Ficha 1b - Programação Orientada a Objectos

Jorge Gustavo Rocha

May 30, 2020

1 Programação Orientada a Objetos em Python

Jorge Gustavo RochaDepartamento de Informática, Universidade do Minho27 de abril de 2020 O Python é uma linguagem de programação que complementa os conceitos da Programação Imperativa com a Programação Orientada a Objectos.

Esta ficha introduz apenas conceitos e não tem exercícios.

1.1 Funções

Em Python estamos recorrentemente a usar funções predefinidas, como `print()`, `type()`, `len()`, etc ou mesmo a definir novas funções. As funções são uma peça fundamental na **Programação Imperativa** para construir programas, pois permitem agregar uma sequência de instruções muito utilizada numa função, que depois se usa quantas vezes for preciso.

Na **Programação Imperativa** tipicamente usam-se funções e módulos (que nos permite dividir os programas em blocos que se podem reutilizar) para construir programas mais complexos.

1.2 Classes e métodos

O Python é também uma Linguagem Orientada a Objectos. Para além das suas características da programação imperativa, permite desenvolver programas à custa da definição de classes e da criação de **objectos** dessas classes. As classes são uma forma de juntar dados e código numa única unidade. Enquanto que uma função agrega apenas código que se pode reutilizar, as classes permitem reutilizar dados e código.

É complicado começar a criar classes e a desenvolver programas orientados a objectos, mas não é difícil perceber um exemplo. Vamos tentar ilustrar a programação orientada a objectos com uma classe `Cão`.

Vamos fazer esta introdução para percebermos que existem **métodos** para além das **funções**, já que terá que usar muitos destes métodos, mesmo nos programas mais simples. Os métodos são muito utilizados em Python.

[1]:
```
from datetime import date

class Cão:

    # variável da classe (comum a todas as instâncias)
    familia = 'Canis lupus familiaris'
```

```
    def __init__(self, nome, nascimento):
        # variáveis próprias de cada instância
        self.nome = nome
        self.nascimento = nascimento

    def ladra(self):
        print("Au, au!")

    def idade(self):
        return date.today().year - self.nascimento;
```

Já está criada a classe `Cão`. A classe dá-nos agora a possibilidade de criar objectos (dessa classe). Vamos criar então um programa com dois cães diferentes .

[2]:
```
x = Cão('Artemisa', 2007)
y = Cão('Apolo', 2018)
```

`x` e `y` são **objectos**. Estes objetos tem propriedades (atributos) e métodos associados (código). Vamos exemplificar:

[3]:
```
x.nome
```

[3]:
```
'Artemisa'
```

[4]:
```
y.ladra()
```

```
Au, au!
```

[5]:
```
x.idade()
```

[5]:
```
13
```

[6]:
```
y.familia
```

[6]:
```
'Canis lupus familiaris'
```

1.2.1 Métodos

Vamos focar-nos nos métodos por agora, só para reforçar que estes são ligeiramente diferentes das funções. Enquanto que uma função se invoca através do nome da função, os métodos são sempre invocados no contexto de um objecto. No exemplo anterior, usou-se a sintaxe `y.ladra()` para invocar o método `ladra()`.

[7]:
```
y.ladra()
```

```
Au, au!
```

Este introdução à programação orientada a objectos ajuda a perceber porque é que já encontrou nos exemplos anteriores situações em que se usam métodos em vez de funções.

Um exemplo muito simples é o método `str.format()` que já viu ser utilizado. Se é um método da classe `str`, usa-se no contexto de um objecto dessa classe, da mesma forma que o método `ladra()` se usou no contexto do um objeto da classe `Cão`.

Vamos aplicar o método `str.format()` a um objeto da classe `str`. Começamos por confirmar que o objecto é uma instância da classe `str` (só por curiosidade):

[8]:
```
type("O gasóleo está a {:0.2f} €")
```

[8]:
```
str
```

Invocação do método `str.format()`:

[9]:
```
"O gasóleo está a {:0.2f} €".format(1.214)
```

[9]:
```
'O gasóleo está a 1.21 €'
```

Exercício (resolvido) Use o método `str.upper()` aplicado ao exercício anterior, para apresentar a mensagem em maiúsculas.

[10]:
```
"O gasóleo está a {:0.2f} €".upper()
```

[10]:
```
'O GASÓLEO ESTÁ A {:0.2F} €'
```

Da mesma forma, começará a usar outros métodos da classe `str`, como `str.find()` ou `str.strip()`, por exemplo. Começará também a usar métodos de muitas outras classes. Fica, para já, com a noção de que se a invocação for desta forma, trata-se de um método e não de uma função.

Nota: Complicando as coisas, na verdade o método `ladra()` é uma função da classe `Cão`. Por essa razão, é equivalente usar `y.ladra()` e `Cão.ladra(y)`. Mas isto já são coisas complicadas