Universidad de Nacional de San Agustín
Facultad de Ingeniería de Producción y Servicios
Escuela Profesional de Ciencia de la Computación

Laboratorio 05:
Pruebas Funcionales con Selenium y xUnit

Asignatura: **Ingeniería de Software II**

Elaborado por: Sergio Daniel Mogollon Caceres

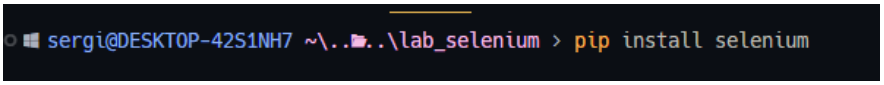
Arequipa - Perú
2023

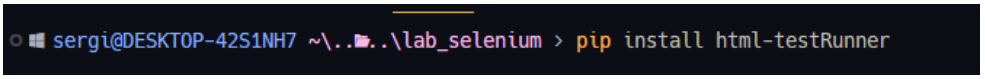
ACTIVIDADES

1. **Crear un proyecto de pruebas funcionales (de acuerdo al lenguaje elegido)**
   Se utilizó Python. Bibliotecas: Selenium y UnitTest

---

2. **Gestionar las dependencias necesarias (Selenium Web Driver y xUnit):** https://www.selenium.dev/documentation/webdriver/getting_started/install_library/





---

3. **Configurar driver de navegador (Browser-specific WebDriver) [https://www.selenium.dev/documentation/webdriver/getting_started/install_drivers/]**
   A partir de las nuevas versiones, *Webdriver* viene habilitado por defecto por el navegador, no es necesario descargar uno extra

---

4. **Implementar las pruebas funcionales de la siguiente funcionalidad:** https://www.calculator.net/percent-calculator.html

   4.1 Diseñar los casos de prueba utilizando alguna estrategia basada en caja negra (p.ej., clases de equivalencia, valores límite, casos de uso, tablas de decisión, ...):

Estrategia basada en Casos de Uso

1 Caso de Uso: Manejar Entradas Positivas de Porcentaje

Descripción: El usuario ingresa al sistema, selecciona la opción de la calculadora de porcentaje, ingresa dos números y solicita el cálculo del porcentaje.

Pasos:

- Usuario abre la aplicación de la calculadora.
- Selecciona la opción de "Calculadora de Porcentaje".
- Ingresa el primer número.
- Ingresa el segundo número.
- Solicita el cálculo del porcentaje.
- El sistema realiza el cálculo y muestra el resultado.

2 Caso de Uso: Manejar Entradas Negativas de Porcentaje

Descripción: El usuario verifica cómo el sistema maneja las entradas de porcentajes negativos.

Pasos:

- Usuario abre la aplicación de la calculadora.
- Selecciona la opción de "Calculadora de Porcentaje".

- Ingresa un número negativo como el primer número.
- Ingresa un número positivo como el segundo número.
- Solicita el cálculo del porcentaje.
- El sistema maneja adecuadamente las entradas negativas y muestra el resultado.

3 Caso de Uso: Validar Entradas con Números Decimales

Descripción: El usuario verifica cómo el sistema maneja las entradas de números decimales.

Pasos:

- Usuario abre la aplicación de la calculadora.
- Selecciona la opción de "Calculadora de Porcentaje".
- Ingresa un número decimal como el primer número.
- Ingresa otro número decimal como el segundo número.
- Solicita el cálculo del porcentaje.
- El sistema valida y maneja correctamente las entradas decimales, mostrando el resultado preciso.

Estrategia basada en Valores Límite

Se centran en los valores límite y en las condiciones límite del conjunto de datos. Puede revelar errores que ocurren en los extremos.

4 Caso de Prueba para Valor Límite Inferior

Escenario de Prueba: Calcular el porcentaje con el valor mínimo permitido.

Valores de Prueba:

- **Primer número:** Valor mínimo permitido (-100000000000000).
- **Segundo número:** Cualquier valor válido (por ejemplo, 50).

Resultado Esperado: Se espera que el sistema maneje correctamente el valor límite inferior y produzca un resultado válido.

5 Caso de Prueba para Valor Límite Superior

Escenario de Prueba: Calcular el porcentaje con el valor máximo permitido.

Valores de Prueba:

- **Primer número:** Cualquier valor válido (100000000000000).
- **Segundo número:** Valor máximo permitido (por ejemplo, 50).

Resultado Esperado: Se espera que el sistema maneje correctamente el valor límite superior y produzca un resultado válido.

Escenario de Prueba | Valores de Prueba | Resultado Esperado

Escenario de Prueba	Valores de Prueba	Resultado Esperado
Validar entradas con números positivos	num1 = 10 num2 = 50	5
Validar entradas con números negativos	num1 = -20 num2 = 100	-20
Validar entradas con números decimales.	num1=7.5 num2=80.5	6.0375
Calcular el porcentaje con el valor mínimo permitido	num1= -100000000000000 num2= 50	-50000000000000
Calcular el porcentaje con el valor máximo permitido	num1= 100000000000000 num2= 50	50000000000000

4.2 Implementar los casos de prueba por medio de scripts de prueba utilizando los elementos proporcionados por Selenium: https://www.selenium.dev/documentation/webdriver/getting_started/first_script/

- *Implementación de los 5 casos de prueba diseñados en la tabla*

Python

```
def test_calculate_percentage_positive_numbers(self):
    # Caso de prueba: Calcular el porcentaje con números positivos
    self.driver.find_element(By.ID, "cpar1").send_keys("10")
    self.driver.find_element(By.ID, "cpar2").send_keys("50")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "5") # 10% de 50 es 5

  def test_calculate_percentage_negative_numbers(self):
    # Caso de prueba: Calcular el porcentaje con números negativos
    self.driver.find_element(By.ID, "cpar1").send_keys("-20")
    self.driver.find_element(By.ID, "cpar2").send_keys("100")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "-20") # -20% de 100 es -20

  def test_calculate_percentage_decimal_numbers(self):
    # Caso de prueba: Calcular el porcentaje con números decimales
    self.driver.find_element(By.ID, "cpar1").send_keys("7.5")
    self.driver.find_element(By.ID, "cpar2").send_keys("80.5")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()
```

```
    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "6.0375") # 7.5% de 80.5 es 6.0375

  def test_calculate_percentage_with_minimum_value(self):
    # Caso de prueba: Calcular el porcentaje con el valor mínimo permitido
    self.driver.find_element(By.ID, "cpar1").send_keys("-100000000000000")
    self.driver.find_element(By.ID, "cpar2").send_keys("50")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "-50000000000000") # -999999999% de 50 es
-1999999998

  def test_calculate_percentage_with_maximum_value(self):
    # Caso de prueba: Calcular el porcentaje con el valor máximo permitido
    self.driver.find_element(By.ID, "cpar1").send_keys("100000000000000")
    self.driver.find_element(By.ID, "cpar2").send_keys("50")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "50000000000000") # 999999999% de 999999999 es
999999999
```

4.3 Actualizar (o traducir al lenguaje escogido) el script de prueba proporcionado en el archivo adjuntado:

4.4 Cada caso de prueba y su correspondiente script debe ser implementado en base a xUnit: Test Case, ASSERT, SetUp(), TearDown()

- *Implementación completa con ASSERTS, Setup(), tearDown() y test Cases:*

Python

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from HtmlTestRunner import HTMLTestRunner

class TestPercentageCalculator(unittest.TestCase):
  def setUp(self):
    # Configuración: Se ejecuta antes de cada caso de prueba
    self.driver = webdriver.Chrome()
    self.driver.get("http://www.calculator.net/")
    # Hacer clic en Math Calculators
    self.driver.find_element(By.XPATH,
"//*[@id='homelistwrap']/div[3]/div[2]/a").click()
    # Hacer clic en Percent Calculator
    self.driver.find_element(By.XPATH,
"//*[@id='content']/table[2]/tbody/tr/td/div[3]/a").click()
```

```
    # xpath de botones de interaccion
    self.xpath_btn_calculate =
"//*[@id='content']/form[1]/table/tbody/tr[2]/td/input[2]"
    self.xpath_result = "//*[@id='content']/p[2]/font/b"

  def tearDown(self):
    # time.sleep(5)
    self.driver.close()

  def test_calculate_percentage_positive_numbers(self):
    # Caso de prueba: Calcular el porcentaje con números positivos
    self.driver.find_element(By.ID, "cpar1").send_keys("10")
    self.driver.find_element(By.ID, "cpar2").send_keys("50")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "5") # 10% de 50 es 5

  def test_calculate_percentage_negative_numbers(self):
    # Caso de prueba: Calcular el porcentaje con números negativos
    self.driver.find_element(By.ID, "cpar1").send_keys("-20")
    self.driver.find_element(By.ID, "cpar2").send_keys("100")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "-20") # -20% de 100 es -20

  def test_calculate_percentage_decimal_numbers(self):
    # Caso de prueba: Calcular el porcentaje con números decimales
    self.driver.find_element(By.ID, "cpar1").send_keys("7.5")
    self.driver.find_element(By.ID, "cpar2").send_keys("80.5")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "6.0375") # 7.5% de 80.5 es 6.0375

  def test_calculate_percentage_with_minimum_value(self):
    # Caso de prueba: Calcular el porcentaje con el valor mínimo permitido
    self.driver.find_element(By.ID, "cpar1").send_keys("-100000000000000")
    self.driver.find_element(By.ID, "cpar2").send_keys("50")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "-50000000000000") # -999999999% de 50 es
-1999999998

  def test_calculate_percentage_with_maximum_value(self):
    # Caso de prueba: Calcular el porcentaje con el valor máximo permitido
    self.driver.find_element(By.ID, "cpar1").send_keys("100000000000000")
    self.driver.find_element(By.ID, "cpar2").send_keys("50")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()
```

```
    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "50000000000000") # 999999999% de 999999999 es
999999999

  def test_calculate_percentage_with_zero(self):
    # Caso de prueba: Calcular el porcentaje con un valor igual a 0
    self.driver.find_element(By.ID, "cpar1").send_keys("0")
    self.driver.find_element(By.ID, "cpar2").send_keys("100")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "0") # 0% de 100 es 0

  def test_calculate_percentage_with_both_zeros(self):
    # Caso de prueba: Calcular el porcentaje con ambos valores igual a 0
    self.driver.find_element(By.ID, "cpar1").send_keys("0")
    self.driver.find_element(By.ID, "cpar2").send_keys("0")
    self.driver.find_element(By.XPATH, self.xpath_btn_calculate).click()

    result = self.driver.find_element(By.XPATH, self.xpath_result).text
    self.assertEqual(result, "0") # 0% de 0 es 0


if __name__ == "__main__":
  report_name = 'reporte_de_pruebas.html'
  suite =
unittest.TestLoader().loadTestsFromTestCase(TestPercentageCalculator)

  # Configura HTMLTestRunner con el nombre personalizado
  with open(report_name, 'w') as f:
    runner = HTMLTestRunner(stream=f, verbosity=2,
report_name="reporte_de_pruebas")
    result = runner.run(suite)
  print(result)
```

4.5 Reportar el resultado de ejecución de los casos de prueba por medio de xUnit.

- *Reporte de Tests generado en formato txt*

```
Unset

Running tests...
----------------------------------------------------------------------
 test_calculate_percentage_decimal_numbers
(__main__.TestPercentageCalculator.test_calculate_percentage_decimal_numbers
) ... OK (5.554131)s
```

```
 test_calculate_percentage_negative_numbers
(__main__.TestPercentageCalculator.test_calculate_percentage_negative_number
s) ... OK (5.362554)s
 test_calculate_percentage_positive_numbers
(__main__.TestPercentageCalculator.test_calculate_percentage_positive_number
s) ... OK (5.398499)s
 test_calculate_percentage_with_both_zeros
(__main__.TestPercentageCalculator.test_calculate_percentage_with_both_zeros
) ... OK (5.586120)s
 test_calculate_percentage_with_maximum_value
(__main__.TestPercentageCalculator.test_calculate_percentage_with_maximum_va
lue) ... OK (5.284944)s
 test_calculate_percentage_with_minimum_value
(__main__.TestPercentageCalculator.test_calculate_percentage_with_minimum_va
lue) ... OK (5.373235)s
 test_calculate_percentage_with_zero
(__main__.TestPercentageCalculator.test_calculate_percentage_with_zero) ...
OK (5.050671)s

----------------------------------------------------------------------
Ran 7 tests in 0:00:52

OK



Generating HTML reports...
reports\reporte_de_pruebas___main__.TestPercentageCalculator_2023-11-14_17-5
4-41.html
```

- *Reporte generado en formato HTML por unittest de python y html-test-runner*