

Trabajo Práctico Obligatorio – Automatización de Casos de Prueba

Actividades de Testing en un proyecto Agile (Scrum)

Este Trabajo Práctico Obligatorio (TPO) simula las actividades que realiza un equipo de Calidad, con los roles de QC - QA, QA Automation / SDET dentro de un equipo de Ingeniería, para un proyecto de una plataforma de E-commerce.

Objetivos Generales del TPO

- ☐ Conocer, Entender y Ejecutar algunas de las actividades que Ingenieros de QA, QA Automation o SDET realizan en un equipo de ingeniería
- ☐ Utilizar herramientas de desarrollo que suelen utilizar los equipos de Calidad en proyectos de Software

OBJETIVOS DEL TPO

Organización de los equipos, Setup de Herramientas y Workflow

- Definir los equipos para realizar el Trabajo Práctico Obligatorio
- Instalación de las herramientas:
 - o IDE para desarrollo (se presenta PyCharm como ejemplo)
 - o Selección de la aplicación a usar para llevar a cabo la automatización (aplicación web o local)
- Actividades de Calidad del Software:
 - o Creación de Casos de Prueba (que después automatizarán)
 - o Ejecución de Casos de Prueba (para obtener el script de pytest con selenium)
 - o Automatización de Casos de Prueba
 - o Creación de una presentación con información del TPO

Cómo está presentada esta guía:

1. Actividades del Trabajo Práctico Obligatorio: Automatización de Casos de Prueba
2. Recursos: Guía de instalación de las herramientas (Python, Selenium, IDE de desarrollo)

Fecha de Entrega límite del TPO: Última semana de Clases



Trabajo Práctico Obligatorio

Automatización de Casos de Prueba

Índice

TPO: Automatización de Casos de Prueba	3
Actividades a realizar.....	3
Aplicaciones web sobre las que realizar el TPO	3
Presentación y Entrega del TPO	4
Recursos: Guías de Instalación	5
Herramientas necesarias para automatizar Casos de Prueba	5
Parte A - Pycharm IDE.....	5
Parte B - Webdriver	13
Parte C - Selenium IDE.....	14
Parte D – Assertions (Aserciones)	18

TPO: Automatización de Casos de Prueba

Actividades a realizar

- **Crear al menos 5 Casos de Prueba (TC, Test Cases) sobre alguna de las aplicaciones web presentadas debajo**
 - El formato debe ser similar a los casos de prueba presentados para el ejercicio de la Calculadora (Con precondiciones, pasos de reproducción, resultados esperados, etc)
- **Crear al menos 4 Casos de Prueba Automatizados sobre alguna de las aplicaciones web presentadas debajo**
 - De los casos de prueba que hayan creado seleccionen algunos, graben los pasos de reproducción con Selenium IDE y luego exporten el archivo pytest
 - **Agreguen las aserciones** que les permitan verificar que el TC automatizado hace las mismas (o más) verificaciones que el Caso de Prueba manual
 - **Esto es muy importante, y la aprobación del Trabajo Práctico dependerá en mayoritariamente de esto**
 - Nota: Aquellos elementos dinámicos, o pop-ups en la página (por ejemplo, algún menú que se despliegue para cerrar sesión) son un poco más complejos de automatizar. Por eso, se recomienda que comiencen con TCs simples (Test Cases de navegación por las distintas páginas, acceder al carrito de compra, o completar la compra de algún producto)
- **Crear una presentación** en la que documenten el proceso de resolución del TPO (los detalles de la presentación se presentan en este documento)

Aplicaciones web sobre las que realizar el TPO

Antes que nada, deberán elegir qué aplicación web van a utilizar para realizar el TPO:

1. <https://www.saucedemo.com>
2. <https://www.demoblaze.com/index.html>
3. <https://magento.softwaretestingboard.com/>

Las 3 páginas web son similares (son sitios de e-commerce)

Presentación y Entrega del TPO

1. Crear una presentación en la que deberán especificar:
 - Integrantes del Equipo
 - Plataforma seleccionada para llevar a cabo la automatización (aplicación web, o aplicación local)
 - ¿Por qué eligieron esta aplicación?
 - Presentación de los Casos de Prueba creados:
 - ¿Cuáles son las funcionalidades de la aplicación que cubren los casos de prueba?
 - ¿Por qué seleccionaron ésta / éstas?
 - ¿Identificaron otras funcionalidades para las que no crearon casos de prueba?
 - ¿Utilizaron alguna técnica para identificar o diagramar las funcionalidades de la aplicación? (por ejemplo, transición de estados o diagramas de actividad)
 - ☐ De ser así, presenten los diagramas correspondientes
 - Sobre los Casos de Prueba automatizados:
 - Justificación de la selección de los Casos de Prueba que automatizarán
 - ☐ ¿Prioridad, facilidad de la automatización u otros criterios?
 - ☐ Indiquen el orden en que automatizaron los casos de prueba
2. Además, deberán presentar los scripts de los casos de prueba automatizados.
 - a. El / los scripts a presentar deben ser ejecutables sin errores de compilación
 - b. Los casos de prueba automatizados deberán tener las aserciones (validaciones) que realiza el caso de prueba manual

NOTA: Todo lo presentado (presentación y scripts de código) deberá ser agregado al canal de Teams creado específicamente para el equipo

Recursos: Guías de Instalación

En esta sección se presentan guías de cómo instalar las herramientas necesarias para desarrollar el Trabajo Práctico Obligatorio. La propuesta es que comiencen a explorar algunas de las herramientas que existen en el mercado para realizar la automatización de Casos de Prueba.

Como trabajaremos sobre una aplicación web podemos utilizar herramientas como Selenium, que nos permitirán automatizar acciones a realizar sobre un explorador web. Luego, podremos crear Casos de Prueba automatizados para disminuir el esfuerzo de ejecución manual.

Herramientas necesarias para automatizar Casos de Prueba

Parte A - PyCharm IDE

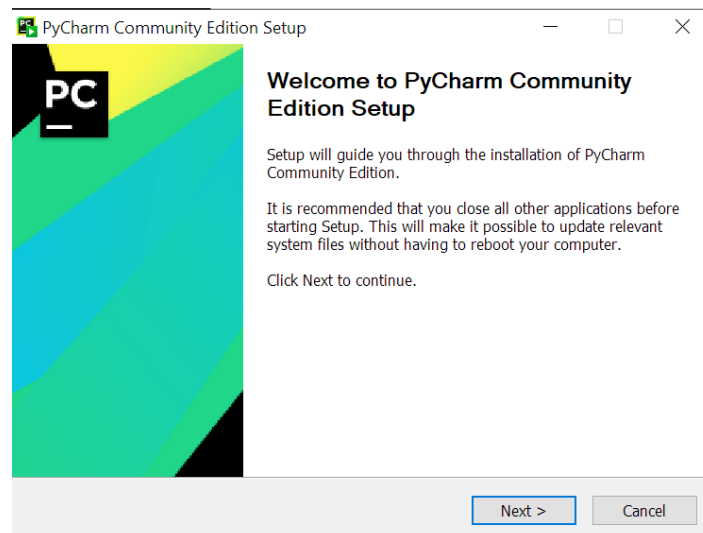
PyCharm es un entorno de desarrollo integrado (IDE) utilizado en [programación informática](#), concretamente para el lenguaje de programación [Python](#). Está desarrollado por la empresa [checa JetBrains](#) (antes conocida como IntelliJ). Proporciona análisis de código, un depurador gráfico, un probador de unidades integrado, integración con sistemas de control de versiones (VCS), y soporta el desarrollo web con Django, así como la ciencia de datos con Anaconda.

PyCharm es [multiplataforma](#), con versiones para [Windows](#), [macOS](#) y [Linux](#). La Community Edition (edición de comunidad) se publica bajo la Licencia Apache, y también hay una Professional Edition (edición profesional) con características adicionales publicada bajo una [licencia propietaria financiada por suscripción](#) y también una versión educativa.

Instalación

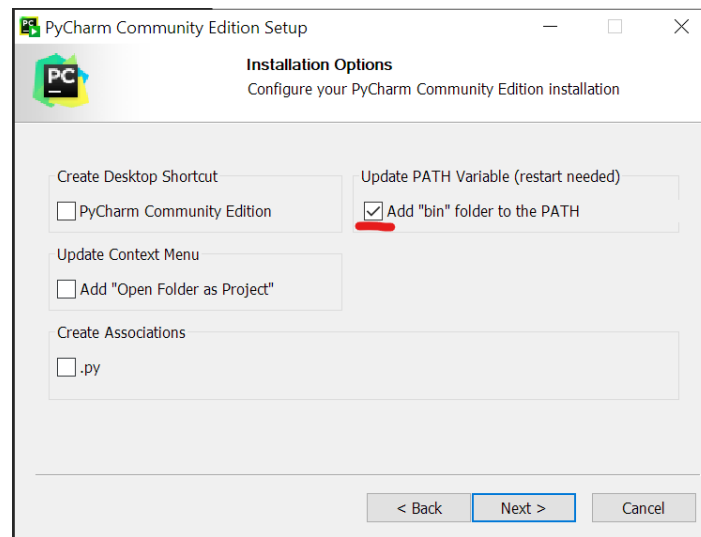
Nota: La siguiente es una guía de cómo instalar Pycharm Community Edition. Si utilizan otro IDE (como Visual Studio Code, por ejemplo) no es necesario que hagan estos pasos

1. Descargar e Instalar Pycharm IDE – Community Edition: <https://www.jetbrains.com/pycharm/>



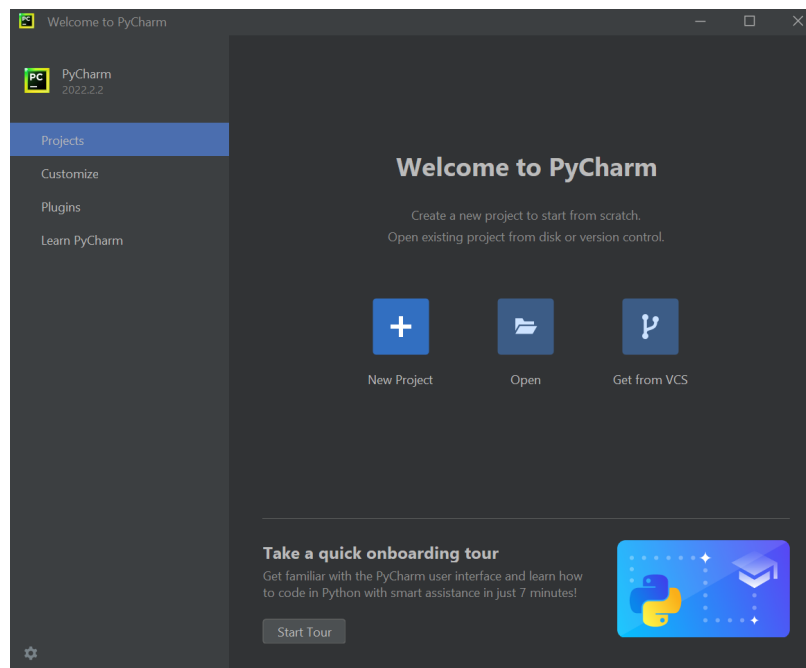
Trabajo Práctico Obligatorio

Automatización de Casos de Prueba



Configuración de Pycharm

- Abrir PyCharm Community Edition



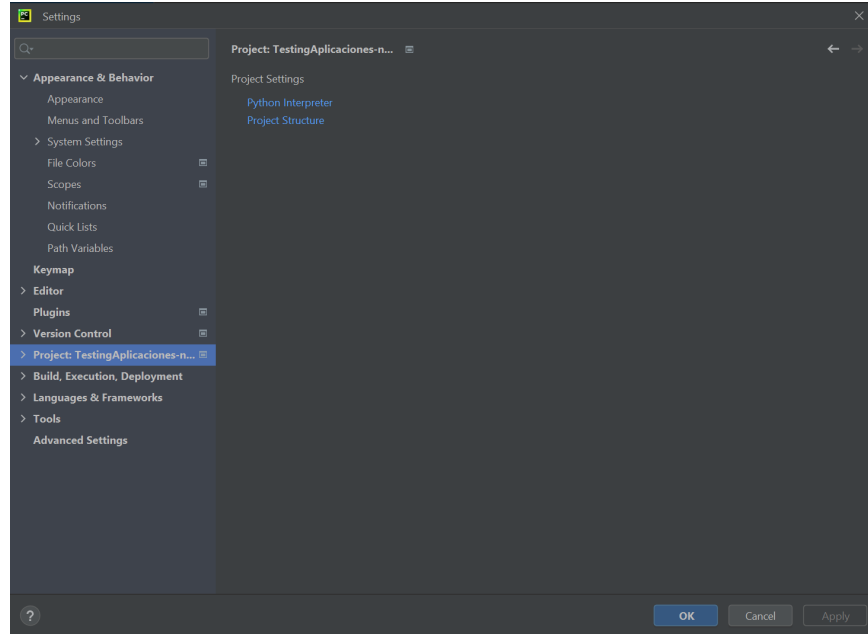
- Crear un nuevo proyecto, que será donde copien los scripts de pytest que exportarán desde Selenium IDE

Trabajo Práctico Obligatorio

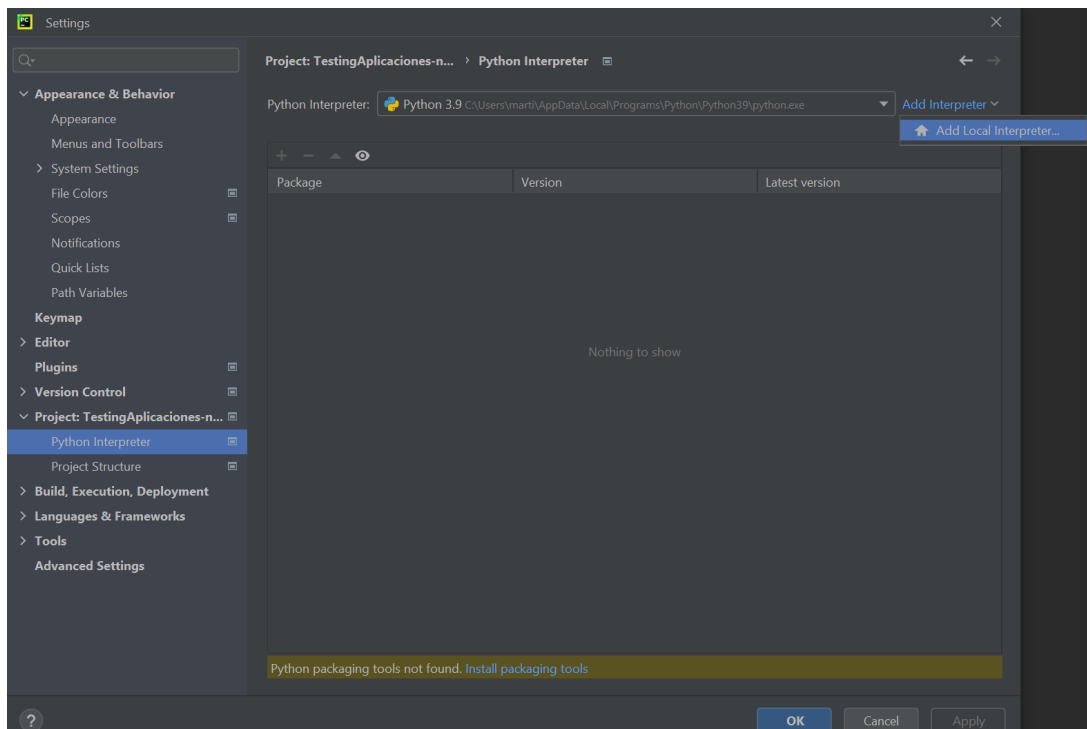
Automatización de Casos de Prueba

Configurar el Intérprete de Python para el proyecto

Una vez creado el Proyecto, seleccionen la opción File > Settings > Project > Project Interpreter. Aquí podrán configurar el intérprete de Python que el proyecto usará.

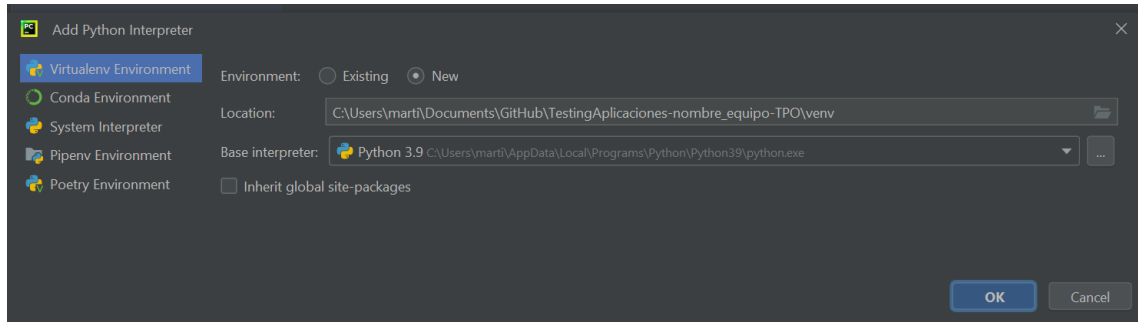


Seleccionen la Opción “Add Interpreter > Add Local Interpreter”



Crear un entorno Virtualenv

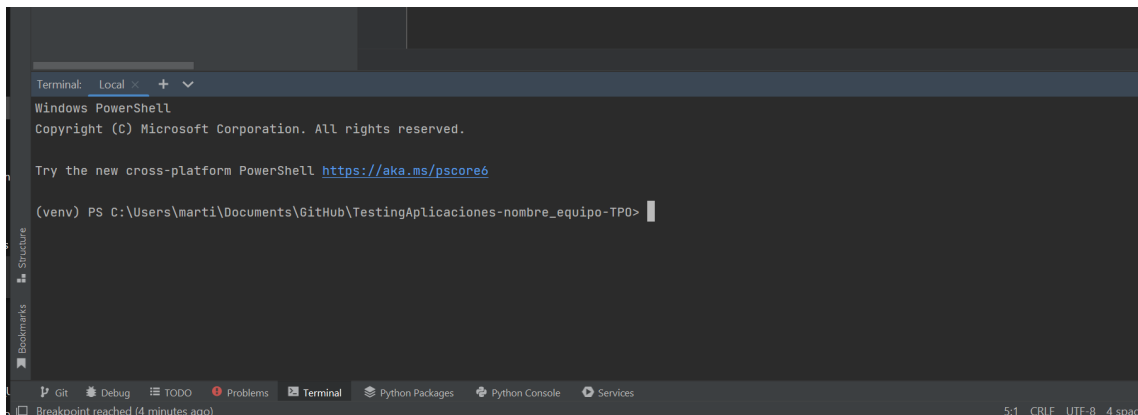
- Revisar la carpeta donde se va a crear (puede ser dentro de la carpeta del repositorio/venv)



Instalar las librerías pytest y selenium

Abrir la terminal de PyCharm e instalar las librerías “pytest” y “selenium”

Importante: Verificar que la línea de ejecución comience con (venv) (esto nos dice que las librerías serán instaladas en la versión de Python que estamos utilizando en el proyecto)



Comandos de Instalación:

- pip install pytest
- pip install selenium

Nota: ¡si tienen problemas de permisos al momento de instalar las librerías, avisen!

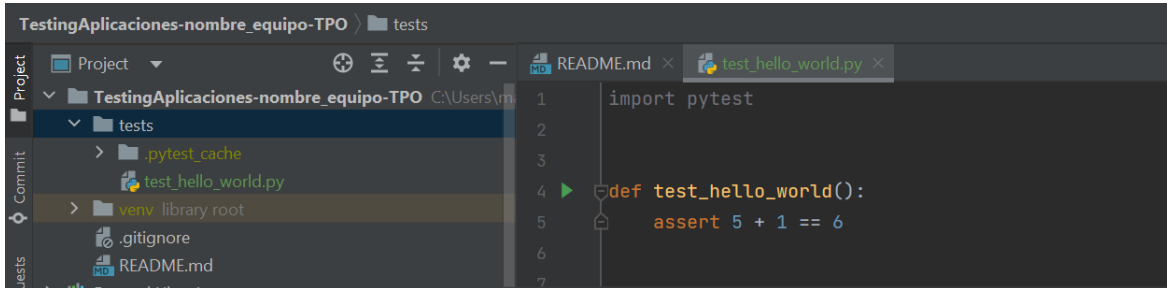
Trabajo Práctico Obligatorio

Automatización de Casos de Prueba

Ejercicio: Hello World en Python con Pytest

A modo de ejemplo, se presenta la creación de un pequeño test de pytest, para verificar que la instalación de Python y las librerías fue exitosa.

- Creen la carpeta “test_pytest” dentro del proyecto y el script de Python “test_hello_world.py”

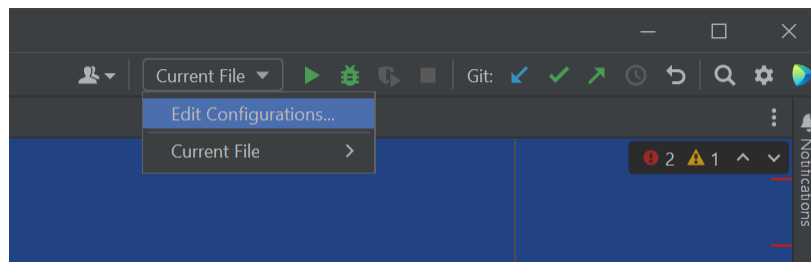


Agreguen al script creado:

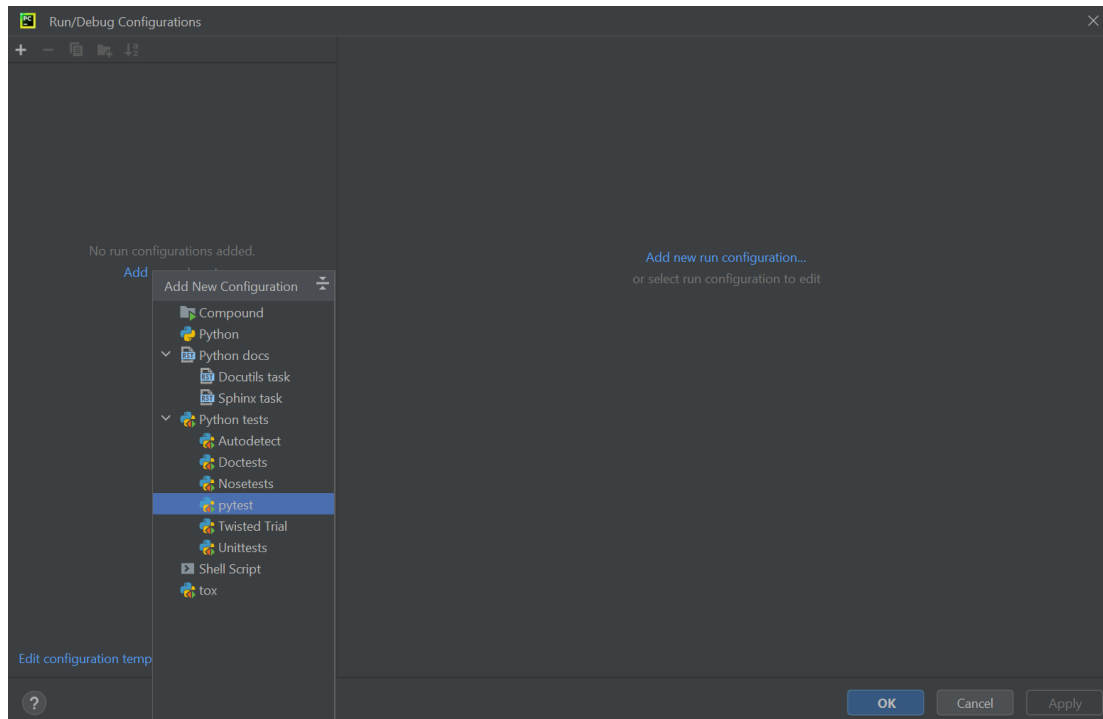
```
import pytest

def test_hello_world():
    assert 5 + 1 == 6
```

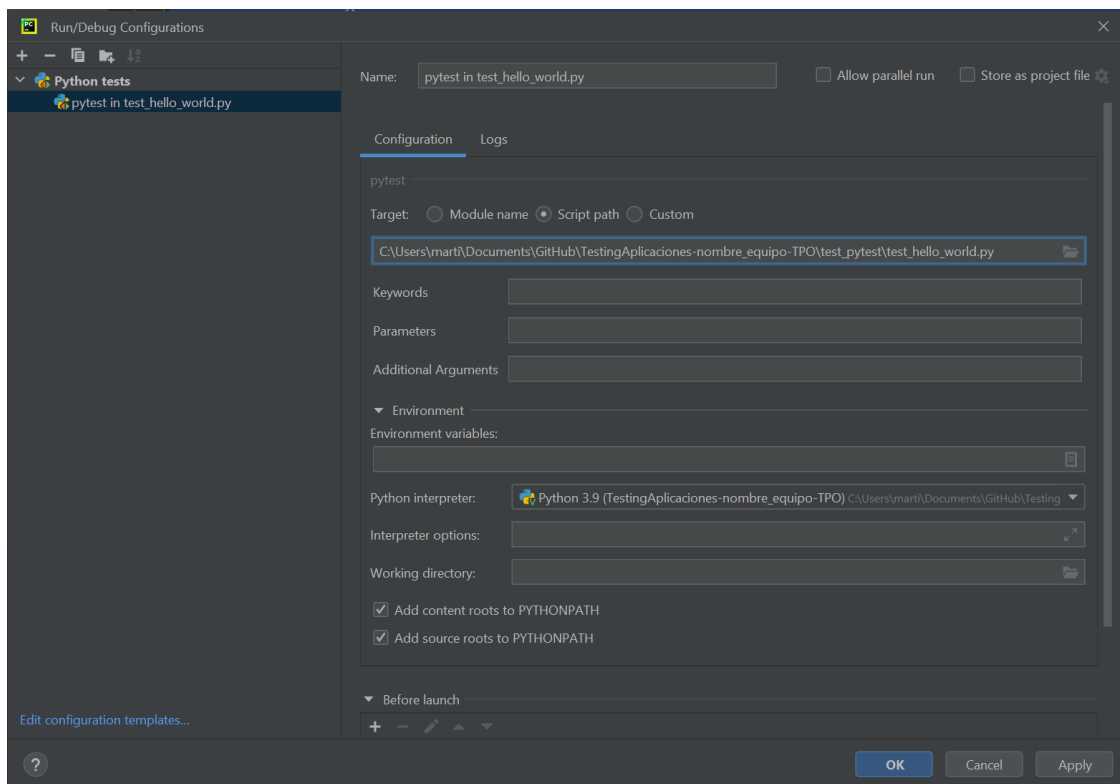
En la parte superior derecha de PyCharm editar la configuración de ejecución



Agregar la configuración para Pytest



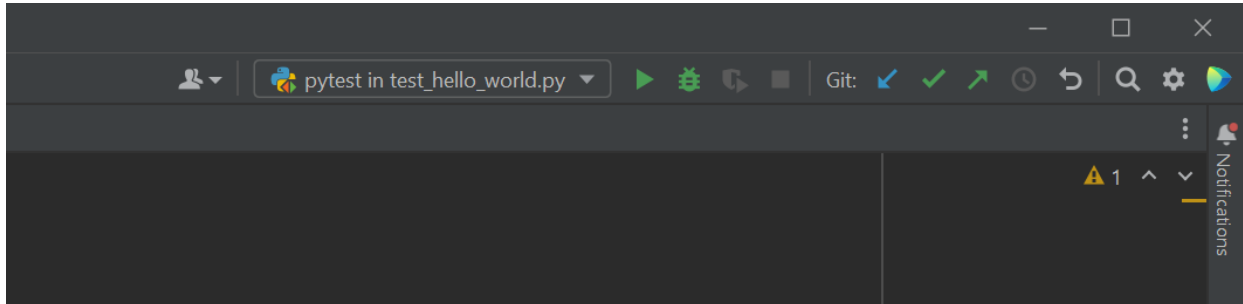
Seleccionar el script de Python y confirmar



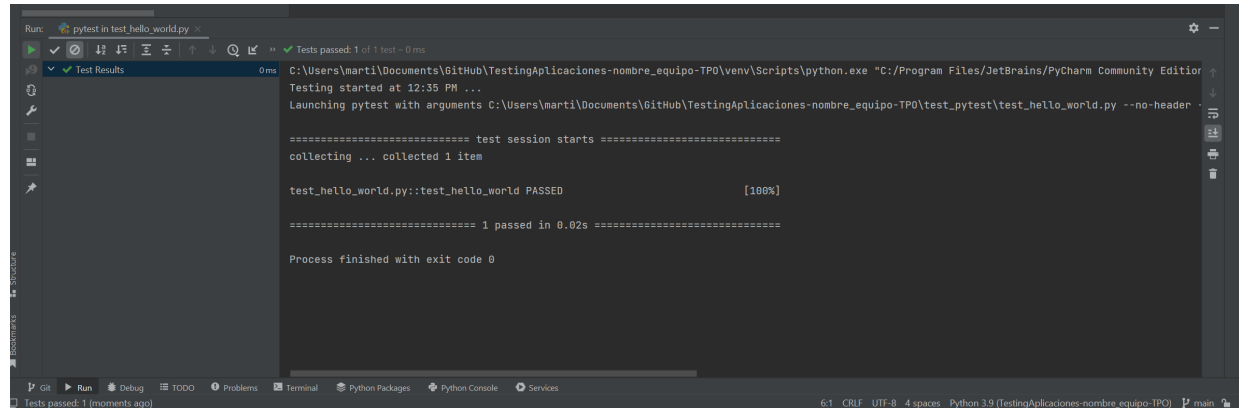
Trabajo Práctico Obligatorio

Automatización de Casos de Prueba

Luego, ejecutar los tests del script con el botón Play



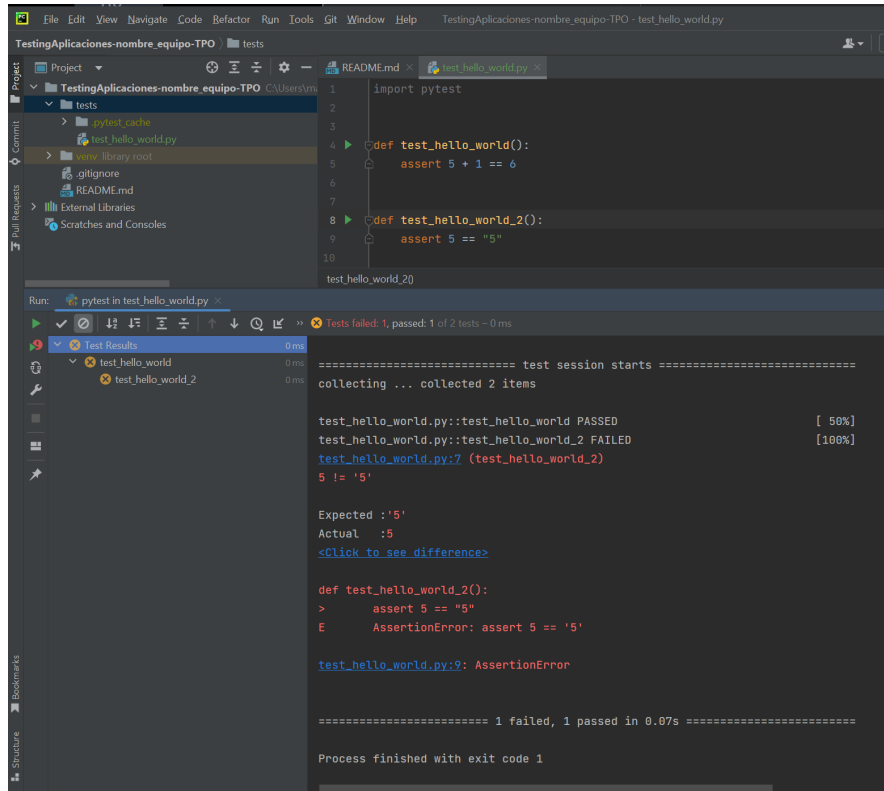
Si todo está bien, verán que la pestaña “Run” de PyCharm les mostrará el test ejecutado y el resultado PASSED



Trabajo Práctico Obligatorio

Automatización de Casos de Prueba

¿Y si agregamos un nuevo test? Supongamos que agregamos un test que FALLA:



```
1 import pytest
2
3
4 def test_hello_world():
5     assert 5 + 1 == 6
6
7
8 def test_hello_world_2():
9     assert 5 == "5"
10
11 test_hello_world_2()
```

Run: pytest in test_hello_world.py

Test Results

- test_hello_world 0 ms
- test_hello_world_2 0 ms

test session starts
collecting ... collected 2 items

test_hello_world.py::test_hello_world PASSED [50%]
test_hello_world.py::test_hello_world_2 FAILED [100%]
test_hello_world.py:7 (test_hello_world_2)
5 != '5'

Expected : '5'
Actual : 5
<Click to see difference>

def test_hello_world_2():
> assert 5 == "5"
E AssertionError: assert 5 == '5'

test_hello_world.py:9: AssertionError

===== 1 failed, 1 passed in 0.07s =====

Process finished with exit code 1

Parte B - Webdriver

Para poder ejecutar scripts que utilicen Selenium en sus PC necesitamos:

- ☐ Un **webdriver** para el explorador web que vamos a utilizar
- ☐ Instalar las **librerías de Selenium**, para el lenguaje de programación que vayamos a utilizar (en nuestro caso será Python)

Selenium Webdriver

Selenium WebDriver es una colección de enlaces que nos permite controlar un navegador de forma nativa, como lo haría un usuario. Es también una interfaz de programación simple y concisa que permite automatizar aplicaciones web para fines de prueba u otros propósitos.

Se mencionan a continuación los pasos para instalar el webdriver de Google Chrome:

- ☐ Verificar qué versión de Google Chrome tenemos en nuestra PC (la versión 118 es la última al momento que esta guía fue escrita)
- ☐ Ir a <https://googlechromelabs.github.io/chrome-for-testing/>
- ☐ Descargar la versión del **chromedriver** correspondiente a nuestro Sistema Operativo. Lo más común es descargar la versión "Stable"
 - Windows: chromedriver_win32
 - Mac OS (Intel): chromedriver_mac-x64
 - Mac OS (Apple Silicon): chromedriver_mac_arm64

Luego, descomprimir el archivo y configurarlo como ejecutable por nuestro sistema operativo.

Tutorial de cómo configurarlo: <https://zw.betz.com/download-chromedriver-binary-and-add-to-your-path-for-automated-functional-testing/>

- ☐ En caso de problemas, consultar con su equipo o con el profesor

Para verificar que el webdriver es reconocido como ejecutable, pueden abrir una terminal y ejecutar:

```
Windows > chromedriver.exe -v
```

```
Mac OS > chromedriver -v
```

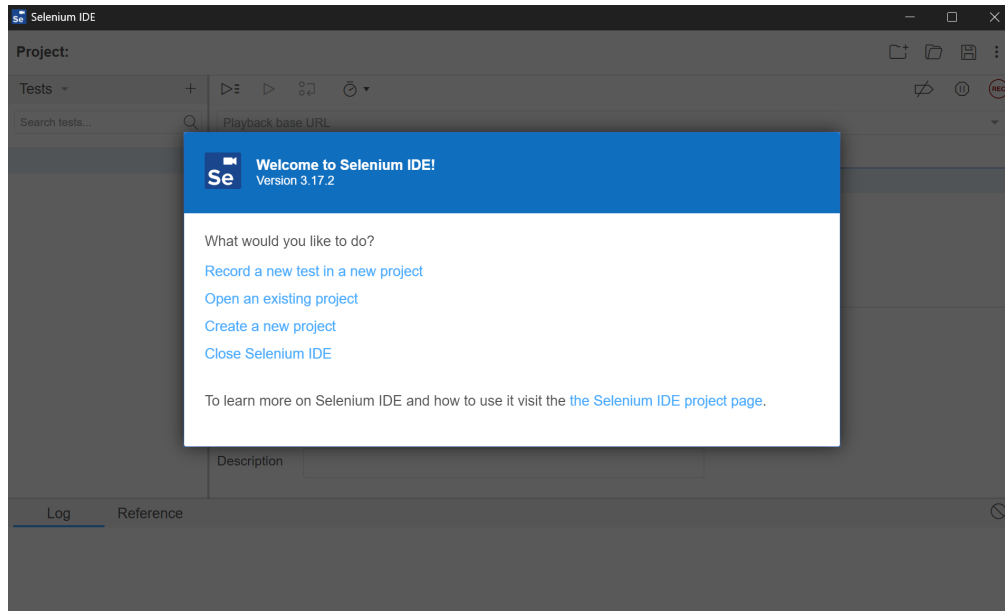
Importante:

- ☐ En Windows, puede ocurrir que el comando mencionado les tire un error (que les diga que "no se reconoce el comando"). Si les pasa esto incluso luego de haber agregado al PATH la carpeta donde guardaron el chromedriver, prueben ejecutando el script automatizado que se encuentra en la sección siguiente. Y si el browser se abre al ejecutarlo, quiere decir que el chromedriver es reconocido exitosamente)

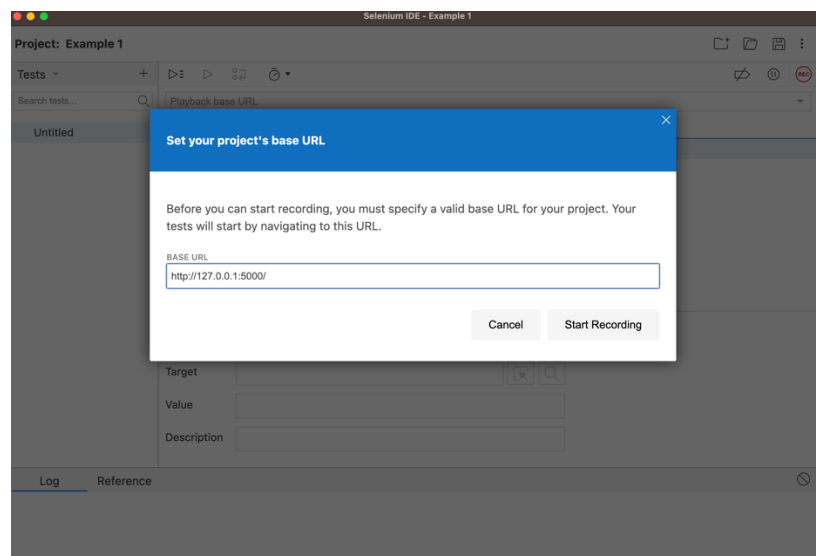
Parte C - Selenium IDE

Abran un explorador web (por ejemplo, Google Chrome o Microsoft Edge)

- ☐ Instalar la extensión “Selenium IDE” para Google Chrome (y otros exploradores basados en Chromium): <https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd?hl=en>
- ☐ Luego, les aparecerá un ícono de “extensiones” a la derecha de la barra de búsqueda del explorador. Al hacerle click se abrirá la ventana de Selenium IDE



- ☐ Seleccionar “Record a new test in a new project”. Escriban un nombre para el proyecto
- ☐ Ingresen la URL de una página web de Prueba: <https://www.wikipedia.org/>, por ejemplo



Trabajo Práctico Obligatorio

Automatización de Casos de Prueba

- ☐ Luego de presionar OK, se abrirá una nueva ventana de Chrome. Y todos los pasos que realicen se grabarán

Exportar un script de prueba

Cuando Selenium IDE esté grabando, realicen distintas acciones. Verán que los pasos que realizan se van agregando a la ventana de Selenium IDE. Algunas opciones que pueden hacer son:

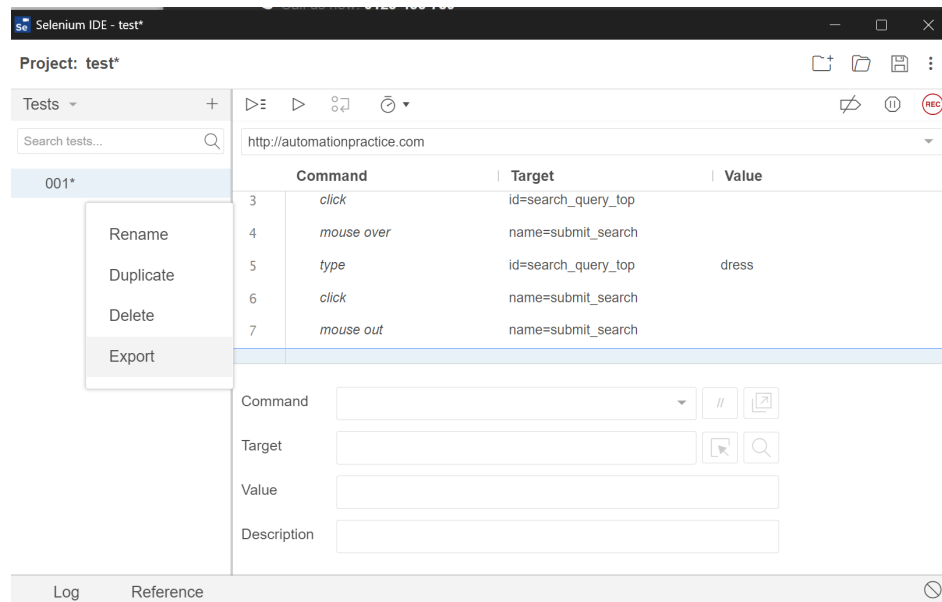
- ☐ Hacer click en menús o links
- ☐ Escribir en campos de texto
- ☐ Hacer clicks en botones, por ejemplo, de búsqueda

Luego de que hayan realizado acciones, ir a la ventana de Selenium IDE y detener la grabación. Guarden la grabación con algún nombre. Pueden reproducir la grabación, y verán que se van a realizar las mismas acciones que hicieron ustedes.

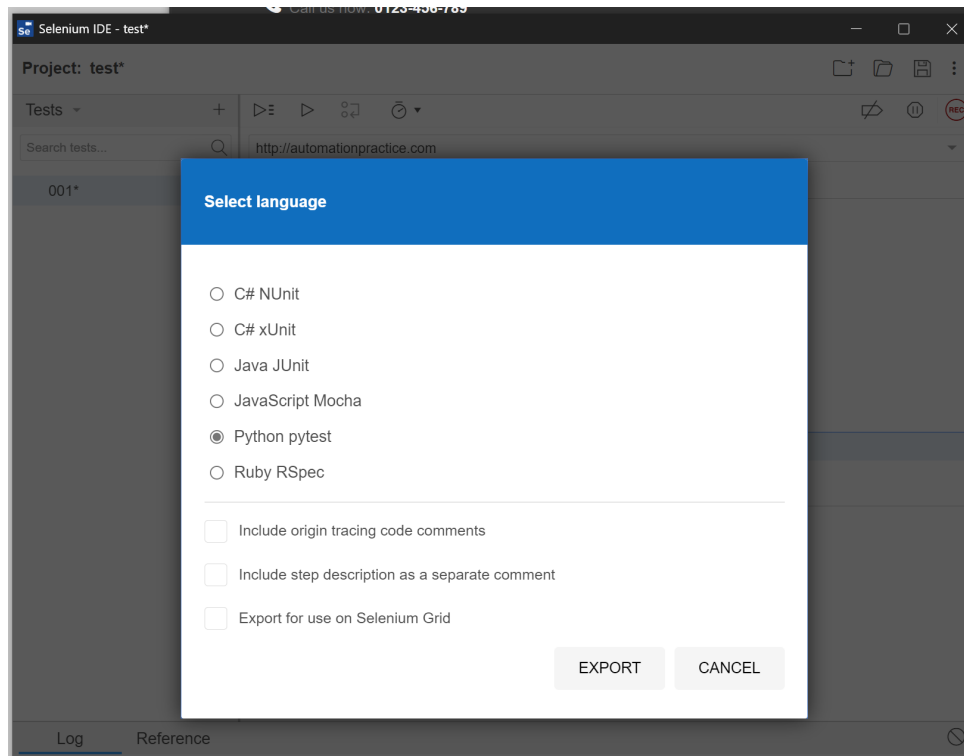
Exportar la grabación a un script de Python

Para exportar la grabación como un script de Python:

- ☐ Verifiquen que la opción “Tests” esté seleccionada, y luego seleccionar la opción “Export” que aparece en el menú de la grabación que acaban de hacer



- Exporten el archivo



Este script lo podrán modificar en el IDE que utilicen.

Ejemplo de un script

Se presenta a continuación un script de la ejecución de la compra de un producto en la página de prueba

<https://www.demoblaze.com/>

```
# Generated by Selenium IDE
import time

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

class Test01():
    def setup_method(self, method):
        self.driver = webdriver.Chrome()
        self.vars = {}

    def teardown_method(self, method):
        self.driver.quit()

    def test_01(self):
        self.driver.implicitly_wait(10)
        self.driver.get("https://www.demoblaze.com/")
        self.driver.find_element(By.LINK_TEXT, "Nokia lumia 1520").click()
        self.driver.find_element(By.LINK_TEXT, "Add to cart").click()
        time.sleep(1)
        assert self.driver.switch_to.alert.text == "Product added"
        # click on the alert's OK button
        self.driver.switch_to.alert.accept()
        self.driver.find_element(By.ID, "cartur").click()
        self.driver.find_element(By.CSS_SELECTOR, ".btn-success").click()
        self.driver.find_element(By.ID, "name").click()
        self.driver.find_element(By.ID, "name").send_keys("name")
        self.driver.find_element(By.ID, "country").send_keys("country")
        self.driver.find_element(By.ID, "city").send_keys("city")
        self.driver.find_element(By.ID, "card").send_keys("123456")
        self.driver.find_element(By.ID, "month").send_keys("03")
        self.driver.find_element(By.ID, "year").send_keys("2024")
        self.driver.find_element(By.CSS_SELECTOR, "#orderModal .btn-
primary").click()
        time.sleep(1)
        self.driver.find_element(By.CSS_SELECTOR, ".confirm").click()
        time.sleep(1)
        self.driver.find_element(By.CSS_SELECTOR, ".active > .nav-link").click()
```

- ☐ Copien este script a un archivo de extensión .py en sus PC
- ☐ Ejecuten este archivo desde su IDE o una terminal
 - Pueden ejecutarlo desde el IDE o desde una terminal, con el comando “`pytest <nombre_del_archivo.py>`”
 - Al ejecutarlo, debería abrirse una ventana de explorador y se debería abrir un explorador web.



Trabajo Práctico Obligatorio Automatización de Casos de Prueba

Parte D – Assertions (Aserciones)

El script de ejemplo sólo realiza acciones sobre un explorador (se selecciona un producto y se realiza una compra), pero no realiza ningún tipo de validación (excepto la validación de que el alert del browser diga 'Product added'). No se valida que el producto seleccionado sea efectivamente el que se está comprando, por ejemplo.

Es por eso que necesitamos darle “inteligencia” a nuestros casos de prueba automatizados, para que “sepan” qué validaciones tienen que hacer para verificar que un caso de prueba es PASS o FAIL y, más importante aún, para que el caso de prueba automatizado sea equivalente al caso de prueba manual. Éste va a ser el objetivo del Trabajo Práctico Obligatorio.

Existe mucha documentación sobre Aserciones, y cómo hacerlas para un caso de prueba automatizado con Selenium. Pueden consultar en internet, e incluso utilizar herramientas como ChatGPT o Google Bard para analizar la mejor forma de utilizar aserciones.