07-map filter reduce

July 25, 2020

1 Introdução

O Python provê três funções que são largamente aplicadas para implementar soluções em estilo de programação funcional: **map**, **filter** e **functools.reduce**.

Estas função são de **Alta Ordem**, que recebem um iterável (coleção) **it**, e uma outra função, **fn**, e aplica **fn** de diferentes maneiras no iterável **it**.

Nota: Funções de Alta Ordem são funções que recebem e/ou retornam outras funções e serão vistas com mais detalhes na próxima semana.

2 filter

```
it2 = filter(fn, it)
```

Retorna um iterável, **it2**, contendo apenas os elementos de **it** selecionados pela função de teste **fn**. A função de teste, **fn**, deve receber um valor (que será um item da coleção **it**) e retornar **True** caso o item deva ser incluído no iterável retornado ou **False** caso contrário.

Observe as duas funções abaixo:

- eh_par(valor): retorna **True** se **valor** é par
- eh_impar(valor) retorna **True** se **valor** é impar

[1]:
```
def eh_par(valor):
    return (valor % 2) == 0

def eh_impar(valor):
    return (valor % 2) == 1

lista = [x for x in range(0, 11)]  # equivalente a: lista = [0, 1, 2, 3, 4, 5,␣
↪6, 7, 8, 9, 10]
print("Todos:", lista)
print("Par:", list(filter(eh_par, lista)))
print("Ímpar:", list(filter(eh_impar, lista)))
```

```
Todos: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Par: [0, 2, 4, 6, 8, 10]
Ímpar: [1, 3, 5, 7, 9]
```

Note que chamamos o construtor **list** passando o resultado de **filter** já que no Python 3 o retorno do **filter** é um iterável e queremos, por conveniência, usar a forma legível de se imprimir uma lista. A linha abaixo apresenta o código para a impressão do retorno da função **filter**, porém sem a utilização do **list**.

[2]:
```
print("Par:", filter(eh_par, lista))
```

```
Par: <filter object at 0x7f6f70d63190>
```

3 Funções lambda

Também podemos passar funções **lambda** para **filter**, **map** e **functools.reduce**. Isto é, ao invés de uma função definida for **def** podemos usar funções **lambda**. Funções **lambda** são funções anônimas e a sua construção minimalística é extremamente útil para pequenas funções.

Observe a implementação do filtro de números pares com Funções **lambda**:

[3]:
```
print(list(filter(lambda x: x % 2 == 0, lista)))
```

```
[0, 2, 4, 6, 8, 10]
```

4 map

```
it2 = map(fn, it, ...)
```

O **map** aplica a função **fn** a cada elemento do iterável **it** (ou iteráveis, como indicado por ...) e retorna um novo iterável **it2** com o resultado gerado pela aplicação da função.

A função **fn** recebe um item do iterável (ou de cada iterável, como será abordado a seguir) e deve retornar um valor a ser acrescentado no iterável de saída.

Observe o exemplo abaixo onde uma função **lambda** que dobra o valor dos itens do iterável é utilizada:

[4]:
```
lista = [1,2,4,6,8]
print(list(map(lambda x: x*2, lista)))
```

```
[2, 4, 8, 12, 16]
```

O **map** pode receber como parâmetro vários iteráveis, neste caso a função **fn** deve receber o mesmo número de parâmetros que o número de iteráveis. No exemplo abaixo fazemos a soma dos elementos na mesma posição.

[5]:
```
lista2 = [5, 6, 7, 8]
print(list(map(lambda x, y: x+y, lista, lista2)))
```

```
[6, 8, 11, 14]
```

Observe que a quantidade de item dos iteráveis passados como parâmetros não necessitam ser as mesmas. Caso os iteráveis tenham quantidades variáveis de elementos, será considerado o menor iterável e os demais elementos serão descartados.

5 functools.reduce

```
valor = functools.reduce(fn, it [, init])
```

A função **functools.reduce** aplica a função **fn** a cada elemento do iterável **it** de forma a combinar todos os elementos em um único valor (sumário). A função **fn** deve receber como argumento dois valores, o primeiro é o valor já computado dos itens anteriores e o segundo o valor do item atual. Opcionalmente, pode-se definir um valor inicial padrão com **init** (conforme indicado por **[, init]**

Para utilizar functools.reduce é necessário importar a função já que ela foi removida do **core** de funções Python.

[6]:
```
from functools import reduce

lista = [1, 3, 5, 7]
print(reduce(lambda a, b: a+b, lista))
```

```
16
```

O código acima retorna o somatório dos itens da lista. Três chamadas a função lambda foram realizadas:

- 1, 3 que retornou 4;
- 4, 5 que retornou 9; e
- 9, 7 que retornou 16.

6 Bibliografia

John Hunt. **A Beginners Guide to Python 3 Programming**. Undergraduate Topics in Computer Science. Springer, 2019.

Kent D. Lee. **Python Programming Fundamentals**. Undergraduate Topics in Computer Science. Springer, 2014.