COLOMBIA POTENCIA DE LA VIDA

TIC

Módulo 1

LECCIÓN 3

Funciones lambda

Lambda Functions

La función lambda es un concepto tomado de las matemáticas, más específicamente, de una parte llamada el Cálculo Lambda, pero estos dos fenómenos no son iguales.

Una función lambda es una función sin nombre (también puedes llamarla una función anónima).

Se definen en una línea y cuyo código a ejecutar suele ser pequeño.

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.

```
[ ]  #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
```

- 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.

```
[ ]  #Función tradicional
     def suma(a, b):
         return a+b

     #Versión lambda
     lambda a, b : a + b
```

```
     #El resultado de la función se puede almacenar en una variable
     suma = lambda a, b: a + b

     #O llamarla directamente
     (lambda a, b: a + b)(2, 4)
```

```
6
```

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.

```
[ ]  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
```

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

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.

Alcance/Scope/Ambito de las variables en Python

En Python, las variables solo existen dentro de la región que ha sido creada, a esto se le conoce como el scope.

Una variable que es creada dentro de una función, solo puede ser utilizada dentro de dicha función, esto se conoce como scope local.

Del ejemplo anterior se puede identificar las variables a y b las cuales solo existen dentro del bloque de la función, ya que por fuera de ella las variables no están definidas o tienen un valor y tipo de datos diferente.

```
[ ]  a = 10
     b = 100
     print(a, b)

     def imprimir(a):
       print(f'Aquí la variable a vale {a}')
       print(f'y puedo acceder a b que vale {b}')

     imprimir(50);

     print(a,b)
```

```
10 100
Aquí la variable a vale 50
y puedo acceder a b que vale 100
10 100
```