



Automation Testing

1



Fundamentals of Software Testing

2

Definition

- The practice of investigating a software / system under test so as to ensure that it is of the highest quality.
- The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of a component or system and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

3

3

What is software testing?

- when done correctly, can increase overall software quality of conformance by testing that the product conforms to its requirements.
- begins with the software engineer in early stages, but later specialists may be involved in the testing process.
- After generating source code, the software must be tested to uncover and correct as many errors as possible before delivering it to the customer.
- It's important to test software because a customer, after running it many times would finally uncover an error.

4

4

Software Testing Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

5

5

Software Testing Terms

- Verification : You Build Software Right
- Validation : You Build Right Software
- Testing : Find & Make Sure
- Debugging : Find & Fixed Bug

6

6

Software Verification & Validation

Criteria	Verification	Validation
<i>Objective</i>	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.	To ensure that the product actually meets the user's needs and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment.

7

7

Software Verification & Validation

Criteria	Verification	Validation
<i>Question</i>	Are we building the product <i>right</i> ?	Are we building the <i>right</i> product?
<i>Evaluation Items</i>	Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product / software.
<i>Activities</i>	<ul style="list-style-type: none"> • Reviews • Walkthroughs • Inspections 	<ul style="list-style-type: none"> • Testing

8

8

Testing vs Debugging

Criteria	Testing	Debugging
<i>Definition</i>	The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of a component or system and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.	The process of finding, analyzing and removing the causes of failures in a component or system.

9

9

Testing vs Debugging

Criteria	Testing	Debugging
Objective	The objective of testing is limited to finding defects (and confirming that the defects are fixed).	In addition to finding defects, the objective of debugging is to analyze and resolve them.
Responsibility	Normally independent testers are responsible for testing.	Developers are responsible for debugging.
Knowledge	Testing can be performed without the knowledge of how the software is built.	Successful debugging requires the knowledge of the software's architecture and technical design.

10

10

Why Software Testing?

- Delay / Loss of time
- Futility / Loss of effort
- Wastage / Loss of money
- Shame / Loss of business reputation
- Injury or death

11

11

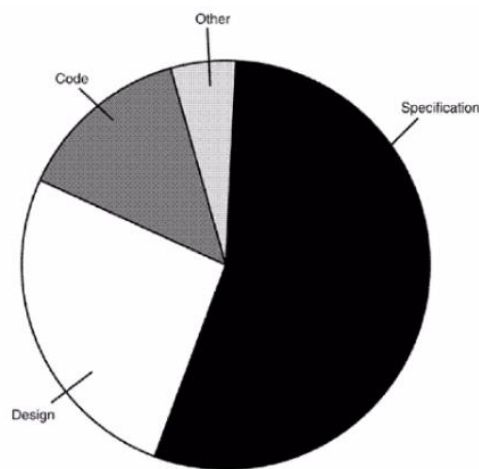
Software Bug

- The software doesn't do something that the product specification says it should do.
- The software does something that the product specification says it shouldn't do.
- The software does something that the product specification doesn't mention.
- The software doesn't do something that the product specification doesn't mention but should.
- The software is difficult to understand, hard to use, slow or in the software tester's eyes will be viewed by the end user as just plain not right.

12

12

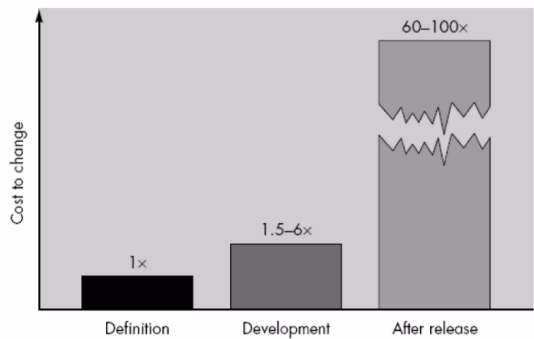
Why do bugs occur?



13

13

Why do bugs occur?



Activity	Resources	Name(s)	Duration	Cost
	Project sponsor	Salim	N/A	N/A
	Key business owner	Karen	N/A	N/A
Requirements	1 business analyst	Brian	½ month	\$7,500
Design	1 software architect	Angela	1 month	\$20,800
Construction	4 developers	Reiko,Tim, Hua, Mike	2½ months	\$138,600
System Testing	1 tester	Ian	½ month	\$6,100
User Testing	1 end user	Emily	½ month	\$4,300
Rework	4 developers	(as above)	½ month	\$27,700
Project Management	½ project manager	Phil	4 months	\$34,700
TOTAL				\$239,700

14

14

Source of problems

- Requirements Definition
- Design
- Implementation
- Support Systems
- Inadequate Testing of Software
- Evolution

15

15

What does a software tester do?

- The goal of a software tester is to find bugs.
- The goal of a software tester is to find bugs and find them as early as possible.
- The goal of a software tester is to find bugs, find them as early as possible, and make sure they get fixed.

16

16

Software is...

- Requirement Specification
- Design Document
- Source Code
- Test Suites and Test Plans
- Interface to Hardware and Software Operating Environment
- Internal and External Documentation

17

17

When does testing occur?

- Customer Requirement
- Software Specification
- Software Design
- Source Code

18

18

Test Documents

- Test Plan
- Test Case
- Bug Report
- Test Tools and Automation
- Metrics, Statistics and Summary

19

19

Challenges for testers

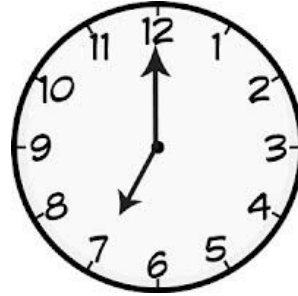
- No Specification means no testing.
- Market/ Entrepreneurial pressures not to test.
- Lack of Trained Testers.

20

20

Economics of testing

- Find bugs in phase
- Time-to-market
- Time-to-profit



21

21

Software Testing Type

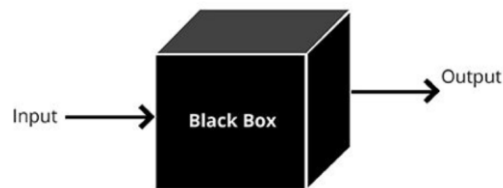
- There are two major types of software testing
 - **Black box testing :**
focuses on input, output, and principle function of a software module.
 - **White box testing/ Glass box testing :**
Glass box testing looks into the structural testing or glass box testing, statements paths and branches are checked for correct execution.

22

22

Black Box Testing

- Blackbox Testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.
- These tests can be functional or non-functional, though usually functional.



23

23

Black Box Testing

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

24

24

Black Box Testing - Techniques

- Equivalence Partitioning (EP)
- Boundary Value Analysis (BVA)
- Cause-Effect Graphing

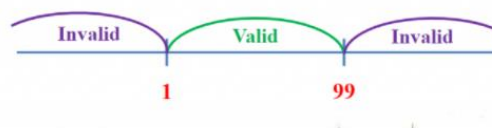
25

25

Black Box Testing - Equivalence Partitioning

Example: A textbox can input integer numbers from 1 to 99

- Partition 1 – invalid: < 1
- Partition 2 – valid: $1 \leq \text{ } \leq 99$
- Partition 3 – invalid: > 99



Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

26

26

Black Box Testing - Equivalence Partitioning

Price

฿ 2,150.00

Product options

-- Select product options --

Quantity

1

 BUY NOW

Equivalence partitioning		
Invalid input	Valid Input	Invalid input
<1	1-10,000	>10,000

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

27

27

Black Box Testing - Boundary Value Analysis

- The maximum and minimum values of partition are its boundary values.
- Boundaries define 3 set of data:
 - In-bound
 - Out-of-bound
 - On-bound



Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

28

28

Black Box Testing - Boundary Value Analysis

Example: A textbox can input integer numbers from 1 to 99

- Select value for test case: 0; 1; 99; 100



Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

29

29

Black Box Testing - Decision Table

- Equivalent partitioning and boundary value analysis are usually applied to an input. In the case of combining multiple inputs in a function, it is difficult to use an equivalent partition or boundary value analysis.
- In this case, we can use Decision table or State transition

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

30

30

Black Box Testing - Decision Table

- Row of Cause (Condition) and Effect (Action/Expected Result)
- Columns of possible combinations

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Condition 1(n)	Yes	Yes	No	No
Condition 2	Yes	No	Yes	No
Actions				
Expected Result 1	x		x	
Expected Result 1		x		x

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

31

31

Black Box Testing - Login Form Case Example

Conditions	Value 1	Value 2	Value 3	Value 4
Username	T	F	T	F
Password	T	T	F	F
Actions				
Message	Login Successfull y	Please enter your username.	Please enter your password.	Please enter valid username and password.

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

32

32

Black Box Testing - Advantages

- Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.
- Tester need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
- Test cases can be designed as soon as the specifications are complete.

33

33

Black Box Testing - Disadvantages

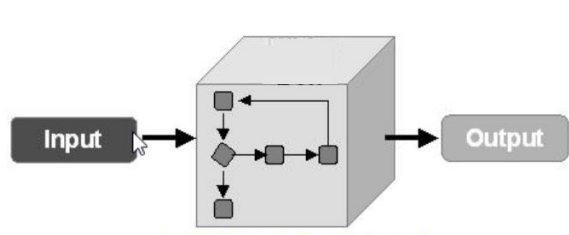
- Only a small number of possible inputs can be tested and many program paths will be left untested.
- Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
- Tests can be redundant if the software designer/developer has already run a test case.
- Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

34

34

White Box Testing

- white-box testing: Testing based on an analysis of the internal structure of the component or system.
- white-box test design technique: Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.



Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

35

35

White Box Testing - Advantages

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

36

36

White Box Testing - Disadvantages

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tested, tools to cater to every kind of implementation/platform may not be readily available.

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

37

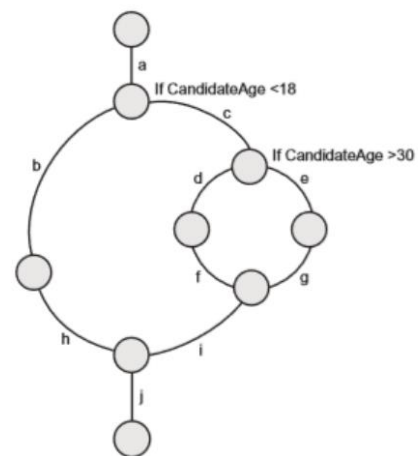
37

White Box - Test Case Design (Decision coverage)

```

1  Program Age Check
2
3  CandidateAge: Integer;
4
5  Begin
6
7  Read(CandidateAge)
8
9  If CandidateAge < 18
10 Then
11   Print ("Candidate is too young")
12 Else
13   If CandidateAge > 30
14   Then
15    Print ("Candidate is too old")
16   Else
17    Print("Candidate may join Club 18–30")
18   Endif
19 Endif

```



Picture Credit : software testing (An ISTQB-ISEB Foundation Guide) Book

38

38

White Box - Test Case Design (Decision coverage)

Condition	CandidateAge	Path
CandidateAge < 18	16	a->b->h->j
18 > CandidateAge < 30	22	a->c->d->f-> <u>i</u> ->j
CandidateAge < 30	35	a->c->e->g-> <u>i</u> ->j

Picture Credit : software testing (An ISTQB-ISEB Foundation Guide) Book

39

39

White Box - Test Case Design

- Statement Coverage
- Decision Coverage
- Condition Coverage
- Decision/ Condition Coverage
- Multiple - Condition Coverage
- MC/DC Coverage

Content Credit : <https://www.lotus-qa.com/black-box-test-design-techniques/>

40

40

Black Box Testing vs White Box Testing

Criteria	Black Box Testing	White Box Testing
Levels Applicable To	Mainly applicable to higher levels of testing: Acceptance Testing & System Testing	Mainly applicable to lower levels of testing: Unit Testing & Integration Testing
Responsibility	Software Testers	Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

41

41



42

What is Automation?

- Automation Testing is use of Tools or Software to perform Software Testing.
- Developing and executing tests that can run and compare the actual to expected results.
- The automation software can also enter test data into the System Under Test , compare expected and actual results and generate detailed test reports

43

43

Why to Automate?

- Automation improves accuracy, increases coverage
- To make test execution faster, accurate and improves quality

44

44

Benefits of Test Automation?

- Manual testing takes time and money to test all procedures, fields, and bad cases.
- Manually testing for multilingual sites is challenging.
- In software testing, test automation eliminates the need for human interaction.
- Test automation accelerates the execution of tests.
- Increased Test Coverage is aided by automation.
- Manual testing may become tedious and, as a result, prone to errors.

45

45

Process of Automated Testing

- Choose a test tool
- Define the Automation Scope
- Design, Planning, and Development
- Execution of the Test
- Maintenance

46

46

Introduction to Selenium

- Selenium is an open-source and a portable automated software testing tool for testing web applications.
- It has capabilities to operate across different browsers and operating systems.
- Tool : Selenium IDE, Selenium RC, Selenium WebDriver, Selenium Grid

<https://www.tutorialspoint.com/selenium/>

47

47

Selenium - IDE Download

- <https://www.selenium.dev/downloads/>



Selenium IDE

Selenium IDE is a Chrome, Firefox and Edge plugin which records and plays back user interactions with the browser. Use this to either create simple scripts or assist in exploratory testing.

Download latest released version for [Chrome](#) or [Firefox](#) or [Edge](#). View the [Release Notes](#).

Download previous IDE [versions](#).


48

48

Selenium - IDE Download



Selenium IDE by Selenium

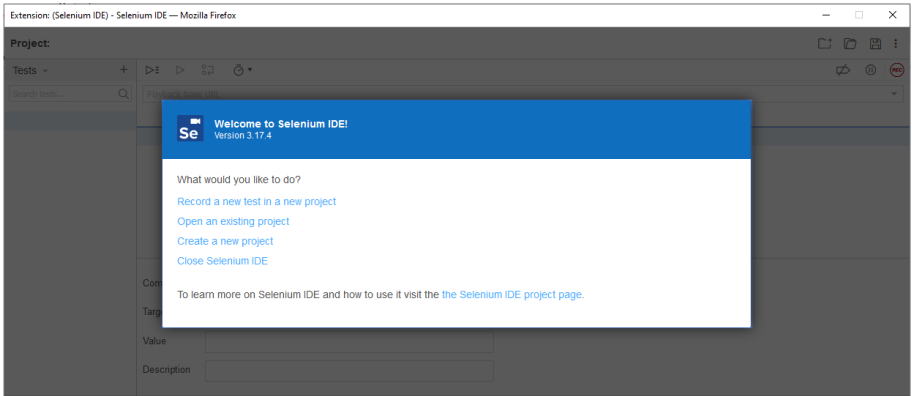
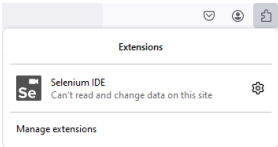
 This add-on is not actively monitored for security by Mozilla. Make sure you trust it before installing.
[Learn more](#)

Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests.

49

49

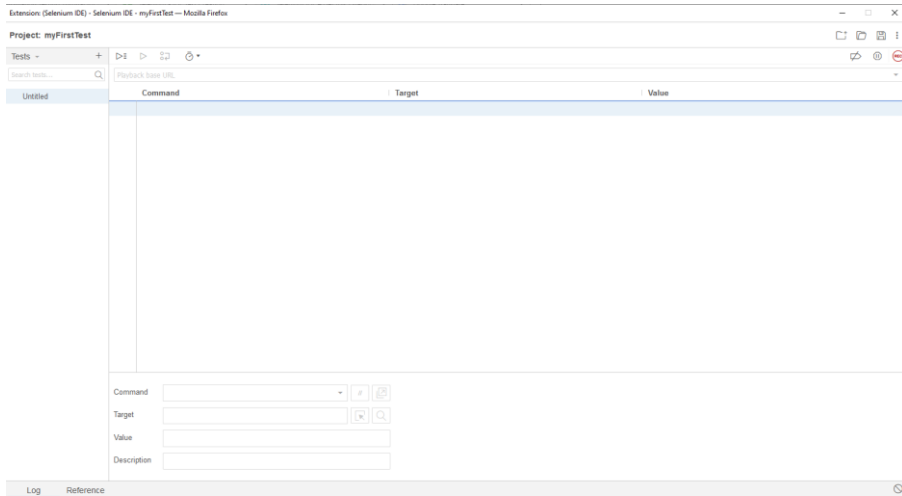
Selenium - IDE



50

50

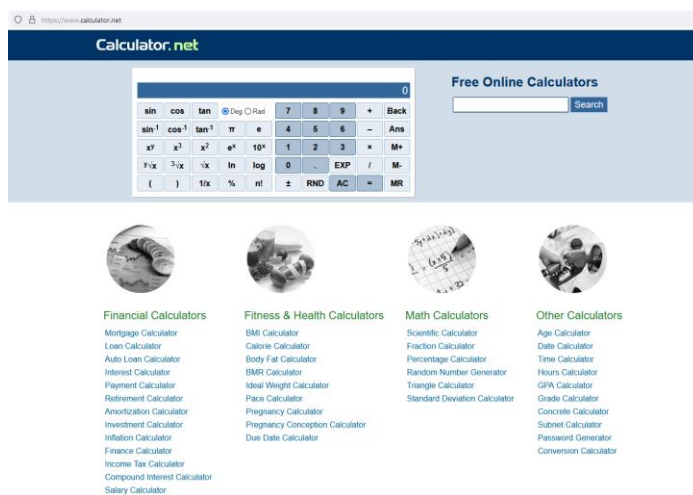
Selenium - IDE



51

51

Selenium - IDE : Test Creation



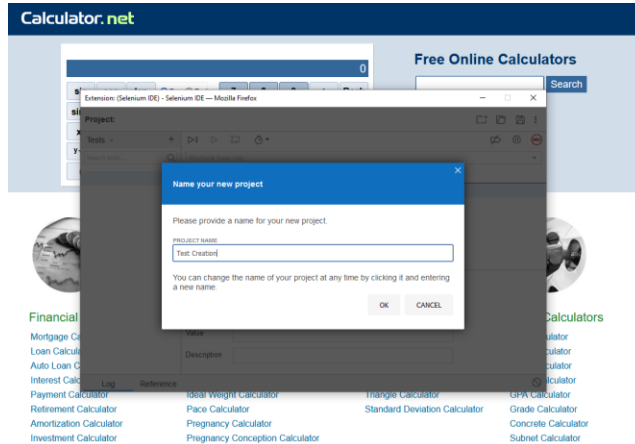
<https://www.calculator.net/>

52

52

Selenium - IDE : Test Creation

- Open Selenium – IDE ----> New Project : Test Creation



53

53

Selenium - IDE : Test Creation

- Set your project's base URL : <https://www.calculator.net/>

Set your project's base URL

Before you can start recording, you must specify a valid base URL for your project. Your tests will start by navigating to this URL.

BASE URL

START RECORDING
CANCEL

54

54

Selenium - IDE : Test Creation

- Create Test Case 1



Math Calculators

Scientific Calculator

Fraction Calculator

Percentage Calculator

Random Number Generator

Triangle Calculator

Standard Deviation Calculator

Percentage Calculator

Please provide any two values below and click the "Calculate" button to get the third value.

Result: 0.5

10% of 5 = **0.5**

Steps:

10% of 5 = $0.1 \times 5 = 0.5$

55

55

Selenium - IDE : Test Creation

- Result Create Test Case 1

Extension: [Selenium IDE] - Selenium IDE - Test Creation* — Mozilla Firefox

Project: Test Creation*

Command	Target	Value
1. open	/	
2. set window size	1297x787	
3. click	linkText=Percentage Calculator	
4. click	id=cpar1	
5. type	id=cpar1	10
6. click	id=cpar2	
7. type	id=cpar2	5
8. click	name=x	

56

56

Selenium - IDE : Test Creation

- Create Test Case 2



Math Calculators

Scientific Calculator

Fraction Calculator

Percentage Calculator

Random Number Generator

Triangle Calculator

Standard Deviation Calculator

Percentage Calculator

Please provide any two values below and click the "Calculate" button to get the third value.

Result: 2

20% of 10 = **2**

Steps:
20% of 10 = $0.2 \times 10 = 2$

57

57

Selenium - IDE : Test Creation

- Result Create Test Case 2

Extension: (Selenium IDE) - Selenium IDE - Test Creation — Mozilla Firefox

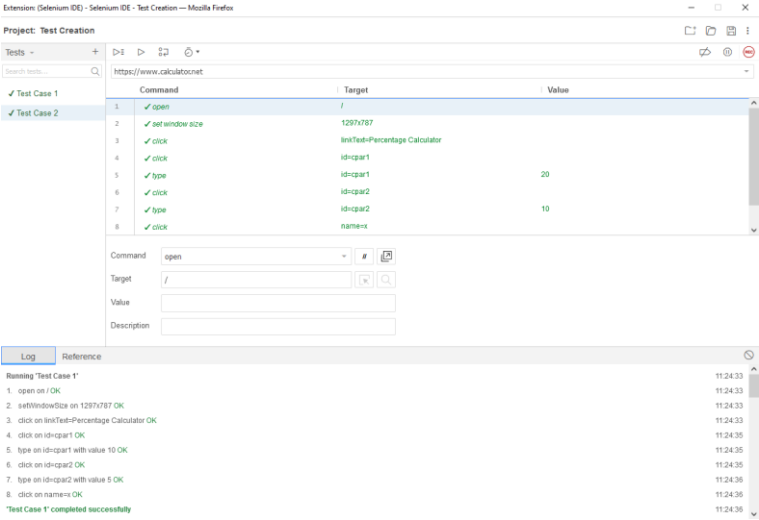
Project: Test Creation

Tests	Command	Target	Value
Test Case 1	1 open	/	
Test Case 2	2 set window size	1297x787	
	3 click	linkText=Percentage Calculator	
	4 click	id=cpar1	
	5 type	id=cpar1	20
	6 click	id=cpar2	
	7 type	id=cpar2	10
	8 click	name=x	

58

58

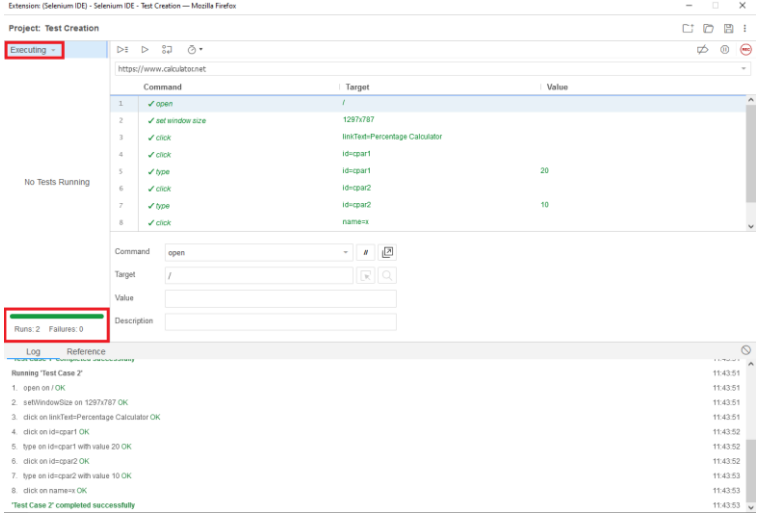
Executing the Recorded Test



59

59

Executing the Recorded Test



60

60

Selenium - IDE Debugging

Extension: (Selenium IDE) - Selenium IDE - Test Creation* — Mozilla Firefox

Project: Test Creation*

Tests - + [Icons]

Search tests... [Search Icon] https://www.calculator.net

	Command	Target	Value
✓ Test Case 1	1 open	/	
✓ Test Case 2	2 ✓ set window size	1297x787	
	3 ✓ click	linkText=Percentage Calculator	
	4 ✓ click	id=cpar1	
	5 ✓ type	id=cpar1	10
	6 ✓ click	id=cpar2	
	7 ✓ type	id=cpar2	
	8 ✓ click	name=+	

Command type [Dropdown] [Icons]

Target id=cpar1 [Dropdown] [Icons]

Value 10 [Input]

Description [Text Area]

Log Reference

Running 'Test Case 1'

- open on / OK
- setWindowSize on 1297x787 OK
- click on linkText=Percentage Calculator OK
- click on id=cpar1 OK

Context Menu:

- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Delete Del
- Insert new command
- Clear all commands
- Toggle breakpoint B**
- Play to this point S
- Record from here U
- Play from here

61

61

Selenium Python Installation

- Installation

```
> python --version
Python 3.11.2
> python -m pip install selenium
```

- Download Driver (<https://github.com/mozilla/geckodriver/releases>)

Assets 12

geckodriver-v0.33.0-linux-aarch64.tar.gz	2.93 MB	Apr 3
geckodriver-v0.33.0-linux32.tar.gz	3.02 MB	Apr 3
geckodriver-v0.33.0-linux32.tar.gz.asc	833 Bytes	Apr 3
geckodriver-v0.33.0-linux64.tar.gz	2.93 MB	Apr 3
geckodriver-v0.33.0-linux64.tar.gz.asc	833 Bytes	Apr 3
geckodriver-v0.33.0-macos-aarch64.tar.gz	1.85 MB	Apr 3
geckodriver-v0.33.0-macos.tar.gz	2.05 MB	Apr 3
geckodriver-v0.33.0-win-aarch64.zip	1.47 MB	Apr 3
geckodriver-v0.33.0-win32.zip	1.54 MB	Apr 3
geckodriver-v0.33.0-win64.zip	1.59 MB	Apr 3
Source code (zip)		Apr 3
Source code (tar.gz)		Apr 3

```
PS D:\Selenium>
>> setx path "%path%;C:/selenium/geckodriver.exe"
```

62

62

Selenium Python – Get Web Page

```
# import webdriver
from selenium import webdriver

# create webdriver object
driver = webdriver.Firefox()
# get google.co.th
driver.get("https://google.co.th")
```

63

63

Selenium Python – Navigating Links

```
# import webdriver
from selenium import webdriver

# create webdriver object
driver = webdriver.Firefox()

# get google.co.in
driver.get("https://www.google.co.th/search?q=the+sportory")
```

64

64

Selenium Python – Interacting With Web Page

```
# import webdriver
from selenium import webdriver
from selenium.webdriver.common.by import By

# create webdriver object
driver = webdriver.Firefox()

# get web page
driver.get("http://www.python.org")

# get element
element = driver.find_element(By.ID, "id-search-field")

# send keys
element.send_keys("pycon")
```

65

65

Selenium Python – Find Element By Name

```
# import webdriver
from selenium import webdriver
from selenium.webdriver.common.by import By

# create webdriver object
driver = webdriver.Firefox()

# enter keyword to search
keyword = "pycon"

driver.get("http://www.python.org")

# get element
element = driver.find_element(By.NAME, "q")

# print complete element
print(element)
```

66

66



Thank you

67