TIC

Módulo 1

LECCIÓN 2

Map, Filter, Reduce y lambda.

Funciones Lambda

Una función lambda es una función sin nombre (también puedes llamarla una función anónima). Por supuesto, tal afirmación plantea inmediatamente la pregunta: ¿cómo se usa algo que no se puede identificar?

Afortunadamente, no es un problema, ya que se puede mandar llamar dicha función si realmente se necesita, pero, en muchos casos la función lambda puede existir y funcionar mientras permanece completamente de incógnito.

La declaración de la función lambda no se parece a una declaración de función normal. La siguiente cláusula devuelve el valor de la expresión al tomar en cuenta el valor del argumento lambda actual.

Python

```
#lambda parameters: expression

two = lambda: 2
sqr = lambda x: x * x
pwr = lambda x, y: x ** y

for a in range(-2, 3):
        print(sqr(a), end=" ")
        print(pwr(a, two()))

#4 4
#1 1
#0 0
#1 1
#4 4
```

Fuente: Elaboración propia

- **La primera** lambda es una función anónima sin parámetros que siempre devuelve un 2. Como se ha asignado a una variable llamada two, podemos decir que la función ya no es anónima, y se puede usar su nombre para invocarla.

- **La segunda** es una función anónima de un parámetro que devuelve el valor de su argumento al cuadrado. Se ha nombrado sqr.
- **La tercera** lambda toma dos parámetros y devuelve el valor del primero elevado al segundo. El nombre de la variable que lleva la lambda habla por sí mismo.

PEP 8, la Guía de Estilo para Código Python, recomienda que las funciones lambdas no deben asignarse a variables, sino que deben definirse como funciones.

Esto significa que es mejor utilizar una sentencia def, y evita usar una sentencia de asignación que vincule una expresión lambda a un identificador.

Analiza el código a continuación:

Python

```
# Recomendado:
def f(x): return 3*x


# No recomendado:
f = lambda x: 3*x
```

Fuente: Elaboración propia

La parte más interesante de usar lambdas aparece cuando puedes usarlas en su forma pura: Como partes anónimas de código destinadas a evaluar un resultado.

Imagina que necesitamos una función (la nombraremos print_function) que imprime los valores de otra función dada para un conjunto de argumentos seleccionados.

Python

```
def print_function(args, fun):
        for x in args:
        print('f(', x,') = ', fun(x), sep='')


def poly(x):
        return 2 * x**2 - 4 * x + 2


print_function([x for x in range(-2, 3)], poly)

#Ahora utilizando lambda para evitar crear la funnción poly() solo para definir el polinomio.

print_function([x for x in range(-2, 3)], lambda x: 2 * x**2 - 4 * x + 2)

#La salida es la misma

#f(-2) = 18
#f(-1) = 8
#f(0) = 2
#f(1) = 0
#f(2) = 2
```

Fuente: Elaboración propia

Como se puede apreciar en el segundo llamado de print_function, dado que la función poly requiere de un argumento para poderse ejecutar, es necesario utilizar una función anónima para poder hacer uso del argumento.