Ejercicios de Python

Semana 5

14.07.2025 - 20.07.2025

Caso General

Gestión de una Biblioteca Digital

Las bibliotecas, tanto físicas como digitales, necesitan un sistema robusto para catalogar sus libros, gestionar los préstamos y devoluciones, y saber qué libros están disponibles. La automatización de estos procesos es esencial para un servicio eficiente.

- Yamil Ayma

Nivel Fácil: Buscar un Libro por Título

Caso

Un usuario de la biblioteca quiere saber si un libro específico está en el catálogo. Esta es la función de búsqueda más fundamental de cualquier sistema bibliotecario.

Objetivo

Escribe una función en Python llamada buscar_libro_por_titulo que reciba un catalogo (una lista de diccionarios, donde cada diccionario es un libro) y un titulo_buscado. La función debe buscar en el catálogo y devolver el diccionario completo del libro si lo encuentra. Si el libro no existe en el catálogo, debe devolver None.

Ejemplo real

Un estudiante entra a la página web de la biblioteca de su universidad y utiliza la barra de búsqueda para encontrar el libro "Cien Años de Soledad".

Pruebas de validación

```
def probar_buscar_libro_por_titulo():
    catalogo = [
        {'titulo': 'Cien Años de Soledad', 'autor': 'Gabriel García Márquez', 'disponible':
True},
        {'titulo': 'El Señor de los Anillos', 'autor': 'J.R.R. Tolkien', 'disponible': False},
        {'titulo': '1984', 'autor': 'George Orwell', 'disponible': True}
    ]

    # Prueba 1: Buscar un libro que existe
    libro_encontrado = buscar_libro_por_titulo(catalogo, '1984')
    esperado1 = {'titulo': '1984', 'autor': 'George Orwell', 'disponible': True}
    print(f"Prueba 1: {libro_encontrado == esperado1}")

    # Prueba 2: Buscar un libro que no existe
    libro_no_encontrado = buscar_libro_por_titulo(catalogo, 'Fahrenheit 451')
    print(f"Prueba 2: {libro_no_encontrado is None}")

    # Prueba 3: Buscar en un catálogo vacío
    libro_en_catalogo_vacio = buscar_libro_por_titulo([], 'Cien Años de Soledad')
    print(f"Prueba 3: {libro_en_catalogo_vacio is None}")

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_buscar_libro_por_titulo()
```

Copie el código

Pegue estas lineas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 👩‍👩‍👧

Nivel Medio: Prestar y Devolver un Libro

Caso

Un bibliotecario necesita procesar el préstamo de un libro a un usuario. El sistema debe verificar si el libro está disponible y actualizar su estado. De la misma forma, debe poder procesar la devolución.

Objetivo

Crea una función llamada gestionar_prestamo que reciba un catalogo, el titulo del libro y una accion (que puede ser "prestar" o "devolver").

- Si la accion es "prestar": la función debe verificar si el libro existe y está disponible. Si es así, cambia su estado 'disponible' a False y devuelve True. De lo contrario, devuelve False.
- Si la accion es "devolver": la función debe verificar si el libro existe y no está disponible. Si es así, cambia su estado 'disponible' a True y devuelve True. De lo contrario, devuelve False.

Continua en la siguiente página

Ejemplo real

- En el mostrador de la biblioteca, el bibliotecario escanea un libro para prestarlo, y el sistema actualiza su estado. Días después, el usuario regresa, el libro es escaneado de nuevo, y el sistema lo marca como disponible otra vez.

Pruebas de validación

```
def probar_gestionar_prestamo():
    catalogo_base = [
        {'titulo': 'Cien Años de Soledad', 'autor': 'G.G. Márquez', 'disponible': True},
        {'titulo': 'El Señor de los Anillos', 'autor': 'J.R.R. Tolkien', 'disponible': False}
    ]

    # Prueba 1: Prestar un libro disponible
    catalogo_p1 = [libro.copy() for libro in catalogo_base]
    resultado1 = gestionar_prestamo(catalogo_p1, 'Cien Años de Soledad', 'prestar')
    libro_actualizado1 = buscar_libro_por_titulo(catalogo_p1, 'Cien Años de Soledad')
    print(f"Prueba 1 (Prestar éxito): {resultado1 is True and not libro_actualizado1['disponible']}")

    # Prueba 2: Intentar prestar un libro no disponible
    catalogo_p2 = [libro.copy() for libro in catalogo_base]
    resultado2 = gestionar_prestamo(catalogo_p2, 'El Señor de los Anillos', 'prestar')
    print(f"Prueba 2 (Prestar fallo): {resultado2 is False}")

    # Prueba 3: Devolver un libro prestado
    catalogo_p3 = [libro.copy() for libro in catalogo_base]
    resultado3 = gestionar_prestamo(catalogo_p3, 'El Señor de los Anillos', 'devolver')
    libro_actualizado3 = buscar_libro_por_titulo(catalogo_p3, 'El Señor de los Anillos')
    print(f"Prueba 3 (Devolver éxito): {resultado3 is True and libro_actualizado3['disponible']}")

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_gestionar_prestamo()
```

Copie el código

Pegue estas lineas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 👷‍♂️👷‍♀️

Nivel Avanzado: Reporte de Libros Vencidos

Caso

La biblioteca necesita generar un reporte de los libros que no han sido devueltos a tiempo para poder enviar recordatorios a los usuarios. Esto es vital para la gestión del inventario y la disponibilidad de los títulos más solicitados.

Objetivo

Escribe una función llamada reporte_libros_vencidos que reciba un catalogo y una fecha_actual. Algunos libros en el catálogo (los que no están disponibles) tendrán una clave adicional 'fecha_devolucion' con una fecha. La función debe devolver una lista de diccionarios, conteniendo solo los libros cuya fecha_devolucion es anterior a la fecha_actual. Cada diccionario en el resultado debe contener el 'titulo' del libro y los 'dias_retraso'. La lista debe estar ordenada de mayor a menor según los días de retraso.

Continua en la siguiente página

Ejemplo real

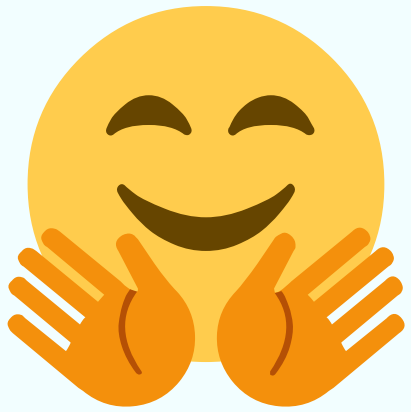
Un sistema automatizado que corre cada mañana, compara la fecha del día con las fechas de devolución y genera un listado para que el personal de la biblioteca contacte a los usuarios morosos.

Pruebas de validación

```
from datetime import date, timedelta

def probar_reporte_libros_vencidos():
    hoy = date(2023, 10, 27)
    catalogo = [
        {'titulo': '1984', 'autor': 'George Orwell', 'disponible': False, 'fecha_devolucion':
date(2023, 10, 20)},
        {'titulo': 'Dune', 'autor': 'Frank Herbert', 'disponible': False, 'fecha_devolucion':
date(2023, 10, 30)},
        {'titulo': 'El Hobbit', 'autor': 'J.R.R. Tolkien', 'disponible': True},
        {'titulo': 'Fahrenheit 451', 'autor': 'Ray Bradbury', 'disponible': False, 'fecha_devolucion':
date(2023, 9, 27)}
    ]

    # Prueba 1: Generar reporte de vencidos
    reporte = reporte_libros_vencidos(catalogo, hoy)
    # Esperado: Fahrenheit (30 días de retraso), 1984 (7 días de retraso)
    esperado1 = [
        {'titulo': 'Fahrenheit 451', 'dias_retraso': 30},
        {'titulo': '1984', 'dias_retraso': 7}
    ]
    print(f"Prueba 1: {reporte == esperado1}")

    # Prueba 2: No hay libros vencidos
    catalogo_sin_vencidos = [
        {'titulo': 'Dune', 'autor': 'Frank Herbert', 'disponible': False, 'fecha_devolucion':
date(2023, 11, 1)}
    ]
    reporte2 = reporte_libros_vencidos(catalogo_sin_vencidos, hoy)
    print(f"Prueba 2: {reporte2 == []}")

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_reporte_libros_vencidos()
```

Copie el código

Pegue estas lineas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 👨‍👩‍👧



Éxitos en los ejercicios

Hay dos maneras de construir un diseño de software. Una es simplificarlo tanto que no presente deficiencias evidentes. Y la otra es complicarlo tanto que no presente deficiencias evidentes.

- CAR Hoare

Aprende mucho