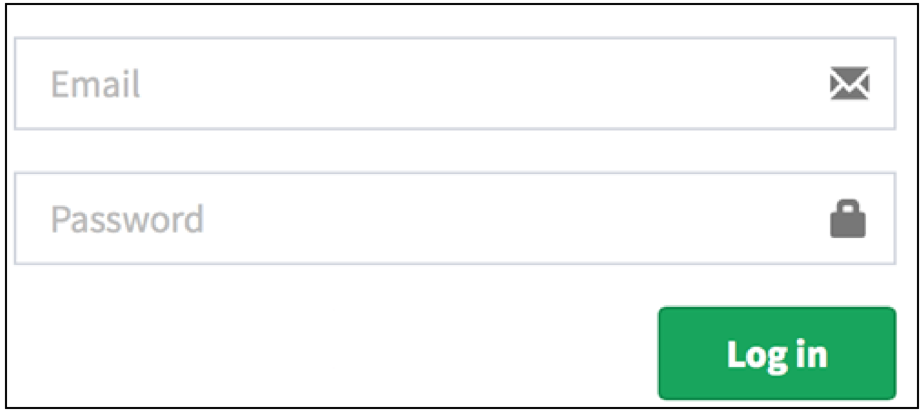
Black box testing techniques

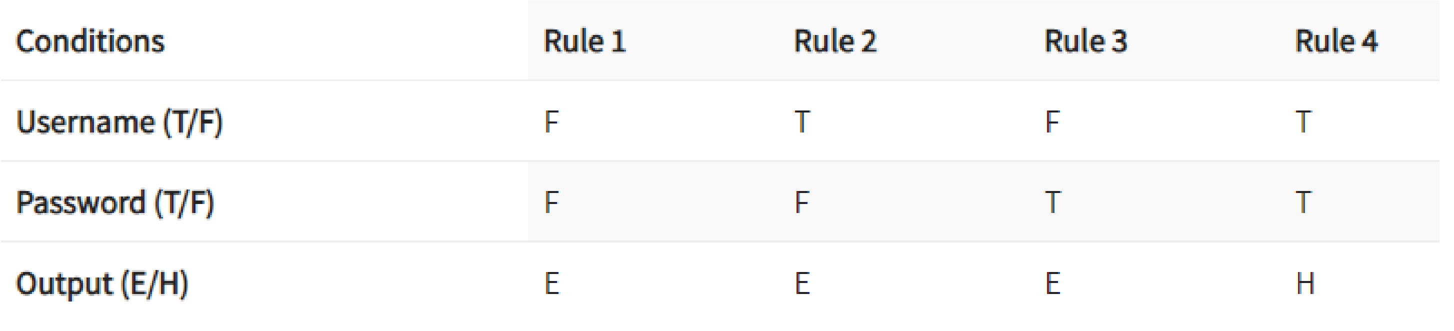
2. Boundary value analysis(BVA)
   - EX: range of data -> if VALUE is between 1 and 100(inclusive). then print ‘Pass’
   - Find the boundary.
     - Lower boundary.
     - Upper boundary.
   - Test one value above and below it.
   - Applicable for numeric fields and date.



Black box testing techniques

- Decision Table Testing
  - Used to test system behavior for different input combinations.
  - **Decision Table** is a tabular representation of inputs versus rules/cases/test conditions.
- Ex:

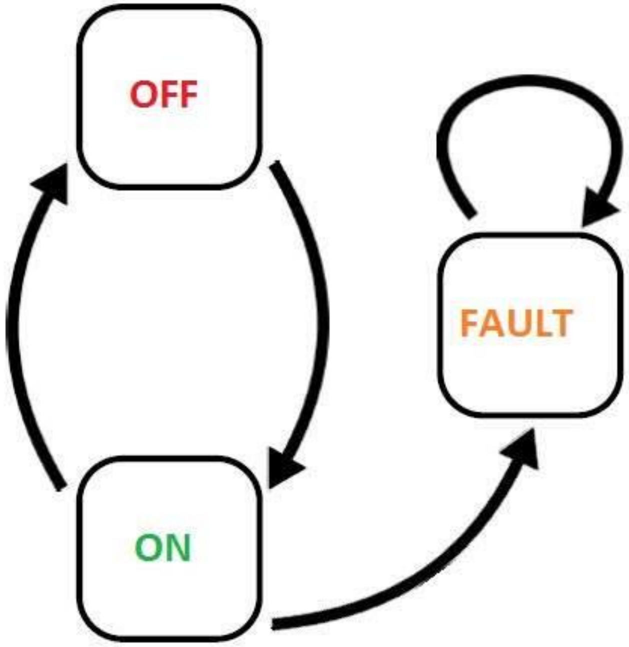


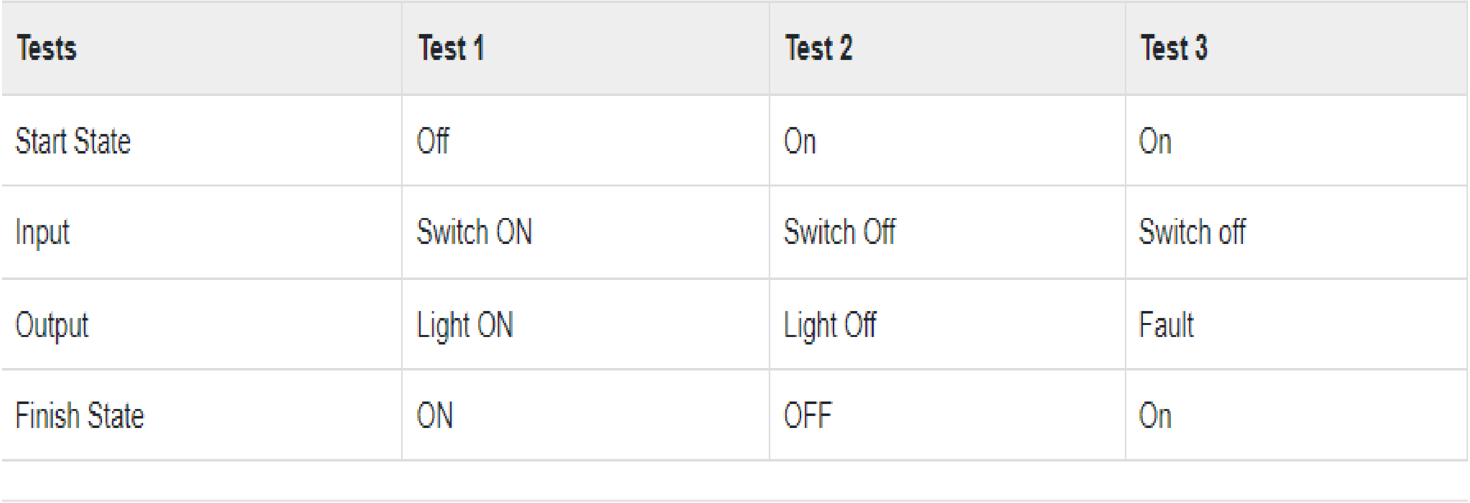


Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

Black box testing techniques

- State Transition Testing
  - Used in which outputs are triggered by changes to the input conditions.
  - Designed to execute valid and invalid state transitions.
- Ex:

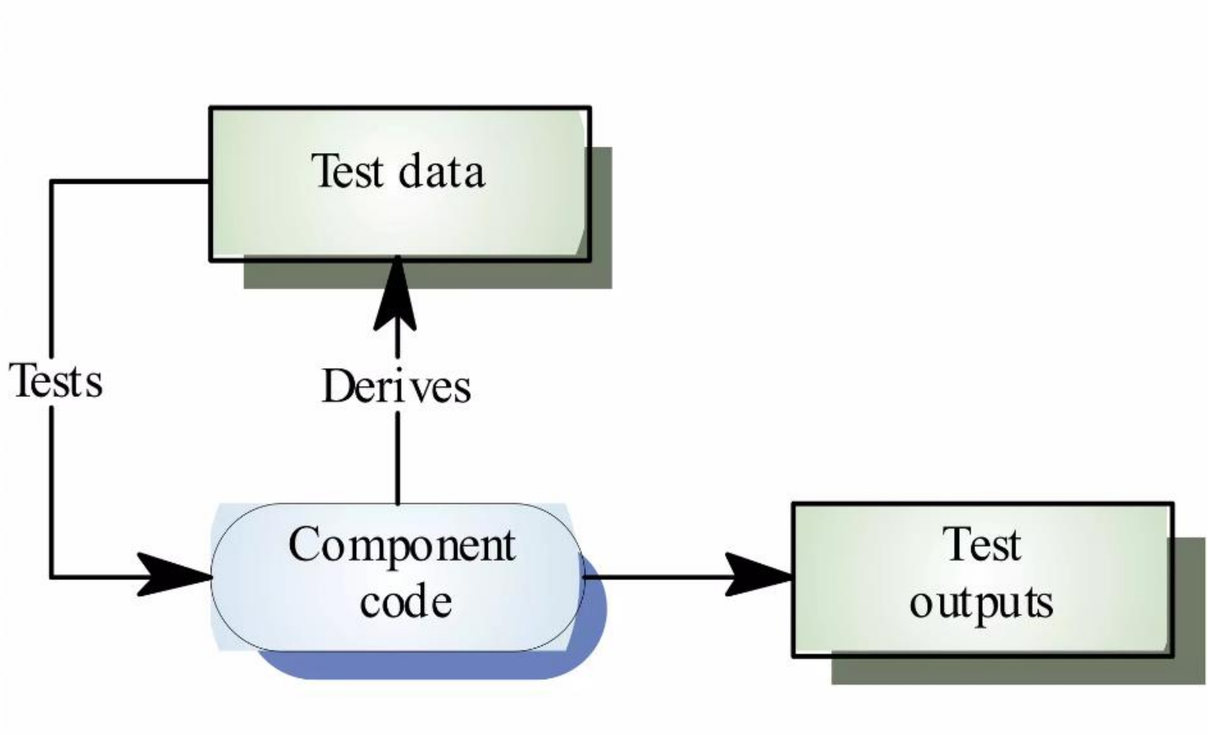




Tests	Test 1	Test 2	Test 3
Start State	Off	On	On
Input	Switch ON	Switch Off	Switch off
Output	Light ON	Light Off	Fault
Finish State	ON	OFF	On

White box testing

Testing based on analysis of the internal structure of the component or system



White box testing techniques

- The goal of White Box testing in software engineering is to verify all the decision branches, loops, and statements in the code.
- ***Statement Coverage:***
    - we would only need one test case to check all the lines of the code.
    - If I consider *TestCase_01 to be (A=40 and B=70),* then all the lines of code will be executed.

```
INPUT A & B
C = A + B
IF C>100
PRINT “ITS DONE”
```

White box testing techniques

- ***Branch Coverage:***
  - which will evaluate the “FALSE” conditions.
  - Branch coverage to ensure maximum coverage**.**
- **TestCase_01**: A=33, B=45
- **TestCase_02**: A=25, B=30

```
INPUT A & B
C = A + B
IF C>100
PRINT “ITS DONE”
Else
print “Its Pending”
```

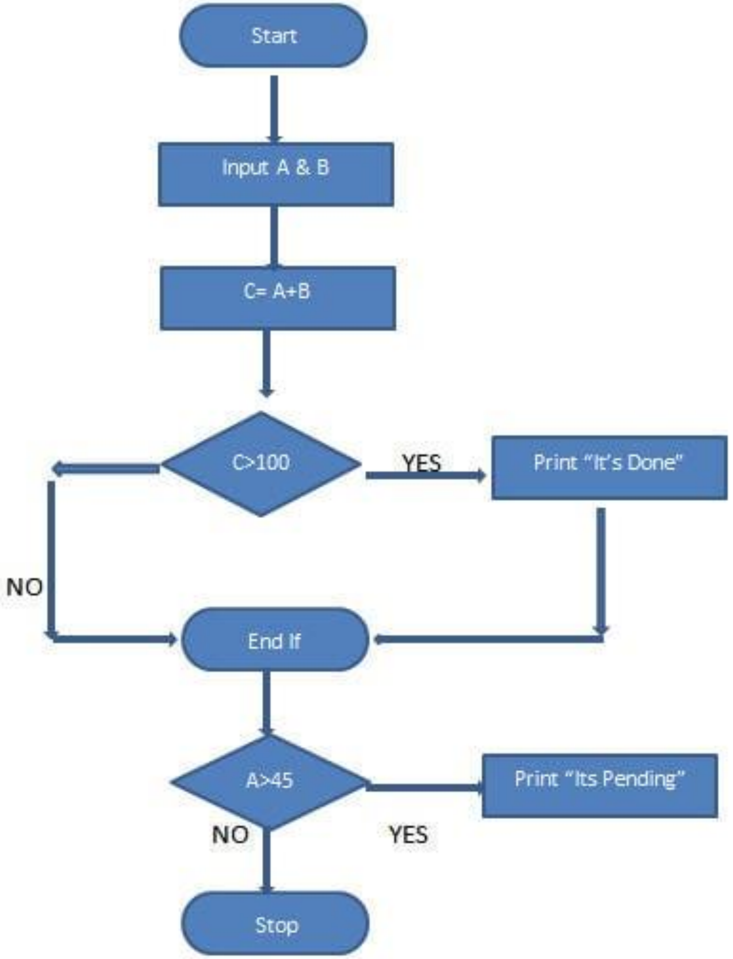
White box testing techniques

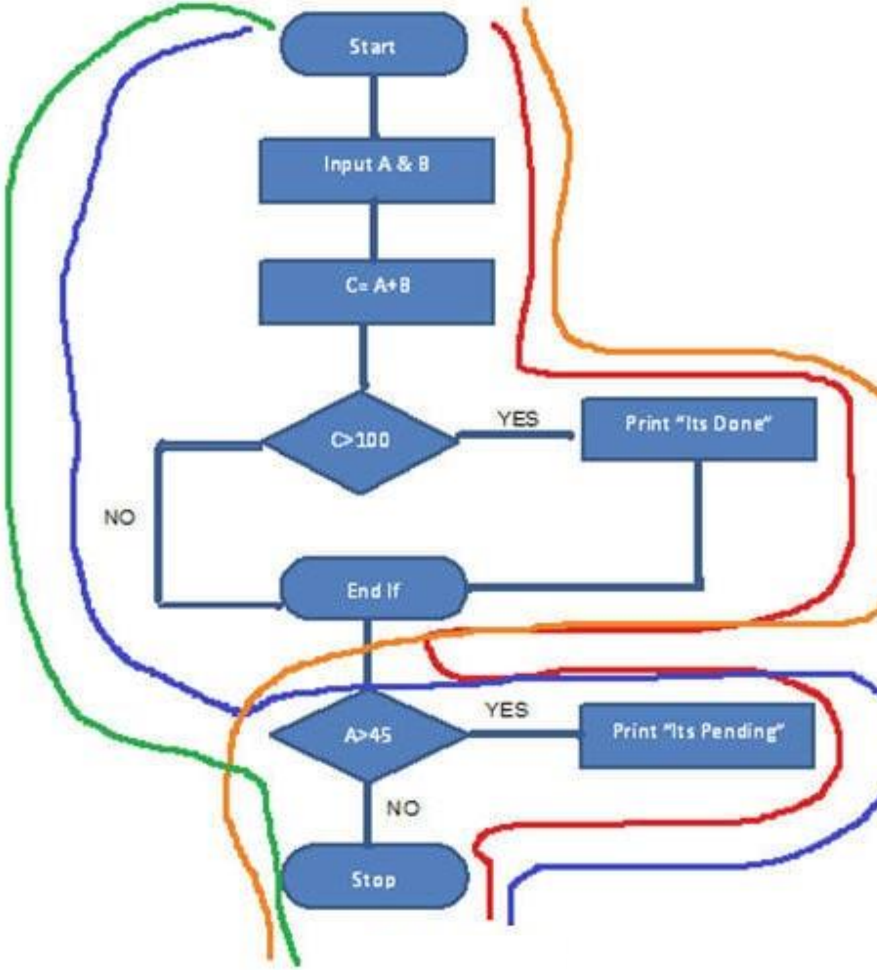
- ***Path Coverage:***
  - used to test the complex code snippets.
  - which basically involve loop statements or combination of loops and decision statements.

```
INPUT A & B
C = A + B
IF C>100
PRINT “ITS DONE”
END IF
IF A>50
PRINT “ITS PENDING”
END IF
```

- To ensure maximum coverage, we would require 4 test cases.

TestCase_01: A=50, B=60
TestCase_02: A=55, B=40
TestCase_03: A=40, B=65
TestCase_04: A=30, B=30



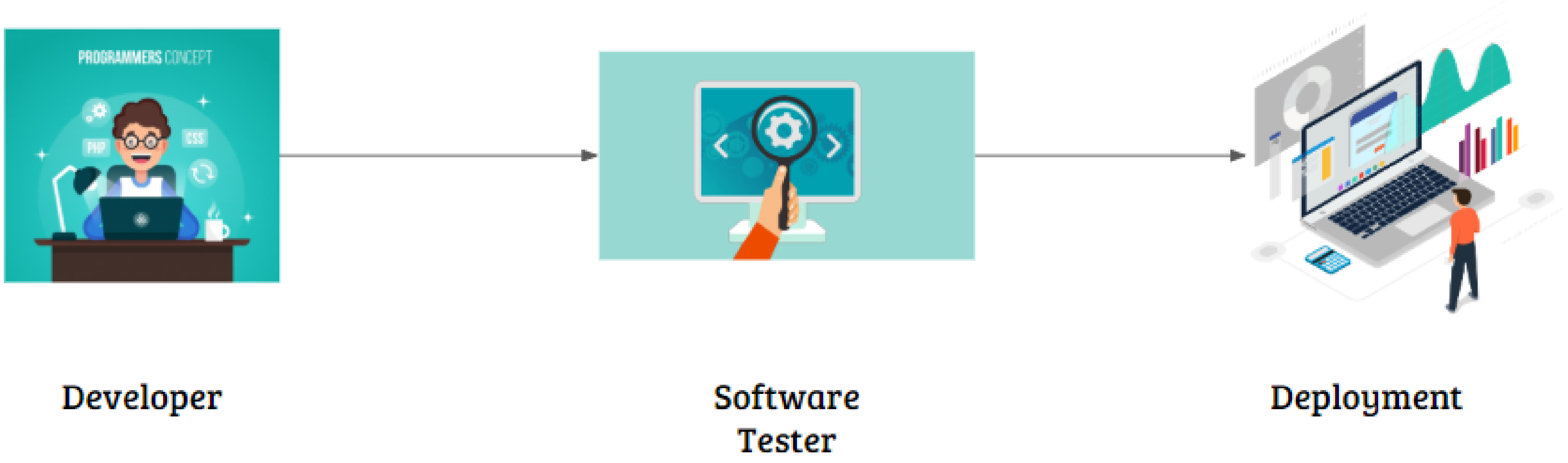


Test cases objectives

- **Test to pass(Happy scenario)**
  - Assures that the software minimally works
  - Applies simple and straightforward test cases

- **Test to fail(Bad scenario)**
  - Choose test cases to appear the weaknesses in software
  - Designing test cases with the sole purpose of breaking the software

Software Deployment Cycle



Developer

Software Tester

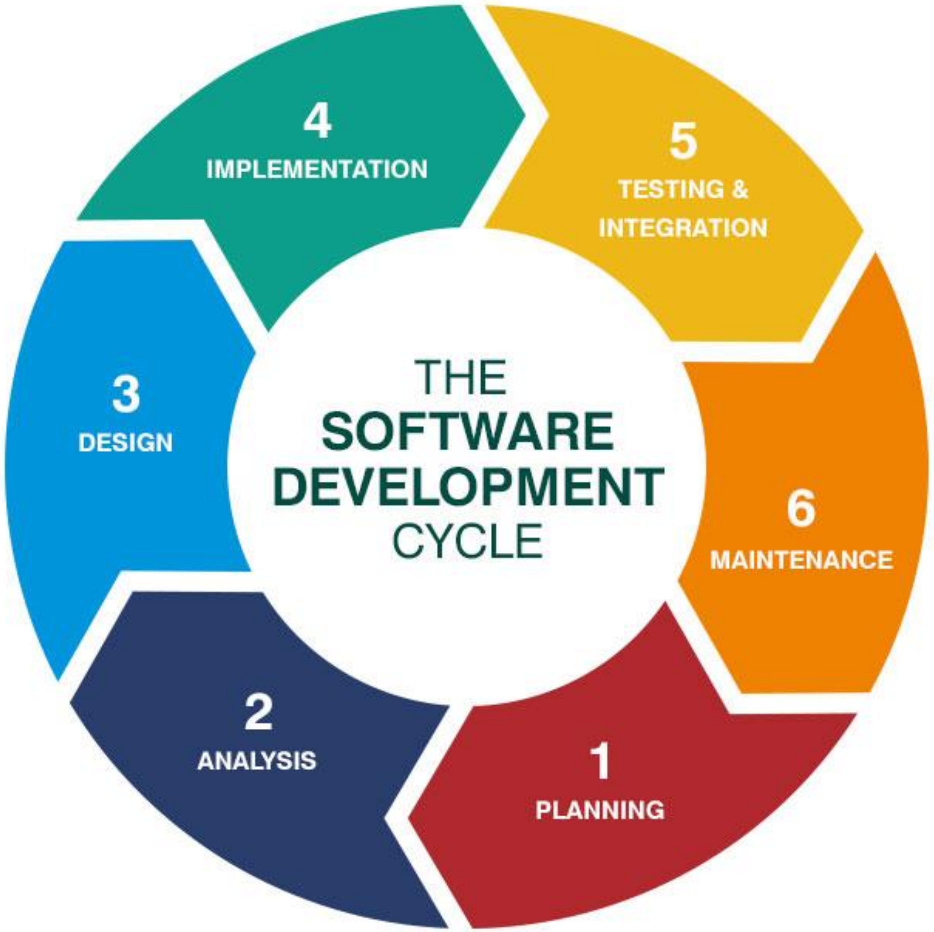
Deployment

SDLC and STLC

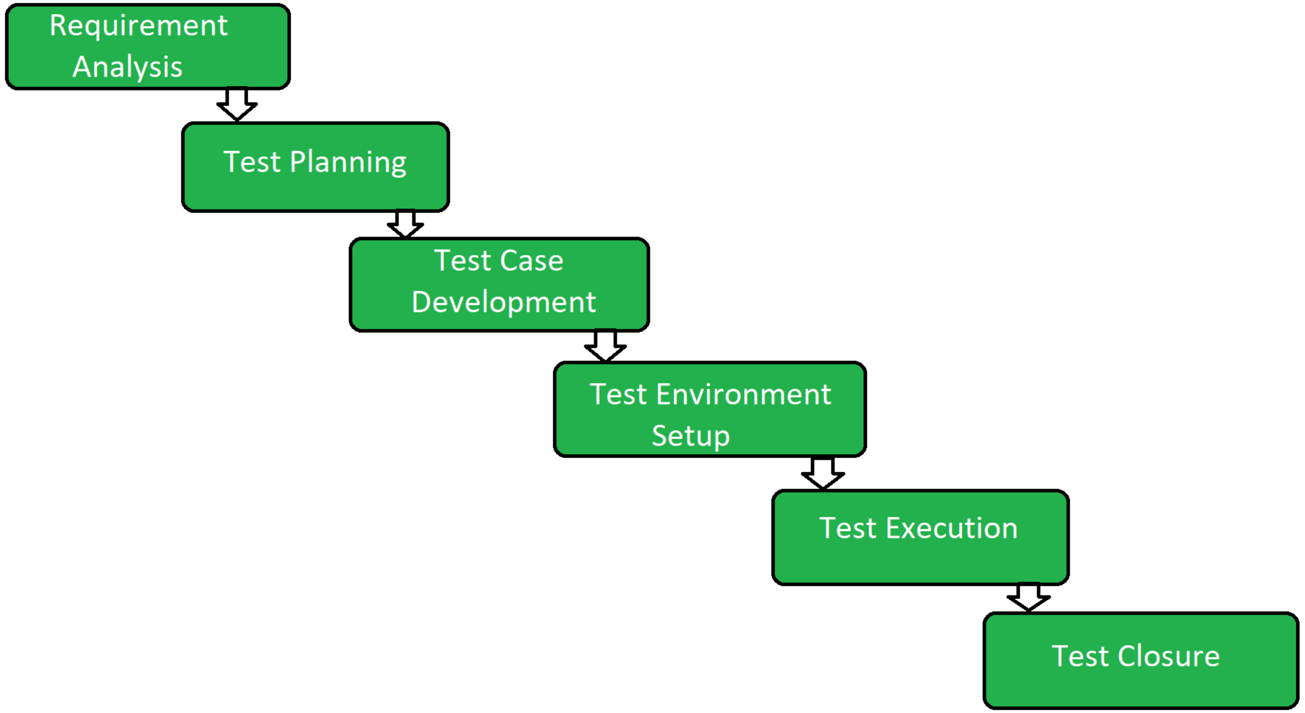
- **SDLC:** SDLC is Software Development Life Cycle. It is the sequence of activities carried out by Developers to design and develop high-quality software.

- **STLC:** STLC is Software Testing Life Cycle. It consists of a series of activities carried out by Testers methodologically to test your software product.

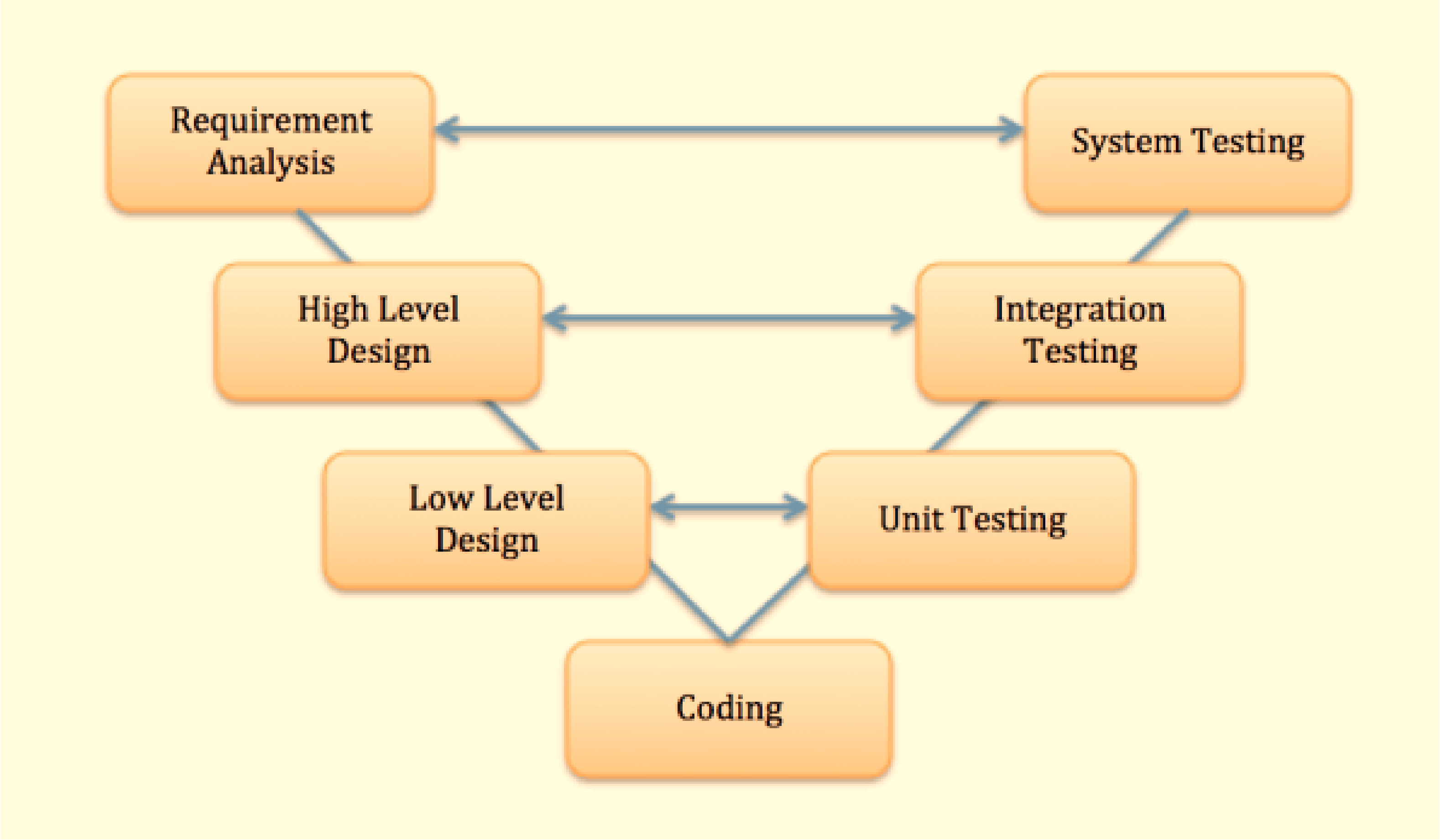
SDLC


4
IMPLEMENTATION
5
TESTING &
INTEGRATION
3
DESIGN
THE
SOFTWARE
DEVELOPMENT
CYCLE
6
MAINTENANCE
2
ANALYSIS
1
PLANNING

STLC Life Cycle



V Model



Requirement Analysis

System Testing

High Level Design

Integration Testing

Low Level Design

Unit Testing

Coding

Thanks