Tema 11: La función FILTER

Lectura fácil

1. ¿Qué es `filter`?

La función `filter()` en Python se utiliza para **filtrar** elementos de una lista o cualquier estructura iterable (como tuplas o conjuntos), seleccionando únicamente aquellos que cumplen con una **condición** específica. Esto es posible gracias a una **función** que pasamos como argumento, la cual define qué elementos deben ser incluidos en el resultado filtrado.

Sintaxis:

```
filter(funcion, iterable)
```

- **funcion**: Una función que toma un elemento y devuelve `True` si el elemento cumple la condición o `False` si no lo cumple.
- **iterable**: La estructura de datos que vamos a filtrar (por ejemplo, una lista o una tupla).

2. Ejemplos de Uso de `filter()`

Filtrar números pares

Supongamos que queremos quedarnos solo con los números pares de una lista de números.

```
# Lista de números
```

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]


# Definir una función que devuelve True si el número es par

def es_par(n):

    return n % 2 == 0  ← Devolverá True o False


# Usar filter() para obtener solo los números pares

numeros_pares = filter(es_par, numeros)


# Convertir el resultado en una lista y mostrarlo

print(list(numeros_pares))
```

Salida:

```
[2, 4, 6, 8, 10]
```

Aquí definimos una función `es_par()` que retorna `True` si el número es divisible entre 2. Luego, usamos `filter()` para aplicar esta función a la lista de números, manteniendo solo los números pares.

Uso de filter con diccionarios

Seleccionar los productos cuyo precio es mayor de 1.20

```
productos = {
   "pan": 1.00,
```

```
    "leche": 1.20,
    "huevos": 2.50,
    "queso": 4.00
}
def precio_mayor_120(item):
    producto, precio = item
    return precio > 1.20

# Filtramos pares (producto, precio)
productos_filtrados = dict(filter(precio_mayor_120, productos.items()))

print(productos_filtrados)
```

3. ¿Qué es `map`?

La función `map()` también es una función de orden superior en Python, pero en lugar de **filtrar** elementos, `map()` se usa para **transformar** o **aplicar una operación** a todos los elementos de un iterable, y devolver un nuevo iterable con los resultados.

Sintaxis:

```
map(funcion, iterable)
```

- **funcion**: Una función que define cómo se transformará cada elemento del iterable.
- **iterable**: La estructura de datos cuyos elementos serán transformados.

---

4. Ejemplos de Uso de `map()`

Multiplicar cada número por 2

Vamos a usar `map()` para multiplicar todos los números de una lista por 2.

```
# Lista de números
numeros = [1, 2, 3, 4, 5]

# Definir una función que multiplica el número por 2
def multiplicar_por_2(n):
    return n * 2

# Usar map() para aplicar la función a todos los números
numeros_multiplicados = map(multiplicar_por_2, numeros)

# Convertir el resultado en una lista y mostrarlo
print(list(numeros_multiplicados))
```

Salida:

```
[2, 4, 6, 8, 10]
```

Aquí, la función `multiplicar_por_2()` toma cada número de la lista y lo multiplica por 2. `map()` aplica esta transformación a todos los elementos y devuelve un nuevo iterable con los resultados.

---

5. Diferencias entre `filter()` y `map()`

Aunque ambas funciones son útiles para trabajar con colecciones de datos, tienen **propósitos diferentes**:

Característica	`filter()`	`map()`
Propósito	Filtrar elementos que cumplen una condición	Transformar todos los elementos de un iterable
Resultado	Solo los elementos que cumplen con la condición	Un nuevo iterable con todos los elementos transformados
Tipo de función usada	La función debe devolver `True` o `False`	La función devuelve el resultado de la transformación
Ejemplo de uso	Filtrar números pares de una lista	Multiplicar cada número de una lista por 2

Ejemplo visual	Elimina algunos elementos	Aplica un cambio a todos los elementos

Ejemplo Comparativo:

Imaginemos que tenemos una lista de números y queremos hacer dos cosas:

1. **Filtrar** los números mayores que 5 (usando `filter()`).
2. **Multiplicar** todos los números por 3 (usando `map()`).

```
# Lista de números
numeros = [2, 4, 6, 8, 10]

# Usar filter() para filtrar números mayores que 5
def mayor_a_5(n):
    return n > 5  ← Devolverá True o False

numeros_mayores_5 = filter(mayor_a_5, numeros)
print(list(numeros_mayores_5))  # [6, 8, 10]

# Usar map() para multiplicar cada número por 3
def multiplicar_por_3(n):
    return n * 3

numeros_multiplicados = map(multiplicar_por_3, numeros)
```

```
print(list(numeros_multiplicados))  # [6, 12, 18, 24, 30]
```

En el ejemplo anterior:

- `filter()` devuelve solo los números que son mayores que 5, eliminando los que no cumplen la condición.
- `map()` devuelve una nueva lista donde cada número ha sido multiplicado por 3, sin eliminar ninguno.

---

6. Cuándo Usar `filter()` y Cuándo Usar `map()`

- **Usa `filter()`** cuando quieras **eliminar elementos** de una lista u otro iterable según una condición.
  Ejemplo: Filtrar solo los números positivos de una lista.
- **Usa `map()`** cuando necesites **aplicar una transformación** a cada elemento de una lista u otro iterable, sin eliminar nada.
  Ejemplo: Convertir todas las temperaturas de grados Celsius a Fahrenheit.

---

7. Ejercicios Prácticos

Ejercicio 1: Usar filter() para eliminar los números menores a 10
Dada la siguiente lista de números:

```
numeros = [4, 9, 16, 25, 1, 7, 12]
```

Define una función `mayor_a_10()` que devuelva `True` para los números mayores que 10 y úsala con `filter()` para quedarte solo con esos números.

Ejercicio 2: Usar `map()` para convertir metros a centímetros
Dada la siguiente lista de alturas en metros:

```
alturas_metros = [1.60, 1.75, 1.80, 1.50]
```

Define una función `metros_a_centimetros()` que multiplique cada altura por 100 para convertirla a centímetros, y úsala con `map()`.

---

8. Glosario

- **`filter()`**: Función que devuelve solo los elementos que cumplen con una condición.
- **`map()`**: Función que transforma todos los elementos de un iterable aplicando una función.
- **Iterable**: Cualquier estructura de datos que puede ser recorrida (como listas o tuplas).
- **Función**: Bloque de código que realiza una tarea específica y devuelve un resultado.