Ejercicios de Python

Semana 3

07.07.2025 - 13.07.2025

Caso General

Sistema de Gestión de Tareas

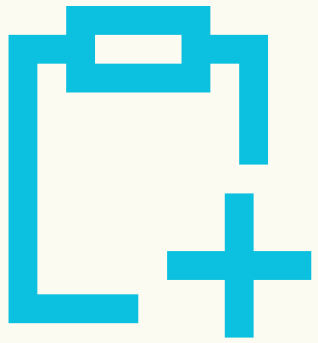
Organizar las tareas es fundamental tanto en la vida personal como en la profesional. Un sistema de gestión de tareas permite llevar un registro de lo que hay que hacer, marcar el progreso y priorizar el trabajo. La lógica detrás de una simple "To-Do List" es la base de software complejo como Trello, Asana o Jira.

- Yamil Ayma

Nivel Fácil: Agregar una Nueva Tarea

Caso

La funcionalidad más básica de cualquier lista de tareas es poder añadir un nuevo elemento. Sin esto, el sistema no tiene utilidad.



Objetivo

Escribe una función en Python llamada agregar_tarea que reciba una lista de tareas (lista_tareas) y la descripción de una nueva tarea (descripcion). Cada tarea en la lista es un diccionario. La función debe añadir un nuevo diccionario a la lista con la 'descripcion' proporcionada y un estado 'completada' inicializado en False. La función debe devolver la lista de tareas actualizada.

Ejemplo real

Un usuario abre su aplicación de notas y escribe "Comprar pan" para añadirlo a su lista de la compra.

Pruebas de validación

```
def probar_agregar_tarea():
    # Prueba 1: Agregar a una lista vacía
    tareas = []
    tareas_actualizadas = agregar_tarea(tareas, "Estudiar Python")
    print(f"Prueba 1: {tareas_actualizadas == [{'descripcion': 'Estudiar Python', 'completada':
False}]}")

    # Prueba 2: Agregar a una lista existente
    tareas_existentes = [{'descripcion': 'Hacer ejercicio', 'completada': True}]
    tareas_actualizadas_2 = agregar_tarea(tareas_existentes, "Llamar al dentista")
    esperado = [
        {'descripcion': 'Hacer ejercicio', 'completada': True},
        {'descripcion': 'Llamar al dentista', 'completada': False}
    ]
    print(f"Prueba 2: {tareas_actualizadas_2 == esperado}")

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

Copie el código

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

Nivel Medio: Marcar Tarea como Completada y Filtrar por Estado

Caso

A medida que se completan las tareas, es esencial poder marcarlas como finalizadas. Además, un usuario a menudo querrá ver solo las tareas pendientes para enfocarse, o revisar las que ya ha completado.

Objetivo

Crea una función llamada marcar_y_filtrar que reciba tres argumentos: una lista_tareas, la descripcion de la tarea a marcar, y un estado_filtro (que puede ser "pendientes", "completadas" o "todas").

- La función debe buscar en la lista_tareas la tarea cuya descripción coincida y cambiar su estado 'completada' a True. Si no la encuentra, no hace nada.
- Luego, debe devolver una nueva lista que contenga únicamente las tareas que coincidan con el estado_filtro.

Continua en la siguiente página

Ejemplo real

En una app de tareas, marcas la casilla de "Lavar el coche" y luego usas el filtro "Pendientes" para ver qué te queda por hacer.

Pruebas de validación

```
def probar_marcar_y_filtrar():
    lista_base = [
        {'descripcion': 'Comprar leche', 'completada': False},
        {'descripcion': 'Pagar facturas', 'completada': False},
        {'descripcion': 'Sacar al perro', 'completada': False}
    ]

    # Prueba 1: Marcar una tarea y filtrar por completadas
    tareas_p1 = [t.copy() for t in lista_base]
    resultado1 = marcar_y_filtrar(tareas_p1, 'Pagar facturas', 'completadas')
    print(f"Prueba 1: {resultado1 == [{'descripcion': 'Pagar facturas', 'completada': True}]}")

    # Prueba 2: Marcar una tarea y filtrar por pendientes
    tareas_p2 = [t.copy() for t in lista_base]
    resultado2 = marcar_y_filtrar(tareas_p2, 'Comprar leche', 'pendientes')
    esperado2 = [
        {'descripcion': 'Pagar facturas', 'completada': False},
        {'descripcion': 'Sacar al perro', 'completada': False}
    ]
    print(f"Prueba 2: {resultado2 == esperado2}")

    # Prueba 3: Intentar marcar una tarea que no existe y filtrar todo
    tareas_p3 = [t.copy() for t in lista_base]
    resultado3 = marcar_y_filtrar(tareas_p3, 'Leer un libro', 'todas')
    # La lista original no debe cambiar, porque la tarea no se encontró
    print(f"Prueba 3: {resultado3 == lista_base}")


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

Copie el código

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

Nivel Avanzado: Asignar Prioridad y Ordenar Tareas Pendientes

Caso

En proyectos complejos, no todas las tareas tienen la misma importancia. Es crucial poder asignar prioridades y ordenar la lista de tareas para que las más urgentes aparezcan primero.

Objetivo

Escribe una función llamada reporte_prioritario que reciba una lista_tareas. Algunas tareas en la lista pueden tener una clave adicional 'prioridad' con valores "Alta", "Media" o "Baja". Las tareas sin esta clave se consideran de prioridad "Baja". La función debe devolver una nueva lista que contenga únicamente las tareas pendientes ('completada': False), ordenadas por prioridad de la siguiente forma: primero las de prioridad "Alta", luego "Media" y finalmente "Baja".

Continua en la siguiente página

Ejemplo real

Un gestor de proyectos necesita ver un resumen de las tareas críticas (prioridad "Alta") que el equipo aún no ha completado para planificar la semana.

Pruebas de validación

```
def probar_reporte_prioritario():
    tareas = [
        {'descripcion': 'Revisar código', 'completada': False, 'prioridad': 'Media'},
        {'descripcion': 'Diseñar base de datos', 'completada': False, 'prioridad': 'Alta'},
        {'descripcion': 'Actualizar documentación', 'completada': True, 'prioridad': 'Baja'},
        {'descripcion': 'Corregir bug crítico', 'completada': False, 'prioridad': 'Alta'},
        {'descripcion': 'Enviar email de seguimiento', 'completada': False}, # Prioridad Baja por
defecto
        {'descripcion': 'Reunión de equipo', 'completada': False, 'prioridad': 'Media'}
    ]

    # Prueba 1: Generar reporte ordenado
    reporte = reporte_prioritario(tareas)
    # Esperado: 2 Altas, 2 Medias, 1 Baja. Las completadas se ignoran.
    # El orden relativo entre tareas de la misma prioridad no importa para la prueba.
    nombres_ordenados = [t['descripcion'] for t in reporte]

    es_valido = (
        len(reporte) == 5 and
        reporte[0]['prioridad'] == 'Alta' and reporte[1]['prioridad'] == 'Alta' and
        reporte[2]['prioridad'] == 'Media' and reporte[3]['prioridad'] == 'Media' and
        reporte[4]['prioridad'] == 'Baja'
    )
    print(f"Prueba 1: {es_valido}")

    # Prueba 2: Lista con solo tareas completadas
    tareas_completadas = [{'descripcion': 'Tarea 1', 'completada': True, 'prioridad': 'Alta'}]
    reporte2 = reporte_prioritario(tareas_completadas)
    print(f"Prueba 2 (solo completadas): {reporte2 == []}")

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

Copie el código

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

Éxitos en los ejercicios

La simplicidad no precede a la complejidad, pero la sigue.
- Alan Perlis. Epigramas de la Programación

Aprende mucho