
LOGIN
Username :
Password :
forgot your password ?

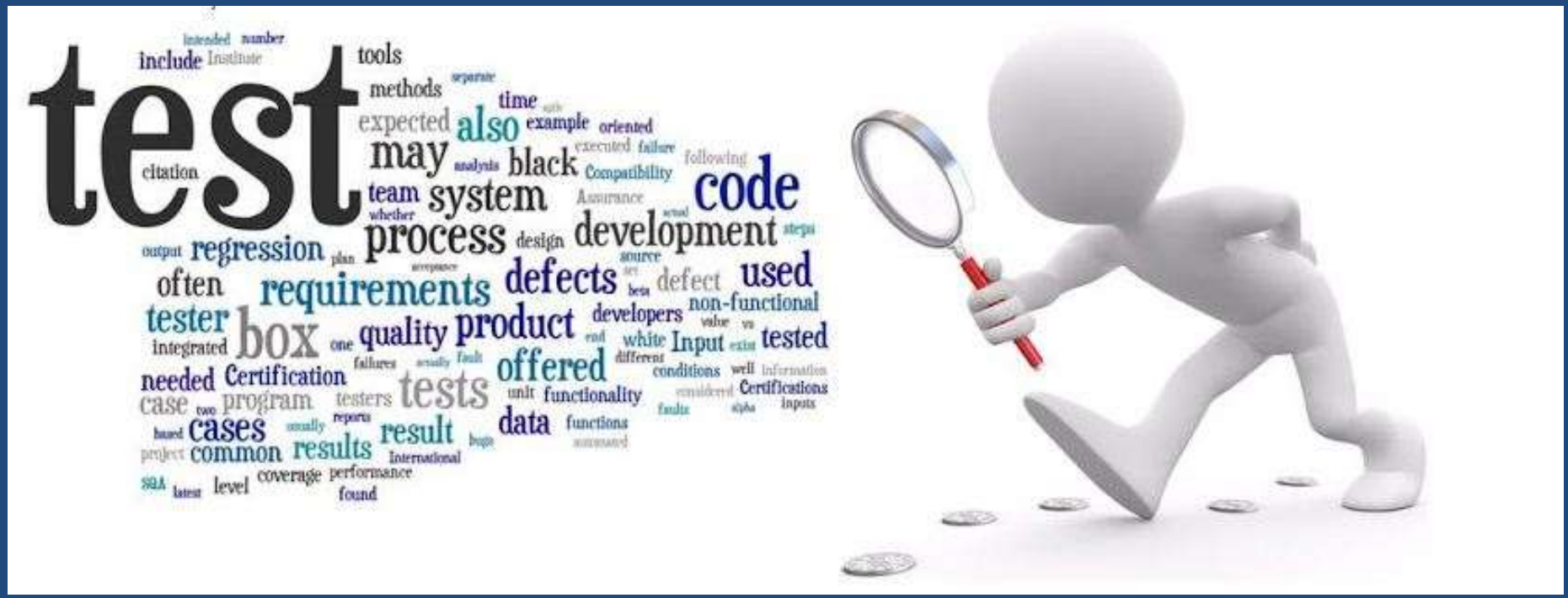

Powerpoint Templates
I want Login form with signup attached there and in black color????


Login
Email address
Password
Login
Create an account

Introduction

- According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

- In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.


test
include
tools
methods
time
expected
also
example
oriented
may
black
Compatibility
citation
team
system
code
process
development
regression
design
defects
defect
used
often
requirements
non-functional
tester
box
quality
product
developers
tested
integrated
white
Input
needed
Certification
offered
conditions
Certifications
case
program
testers
tests
functionality
cases
result
data
functions
common
results
coverage
performance
level
found

Objectives

- Finding defects which may get created by the programmer.
- Gaining confidence in and providing information about the level of quality.
- To prevent defects.

- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

Who does Testing?

- It depends on the process and the associated stakeholders of the project(s).
- Different companies have different designations for people who test the software

- In most cases, the following professionals are involved.
    - Software Tester
    - Software Developer
    - Project Lead/Manager
    - End User

Testing Methods

- Black Box Testing
- White Box Testing
- Gray Box Testing

Black Box Testing

- Testing without having any knowledge of the interior working of the application .
- Tester doesn’t have the access of source code
- Needn’t not know the internal working of the application.
- Main focus in the functionality of the system as a whole
- This is known as the behavior testing .
- focuses on the functional requirements of the software

Black Box Testing

- Black-box testing attempts to find errors in the following categories:

(1) incorrect or missing functions

(2) interface errors

(3) errors in data structures or external database access

(4) behavior or performance errors

Methods of Black Box Testing

- Graph Based Testing
- Equivalent Partitioning
- Boundary Value Analysis
- Orthogonal Array Testing

White Box Testing

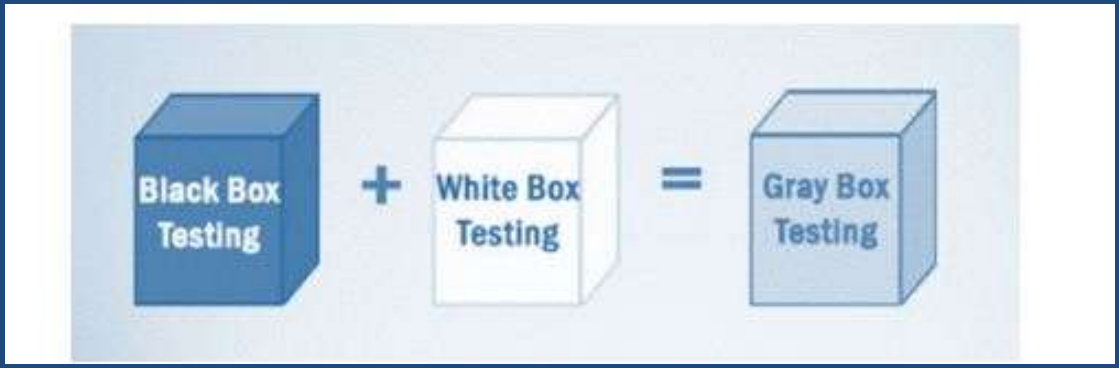
- Internal details and system is made visible
- White Box technique is used by both developers as well as testers.
- Highly efficient method to find bugs .
- Known as Glass Box ,Open Box ,Transparent Box Testing.
- Main focus on the structural part of the application.

Methods

- Basis Path Testing: Flow Graph, Independent program path
- Control Structure Testing : Condition Testing, Data flow testing, Condition Testing, loop testing

Gray Box Testing

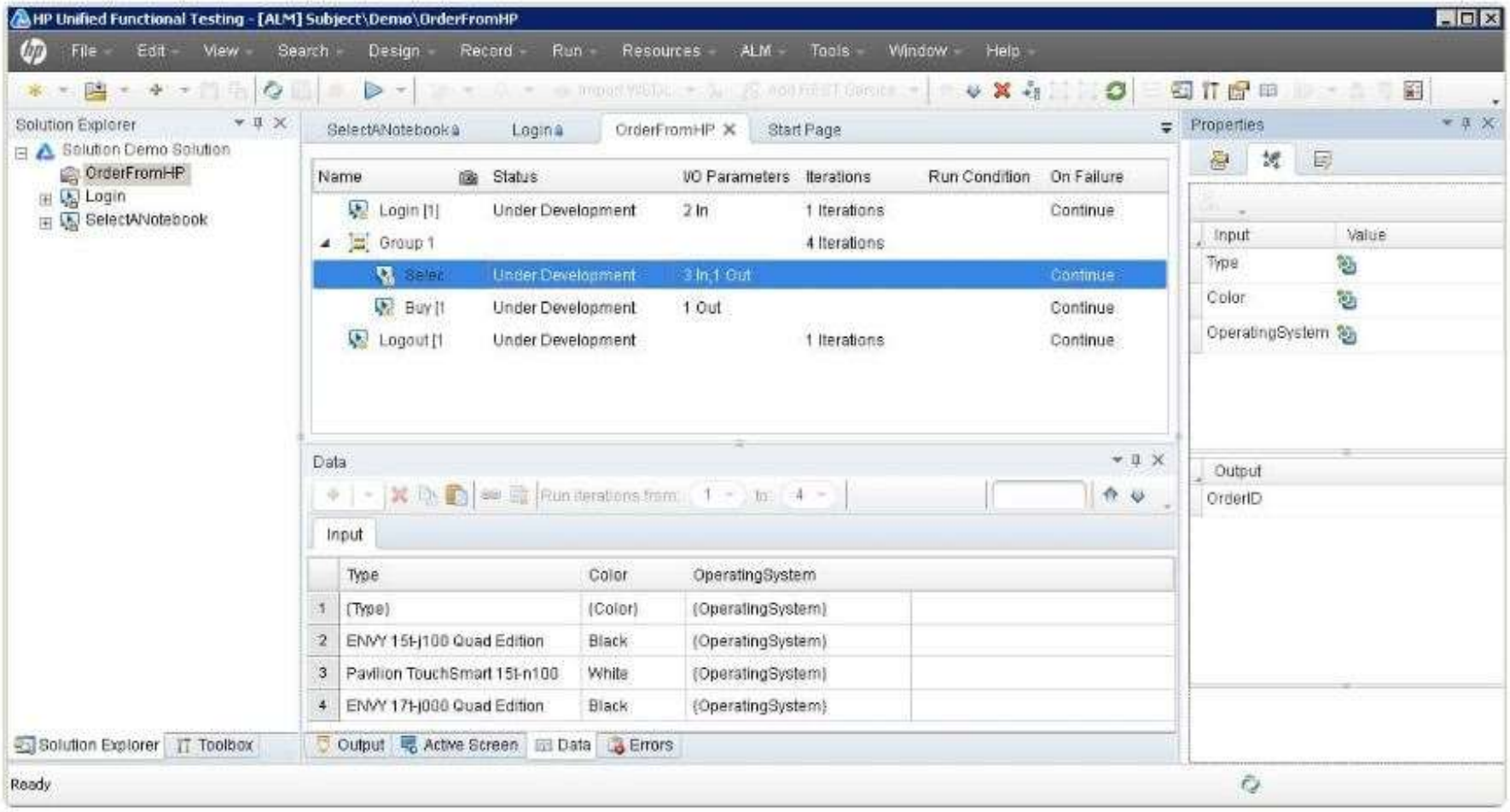
- Gray-box testing is a combination of white-box testing and black-box testing.
- The aim of this testing is to search for the defects.
- Gray-box testing is beneficial.


Black Box Testing
+
White Box Testing
=
Gray Box Testing

Software Testing Tools

- There are numerous tools available in the market .
- 1.Katalon Studio
- 2. HP Unified Functional Testing (UFT)
- 3. IBM Rational Functional Tester

HP Unified Functional Testing (UFT)



Boundary Value Analysis

- A greater number of errors occurs at the boundaries of the input domain rather than in the “center.”
- It is for this reason that *boundary value analysis* (BVA) has been developed as a testing technique.
- Boundary value analysis leads to a selection of test cases that exercise bounding values.
- Boundary value analysis is a testcase design technique that complements equivalence partitioning.
- Rather than selecting any element of an equivalence class, BVA leads to the selection of test cases at the “edges” of the class.
- Rather than focusing solely on input conditions, BVA derives test cases from the output domain as well

Boundary Value Analysis

Example 1:

AGE: [ ] (accepts 18 to 56)

BOUNDARY VALUE ANALYSIS		
INVALID (min - 1)	**VALID** (min, +min, -max, max)	**INVALID** (max + 1)
17	18, 19, 55, 56	57

Valid Test Cases

Valid Test Cases:

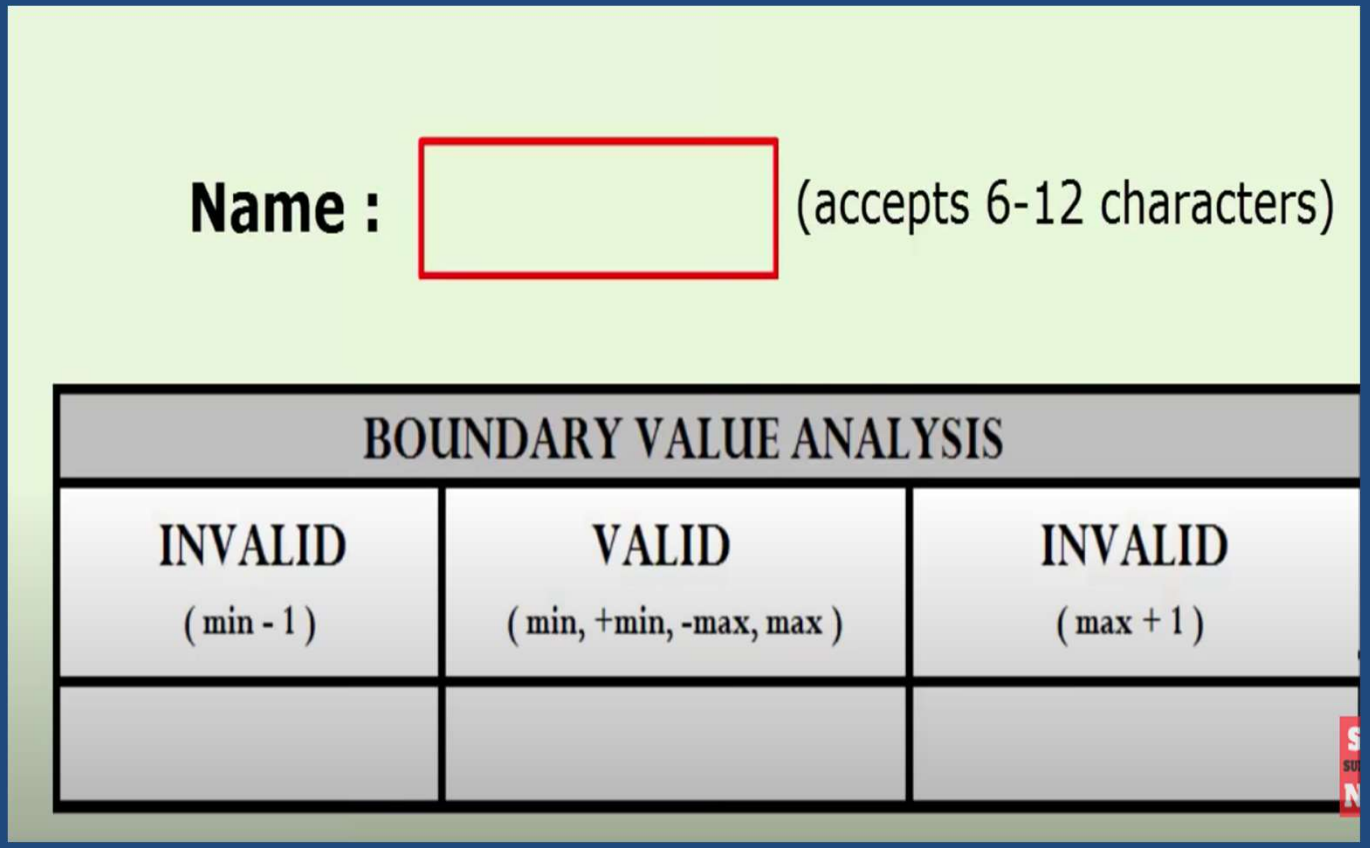
Enter the value 18
Enter the value 19
Enter the value 55
Enter the value 56

Invalid Test case

Invalid Test Cases:

Enter the value 17
Enter the value 57

Boundary Value Analysis



Valid Test Cases

Valid Test Cases:

Text length of 6
Text length of 7
Text length of 11
Text length of 12

Invalid Test Case

Invalid Test Cases:

Text length of 5
Text length of 13

Equivalent Partitioning

AGE: [ ] (accepts 18 to 56)

EQUIVALANCE PARTITIONING		
INVALID	VALID	INVALID
<=17	18-56	>=57

Equivalent Partitioning

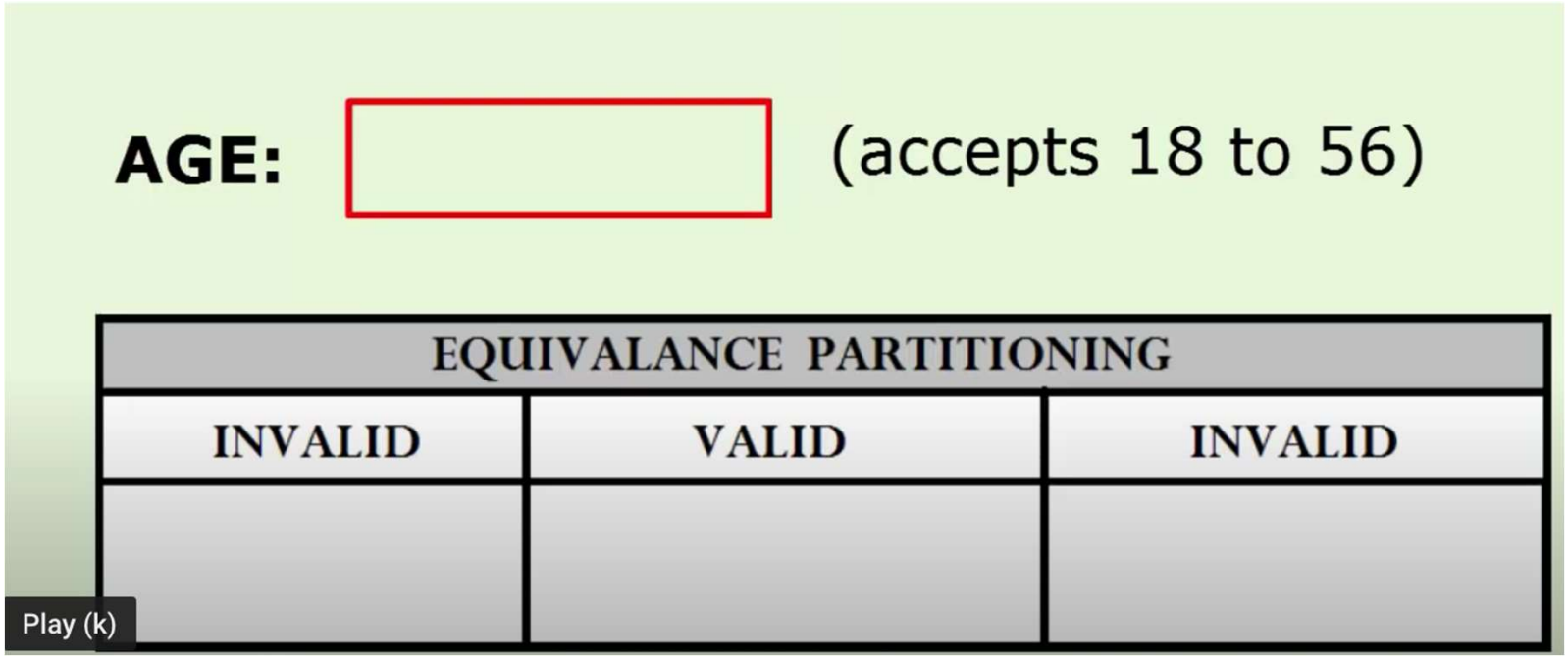
- *Equivalence partitioning* is a **black-box testing** method that divides the input domain of a program into classes of data from which test cases can be derived.

- Test-case design for equivalence partitioning is based on an evaluation of *equivalence classes* for an input condition

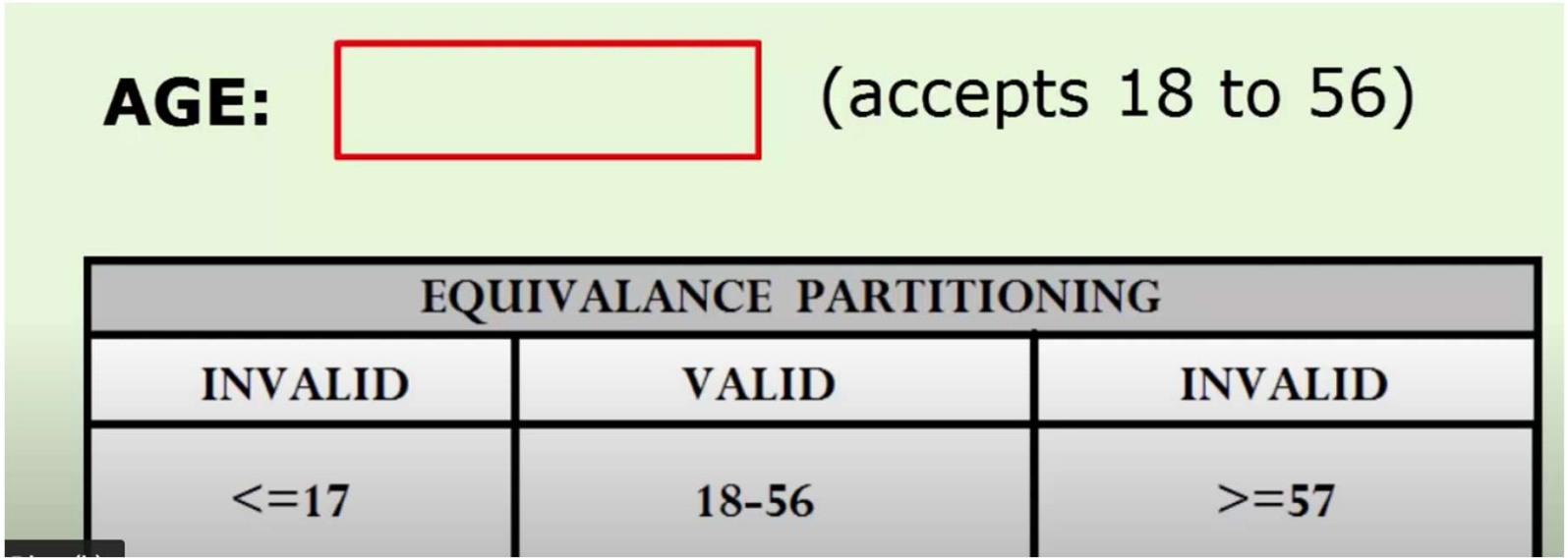
- Equivalence classes may be defined according to the following guidelines:

1. If an input condition **specifies a range**, one valid and two invalid equivalence classes are defined.
2. If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
3. If an input condition specifies a member of a set, one valid and one invalid equivalence class are defined.
4. If an input condition is Boolean, one valid and one invalid class are defined.

Equivalent Partitioning



Equivalent Partitioning



Basis Path Testing

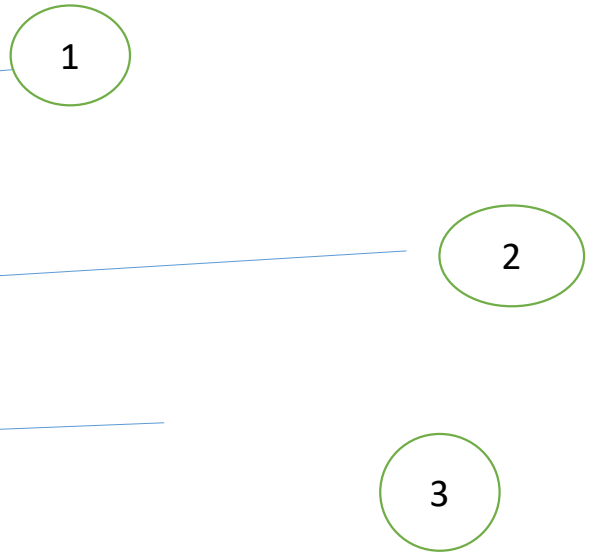
The following steps can be applied to derive the basis set:

1. Using the design or code as a foundation, draw a **corresponding flow graph.**
2. Determine the **cyclomatic complexity** of the resultant flow graph.
3. Determine a basis set of linearly **independent paths.**
4. **Prepare test cases** that will force execution of each path in the basis set.

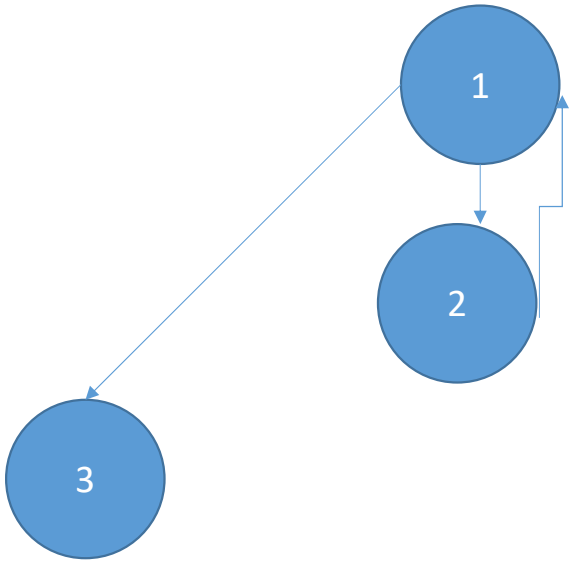
Basis Path testing method

While loop

```
while (x<y)
        {
          Print{“the value”};
         }
```



Basis Path Testing



Basis Path Testing

Cyclomatic Complexity

- It is a source code complexity measurement that is being correlated to a number of coding errors.
- It is calculated by developing a **control Flow Graph** of the code that measures the number of linearly-independent paths through a program module.
- The cyclomatic complexity V(G) can be calculated as:

  V(G) = regions

  V(G)= predicate nodes (P) + 1

  V(G)= No. of edge (E)- No. of nodes (N)+2

Basis Path Testing

- For above case

V(G) = 2 regions

V(G) = Predicate Node +1 = 1+1 = 2

V(G) = 3edges- 3 nodes +2 = 2

Basis Path Testing

- Linearly Independent Path

  Path 1 = 1-2-1-3

  Path 2 = 1-3

Basis Path Testing

- Prepare test case

For Path 1 = 1-2-1

A test case is x= 5 and y = 7

For Path 2 = 1-3

A test case is x =7 and y =5

Basis Path Testing

```
    If (x<y)
            {
           Print(“”success)
           }
else
{
Print(“fail”)
}
```