

Grado en Ingeniería del Software  
Doble Grado en Matemática Computacional e Ingeniería del Software  
Doble Grado en Física Computacional e Ingeniería de Software

# Verificación de Software



## Práctica Técnicas BDD con Behave y Selenium

Alonso Álvarez García  
Rafael Socas Gutiérrez

Curso 2023/24



## Datos de los alumnos

#	Nombre y apellidos	Curso
1		
2		
3		
4		
5		

## Instrucciones

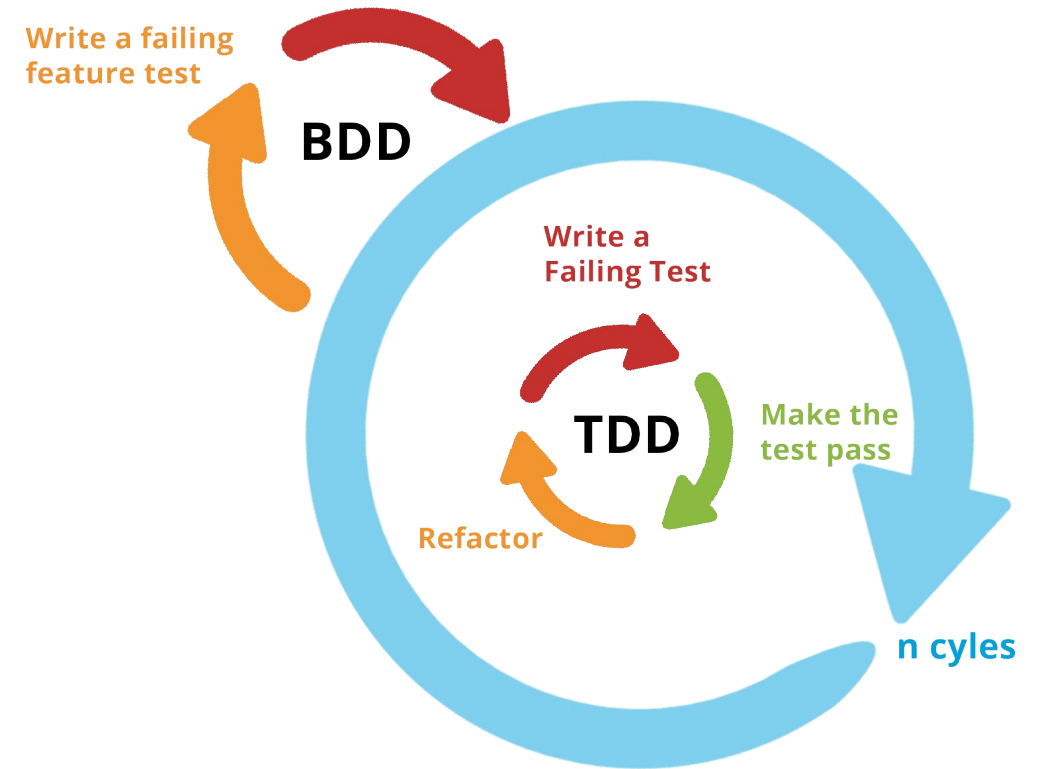
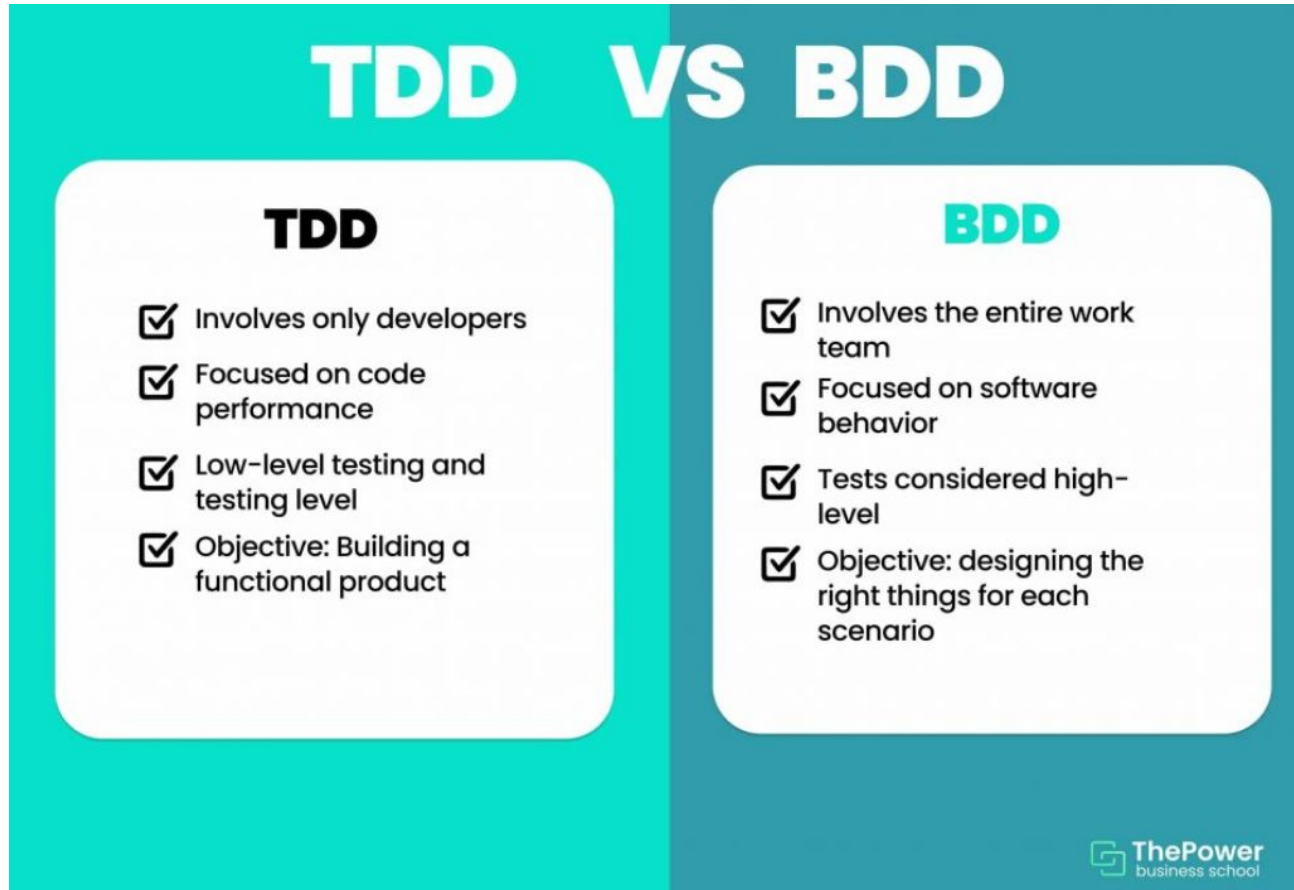
- Completa la práctica en este mismo Power Point rellenando las páginas en blanco o incluyendo más páginas si necesitas más espacio para los pantallazos y las explicaciones.
- Una vez completado el Power Point, guárdalo en formato pdf. **A la plataforma BB sube el pdf resultante.**
- Sube a BB también los ficheros **prestamos.feature** y **web\_steps.py** con las definiciones y el código Python ya depurado y asegurándote que funciona correctamente todos los tests.
- Rellene el nombre/apellidos y el curso de los participantes del grupo.
- **IMPORTANTE:** Recordad que en un contexto profesional importa mucho la forma, además del contenido. No se trata únicamente de hacer bien el trabajo, hay que saber transmitirlo adecuadamente. Es decir, cuidad la presentación de resultados. Además, siempre que sea posible, haremos una miniexposición en clase. **Esta parte supone el 20% de la nota.**
- **Fecha máxima de entrega: lunes 6 mayo 23:59.**

2

Puntos

## Contexto

## Filosofía BDD y su relación con TDD



## Contexto

Behavior-Driven Development (or BDD) is an agile software development technique that encourages collaboration between developers, QA and non-technical or business participants in a software project. **Behave uses tests written in a natural language style, backed up by Python code.**

### Gherkin

- A natural language
- Only a specification

### Gherkin / Specification

- Feature
  - Scenario (use cases)
    - Given
    - When
    - Then
- Scenario Outline

### Gherkin

**Feature:** Some terse yet descriptive text of what is desired

**@BusinessCritical @Meh**

**Scenario:** Some determinable business situation

**Given** some precondition

**And** some other precondition

**When** some action by the actor


**And** some other action

**And** yet another action

**Then** some testable outcome is achieved

**And** something else we can check happens too

### Gherkin

- Implementations:
  - **Behave (python)** 
  - Jcucumber (java)
  - Cucumber-JS (nodejs)
  - Cucumber (ruby)

## Contexto

### Framework Behave

#### Behave

- Python implementation
  - pip install behave (python 2.7+)
- Decorators for Gherkin
  - @given
  - @when
  - @then
  - @step

#### Behave Context

- Context Levels
  - All
  - Feature
  - Scenario
- Automatic Cleanup on same level
- Context inheritance
  - All → Feature → Scenario

#### Behave

- features/\*.feature
- features/steps/\*.py
- features/environment.py

#### Behave Lifecycle

- features/environment.py
  - All (before\_all, after\_all)
  - Feature (before\_feature, after\_feature)
  - Scenario (before\_scenario, after\_scenario)
  - Step (before\_step, after\_step)
  - Tag (before\_tag, after\_tag)
- features/steps/\*.py
  - @step/@given/@when/@then

#### Behave Run

- Behave
  - -t (specify tags to run, not = ~)
  - --no-capture



## Contexto

### Ejemplo: BDD basado en Behave

```
# -- FILE: features/example.feature  
Feature: Showing off behave
```

```
Scenario: Run a simple test  
  Given we have behave installed  
  When we implement 5 tests  
  Then behave will test them for us!
```

```
# -- FILE: features/steps/example_steps.py  
from behave import given, when, then, step  
  
@given('we have behave installed')  
def step_impl(context):  
    pass  
  
@when('we implement {number:d} tests')  
def step_impl(context, number): # -- NOTE: number is converted into integer  
    assert number > 1 or number == 0  
    context.tests_count = number  
  
@then('behave will test them for us!')  
def step_impl(context):  
    assert context.failed is False  
    assert context.tests_count >= 0
```

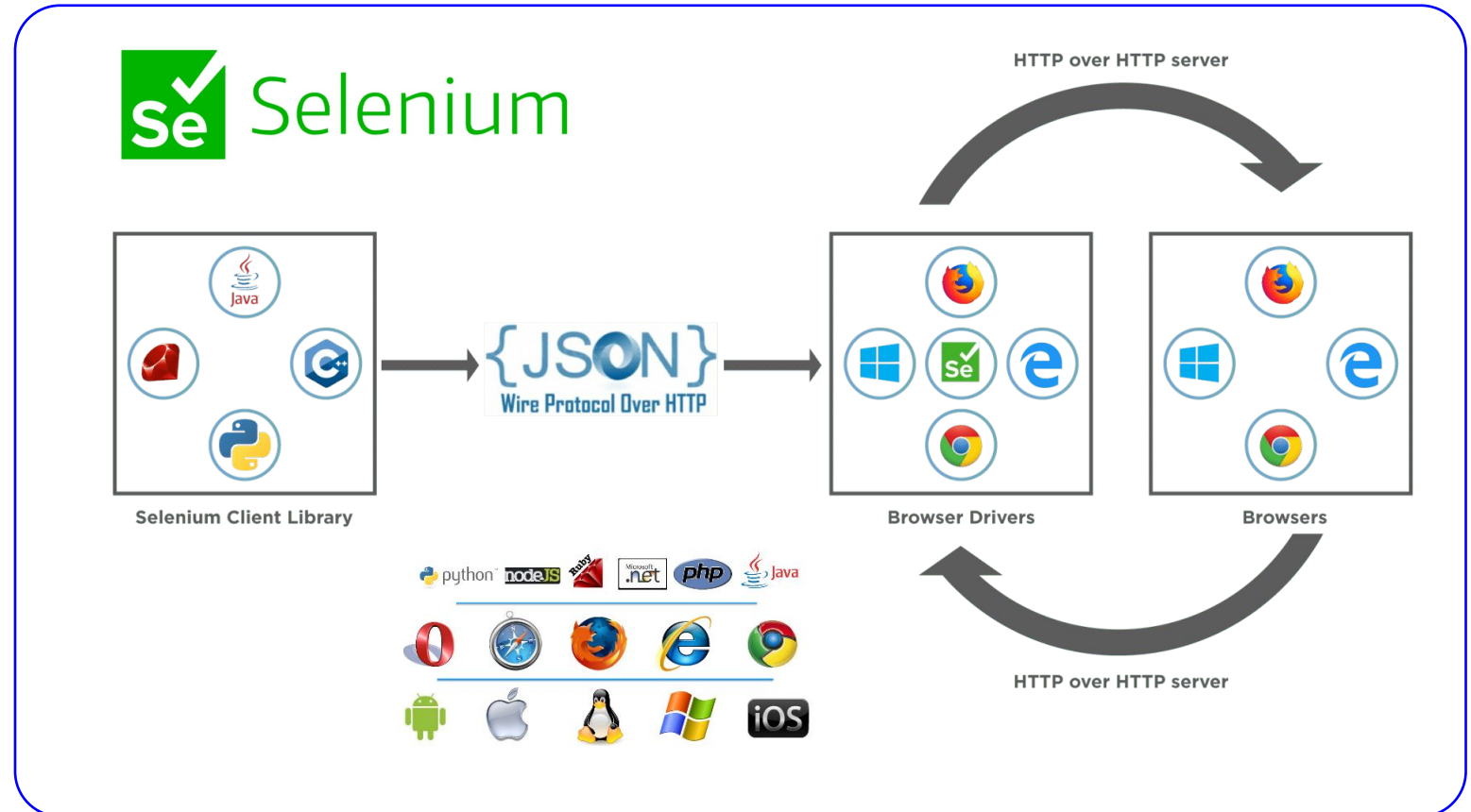
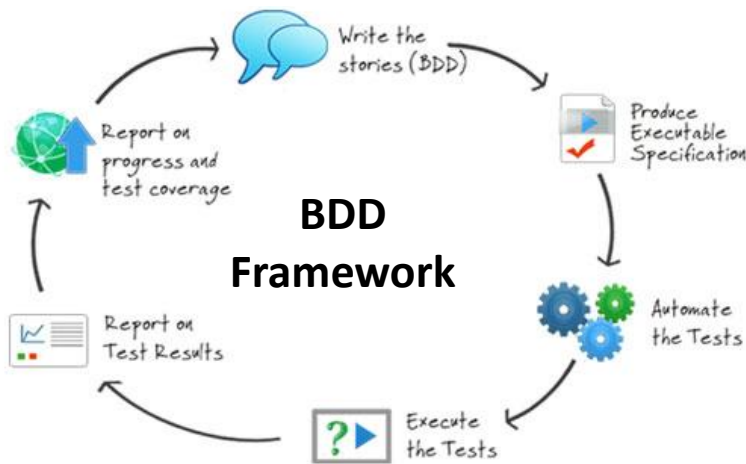
### Tests BDD con Behave

```
$ behave  
Feature: Showing off behave # features/example.feature:2  
  
Scenario: Run a simple test # features/example.feature:4  
  Given we have behave installed # features/steps/example_steps.py:4  
  When we implement 5 tests # features/steps/example_steps.py:8  
  Then behave will test them for us! # features/steps/example_steps.py:13  
  
1 feature passed, 0 failed, 0 skipped  
1 scenario passed, 0 failed, 0 skipped  
3 steps passed, 0 failed, 0 skipped, 0 undefined
```



## Contexto

If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use **Selenium WebDriver**, a collection of language specific bindings to drive a browser.





## Contexto

## Ejemplo: Uso de la API Selenium para Python

### Example 1:

- open a new Firefox browser
- load the Yahoo homepage
- search for “seleniumhq”
- close the browser

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

browser = webdriver.Firefox()

browser.get('http://www.yahoo.com')
assert 'Yahoo' in browser.title

elem = browser.find_element(By.NAME, 'p') # Find the search box
elem.send_keys('seleniumhq' + Keys.RETURN)

browser.quit()
```

### Example 2:

Selenium WebDriver is often used as a basis for testing web applications. Here is a simple example using Python's standard [unittest](#) library:

```
import unittest
from selenium import webdriver

class GoogleTestCase(unittest.TestCase):

    def setUp(self):
        self.browser = webdriver.Firefox()
        self.addCleanup(self.browser.quit)

    def test_page_title(self):
        self.browser.get('http://www.google.com')
        self.assertIn('Google', self.browser.title)

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

## Objetivos

- Definir test de alto nivel en lenguaje natural (mediante la sintaxis Gherkin) y aplicarlos mediante técnicas BDD.
- Instalar y configurar el framework **Behave** para aplicar metodologías BDD en lenguaje Python.
- Usar el entorno **Selenium** para automatizar test BDD en app Web mediante Python y el **Webdriver** de Chrome.
- Definir y ejecutar diferentes escenarios de test para una app Web.

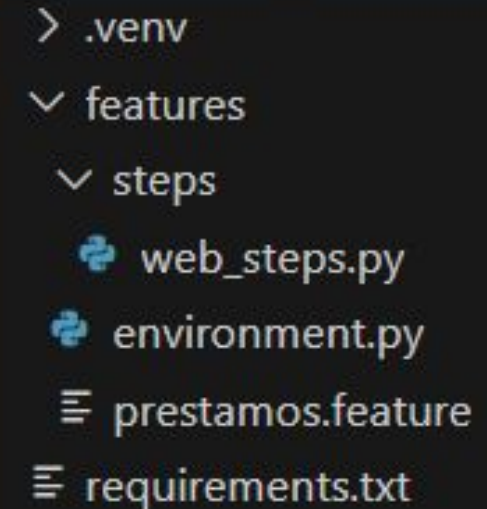


## Tarea 1: preparación del entorno (1/3)

- Nos apoyaremos en el entorno de desarrollo (IDE) Visual Studio Code y en el intérprete Python 3. Para instalarlo se seguirá el siguiente tutorial <https://code.visualstudio.com/docs/python/python-tutorial> .
- Crear una carpeta con donde se guardarán el código a testear y los tests correspondientes. Los ficheros iniciales a incluir en esa carpeta se aportan junto con este enunciado
- En el IDE, abrir la carpeta y crear un entorno virtual (Ctrl+Shift+P) de tipo Venv (ver tutorial anterior).

## Tarea 1: preparación del entorno (2/3)

- Crear la siguiente estructura de ficheros y carpetas e incluir los ficheros proporcionados en la práctica (se proporcionan junto con el enunciado):
  - `requirements.txt`
  - `environment.py`
- El alumno tiene que crear durante la elaboración de la práctica los ficheros:
  - `prestamos.feature`
  - `web_steps.py`
- Instalar los paquetes Python necesarios para configurar el entorno de test:  
> `pip install -r requirements.txt`



```
> .venv
✓ features
  ✓ steps
    • web_steps.py
    • environment.py
    ≡ prestamos.feature
    ≡ requirements.txt
```

## Tarea 1: preparación del entorno (3/3)

1  
Punto

Instalar el webdriver de **Chorme** y  
poner la ruta en el **PATH** de **Windows**

<https://googlechromelabs.github.io/chrome-for-testing/>

### Stable

Version: 119.0.6045.105 (r1204232)

Binary	Platform	URL	HTTP status
chrome	linux64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/linux64/chrome-linux64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/linux64/chrome-linux64.zip</a>	200
chrome	mac-arm64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-arm64/chrome-mac-arm64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-arm64/chrome-mac-arm64.zip</a>	200
chrome	mac-x64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-x64/chrome-mac-x64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-x64/chrome-mac-x64.zip</a>	200
chrome	win32	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win32/chrome-win32.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win32/chrome-win32.zip</a>	200
chrome	win64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win64/chrome-win64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win64/chrome-win64.zip</a>	200
chromedriver	linux64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/linux64/chromedriver-linux64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/linux64/chromedriver-linux64.zip</a>	200
chromedriver	mac-arm64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-arm64/chromedriver-mac-arm64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-arm64/chromedriver-mac-arm64.zip</a>	200
chromedriver	mac-x64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-x64/chromedriver-mac-x64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/mac-x64/chromedriver-mac-x64.zip</a>	200
chromedriver	win32	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win32/chromedriver-win32.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win32/chromedriver-win32.zip</a>	200
chromedriver	win64	<a href="https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win64/chromedriver-win64.zip">https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/119.0.6045.105/win64/chromedriver-win64.zip</a>	200

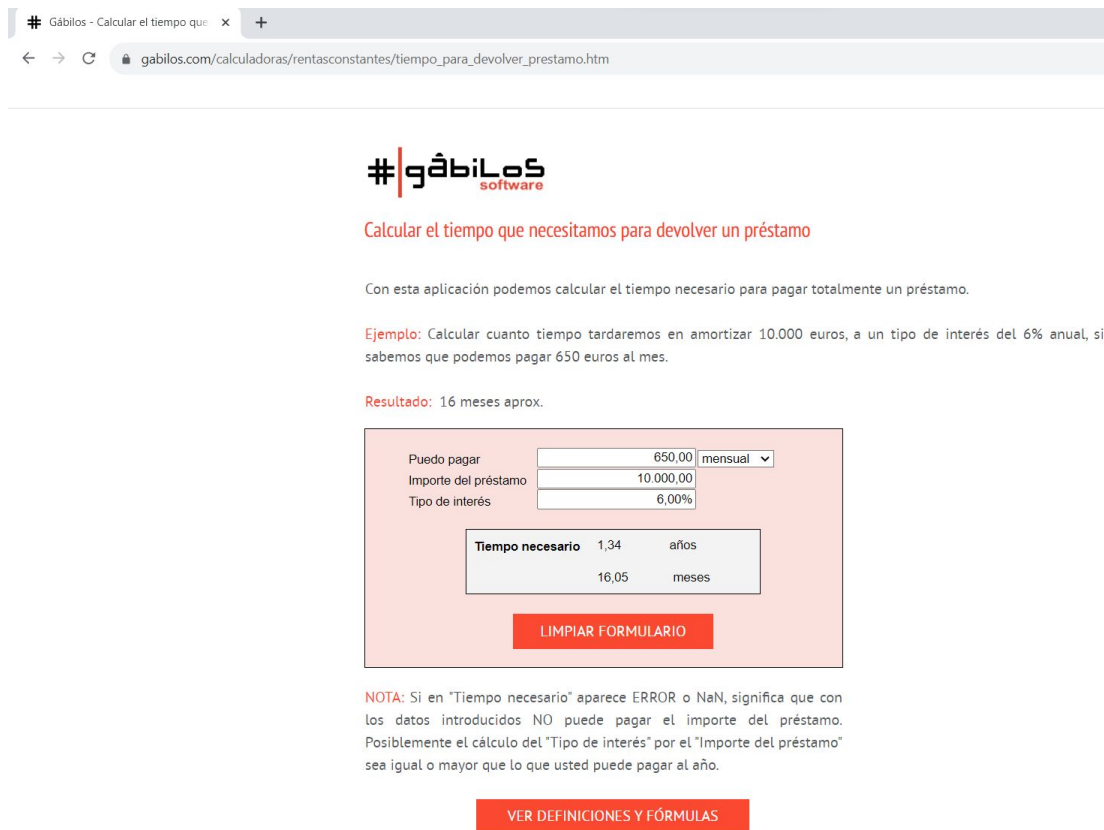






## Aplicación WEB a Testear

[https://www.gabilos.com/calculadoras/rentasconstantes/tiempo\\_para\\_devolver\\_prestamo.htm](https://www.gabilos.com/calculadoras/rentasconstantes/tiempo_para_devolver_prestamo.htm)



# Gábilos - Calcular el tiempo que x +

gabilos.com/calculadoras/rentasconstantes/tiempo\_para\_devolver\_prestamo.htm

#gábilos  
software

Calcular el tiempo que necesitamos para devolver un préstamo

Con esta aplicación podemos calcular el tiempo necesario para pagar totalmente un préstamo.

Ejemplo: Calcular cuanto tiempo tardaremos en amortizar 10.000 euros, a un tipo de interés del 6% anual, si sabemos que podemos pagar 650 euros al mes.

Resultado: 16 meses aprox.

Puedo pagar	650,00	mensual
Importe del préstamo	10.000,00	
Tipo de interés	6,00%	

Tiempo necesario	1,34	años
	16,05	meses

LIMPIAR FORMULARIO

NOTA: Si en "Tiempo necesario" aparece ERROR o NaN, significa que con los datos introducidos NO puede pagar el importe del préstamo. Posiblemente el cálculo del "Tipo de interés" por el "Importe del préstamo" sea igual o mayor que lo que usted puede pagar al año.

VER DEFINICIONES Y FÓRMULAS

Se trata de una WEB que tienen una funcionalidad de cálculo de préstamos de la empresa Gabilos software. Su funcionamiento es muy sencillo:

- Se introduce la cuota a pagar y la periodicidad de las mismas (mensual, trimestral, semestral y anual).
- El importe de préstamo.
- El tipo de interés

Una vez introducidos los datos, la herramienta devuelve el tiempo de devolución del préstamo en años y en meses. También, dispone de un botón para limpiar el formulario e introducir nuevos datos y otro para ir a la ayuda.

## Tarea 2: Definir mediante sintaxis Gherkin y Behave el escenario a testear

1  
Punto

Con esta sintaxis crear un **escenario** en el fichero `prestamos.feature` que analice como mínimo las siguientes fases:

- Vaya a la página WEB.
- Introduzca una cuota de **750€**.
- Un importe del préstamo de **15000€**
- Un tipo de interés del **3%**.
- Y finalmente un periodo de **trimestral**.
- En ese momento, debe aparecer un tiempo necesario para devolver el préstamo de **5,44 años y 21,75 meses**.
- Luego pruebe que el botón de **limpiar el formulario** funciona correctamente.

*Ejemplo*

**IMPORTANTE:** Usar el plan de pruebas desarrollado en la práctica del Plan de Pruebas

Una vez tenga este fichero correctamente definido, ejecute los test con el comando `behave`. Como verá, al no estar definido el fichero `web_steps.py`, **todos los test fallarán**, pero le dará indicaciones de las funciones a definir en el fichero de steps `web_steps.py`.









## Tarea 3: Definir en Python el fichero web\_steps.py con los diferentes steps.

Ahora, ha llegado el momento de definir el código Python a incluir en el fichero `web_steps.py` . Para ello, siga los siguientes pasos:

- Ponga como inicio del fichero las siguientes líneas de código Python

```
from behave import given, when, then
from selenium.webdriver.common.by import By
```

- Luego, programe todos los steps (@given, @when y @then)

- A continuación, se presenta un ejemplo (para uno de los estps de la práctica)

```
@when('Yo pongo el "Capital que puedo pagar" a "750,00" euros')
def step_impl(context):
    element = context.driver.find_element(By.ID, 'p4D6')
    element.clear()
    element.send_keys('750,00')
```

- Analice los siguientes tutoriales que le ayudarán a escribir el código Python
  - <https://selenium-python.readthedocs.io/locating-elements.html>
  - <https://www.selenium.dev/documentation/webdriver/elements/>

## Tarea 3: Definir en Python el fichero web\_steps.py con los diferentes steps.

3  
Puntos

- Para identificar los: *id*, *nombre*, *clase*, etc. de los diferentes elementos, hágalo con el navegador (opción inspect).

The screenshot shows a web application interface on the left and its browser developer tools on the right. The web application has a form with fields for 'Puedo pagar' (650,00), 'Importe del préstamo' (10.000,00), and 'Tipo de interés' (mensual). It also displays 'Tiempo necesario' (1,34 and 16,05) and a 'LIMPIAR FORM' button. A red arrow points from the 'Importe del préstamo' field to the HTML element in the browser's 'Elements' panel. The 'Elements' panel shows the HTML structure, and the 'Styles' panel shows the CSS for the selected element.

Elements panel (HTML structure):

```
<td class="ee119">&nbsp;</td>
<td class="ee120"> Importe del préstamo </td>
<td class="ee121">
  <input value="10.000,00" name="p407" id="p407" type="text"
    onblur="this.value=NumberFormat(this.value, '2', ',',
    '.');recalc_onclick('p407');" tabindex="3" onkeydown="if
    (event.keyCode==13) event.keyCode=9;" class="ee123"
    onkeypress="if (event.keyCode==46) (event.keyCode=44);"
    onkeyup="EvaluateText('%f', this);" => $0
  </td>
<td class="ee120"></td>
<td class="ee124">&nbsp;</td>
</tr>
```

Styles panel (CSS for element .ee123):

```
element.style {
  color: windowtext;
  font-family: Arial;
  font-size: 10.00pt;
  font-style: normal;
  font-weight: 400;
  text-align: right;
  vertical-align: bottom;
}
input[type="text"] {
  writing-mode: horizontal-tb !important;
}
input:not([type="image"]) {
  box-sizing: border-box;
}
```

- Por último, **itere las veces necesarias hasta que se ejecuten todos los test del escenario correctamente**, o no, si la WEB tiene algún bug en su software. **Presente el resultado de los test y comente el resultado.**





## Tarea 4: Definir un nuevo escenario a testear

3  
Puntos

Ahora se trata de definir un nuevo escenario para probar la robustez de la aplicación. Recuerde que, cuantos más escenarios de prueba se diseñen mejor probada estará la aplicación.

El nuevo escenario trata de probar cómo funciona la APP con interés negativo. Para ello, analice como mínimo las siguientes fases:

- Vaya a la página WEB
- Introduzca una cuota de **250€**
- Un importe del préstamo de **10000€**
- Un tipo de interés del **-2%**
- Y finalmente un periodo de **trimestral**
- En ese momento, debe aparecer un tiempo necesario de **3,22 años y 38,69 meses**
- Luego pruebe que el botón de **limpiar el formulario** funciona correctamente

*Ejemplo*

**IMPORTANTE:** Usar el plan de pruebas desarrollado en la práctica del Plan de Pruebas

En este caso, complete los dos ficheros: `prestamos.feature` (con el nuevo escenario) y `web_steps.py` con los nuevos steps (los que ya existan del escenario anterior y sea reutilizables, el sistema los reutilizará). Ejecute `behave`, presente y comente el resultado obtenido en los test.











 Calle Playa de Liencres, 2 bis  
(entrada por calle Rozabella)  
Parque Europa Empresarial  
Edificio Madrid  
28290 Las Rozas, Madrid

 900 373 379  [info@u-tad.com](mailto:info@u-tad.com)

 [SOLICITA MÁS INFORMACIÓN](#)



CENTRO ADSCRITO A:



PROYECTO COFINANCIADO POR:

