MBA em Ciência de Dados

Programação para Ciência de Dados

Avaliação Python Parte III

Material Produzido por Luis Gustavo Nonato
Cemeai - ICMC/USP São Carlos

Exercício 1)

Considere a lista `numeros` abaixo. Utilize o comando map para gerar um nova lista `par_impar` contendo valores **True** ou **Falso** dependendo sebo elemento da lista `numeros` é par ou ímpar, respectivamente. Por exemplo:

```
numeros = [7, 3, 2, 13,...]
par_impar = [False, False, True, False,...]
```

Qual das opções abaixo gera a lista `par_impar` corretamente:
a) map(lambda x: x%2==0,list(numeros))
b) list(map(lambda x: x%2==0,numeros))
c) map(lambda x: x%2==0,numeros)
d) list(lambda x: x%2==0,map(numeros))

In [5]:
```
numeros = [7, 3, 2, 13, 44, 3, 30, 47, 28, 10, 4, 17, 7, 37, 21, 32, 44, 2, 33, 9, 26,
           9, 29, 9, 49, 11, 8, 42, 26, 23, 17, 16, 37, 26, 19, 26, 8, 27, 15, 10, 31,
           41, 42, 10, 4, 9, 7, 18, 44, 12]

par_impar = list(map(lambda x: x%2==0,numeros))
print(par_impar)
```

```
[False, False, True, False, True, False, True, False, True, True, True, False, False, False, False, True, True, True, False, False, True, False, False, False, False, True, True, True, False, False, True, False, True, False, True, True, False, False, True, False, False, True, True, True, False, False, True, True]
```

Exercício 2)

Considere a lista `numeros` abaixo. Utilize o comando filter para gerar um nova lista `impares` contendo somente os elementos de `numeros` que são ímpares. A lista resultante será:
a) [7, 3, 13, 3, 47, 7, 21, 9, 29, 49, 11, 19, 41, 9]
b) [7, 3, 13, 3, 7, 17, 21, 9, 39, 49, 11, 19, 41, 9]
c) [7, 3, 13, 3, 7, 7, 29, 21, 31, 43, 11, 29, 41, 9]
d) [7, 3, 13, 3, 3, 19, 23, 11, 39, 49, 11, 19, 41, 9]

In [4]:
```
numeros = [7, 3, 2, 13, 44, 3, 30, 47, 28, 10, 4, 12, 7, 32, 21, 32, 44, 2, 36,
9, 26,
           6, 29, 36, 49, 11, 8, 42, 26, 20, 6, 16, 38, 26, 19, 26, 8, 22, 14,
10, 30,
           41, 42, 10, 4, 9, 2, 18, 44, 12]

impares = list(filter(lambda x: x%2!=0,numeros))
print(impares)
```

```
[7, 3, 13, 3, 47, 7, 21, 9, 29, 49, 11, 19, 41, 9]
```

Exercício 3)

Considere a lista `numeros` abaixo. Utilize o comando reduce para somar todos os elementos da lista. O resultado da soma será:
a) 1144
b) 1141
c) 1041
c) 1044

In [8]:
```
from functools import reduce

numeros = [7, 3, 2, 13, 44, 3, 30, 47, 28, 10, 4, 17, 7, 37, 21, 32, 44, 2, 33,
9, 26,
           9, 29, 9, 49, 11, 8, 42, 26, 23, 17, 16, 37, 26, 19, 26, 8, 27, 15,
10, 31,
           41, 42, 10, 4, 9, 7, 18, 44, 12]

soma = reduce(lambda x,y: x+y,numeros)
print(soma)
```

```
1044
```

Exercício 4)

Considere a lista `numeros` abaixo. Utilize uma combinação dos comandos reduce e filter para somar todos os elementos da lista que são pares. O código que gera a soma corretamente será:
a) list(reduce(lambda x,y: x+y,list(filter(lambda x: x%2==0,numeros))))
b) reduce(lambda x: x%2==0,list(filter(lambda x,y: x+y,numeros)))
c) reduce(lambda x,y: x+y,filter(lambda x: x%2==0,numeros))
d) reduce(lambda x,y: x+y,list(filter(lambda x: x%2==0,numeros)))

In [9]:
```
from functools import reduce

numeros = [7, 3, 2, 13, 44, 3, 30, 47, 28, 10, 4, 17, 7, 37, 21, 32, 44, 2, 33,
9, 26,
           9, 29, 9, 49, 11, 8, 42, 26, 23, 17, 16, 37, 26, 19, 26, 8, 27, 15,
10, 31,
           41, 42, 10, 4, 9, 7, 18, 44, 12]

soma_pares = reduce(lambda x,y: x+y,list(filter(lambda x: x%2==0,numeros)))
print(soma_pares)
```

```
514
```

Exercício 5)

Assinale a alternativa que melhor explica o que significa `x e i` na declaração `x(i)` que está dentro do comando map no código abaixo:

a) `x` corresponde a lista de números [0,1,2,3,4] e `i` é um dos elementos da lista.

b) `x` corresponde a lista `func e i` é um dos elementos desta lista.

c) `x` corresponde a um dos elementos da lista `func` e `i` é um número da lista [0,1,2,3,4].

d) `x` corresponde a `func e i` é um dos elementos de `func` .

In [15]:
```
def quadrado(x):
    return(x**2)

def cubo(x):
    return(x**3)

funcs = [quadrado, cubo]
for i in range(5):
    res = map(lambda x: x(i),funcs)
    print(list(res))
```

```
[0, 0]
[1, 1]
[4, 8]
[9, 27]
[16, 64]
```