

## 6.2 Funções de ordem superior




1. A função `somatorio` apresentada no livro

```
def somatorio(l_inf, l_sup, calc_termo, prox):
    soma = 0
    while l_inf <= l_sup:
        soma = soma + calc_termo(l_inf)
        l_inf = prox(l_inf)
    return soma
```

é apenas a mais simples de um vasto número de abstracções semelhantes que podem ser capturadas por funções de ordem superior. Por exemplo, podemos usar a função `somatorio` para somar os quadrados dos múltiplos de 3 entre 9 e 21:

```
somatorio(9, 21 lambda x: x ** 2, lambda x: x + 3)
```

- (a) Diga o que fazem as seguintes utilizações dessa função:

- i. `somatorio(4, 500, lambda x: x, lambda x: x + 1)` 
- ii. `somatorio(5, 500, lambda x: x * x, lambda x: x + 5)` 
- iii. `somatorio(1,`   
`5,`  
`lambda x: somatorio(1,`  
`x,`  
`lambda x: x,`  
`lambda x: x+1),`  
`lambda x: x+1)`

- (b) Defina uma função `piatorio` que calcula o produto dos termos de uma função entre dois limites especificados.
- (c) Mostre como definir o `factorial` em termos da utilização da função `piatorio`.

2. Usando recursão, defina os seguintes funcionais sobre listas:

- (a) `filtra(tst, lst)` que devolve uma lista obtida a partir de `lst` mas que apenas contém os elementos que satisfazem o predicado `tst`.
- (b) `transforma(fn, lst)` que devolve uma lista obtida a partir de `lst` cujos elementos correspondem à aplicação da função `fn` aos elementos de `lst`.
- (c) `acumula(fn, lst)` que devolve o valor obtido da aplicação da função `fn` a todos os elementos de `lst`.

3. Usando alguns ou todos os funcionais sobre listas da pergunta anterior, escreva a função `soma_quadrados_impares`, que recebe uma lista de inteiros e devolva a soma dos quadrados dos seus elementos ímpares. A

sua função deve conter apenas uma instrução, a instrução `return`. Não é necessário validar os dados de entrada. Por exemplo:

```
soma_quadrados_impares([1, 2, 3, 4, 5, 6])
35
```

4. Usando funcionais sobre listas, escreva uma função `todos_lista` que recebe uma lista e um predicado unário, e devolve verdadeiro caso todos os elementos da lista satisfaçam o predicado e falso em caso contrário. O corpo da sua função apenas pode ter uma instrução `return`. Por exemplo,

```
>>> todos_lista([4, 5, 6], lambda x: x > 5)
False
>>> todos_lista([4, 5, 6], lambda x: x >= 4)
True
```

5. Considere a seguinte função que recebe uma operação binária, `op` e um inteiro, como argumentos:

```
def concentra(op, num):
    if num < 10:
        return num
    else:
        return op(num % 10, concentra(op, num // 10))
```

Utilize esta função para escrever a função `produto` que recebe um número inteiro e devolve o produto dos seus algarismos. O corpo da sua função apenas pode ter uma instrução `return`. Por exemplo,

```
>>> produto(24)
8
```

6. Considere a função `muda` que recebe uma função unária como argumento, `fn`, e um número inteiro. A função `muda` utiliza a função `num_digitos` como função auxiliar:

```
def muda(fn, num):
    if num < 10:
        return fn(num)
    else:
        nn = fn(num % 10)
        return nn + 10 ** num_digitos(num) * muda(fn, num // 10)

def num_digitos(n):
    if n < 10:
        return 1
    else:
        return 1 + num_digitos(n // 10)
```

Utilize esta função para escrever a função `soma_dois` que recebe um número e devolve o número que se obtém somando dois a cada um dos seus dígitos (um dígito que seja superior a 7 transforma-se em dois dígitos). O corpo da sua função apenas pode ter uma instrução `return`. Por exemplo,

```
>>> soma_dois(457)
679
>>> soma_dois(986)
11108
```

7. Usando funcionais sobre listas, escreva a função `junta_listas`, que recebe uma lista de listas e devolve a lista resultante de juntar as listas da lista recebida. A sua função deve conter apenas uma instrução, a instrução `return`. Por exemplo:

```
>>> junta_listas([[1, 2], [[3]], [4, 5]])
[1, 2, [3], 4, 5]
```

8. Escreva uma função de ordem superior, `nenhum_p`, que recebe um número inteiro positivo `n` e um predicado unário `p`, e devolve verdadeiro se nenhum inteiro positivo menor ou igual a `n` satisfaz `p` e falso no caso contrário. Por exemplo,

```
>>> nenhum_p(87, lambda x: x % 100 == 0)
True
>>> nenhum_p(187, lambda x: x % 100 == 0)
False
```

9. Usando funcionais sobre listas, escreva a função `soma_quadrados`, que recebe um inteiro positivo `n` e devolve a soma

$$1^2 + 2^2 + 3^2 + \dots + n^2$$

A sua função deve conter apenas uma instrução, a instrução `return`.

10. Usando funcionais sobre listas, escreva a função `lista_digitos`, que recebe um inteiro positivo `n` e devolve a lista cujos elementos são os dígitos de `n`. A sua função deve conter apenas uma instrução, a instrução `return`. SUGESTÃO: transforme o número numa cadeia de caracteres. Por exemplo:

```
>>> lista_digitos(123)
[1, 2, 3]
```