FUNCIONES LAMBDA

SE TRATA DE CREAR FUNCIONES DE MANERA RÁPIDA, JUST IN TIME, SOBRE LA MARCHA, PARA PROTOTIPOS LIGEROS QUE REQUIEREN ÚNICAMENTE DE UNA PEQUEÑA OPERACIÓN O COMPROBACIÓN. POR LO TANTO, TODA FUNCIÓN LAMBDA TAMBIÉN PUEDE EXPRESARSE COMO UNA CONVENCIONAL (PERO NO VICEVERSA).

```
def a(x, y):
    return x + y

b = lambda x, y: x + y
```

EN LA IMAGEN SE OBSERVA COMO ESTA CONSTITUIDA UNA FUNCION LAMBDA Y EN COMPARACION A UNA CONVENCIONAL. EN COLOR VERDE SE REMARCAN LOS ARGUMENTOS, Y EN ROJO EL VALOR DE RETORNO. EN BASE A ESTO, SE CONFORMA DE LA SIGUIENTE MANERA:

- `lambda argumentos: resultado`

O BIEN:

- `f = lambda argumentos: resultado`

Y ES EQUIVALENTE A:

```
def f(argumentos):
    return resultado
```

AL IGUAL QUE EN LAS CONVENCIONALES, EN LAS FUNCIONES LAMBDA PUEDEN UTILIZARSE ARGUMENTOS OPCIONALES O BIEN NINGUNO, RETORNAR CUALQUIER TIPO DE VALOR O BIEN NONE. ALGUNOS EJEMPLOS Y SUS EQUIVALENCIAS:

```
>>> def f(x, y, z=1):
...     return (x+y) * z
>>> f(5, 6)
11
>>> f(5, 6, 7)
77
>>> f = lambda x, y, z=1: (x+y) * z
>>> f(5, 6)
11
>>> f(5, 6, 7)
77
```

```
>>> def f():
...     return
>>> print(f())
None
>>> f = lambda: None
>>> print(f())
None
```

SIN EMBARGO, LAS FUNCIONES ESTÁN SIEMPRE DIFERENCIADAS, AUNQUE TENGAN COMPORTAMIENTO Y RESULTADOS EQUIVALENTES.:

```
>>> def a(x):
...     return x * 5
>>> b = lambda x: x * 5
>>> a
<function a at 0x0238C7B0>
>>> b
<function <lambda> at
0x0238CA70>
>>> a == b
False
>>> a is b
False
```

LA RAZÓN POR LA CUAL QUERRÁS EMPLEAR ESTE TIPO DE FUNCIONES ES SIMPLEMENTE POR RAPIDEZ (DURANTE EL DESARROLLO, NO LA EJECUCIÓN) Y, EN CIERTOS CASOS, CLARIDAD DEL CÓDIGO. VEAMOS UN EJEMPLO PRÁCTICO.

LA FUNCIÓN FILTER(FUNCION, ITERABLE) CONSTRUYE UNA LISTA CON AQUELLOS ELEMENTOS EN LOS CUALES FUNCION(ITERABLE[I]) RETORNA TRUE. POR LO TANTO, SI SE QUIEREN MANTENER LOS NÚMEROS MAYORES A CERO DE UNA DETERMINADA LISTA, PODRÍA UTILIZARSE:

```
>>> a = [0, 1, -1, -2, 3, -4, 5, 6, 7]
>>> b = filter(lambda x: x > 0, a)
>>> b
[1, 3, 5, 6, 7]
```

QUE ES EQUIVALENTE A:

```
def f(x):
    return x > 0
b = filter(f, a)
```

COMO SE PUEDE OBSERVAR, LA SEGUNDA OPCIÓN SE VE MÁS EXPLÍCITA (TRES LÍNEAS), AUNQUE LA PRIMERA ACTÚA DE FORMA SIMILAR EN MENOR CANTIDAD DE CÓDIGO (UNA LÍNEA)