Doctest

Antonio Espín Herranz

Pruebas

- Dentro del mundo python las pruebas unitarias se pueden realizar con **unittest**, a menudo combinado con **doctest** para pruebas más sencillas.

- Los paquetes **unittest** y **doctest** están incluidos en la librería estándar de Python.

doctest

- **doctest** permite integrar las pruebas con la documentación.

- Esto permite tener ejemplos de código en la propia documentación y a la vez probar el código sin escribir otro código aparte.

- **Como funciona:**
  - Cuando doctest encuentra una **línea** en la **documentación** que **comienza con '>>>'** se **asume que** lo que le sigue **es código** Python a ejecutar,
  - La respuesta esperada se encuentra en la línea o líneas siguientes, sin >>>.
  - El texto de la prueba termina cuando se encuentra una línea en blanco.

Ejemplo

- Partiendo de la función:

```
def cuadrados(lista):
    """Calcula el cuadrado de los números de una lista"""
return [n ** 2 for n in lista]
```

- Añadimos documentación con pruebas:

```
def cuadrados(lista):
    """Calcula el cuadrado de los números de una lista
    >>> l = [0, 1, 2, 3]
    >>> cuadrados(l)
    [0, 1, 4, 9]
    """
    return [n ** 2 for n in lista]
```

Ejemplo (2ª parte)

```
def _test():
    import doctest
    doctest.testmod()


    if __name__ == “__main__”:
        _test()
```

doctest

- **Métodos**:
  - **testmod**([name])
    - Sirve para que python ejecute las pruebas.
    - Si no indicamos el nombre de un módulo evalúa el módulo actual.
    - Si va todo bien no muestra ningún mensaje.
    - En caso de error indicará el resultado obtenido y el esperado.
    - Puede indicar más información con el parámetro –v al ejecutar el script el parámetro **verbose=True**.

Ejecución con -v

```
Trying:
    l = [0, 1, 2, 3]
Expecting nothing
ok
Trying:
    cuadrados(l)
Expecting:
    [0, 1, 4, 9]
ok
2 items had no tests:
__main__
__main__._test
1 items passed all tests:
    2 tests in __main__.cuadrados
2 tests in 3 items.
2 passed and 0 failed.
Test passed.
```

Con errores

```
def cuadrados(lista):
    """Calcula el cuadrado de los números de una lista
    >>> l = [0, 1, 2, 3]
    >>> cuadrados(l)
    [0, 1, 4, 9]
    """
    return [n * 2 for n in lista]
```

- El valor esperado correcto sería: [0,2,4,6]

Salida

****

File “ejemplo.py”, **line 5**, in __main__.cuadrados
Failed example:
cuadrados(l)
Expected:
[0, 1, 4, 9]
Got:
[0, 2, 4, 6]
****

1 items had failures:
1 of 2 in __main__.cuadrados
Test Failed 1 failures.**

Pruebas anidadas

```
def cuadrado(num):
    """Calcula el cuadrado de un numero.
    >>> l = [0, 1, 2, 3]
    >>> for n in l:
    ...     cuadrado(n)
    [0, 1, 4, 9]
    """
    return num ** 2
```

En este caso queremos comprobar
El correcto funcionamiento de la
Función cuadrado que recibe un
Número como argumento y devuelve
Su cuadrado.
Los resultado obtenidos los compara
Con la lista [0,1,4,9]