

Funções e Bibliotecas

Carlos Tavares

September 29, 2024

Funções em Python i

Para melhor compreender a aplicação de funções, comecemos pela solução a um problema artificial. Suponha que a equação $s(t)$, descreve a posição de um corpo em queda livre ao longo do tempo:

$$s(t) = s_0 + v_0 t - \frac{1}{2} g t^2, \text{ onde } g = 9,81, r \text{ é a resistência do ar e } t \text{ é o tempo}$$

Escreva um programa que dados dois coeficientes a , b e um tempo t , seja capaz de calcular a posição de dois objectos A e B, respectivos aos coeficientes a e b , no tempo t .

A resistência do ar r corresponde a $\frac{2}{\{a \text{ or } b\}^2}$. Finalmente, considere que o objecto B se está a afastar a $3mt/s$ do objecto A no eixo x .

Podemos fazer melhor?

Funções i

Sim, com funções!

Uma função é um processo, com um input x e um output $f(x)$



i.e. um mapa de um conjunto de partida para um conjunto de chegada:


$$f : A \rightarrow B$$

O Python permite a definição de funções:

```
def function_name (arguments): # arguments  $\Leftrightarrow \vec{x} \equiv x_1, x_2, \dots, x_n$   
    (...) # definição do processo  
    return ... # devolução do resultado  $f(\vec{x})$ 
```

Identação


As instruções que pertencem à função têm de estar "identadas", é assim que o Python determina a que estrutura de controlo pertence uma instrução

 - Conjunto de espaços ou "tabs".
indentação

Funções iii

Exemplo:

```
def sum (a, b):
```

```
     return a + b  
    identação
```

```
def a_times_b_plus_c (a, b, c):
```

```
    return a * sum (b,c)
```

As funções só são executadas quando invocadas

```
a = 2
```

```
b = 3
```

```
c = 4
```

```
s = sum (a, b)
```

```
s_m = a_times_b_plus_c (a, b, c)
```

```
print (s, s_m)
```

Como poderemos resolver o problema da queda de objectos utilizando funções?

Mais sobre funções i

Uma função pode ser:

- Pura, sem efeitos "secundários";
- "Impura", com efeitos secundários.

Uma função pura

def sum (a,b):

 return a + b

Uma função com "efeitos secundários":

x = 5

def change_state (a, b):

 global x = 6

Mais sobre funções ii

Variáveis globais vs variáveis locais

No programa acima x é uma variável global, que pode ser utilizada por todas as funções do programa...

$c = 2$ # Isto é uma variável global

def f ():

$v = 0$ # Isto é uma variável local

$v = c + 1$

 print (v)

~~$v = 0$~~

... e v é local e "morre" quando a função termina e não pode ser utilizada fora da função.

Mais sobre funções iii

É possível utilizar o conteúdo de uma variável dentro de uma função, desde que não modifique a expressão

```
c = 1
```

```
def f ():  
    print (c)
```

Para modificar uma variável global dentro de uma função é necessário declará-la como global

```
c = 1
```

```
def f ():  
    global c = c * 5  
  
    print (c)
```

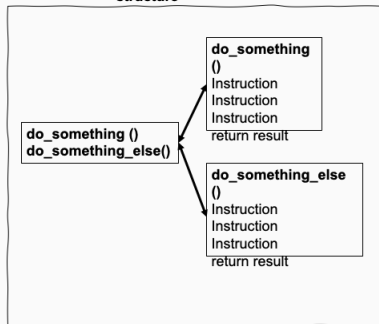
Mais sobre funções iv

As funções transformam o Python numa linguagem de programação estruturada

Sequential structure

```
Instruction  
Instruction  
Instruction  
Instruction  
Instruction  
(...)
```

Structured structure



Mais sobre funções v

Um programa em Python executa todas as instruções que façam parte do programa principal (sem indentação)

Ou

Executa a função main:

Run the main function

```
if __name__ == "__main__":  
    main()
```

Advantages of using functions i

Modularidade - Separação do código em componentes

- Separação de problemas - Permite pensar num problema em sub-problemas;
- Encapsulamento - Isolamento dos efeitos do código
- Testabilidade - É mais fácil testar um componente de cada vez do que testá-los todos ao mesmo tempo.

Abstração - Esconder detalhes desnecessários

- Abordagem dos problema por "camadas".

Advantages of using functions ii

Outras vantagens:

Reutilização do código - É possível escrever programas com menos código

(Regra do polegar) - Menos código é melhor que mais código para as mesmas coisas.

Legibilidade e organização do código - Melhora a legibilidade do código.

Parameterização - Melhora a manageabilidade do código.

Outro problema artificial:

Consideremos que existe um sistema de controlo no avião que obedece às seguintes regras (para leitura dos dados do sensor pode utilizar a função input)

- Leitura do sensor 1: dos dados lidos do sensor, calcular o quadrado do número e dividir por dois;
- Leitura do sensor 2: dos dados lidos do sensor, x , fazer $5^x \bmod 36$;
- Leitura do sensor 3: dos dados lidos do sensor, y , retornar y ;
- Decisão: se a média dos dois primeiros for maior que o terceiro ou a média dos dois últimos for menor que o primeiro: realizar ação de controlo.

Bibliotecas

O Python permite a criação de bibliotecas (pacotes de funções), i.e. ficheiros acessíveis a terceiros com conjuntos de funções reutilizáveis.

Para utilizar as funções de um pacote não é necessário ter acesso ao código, apenas à sua definição.

Existem centenas de pacotes disponíveis com funções para os mais diversos usos.

<https://pypi.org>

Talvez seja a existência de tantos pacotes diferentes que faz do Python tão popular.

Algumas bibliotecas famosas:

Mathematics	numpy, sympy
Optimization	scipy
Data analysis	pandas
Visualization	matplotlib

Como utilizar pacotes em Python? i

É necessário invocar os pacotes em Python:

1. **instrução import**

» `import package_name` # ex. `numpy`, `sympy`

2. **instrução import as**

» `import package_name as p` # `numpy`, `sympy`

3. **Instrução from ... import**

» `from package_name import function`

`# from numpy import cos`

Como utilizar pacotes em Python? ii

Isto só funciona se os pacotes forem instalados. Como instalar pacotes em Python? Utilizar a instrução pip!

» **pip install package_name**

O pip é o gestor de pacotes dentro da vossa instalação do Python.

O pacote numpy i

O numpy disponibiliza uma série de funções matemáticas desde funções trigonométricas, funções com matrizes, etc. É o pacote mais utilizado para matemática em Python.

Importação:

» `import numpy`

Usualmente faz-se

» `import numpy as np`

O pacote random i

O python disponibiliza um pacote para gerir **números pseudo-aleatórios**

- O que significa pseudo-aleatório? Na verdade a função geradora de números aleatórios é determinista!

Documentação completa:

<https://docs.python.org/3/library/random.html>

Importação do pacote para o nosso programa

» **import random**

A função geradora de números aleatórios pode ser "controlada"

» **random.seed (number)**

O pacote random ii

Funções importantes

Gerar um número inteiro aleatório

» **random.randint (a, b)**

Gerar um número real aleatório entre 0 e 1

» **random.random ()**

Gerar um número real num dado intervalo

» **random.uniform (a, b)**

Gerar um número aleatório **sem reposição!**

» **random.sample (populacao, n)**

Outras funções disponível para gerar dados aleatórios a partir de populações:

» **random.choice** (['stone', 'scisor', 'paper'])

» **random.shuffle** (['king', 'queen', 'ace'])

Questões?