

Trabajo Práctico Obligatorio

Actividades de Testing en un proyecto Agile (Scrum)

Este Trabajo Práctico Obligatorio (TPO) simula las actividades que realiza una persona con el rol de Tester / Ingeniero o Analista de QA / SDET dentro de una Célula Scrum, para un proyecto de una plataforma de E-commerce, realizando actividades para cada uno de las iteraciones o Sprints.

Objetivos Generales del TPO

- Conocer, Entender y Ejecutar algunas de las actividades que los Testers, Ingenieros de QA o SDET realizan en un equipo de ingeniería
- Adquirir conocimientos de cómo se lleva adelante un proyecto Agile
- Utilizar herramientas de desarrollo que suelen utilizarse en proyectos de Software
- Conocer cómo se lleva adelante la gestión de las tareas, requerimientos y defectos encontrados durante el desarrollo de una aplicación de Software

Pre-Entrega

Objetivos

Conformación de los Equipos, Setup de Herramientas y Workflow

Actividades a Realizar

- Organización de los equipos de Calidad
- Instalación de las herramientas necesarias
 - IDE para desarrollo (PyCharm)
 - Herramienta de VCS (GitHub)
- Definición y Priorización de las actividades del equipo
- Testing:
 - Especificación de Casos de Prueba
 - Ejecución de Casos de Prueba
 - Reporte de Defectos
 - Automatización de Casos de Prueba

Tarea 1: Equipo

- Crear equipos de no más de 4 personas
- Definir un nombre para su equipo
 - o No hace falta que sea formal! Siempre es bueno agregar un poco de humor!
 - o Eso sí: que sea SFW (safe for work) :P

Tarea 2: Setup/Instalación de Herramientas

NOTA: Todos los miembros del equipo deben realizarla

Qué es Python?

Python es un lenguaje de programación multiparadigma, ya que soporta la orientación a objetos, programación imperativa y la programación funcional. Es un lenguaje interpretado (que no requiere que se compile a lenguaje máquina antes de ser ejecutado), dinámico y multiplataforma.

Se caracteriza por la fluencia de su sintaxis, la cual garantiza una alta legibilidad y comprensión del código tanto para quien desarrolla como para quien lee. Su estructura fundamental incluye características tales como el manejo adecuado de excepciones y tipos de datos de alto nivel. Existen interfaces para hacer llamadas al sistema y librerías, así como diversos sistemas de ventanas. Nuevos módulos pueden ser fácilmente escritos en C o C++ (o algún otro lenguaje, dependiendo de la implementación seleccionada). Python también puede ser usado como un lenguaje para extender aplicaciones escritas en algún otro lenguaje, que necesite ser usado bajo scripting o automatización de la interfaz.

Instalar Python

- Descargar la versión 3 de Python desde <https://www.python.org/downloads/>
- Luego de descargarlo deberá instalarlo

Verificar que Python está correctamente instalado

Luego de instalado, puede verificar que se instaló correctamente si abre una terminal y escribe el comando 'python' (python3 en macOS)

```
C:\Users\marti>python
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
>>> exit()
C:\Users\marti>_
```

```
franco — -zsh — 100x15
~ — -zsh

Last login: Tue Oct  4 21:42:25 on ttys000
franco@Franco-MBP16 ~ % python3
Python 3.10.4 (v3.10.4:9d38120e33, Mar 23 2022, 17:29:05) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more
>>> exit()
franco@Franco-MBP16 ~ %
```

GitHub

GitHub es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprado por Microsoft [en junio del 2018](#). La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

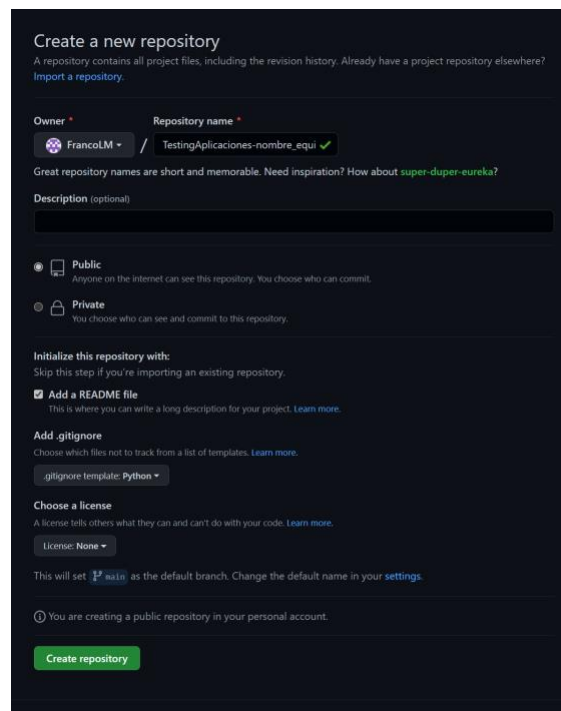
Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por [Linus Torvalds](#). Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se vaya a actualizar.

Así pues, Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs

Crear cuenta en GitHub

Cada miembro del equipo debe crear una cuenta en GitHub: <https://github.com/>

Creación del Repositorio



The screenshot shows the 'Create a new repository' page on GitHub. It includes fields for 'Owner' (FrancoLM) and 'Repository name' (TestingAplicaciones-nombre_equi). There is a 'Description (optional)' text area. Under 'Visibility', 'Public' is selected. The 'Initialize this repository with' section has 'Add a README file' checked. The 'Add .gitignore' section has 'Python' selected. The 'Choose a license' section has 'None' selected. A 'Create repository' button is at the bottom.

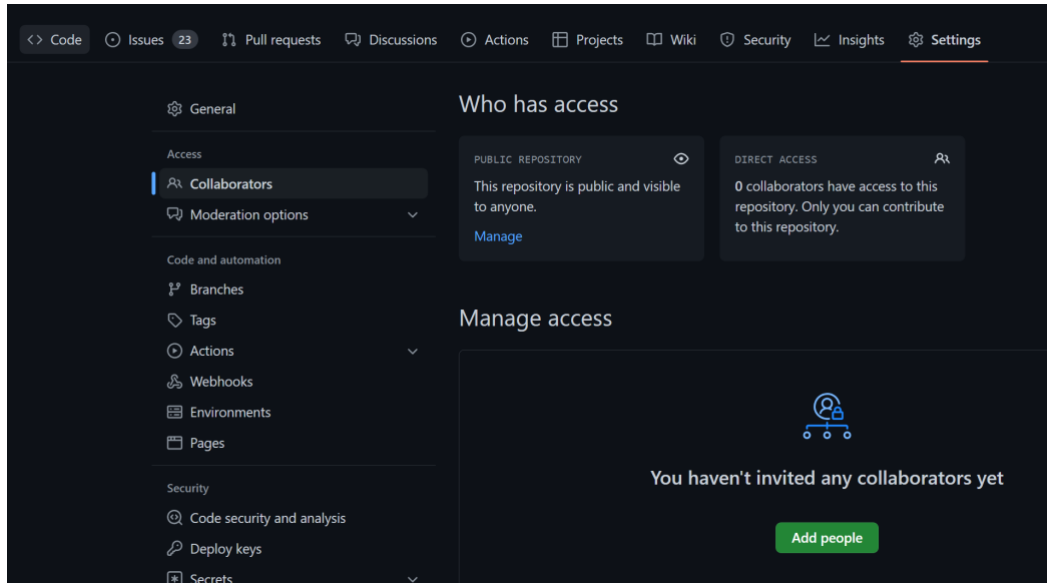
Información del Repositorio

- Nombre: TestingAplicaciones-<nombre_equipo>-TPO
- Público
- Tildar checkbox para incluir README
- .gitignore: Python

Agregar Nombre de Equipo e Integrantes al README del repositorio

Agregar al resto de integrantes del equipo y al profesor como colaborador

- Nombre de usuario de GitHub del profesor: “FrancoLM”

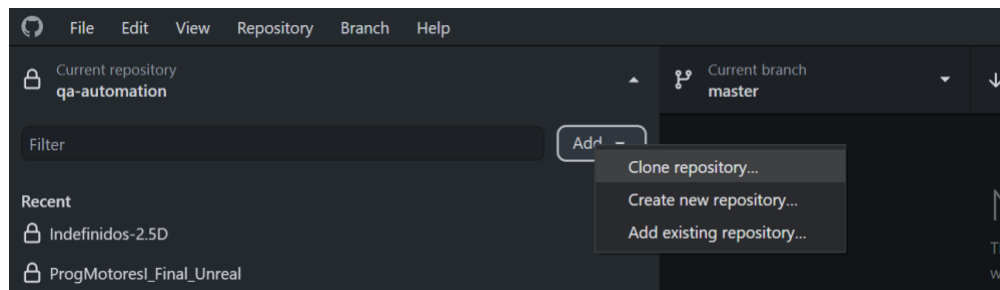


Clonar el Repositorio

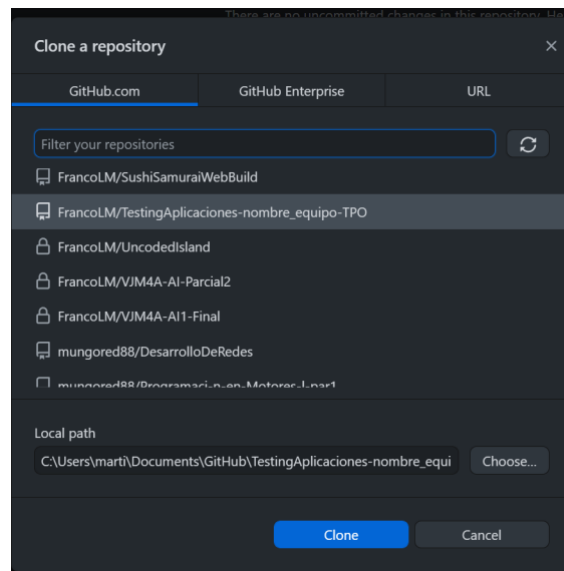
Pueden clonar el repositorio por CLI (línea de comando). Sino, pueden utilizar GitHub Desktop:

<https://desktop.github.com/>

- Instalar GitHub Desktop
- Iniciar sesión con su cuenta
- Seleccionar la opción “Clonar Repositorio”



- Seleccionar el repositorio, seleccionar una carpeta donde clonarlo y confirmar



Pycharm IDE

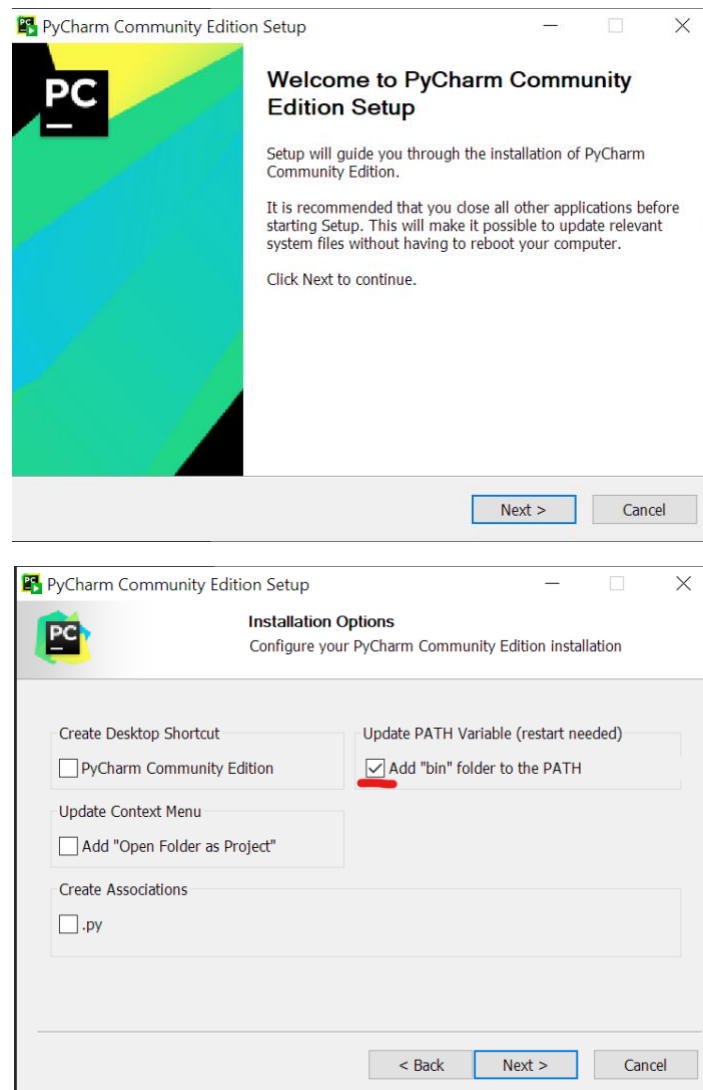
PyCharm es un entorno de desarrollo integrado (IDE) utilizado en [programación informática](#), concretamente para el lenguaje de programación [Python](#). Está desarrollado por la empresa [JetBrains](#) (antes conocida como IntelliJ). Proporciona análisis de código, un depurador gráfico, un probador 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 comunitaria) 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

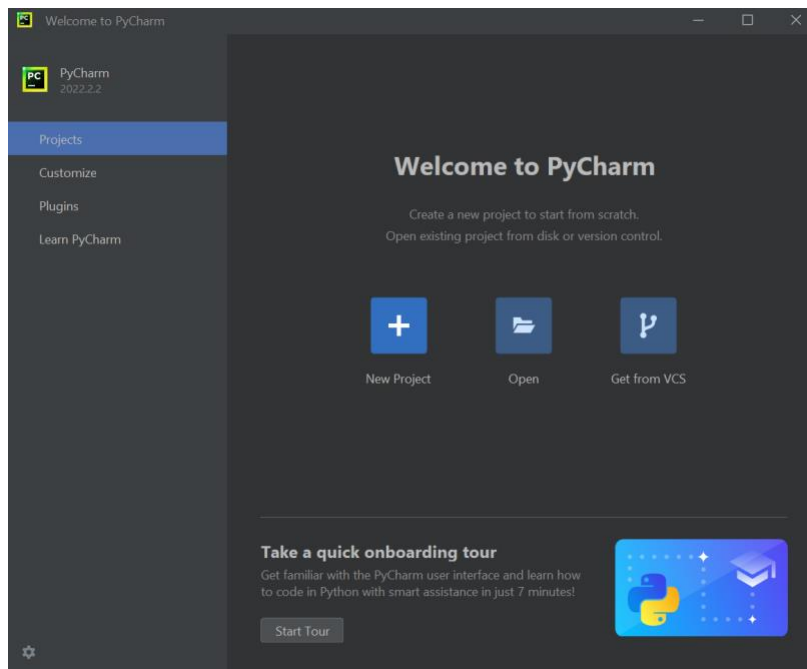
1. Descargar e Instalar Pycharm IDE – Community Edition

<https://www.jetbrains.com/pycharm/>



Configuración de Pycharm

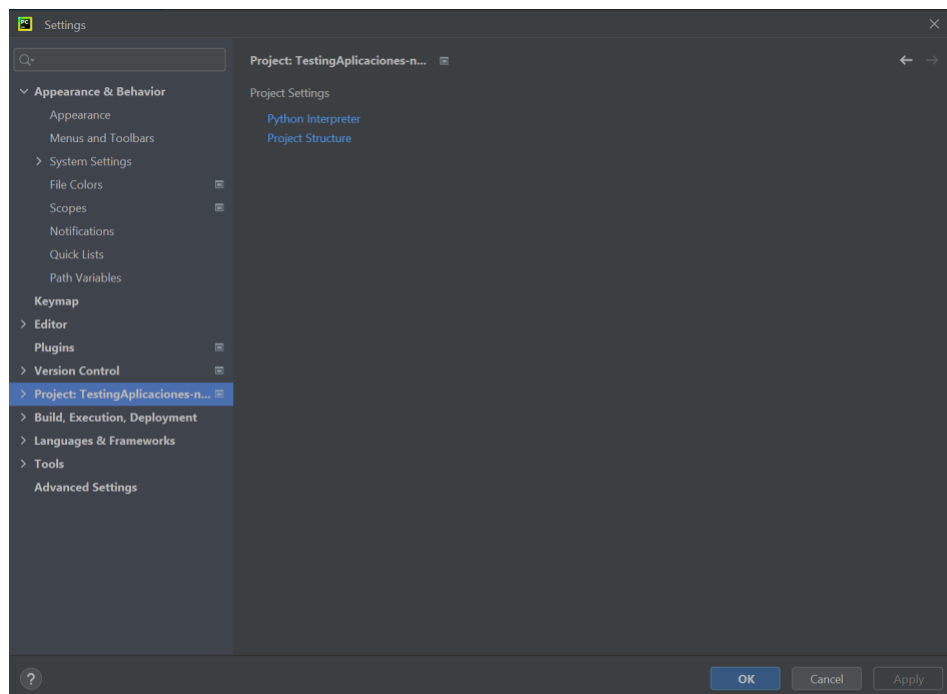
- Abrir PyCharm Community Edition
- Seleccionar “Abrir” (Open)



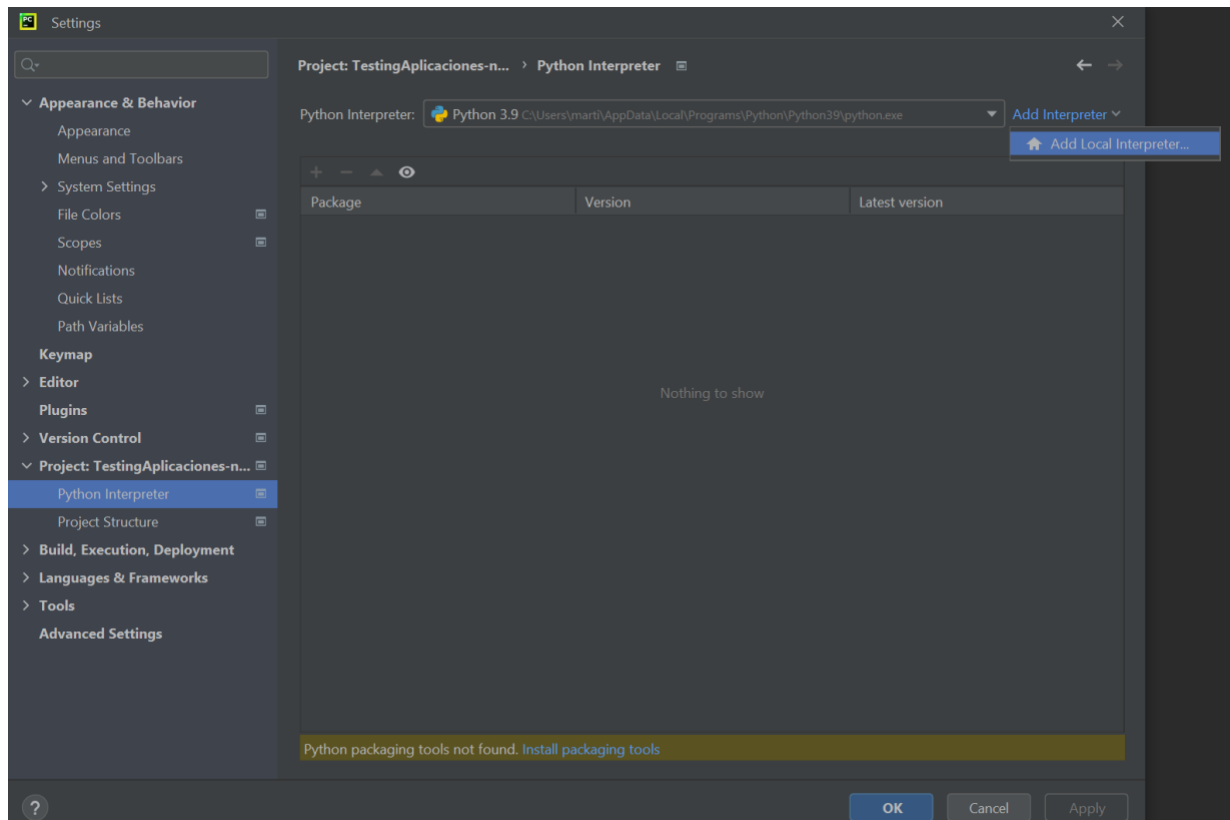
- Buscar la carpeta con el repositorio clonado y Confirmar

Configurar el Intérprete de Python para el proyecto

File > Settings > Project > Project Interpreter

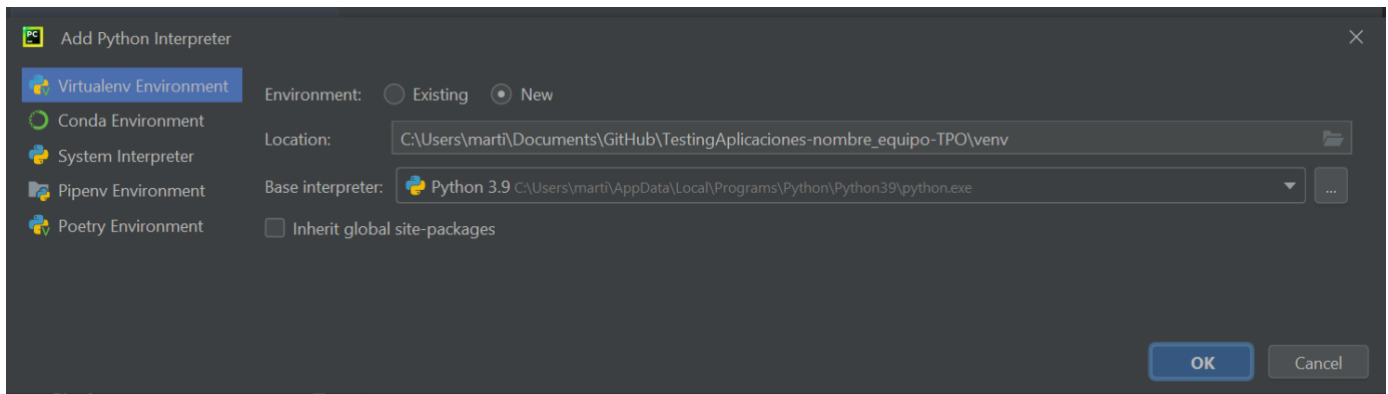


Seleccionar la Opción “Add Interpreter > Add Local Interpreter”



Crear un entorno Virtualenv

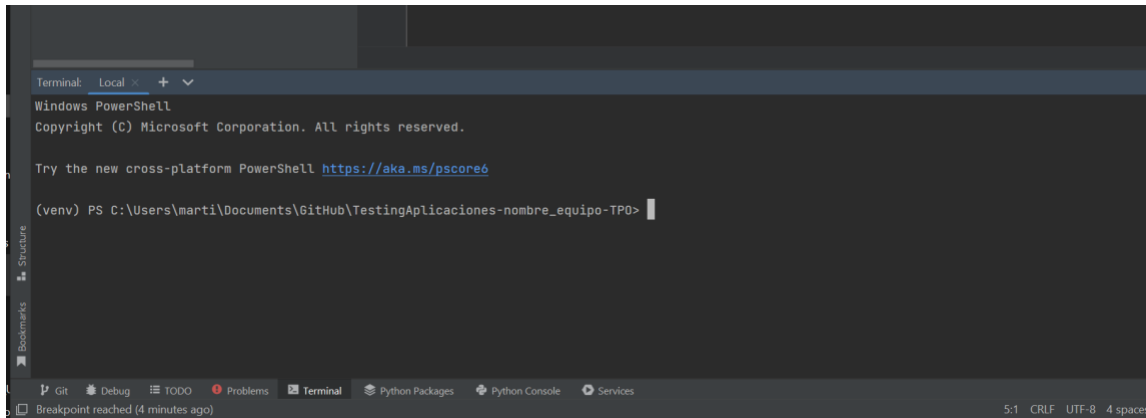
- Revisar la carpeta donde se va a crear (puede ser dentro de la carpeta del repositorio/venv)



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!

Modificar el Archivo README

Uno de los integrantes del equipo deberá modificar el archivo README.md con la siguiente información a completar:

```
# Testing de Aplicaciones: Trabajo Práctico Obligatorio

## Nombre del equipo
- 'nombre del equipo'

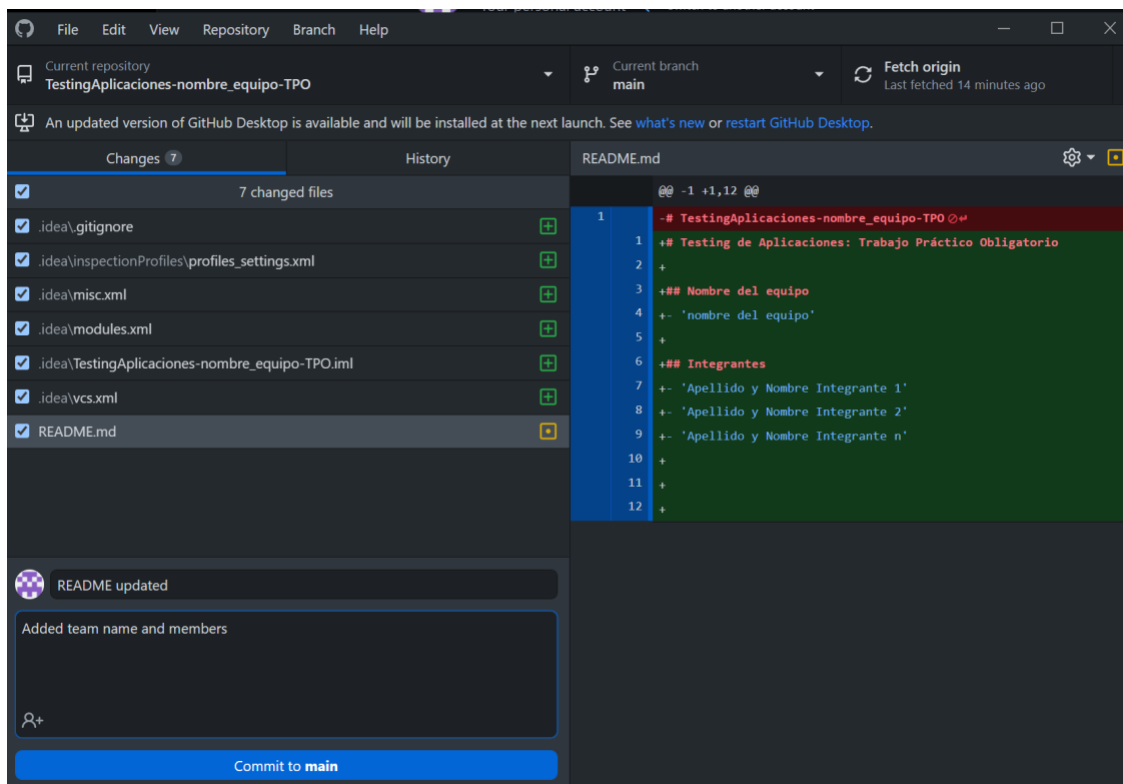
## Integrantes
- 'Apellido y Nombre Integrante 1'
- 'Apellido y Nombre Integrante 2'
- 'Apellido y Nombre Integrante n'
```



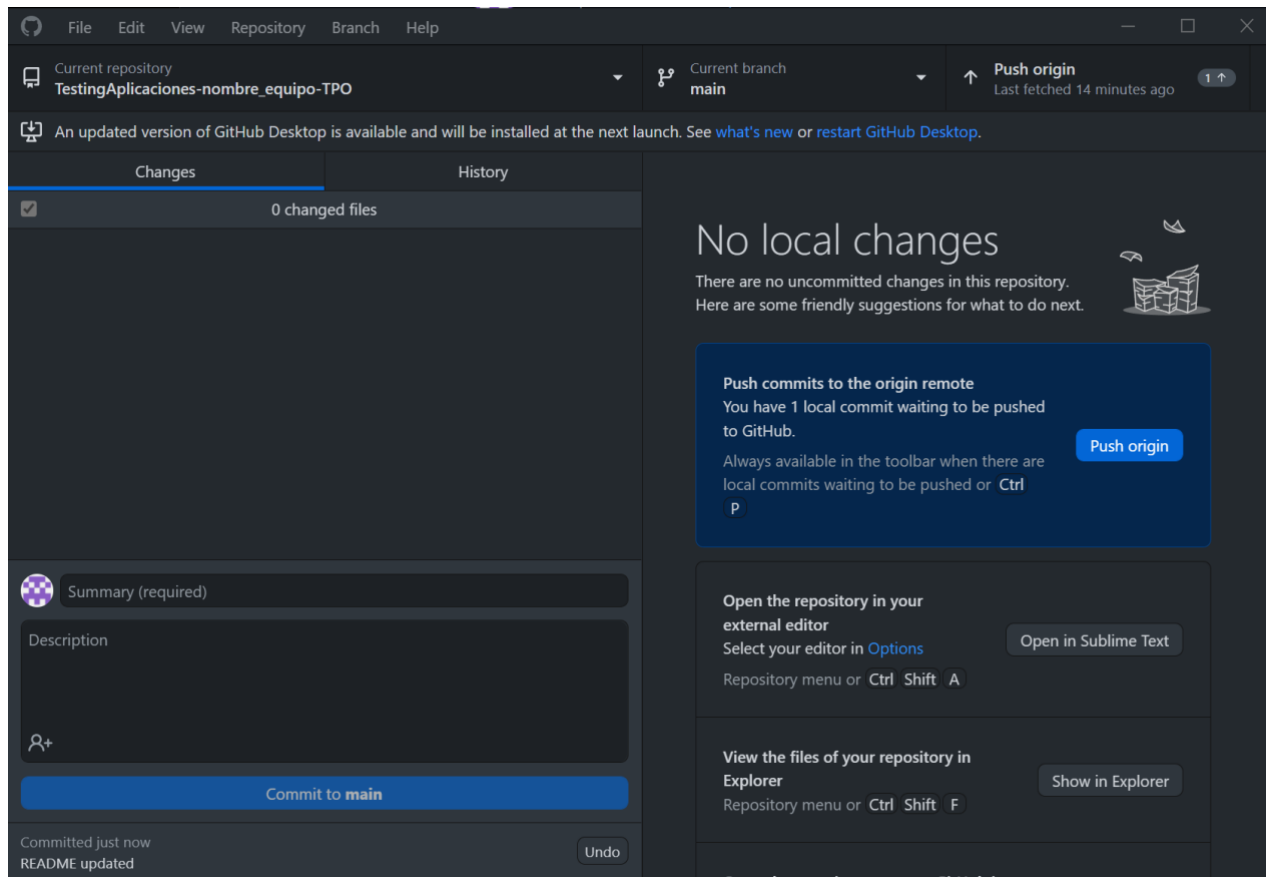

Subir los cambios al Repositorio

Ir a GitHub Desktop y revisar los cambios en el repositorio

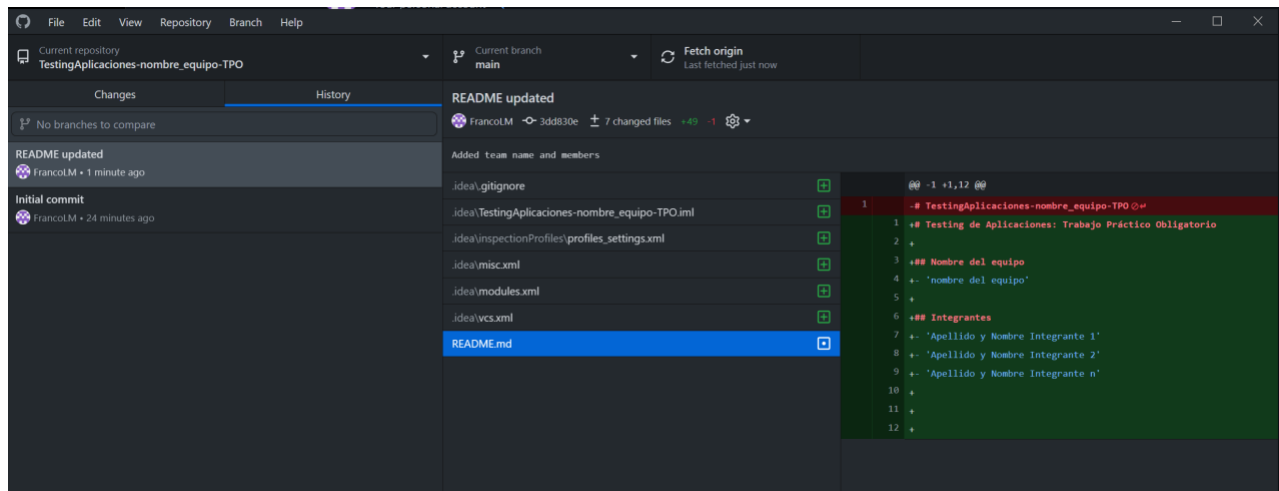
Agregar un “Summary” o resumen y presionar “Commit to main”



La opción “Fetch origin” cambia a “Push origin”. Hacer click para “subir los cambios” a GitHub

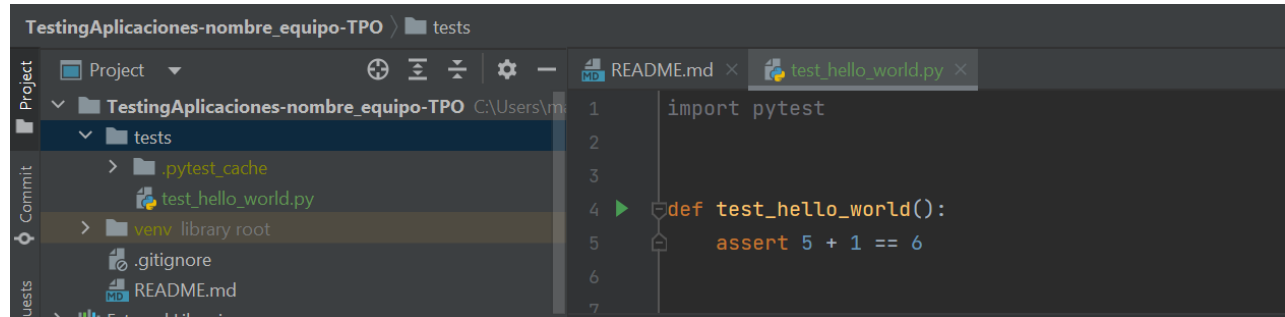


Nota: fíjense que en la parte de “History” verán el nuevo commit



Ejercicio: Hello World en Python con Pytest

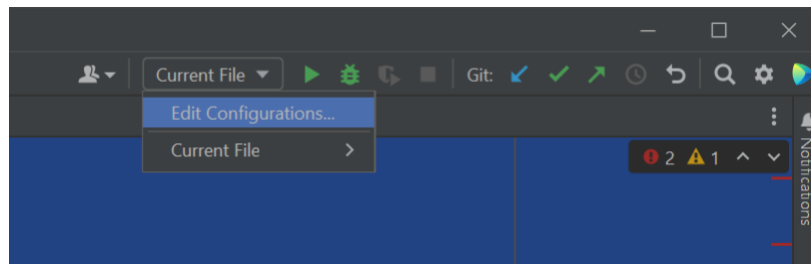
Crear la carpeta “test_pytest” dentro del proyecto y el script de Python “test_hello_world.py”



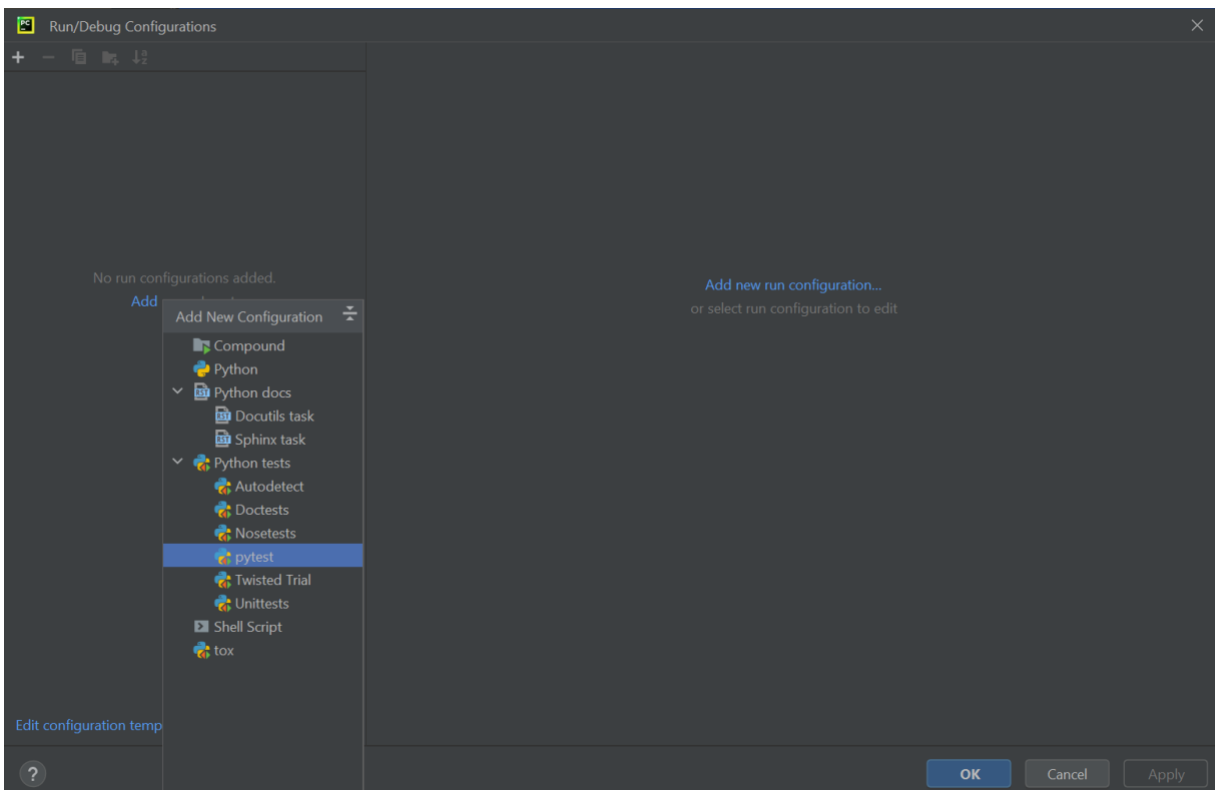
```
import pytest

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

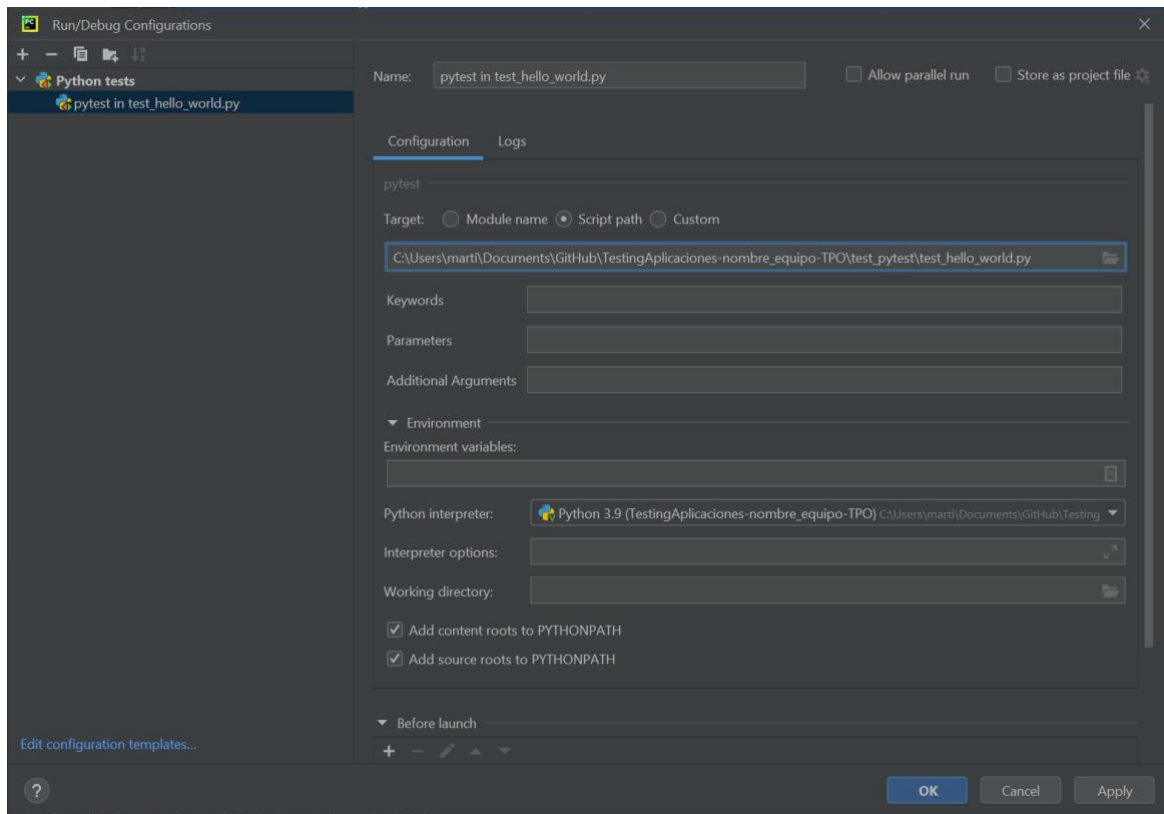
En la parte superior derecha de PyCharm editar la configuración de ejecución



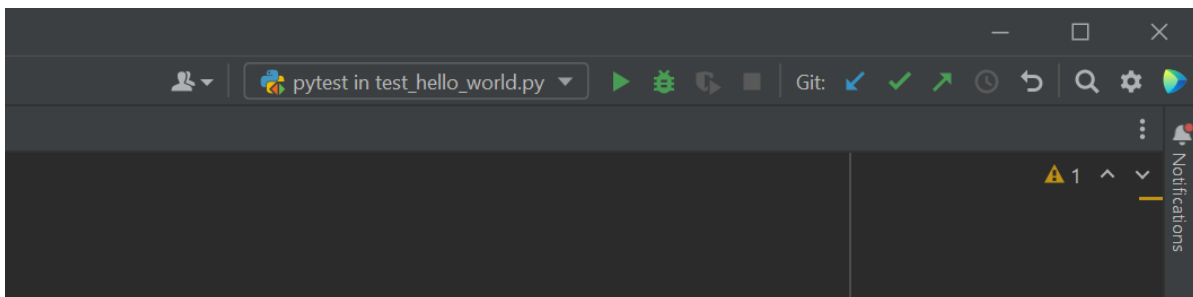
Agregar la configuración para Pytest



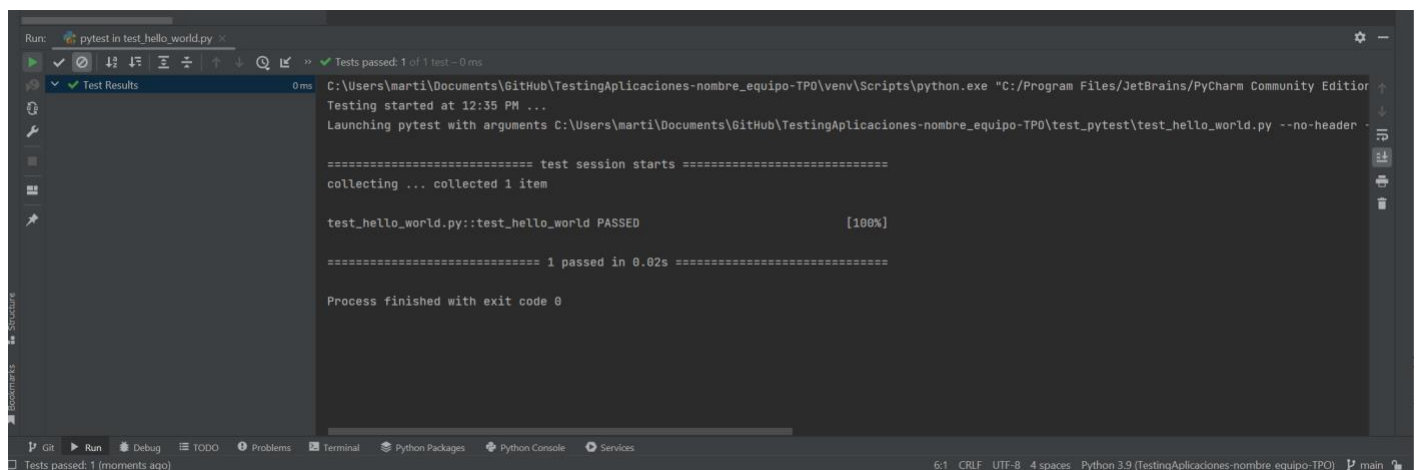
Seleccionar el script de Python y confirmar



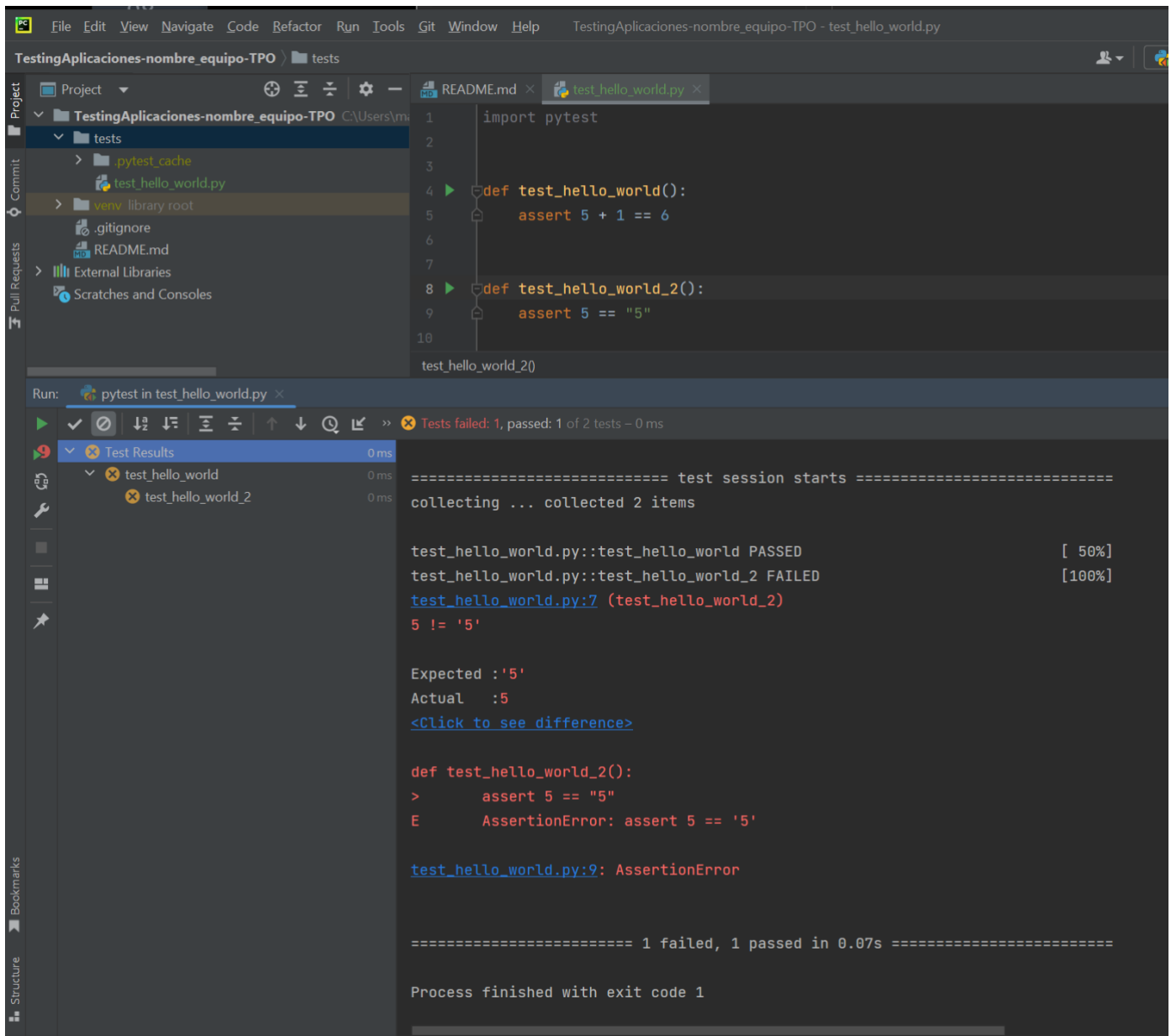
Luego, ejecutar los tests del script con el botón Play



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



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



Criterio de Aceptación de la Entrega

1. TODOS los integrantes del equipo tienen **instalado** PyCharm Community Edition
2. TODOS los integrantes del equipo tienen el **repositorio clonado**
3. El profesor y TODOS los integrantes del equipo son **colaboradores** en el proyecto de GitHub
4. TODOS los integrantes del equipo han podido **ejecutar el script de prueba** exitosamente desde PyCharm