

Fundamentos da Programação LEIC/LETI

Funções de ordem superior

Funções Anônimas (Lambda)

Aula 24

Alberto Abad, Tagus Park, IST, 2021-22

Funções de ordem superior

Funções de ordem superior

- Em aulas anteriores vimos que as funções permitem-nos abstrair algoritmos e procedimentos de cálculo (abstracção procedimental).
- Em Python, tal como nas linguagens puramente funcionais, as funções são entidades de primeira ordem/classe (*first class*):
 - Podemos nomear, utilizar como parâmetro e retornar como valor.
- Isto significa que podemos expressar certos padrões de computação geral a través de funções que manipulam outras funções, conheçios como **funções de ordem superior**:
 - Funções como parâmetros:
 - Funções como métodos gerais (hoje e quarta-feira)
 - Funcionais sobre listas (amanhã)
 - Funções como valor (quarta-feira)

Funções de ordem superior

Funções como parâmetros - Exemplo

```
def soma_naturais(l_inf, l_sup):
    resultado = 0
    for x in range(l_inf, l_sup + 1):
        resultado += x
    return resultado

def soma_quadrado(l_inf, l_sup):
    resultado = 0
    for x in range(l_inf, l_sup + 1):
        resultado += x*x
    return resultado
```

- Qual é o padrão comum?
- Será que podemos abstrair esse padrão comum?

Funções de ordem superior

Funções como parâmetros - Exemplo

- As duas funções `soma_naturais` e `soma_quadrado` diferem apenas na forma como o termo do somatório é calculado, ou seja, a função f :

$$\sum_{n=l_{inf}}^{l_{sup}} f(n) = f(l_{inf}) + f(l_{inf} + 1) + \dots + f(l_{sup})$$

- Por serem as funções entidades de primeira ordem, podemos passara f por parâmetro:

```
In [53]: # Completar com for
def somatorio(l_inf, l_sup, fun):
    resultado = 0
    for x in range(l_inf, l_sup + 1):
        resultado += fun(x)
    return resultado

# Completar com while
def somatorio2(l_inf, l_sup, fun):
    resultado = 0
    i = l_inf
    while i <= l_sup:
        resultado += fun(i)
        i = i + 1
    return resultado
```

Funções de ordem superior

Funções como parâmetros - Exemplo

```
In [113]: # Completar
def quadrado(x):
    return x*x

# Completar
def identidade(x):
    return x

def inv_quadrado(x):
    return 1/(x*x)

# Completar
def somatorio(l_inf, l_sup, fun):
    resultado = 0
    i = l_inf
    while i <= l_sup:
        resultado += fun(i)
        i = i + 1
    return resultado

somatorio(1, 10, inv_quadrado)
```

Out[113]: 1.5497677311665408

Funções de ordem superior

Funções como parâmetros - Exemplo

- E se queremos ter algo mais abstracto?
- Por exemplo, podemos querer avançar o somatório com um passo diferente de somar uma unidade...

```
In [116]: def incl(x):
           return x+1

           def inc2(x):
               return x+2

           def somatorio(l_inf, l_sup, fun_trans, fun_step):
               resultado = 0
               i = l_inf
               while i <= l_sup:
                   resultado += fun_trans(i)
                   i = fun_step(i)
               return resultado

           somatorio(1, 10, quadrado, inc2)
```

Out[116]: 165

Funções de ordem superior

Funções Lambda (funções anónimas)

- O cálculo **lambda** é um modelo de computação universal inventado pelo matemático Alonzo Church em 1941 e que serviu de inspiração a várias linguagens de programação.
 - O cálculo lambda permite-nos modelar funções, e.g.
$$\lambda x. x + 3$$
 - Para avaliar uma função em cálculo lambda escreve-se em geral:
$$(\lambda x. x + 3)3$$
- Em Python, existe a possibilidade de definir funções anónimas recorrendo precisamente a uma notação inspirada no cálculo lambda, em BNF:

<função anónima> ::= lambda <parâmetros formais>: <expressão>

https://en.wikipedia.org/wiki/Lambda_calculus (https://en.wikipedia.org/wiki/Lambda_calculus)

Funções de ordem superior

Funções Lambda (funções anónimas) - Exemplos

- Definição:

```
lambda x : x+3
```

```
lambda x : x*x
```

```
lambda x,y : x+y
```

```
lambda x : 2*x if x%2!=0 else x
```

- Avaliação:

```
(lambda x : x + 3)(3)
```

In [120]:

Out[120]: 5

Funções de ordem superior

Funções Lambda (funções anónimas)

- E para que podem ser úteis estas *funções anónimas*?
- Por exemplo, para definir as funções utilizadas como parâmetros em funções de ordem superior:

```
# soma naturais com incrementos de um em um  
somatorio(1, 10, ???, ???)
```

```
# soma quadrados com incrementos de 1 em 1  
somatorio(1, 10, ???, ???)
```

```
# soma_inv_quadrados_de_dois_em_dois!?!?  
somatorio(1, 10, ???, ???)
```

```
In [126]: def somatorio(l_inf, l_sup, fun_trans, fun_inc):
           resultado = 0
           n = l_inf
           while n <= l_sup:
               resultado += fun_trans(n)
               n = fun_inc(n)
           return resultado

           # soma naturais com incrementos de um em um
           somatorio(1, 10, lambda x : x, lambda x : x+1)

           # soma quadrados com incrementos de 1 em 1
           somatorio(1, 10, lambda x: x*x, lambda x : x+1)

           # soma_inv_quadrados_de_dois_em_dois!?!?
           somatorio(1, 10, lambda x:1/(x*x), lambda x:x+2)
```

Out[126]: 55

Funções de ordem superior

Ordenação e Funções Lambda

Exemplo

- Problema do *checksum* da cifra do Projeto 1:
 - Ordenar por ocorrências primeiro (decrecente)
 - Empates resolvidos em ordem alfabética (crescente)

```
>>> t = (4, 'a'), (7, 'b'), (1, 'c'), (1, 'd'), (1, 'e')
>>> sorted(t)
[(1, 'c'), (1, 'd'), (1, 'e'), (4, 'a'), (7, 'b')]
>>> sorted(t, reverse=True)
[(7, 'b'), (4, 'a'), (1, 'e'), (1, 'd'), (1, 'c')]
```

- Podemos utilizar uma função para customizar a ordem do sort!!

```
>>> help(sorted)
sorted(iterable, /, *, key=None, reverse=False)
Return a new list containing all items from the iterable in ascending order.
```

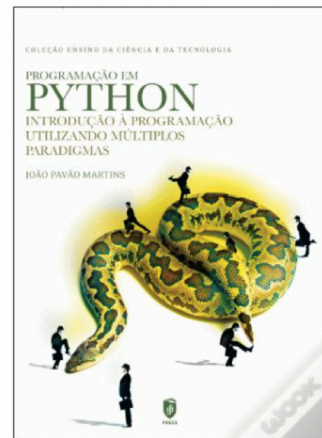
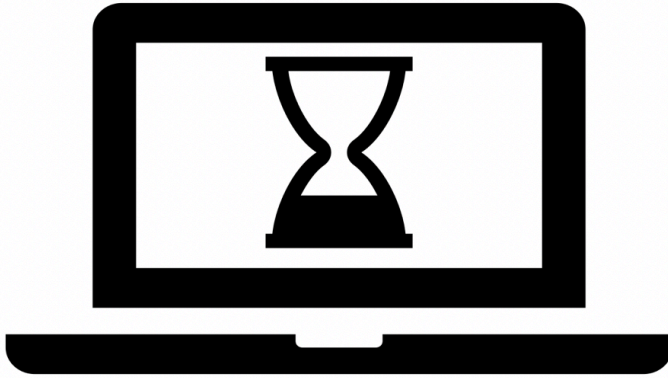
A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.

In []:

Funções de ordem superior

Tarefas próxima aula

- Estudar matéria de hoje:
 - Completar exemplos
- Amanhã nas teóricas **funcionais sobre listas**
- Primeira aula laboratorial da semana --> **Ficha 5** sobre ficheiros e recursão



In []: