PROGRAMACIÓN ORIENTADA A PRUEBAS

Revisado por: Ing. Miguel Cardona.

Aarón Hernández García.

Instalación de Pytest en un Entorno Virtual

14/03/2025

¿Qué es Pytest?

Pytest es un framework de pruebas para Python que facilita la escritura y ejecución de pruebas automatizadas. Se destaca por su sintaxis simple, su capacidad para detectar y ejecutar pruebas automáticamente, y su compatibilidad con otros frameworks de prueba. Además, permite parametrización, ejecución paralela y la integración con herramientas como Selenium y Django, lo que lo hace ideal para pruebas unitarias, de integración y funcionales.

Importancia de las Pruebas Automatizadas en Desarrollo Seguro

Las pruebas automatizadas son esenciales en el desarrollo seguro de software porque ayudan a:

- **Detectar vulnerabilidades temprano:** Evitan que errores de seguridad lleguen a producción.
- **Garantizar integridad del código:** Previenen regresiones al validar cambios automáticamente.
- **Mejorar la eficiencia:** Reducen el tiempo y esfuerzo en pruebas repetitivas.
- **Asegurar cumplimiento de estándares:** Permiten validar reglas de seguridad y buenas prácticas continuamente.

Objetivo de la Actividad

El objetivo de esta actividad es instalar Pytest en un entorno virtual de Python, documentar el proceso paso a paso y generar un archivo requirements.txt para gestionar las dependencias. Además, se incluirán capturas de pantalla del proceso como evidencia.

1. Crear un Entorno Virtual

Antes de instalar Pytest, es recomendable crear un entorno virtual para aislar las dependencias del proyecto. Para ello, ejecuta los siguientes comandos:

```
C:\>python -m venv test_venv

C:\>cd test_venv
```

```
C:\test_venv>Scripts\activate

(test_venv) C:\test_venv>
```

2. Instalar Pytest

Una vez activado el entorno virtual, instala Pytest con el siguiente comando:

```
(test_venv) C:\test_venv>pip install pytest
Collecting pytest
  Downloading pytest-8.3.5-py3-none-any.whl.metadata (7.6 kB)
Collecting colorama (from pytest)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting iniconfig (from pytest)
  Downloading iniconfig-2.0.0-py3-none-any.whl.metadata (2.6 kB)
Collecting packaging (from pytest)
  Downloading packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
Collecting pluggy<2,>=1.5 (from pytest)
  Downloading pluggy-1.5.0-py3-none-any.whl.metadata (4.8 kB)
Downloading pytest-8.3.5-py3-none-any.whl (343 kB)
   ---------------------------------------- 343.6/343.6 kB 5.4 MB/s eta 0:00:00
Downloading pluggy-1.5.0-py3-none-any.whl (20 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Downloading iniconfig-2.0.0-py3-none-any.whl (5.9 kB)
Downloading packaging-24.2-py3-none-any.whl (65 kB)
   ---------------------------------------- 65.5/65.5 kB 3.5 MB/s eta 0:00:00
Installing collected packages: pluggy, packaging, iniconfig, colorama, pytest
Successfully installed colorama-0.4.6 iniconfig-2.0.0 packaging-24.2 pluggy-1.5.0 pytest-8.3.5

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Para verificar que Pytest se instaló correctamente, ejecuta:

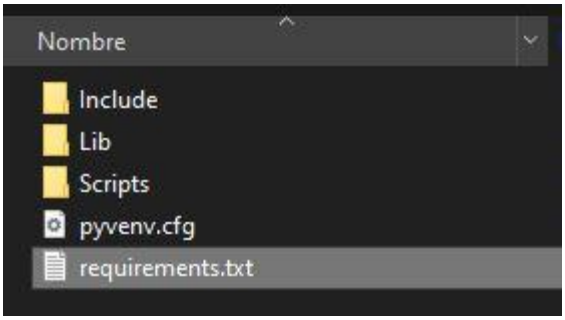
```
(test_venv) C:\test_venv>pytest --version
pytest 8.3.5
```

3. Crear un Archivo requirements.txt

El archivo requirements.txt permite registrar las dependencias del proyecto. Para generarlo automáticamente, ejecuta:

```
(test_venv) C:\test_venv>cat requirements.txt
```

Puedes verificar que se haya creado en tu carpeta test_venv:



5. Importancia de Documentar Procesos Técnicos

Documentar procesos técnicos es esencial para garantizar la claridad, la replicabilidad y la eficiencia en el desarrollo de software. Algunas razones clave incluyen:

- **Facilita la reproducción de procesos:** Permite que otros desarrolladores sigan los pasos sin necesidad de interpretación adicional.
- **Mejora la colaboración:** En proyectos grupales, ayuda a que todos los miembros comprendan los procedimientos y estándares establecidos.
- **Optimiza la resolución de problemas:** Un buen registro de procesos facilita la identificación de errores y la aplicación de soluciones.
- **Favorece la continuidad del proyecto:** Si un desarrollador se ausenta, su trabajo puede ser comprendido y continuado sin contratiempos.

En proyectos colaborativos, la documentación técnica asegura que el conocimiento sea accesible para todos, promoviendo un trabajo más estructurado y eficiente.

Referencias

- VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media.
- Hunt, A., & Thomas, D. (1999). *The Pragmatic Programmer: Your Journey to Mastery*. Addison-Wesley.

1

[1] Elaborado por: Aarón Hernández García. | Revisado por: Ing. Miguel Cardona