

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 TDD (Test Driven Development)

Alonso Álvarez García
Rafael Socas Gutiérrez

Curso 2023/24



Datos de los alumnos

| # | Nombre y apellidos | Curso |
|---|--------------------|-------|
| 1 | Pedro Morales | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

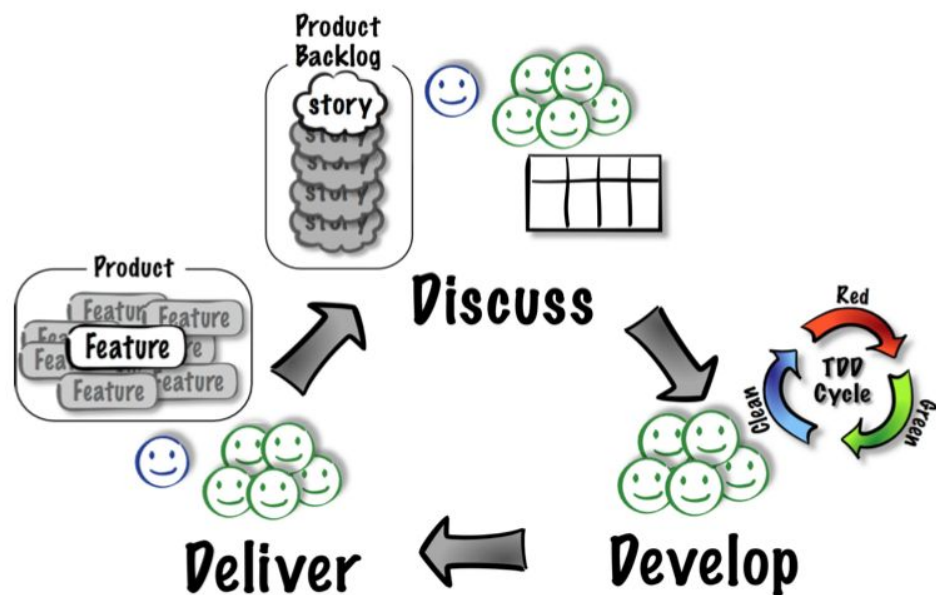
Instrucciones

- 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 **radar_meteorologico.py** y **test_radar_meteorologico.py**, incluyendo las modificaciones realizadas en la práctica.
- 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 29 abril 2024.**

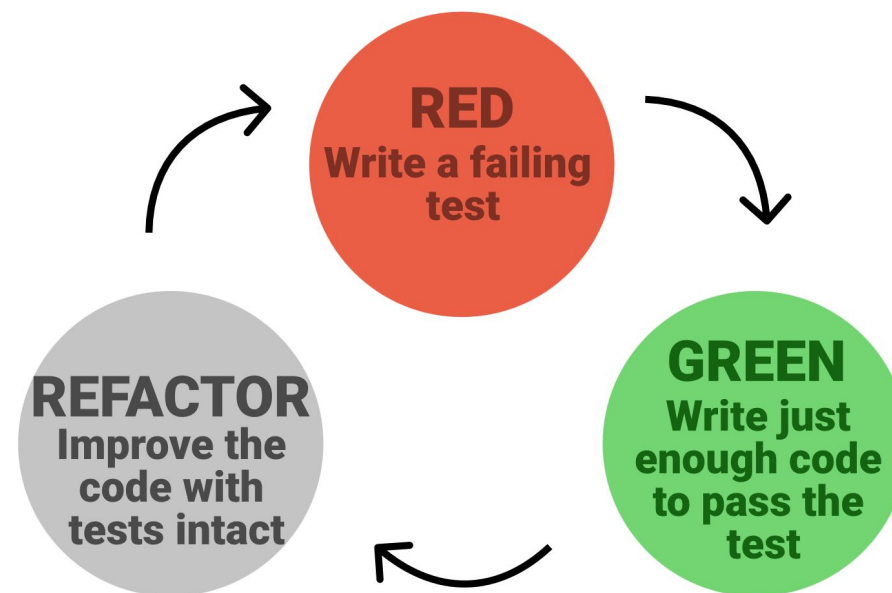
2
Puntos

Contexto

The ATDD cycle



Test Driven Development (TDD)



Test Unitarios

© Image copyright Elisabeth Hendrickson

Objetivos

- Ejecutar casos de test mediante las técnicas TDD.
- Implementar TDD para testear código Python.
- Ejecutar casos de test con los *framework* `unittest` y `nose`.
- Mejorar la calidad del análisis de los test con las funcionalidades `pinocchio` y `coverage`.
- Crear código Python lo más robusto posible gracias a las técnicas tras aplicar la filosofía TDD.



Tarea 1: Preparación del entorno

1
Punto

- 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).
- Instalar el runner nose que mejora la interfaz con el usuario al realizar los test
> `pip install pynose`
- Incluir el módulo pinocchio permite colorear el resultado de los test y así ayudar al diagnóstico
> `pip install pinocchio`
- Por último, instalar módulo coverage que nos ayudará a conocer el número de líneas que se testean
> `pip install coverage`

Código a testear y robustecer

Se pretende testear y robustecer el código Python de un módulo unitario de la aviónica de un Airbus A350. Este módulo, tiene la función de calcular el alcance del radar meteorológico del avión. Con la información obtenida, se presentan la información a escala con la ruta de vuelo en la cabina del avión. El alcance de del radar meteorológico se calcula de la siguiente forma:

$$\text{Alcance} = Co * (T - \tau) / 2$$

Donde:

- *Alcance*: Alcance del radar meteorológica en [km]
- *Co*: velocidad de la luz $3 \cdot 10^5$ [km/s].
- *T*: intervalo de repetición de pulsos [s]. Su rango va de 0 a 0.7 s.
- *tau*: ancho del pulso [μ s]. Su rango va de 0 a 4 μ s.
- IMP: *T* siempre tiene que ser mayor que *tau*.

Siguiendo la filosofía TDD se parte de un código extremadamente simple que no cumpla casi ningún test, solo lo esencial. El fichero se aporta con el enunciado de la práctica

radar_meteorologico.py

```
def alcance_del_radar(T: float, tau: float) -> float:
    """Calcula el alcance del radar meteorológico"""
    """ entrada: T, intervalo de repetición de pulsos [segundos]"""
    """ Entrada: tau, ancho del pulso [microsegundos]"""
    """ Salida: Alcance del radar meteorológico [kilómetros]"""

    #se ponen todas las unidades en segundos, solo en necesario tau,
    #T ya lo está
    tau=tau/pow(10,6)

    Co=3*pow(10,5) #Velocidad de la luz, 300.000 km/s
    return Co*(T-tau)/2 # Calculo del alcance de radar
```

Tarea 2: Test a realizar

1
Punto

Recuerde que la filosofía de TDD se basa en desarrollar el código en base a que superen los test propuestos. En esa línea, cuantos más números y más variados sean los test, más robusto será el código resultante.

Aquí se aportan dos test (1 y 4), se proponen 18 (16 por definir) y 6 categorías de test. Complete esta tabla, e incluso si amplía en el número de test y categorías propuestas será valorado.

| | | Rangos | | |
|-------------|--------------|---------------------|--------------|----------------------------------|
| | | [0-0.7] | [0-4] | |
| Número test | T (segundos) | tau (microsegundos) | Alcance (km) | Categoría del test |
| 1 | 0.5 | 2 | 74999.700 | Valores válidos |
| 2 | | | | |
| 3 | | | | |
| 4 | 0.2 | 5 | No valida | Valores positivos fuera de rango |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | Valores negativos |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | T menor que tau |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | Entrada de strings |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | Entrada de booleanos |
| 17 | | | | |
| 18 | | | | |

Tareas 3: Ejecución de los test, análisis de resultados y aplicación filosofía TDD

6
Puntos

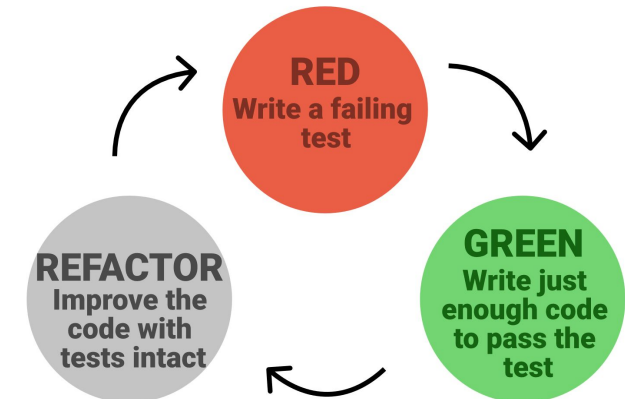
Una vez, se tiene un código base muy sencillo y un conjunto de test, ahora los pasos con técnicas TDD son:

Con un fichero de test, **test_radar_meteorologico.py** (que se proporciona y ya tiene implementado el test 1 y 4) ejecute los test para ver los resultados obtenidos.

1. Los test se ejecutan con los siguientes comandos Python:

```
> python -m nose -v test_radar_meteorologico.py --with-spec --spec-color --with-coverage  
> coverage report -m
```
2. Programe los test en el fichero **test_radar_meteorologico.py**, para ello apóyese en los métodos **assert** de **unittest** (Python unittest Assert Methods) disponibles en <https://www.pythontutorial.net/python-unit-testing/python-unittest-assert/>
3. Modifique el código de **radar_meteorologico.py** para conseguir que se ejecuten todos los test de manera correcta y que todas las líneas de código sean analizadas (comando **coverage report -m**).
4. Vuelva al 1 hasta que un 100% de test superados en el punto 2.

Test Driven Development (TDD)



Test Unitarios



 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

 [SOLICITA MÁS INFORMACIÓN](#)



CENTRO ADSCRITO A:



PROYECTO COFINANCIADO POR:

