Universitat de Lleida
Escola Politècnica Superior

Jordi Virgili
jvirgili@diei.udl.cat

PROCESS MODELS (DEGREE IN COMPUTER SCIENCE)

BUENAS PRÁCTICAS EN PYTHON PT2.

DECORADORES:

Los decoradores son funciones que modifican el comportamiento de otras funciones, de una manera pythonica. Lo habréis visto en algún código, una referencia a clase con @.

Continuando con la función suma, creamos una función que encapsule su llamada:

```
def encapsulador(funcion):
    def nueva_funcion(a, b):
        print("Pre-processing")
        resultado = funcion(a, b)
        print("Post-processing")
        return resultado
    return nueva_funcion

@encapsulador
def suma(a, b):
    print("Vamos a sumar dos nums")
    return a + b

suma(5,8)

Output:
# Pre-processing
# Vamos a sumar dos nums
# 13
# Post-processing
```

Lo que hace esta función de encapsulado, es añadir dos prints envolviendo la función de entrada. Por supuesto la complejidad puede ser mucho mayor, pero esto ilustra una funcionalidad básica. En Python, no es lo mismo llamar a una función que ejecutar una función, la primera te devuelve el objeto función:

```
suma => <function suma at 0x1077bf122>
suma(3,5) => 8
```

Por tanto esta funcionalidad se puede también utilizar pasándole por parámetro la función suma a la función encapsulador, de la manera, creando una nueva función con contenido:

```
fextendida = encapsulador(suma)

funcion_decorada(5, 8)
```

Una de las utilidades más usadas de los decoradores son los **logger**. Su uso nos permite escribir en un fichero los resultados de ciertas operaciones, que funciones han sido llamadas, o cualquier información que en un futuro resulte útil para ver que ha pasado.

```
def log(fichero_log):
    def decorador_log(func):
        def decorador_funcion(*args, **kwargs):
            with open(fichero_log, 'a') as opened_file:
                output = func(*args, **kwargs)
                opened_file.write(f"{output}\n")
        return decorador_funcion
    return decorador_log

@log('ficherosalida.log')
def suma(a, b):
```

```
    return a + b

@log('ficherosalida.log')
def resta(a, b):
    return a - b
```

Nótese que el decorador puede ser usado sobre funciones que tienen diferente número de parámetros de entrada, y su funcionalidad será siempre la misma. Escribir en el fichero pasado como parámetro los resultados de las operaciones.