Clase 2: Funciones Lambda y Programación Funcional en Python 🐍

Introducción a las Funciones Lambda

Las funciones lambda son funciones anónimas que se utilizan para crear pequeñas funciones de manera rápida y sin necesidad de nombrarlas. Se utilizan comúnmente cuando se requiere una función para un corto período de tiempo y son ideales para operaciones sencillas.

Sintaxis de una Función Lambda

```
lambda argumentos: expresión
```

- **argumentos**: los parámetros de entrada de la función.
- **expresión**: el valor que se retorna cuando se llama a la función.

Ejemplo Básico

```
# Definimos una función lambda que suma dos números
suma = lambda x, y: x + y
resultado = suma(10, 5)  # Resultado: 15
print(resultado)
```

En este ejemplo, `suma` es una función lambda que toma dos argumentos (`x` e `y`) y retorna su suma.

Uso de Funciones Lambda en Operaciones Compuestas

A menudo, queremos aplicar funciones a listas. Las funciones lambda son muy útiles en este contexto.

Ejemplo: Elevar al Cuadrado

Vamos a crear una lista de números del 0 al 10 y aplicar una función lambda para obtener el cuadrado de cada número.

```
# Lista de números del 0 al 10
numeros = list(range(11))

# Usamos map para aplicar la función lambda a cada elemento d
e la lista
cuadrados = list(map(lambda x: x**2, numeros))
print("Cuadrados:", cuadrados)  # Resultado: [0, 1, 4, 9, 16,
25, 36, 49, 64, 81, 100]
```

Filtrado de Elementos en una Lista

Podemos utilizar la función `filter()` en combinación con funciones lambda para seleccionar elementos que cumplen ciertas condiciones.

Ejemplo: Filtrar Números Pares

```
# Filtramos los números pares de la lista
pares = list(filter(lambda x: x % 2 == 0, numeros))
print("Números Pares:", pares)  # Resultado: [0, 2, 4, 6, 8,
10]
```

Resumen de Funciones Lambda

1. **Son funciones anónimas**: No necesitan un nombre y se definen en una sola línea.
2. **Útiles para operaciones simples**: Se utilizan principalmente para funciones que requieren poca lógica.
3. **Se pueden usar con `map()` y `filter()`**: Facilitan la aplicación de operaciones en listas de forma concisa.

Ejercicios Prácticos

Ejercicio 1: Sumar Números

Objetivo: Crear una función lambda que sume dos números ingresados por el usuario.

Solución Paso a Paso

1. Definimos la función lambda para sumar.
2. Pedimos al usuario que ingrese dos números.
3. Llamamos a la función y mostramos el resultado.

```
# Definimos la función lambda
suma = lambda x, y: x + y

# Pedimos números al usuario
num1 = float(input("Ingresa el primer número: "))
num2 = float(input("Ingresa el segundo número: "))
```

```
# Calculamos y mostramos el resultado
resultado = suma(num1, num2)
print(f"La suma de {num1} y {num2} es: {resultado}")
```

Ejercicio 2: Cuadrados de Números

Objetivo: Obtener el cuadrado de los números del 1 al 10.

Solución Paso a Paso

1. Creamos una lista con los números del 1 al 10.
2. Usamos `map()` con una función lambda para elevar al cuadrado.
3. Imprimimos el resultado.

```
# Lista de números del 1 al 10
numeros = list(range(1, 11))

# Usamos map para aplicar la función lambda
cuadrados = list(map(lambda x: x**2, numeros))
print("Cuadrados:", cuadrados)  # Resultado: [1, 4, 9, 16, 2
5, 36, 49, 64, 81, 100]
```

Ejercicio 3: Filtrar Números Impares

Objetivo: Filtrar y mostrar los números impares de una lista.

Solución Paso a Paso

1. Creamos una lista de números del 1 al 20.
2. Usamos `filter()` con una función lambda para obtener los números impares.
3. Mostramos el resultado.

```
# Lista de números del 1 al 20
numeros = list(range(1, 21))
```

```
# Filtramos los números impares
impares = list(filter(lambda x: x % 2 != 0, numeros))
print("Números Impares:", impares)  # Resultado: [1, 3, 5, 7,
9, 11, 13, 15, 17, 19]
```

Conclusiones 🎓

Las funciones lambda son herramientas poderosas y concisas para realizar operaciones simples en Python. Se utilizan a menudo en programación funcional, especialmente en combinación con `map()` y `filter()`. Practicar su uso a través de ejemplos reales ayuda a entender su aplicabilidad y versatilidad.