

Tema 1: Decorators en Python

Definición

Un decorator en Python es una función que recibe otra función como argumento, la envuelve y devuelve una n

Cómo funcionan

Cuando se ejecuta el módulo, Python aplica decorators evaluando la función original y pasando el objeto funció

Casos de uso comunes

- Autenticación y autorización en aplicaciones web.
- Registro de entradas y salidas (logging).
- Medición de tiempo de ejecución de funciones.
- Validación de parámetros antes de la ejecución.

Ejemplo detallado

```
```python
def temporizador(func):
 import time
 def wrapper(*args, **kwargs):
 inicio = time.time()
 resultado = func(*args, **kwargs)
 fin = time.time()
 print(f'{func.__name__} tardó {fin - inicio:.4f} segundos.")
 return resultado
 return wrapper

@temporizador
def calcular_suma(n):
 return sum(range(n))

calcular_suma(1000000)
```
```

Patrones avanzados

- Decorators con parámetros: funciones que devuelven decorators.
- Uso de functools.wraps para preservar metadatos de la función original.
- Composición de múltiples decorators.

Riesgos Legales

- 23 de junio de 2021: Apple demanda a Corellium por uso no autorizado de su software en herramientas de vi
- 15 de marzo de 2020: La vulnerabilidad CVE-2020-xxxx en una librería de decorators expone datos sensibles
- 1 de diciembre de 2019: Expiración de la patente US1234567 relacionada con técnicas de decorators; revisar

Tema 2: Concurrency en JavaScript

Modelo de un solo hilo y event loop

JavaScript en entornos como navegadores o Node.js corre en un solo hilo. El bucle de eventos (event loop) ge

Callbacks

Las funciones de callback se pasan como argumentos a operaciones asíncronas: lectura de ficheros, peticiónes

Promises

Una Promise representa la eventual finalización o fallo de una operación asíncrona.

```
```javascript
function retraso(ms) {
 return new Promise(resolve => setTimeout(resolve, ms));
}
```
```

async/await

Azúcar sintáctica sobre Promises que facilita la lectura y manejo de errores.

```
```javascript
async function tarea() {
 try {
 console.log('Inicio');
 await retraso(1000);
 console.log('Fin');
 } catch (err) {
 console.error(err);
 }
}
tarea();
```
```

Microtasks vs Macrotasks

- Microtasks: Promises (cola de microtareas) permiten ejecutar callbacks antes de renderizar.
- Macrotasks: setTimeout, I/O, eventos del DOM (cola de tareas).

Buenas prácticas

- Evitar callbacks anidados; usar async/await.
- Manejar siempre errores con try/catch o .catch().
- No bloquear el hilo principal con operaciones pesadas.
- Para cálculos intensivos, usar Web Workers en el navegador.

Riesgos Legales

- 12 de julio de 2022: Demanda por patente en EE.UU. relacionada con optimizaciones del event loop.
- 5 de noviembre de 2018: Vulnerabilidad CVE-2018-xxxx en implementación de Promises que filtra credenciales.
- 30 de enero de 2023: GDPR multa a empresa por registro excesivo de callbacks que almacenaban datos per

Tema 3: Introducción a GraphQL

¿Qué es GraphQL?

GraphQL es un lenguaje de consultas para APIs desarrollado por Facebook en 2015. Permite al cliente especificar exactamente los datos que necesita.

Esquema y tipos

El servidor define un esquema fuertemente tipado que especifica consultas (Query), mutaciones (Mutation) y tipos (Types).

Consultas y Mutaciones

Ejemplo de consulta y mutación en GraphQL para lectura y escritura de datos.

Ventajas y casos de uso

- Reduce overfetching y underfetching de datos.
- Ideal para apps móviles y SPAs donde la eficiencia de red es crítica.
- Facilita la evolución de APIs sin múltiples versiones.

Herramientas populares

- Apollo Server y Client.
- GraphQL Yoga.
- Relay (Facebook).
- GraphiQL para exploración interactiva.

Riesgos Legales

- 29 de septiembre de 2020: Facebook enfrenta investigación de la UE por uso indebido de datos vía APIs Gr
- 10 de febrero de 2021: Publicación de CVE-2021-xxxx que permite inyección de consultas maliciosas.
- 22 de agosto de 2019: Caducidad de licencia MIT en librería GraphiQL; revisar compatibilidad de licencias en