Fundamentos da Programação

Funções de Ordem Superior: Funções como Valor

Aula 26

José Monteiro

(slides adaptados do Prof. Alberto Abad)

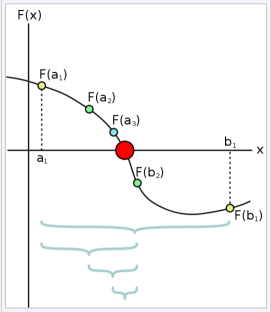
Funções de Ordem Superior

- Em aulas anteriores vimos que as funções permitem-nos abstrair algoritmos e procedimentos de cálculo (*abstração 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 através de funções que manipulam outras funções, conhecidas por **funções de ordem superior**:
  - Funções como parâmetros:
    - Funções como métodos gerais (2ª feira)
    - Funcionais sobre listas (3ª feira)
  - Funções como valor (hoje)

Funções como Parâmetros

Exemplo do Método da Bissecção

- O método da bissecção (baseado no Teorema Bolzano) permite-nos obter a raiz de uma função contínua $f(x)$ situada no intervalo $[a, b]$, sempre que $f(a) <= 0 <= f(b)$ ou $f(b) <= 0 <= f(a)$:



A few steps of the bisection method applied over the starting range [a_1;b_1]. The bigger red dot is the root Wipe of the function.

Funções como Parâmetros

Funções como parâmetros - Exemplo do Método da Bissecção

In [53]:

```
def metodo_bisseccao(f, a, b):
    # Vesão iterativa, pensar em recursiva
    def aproxima_raiz(f, a, b):
        m = (a + b)/2
        while (abs(f(m)) > 1e-9):
            if f(m) > 0:
                a = m
            else:
                b = m
            m = (a + b)/2
        return m

    x = f(a)
    y = f(b)
    if y < 0 < x:
        return aproxima_raiz(f, a, b)
    elif x < 0 < y:
        return aproxima_raiz(f, b, a)
    else:
        raise ValueError("metodo_bisseccao: sig(f(a)) == sig(f(b))!?")


print(metodo_bisseccao(lambda x:x**3 - 2*x, -1, -10))
from math import sqrt
print(sqrt(2))

from math import sin, pi
print(metodo_bisseccao(sin, 2, 4))
print(pi)
```

```
-1.4142135622678325
1.4142135623730951
3.1415926534682512
3.141592653589793
```

Funções como Valor de Funções

Exemplo - Potência geral

- As funções também podem produzir/retornar valores que são funções.

In [112…

```
def make_power_of(n):
    def f_auxiliar(x):
        return x**n

    return f_auxiliar


quadrado = make_power_of(2)
print(quadrado(3))
cubo = make_power_of(3)
print(cubo(2))
print(quadrado(3))
```

9
8
9

- Reparem que neste exemplo o valor do expoente está ligado à função devolvida mesmo após o fim da chamada a `make_power_of` .
- Este tipo de técnica em que uma função mantém valores de *scopes* onde estava encapsulada não estando estes já presentes em memória, em Python e em programação funcional, é conhecida como **closure**.

Funções como Valor de Funções

Exemplo - Cálculo da derivada

- As funções também podem produzir/retornar valores que são funções.
- Consideremos o cálculo da derivada de uma função de variável real f.
  - Por definição:

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a} \tag{1}$$

  - Substituindo $h = x - a$,

$$f'(a) = \lim_{h \to 0} \frac{f(a + h) - f(a)}{h} \tag{2}$$

  - Se dx for um número suficientemente pequeno, podemos considerar a seguinte aproximação:

$$f'(a) \approx \frac{f(a + dx) - f(a)}{dx} \tag{3}$$

Funções como Valor de Funções

Exemplo - Cálculo da derivada

- Definamos primeiro a função de derivada num ponto:

$$f'(a) \approx \frac{f(a + dx) - f(a)}{dx} \tag{4}$$

In [64]:
```
def derivada_num_ponto(f, a):
    delta = 1e-7
    return (f(a+delta) - f(a)) / delta

derivada_num_ponto(lambda x : x**2, 10)
```

Out[64]:
```
19.999999949504854
```

Funções como Valor de Funções

Exemplo - Cálculo da derivada

- Podemos no entanto definir a função de **ordem superior** que retorna a derivada de f da seguinte forma (utilizando funções internas):

In [117…
```
def derivada(f):
    def derivada_num_ponto(a):
        delta = 1e-7
        return (f(a+delta) - f(a)) / delta

    return derivada_num_ponto


df1=derivada(lambda x: x*x*x +3*x - 1)
df1(4)
df1(3)
```

Out[117…
```
30.000000847962838
```

Funções como Valor de Funções

Exemplo - Cálculo da derivada

- Podemos definir a mesma função utilizando funções *lambda* :

In [45]:
```
def derivada_lambda(f):
    delta = 1e-9
    return lambda x:(f(x+delta) - f(x))/delta

g = derivada_lambda(lambda x: x*x)
g(3)
```

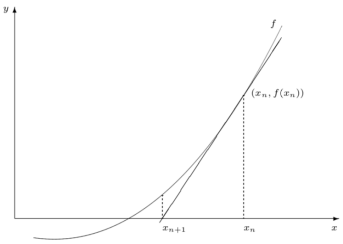
Out[45]:
```
6.000000496442226
```

Funções como Valor de Funções

Exemplo - Método de Newton

- Método para determinar raízes de funções diferenciáveis:
  - Partir de uma aproximação, x_n, para a raiz de uma função f,
  - Calcular, nova aproximação: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- A função matemática que calcula uma nova aproximação é chamada transformada de Newton:

$$t_{Newton}(x) = x - \frac{f(x)}{f'(x)} \tag{5}$$



In [46]:
```
def transformada_newton(f):
    def t_n(x):
        return x - f(x)/derivada(f)(x)
    return t_n

transformada_newton(lambda x:x*x)(-1)
```

Out[46]:
```
-0.4999999858590338
```


Funções como Valor de Funções

Exemplo - Método de Newton

In [47]:

```
def calcula_raizes(f, palpite):
    def bom_palpite(x):
        return abs(x) < 1e-9

    tf_N = transformada_newton(f)

    while not bom_palpite(f(palpite)):
        palpite = tf_N(palpite)

    return palpite


calcula_raizes(lambda x : x * x * x - 2 * x, 1)

# from math import sin
# calcula_raizes(sin, 2)
```

Out[47]:

```
1.4142135623735668
```

In [48]:

```
print(calcula_raizes(lambda x : x * x * x - 2 * x, -5))
%timeit -n 10 calcula_raizes(lambda x : x * x * x - 2 * x, 1)

print(metodo_bisseccao(lambda x:x**3 - 2*x, -1, -10))
%timeit -n 10 metodo_bisseccao(lambda x:x**3 - 2*x, -1, -10)
```

```
-1.4142135623730958
8.45 µs ± 218 ns per loop (mean ± std. dev. of 7 runs, 10 loops each)
-1.4142135622678325
20.5 µs ± 271 ns per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Programação Funcional

Tarefas Próxima Aula

- Estudar matéria de funções de ordem superior
  - **Ficha 6** inclui **funções de ordem superior**

- Próxima aula (última aula teórica!)
  - Tópicos sobre Python
  - Perspetiva sobre a próximas cadeiras do curso