Definição de novas funções em Python

April 28, 2020

1 Definição de funções em Python

Jorge Gustavo RochaDepartamento de Informática, Universidade do Minho27 de abril de 2020 Uma forma de resolver problemas complicados é dividi-los em problemas mais pequenos e assim sucessivamente.

Em Informática, este forma de resolver os problemas é muito utilizada.

Em Python, assim como em muitas outras linguagens de programação, criam-se classes, módulos ou funções para resolver os problemas mais pequenos. Os problems reais, mais complexos, são resolvidos por composição dessas classes, módulos ou funções.

Nestes exercícios vamos aprender a definir novas funções.

1.1 Funções predefinidas

Em Python existem muitas funções incluídas na própria linguagem e que já usou.

Por exemplo, as funções `len()` e `sorted()` já foram usadas para calcular o tamanho de uma lista e ordená-la alfabeticamente, como se ilustra nos exemplos seguintes.

[1]:
```
planetas = [ 'Mercúrio', 'Vénus', 'Terra', 'Marte', 'Júpiter', 'Saturno',␣
 ↪'Urano', 'Neptuno']
len(planetas)
```

[1]:
```
8
```

[2]:
```
sorted(planetas)
```

[2]:
```
['Júpiter',
 'Marte',
 'Mercúrio',
 'Neptuno',
 'Saturno',
 'Terra',
 'Urano',
 'Vénus']
```

1.2 Novas funções

A declaração de novas funções tem a seguinte forma:

```
def nome_da_função( parâmetros_de_entrada):
    corpo_da_função
    return parâmetros_de_saída
```

Esta forma diz-nos qual é a *sintaxe* específica do Python que se tem que utilizar. Contudo, para quem está a começar a programar em Python, ajuda mais começar por ver um exemplo concreto.

Vamos começar por definir uma nova função muito simples. A partir desta definição, vamos explicar cada uma das componentes.

Considere a nova função `euro()`, definida da seguinte forma:

[3]:
```
def euro(valor):
    formatado = "{:0.2f} €".format( float(valor) )
    return formatado
```

Temos uma nova função designada `euro()` que recebe um valor e devolve uma string, devidamente formatada, recorrente ao método `str.format` das strings.

Utilização da nova função A nova função `euro()` pode ser utilizada como qualquer outra função predefinida, da seguinte forma:

[4]:
```
euro(1287.4)
```

[4]:
```
'1287.40 €'
```

Outro exemplo de invocação da função `euro()`:

[5]:
```
preco = input()
com_iva = float(preco) * 1.23
print( euro( com_iva ))
```

```
75
92.25 €
```

1.2.1 Anatomia da função `euro()`

A função `euro()` tem quatro componentes:

Nome da função O nome da função pode ser um identificador qualquer que ainda não esteja a ser usado. Geralmente não se usam nomes começados por maiúscula (esses nomes usam-se para definir novas classes). Neste caso usou-se o nome `euro`.

As palavras reservadas em Python não são muitas. Pode obter a lista dos **nomes que não pode usar** para as suas funções com:

[6]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
```

```
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Parâmetros de entrada Esta função tem um parâmetro de entrada. Está à espera de um número. O número pode ser um inteiro ou um real. Se for um inteiro, é transformado em real com `float(valor)`. Se for já um real, a instrução anterior não altera o que já é um real.

Saída A função tem um parâmetro de saída, que é do tipo `str` (uma string). A saída corresponde à formatação adoptada para valores monetários em Euros.

Corpo da função O corpo da função corresponde ao processamento que é feito com os parâmetros de entrada até se conseguirem os valores para os parâmetros de saída. No caso desta função `euro()` o processamento é mesmo muito simples e está todo numa linha apenas:

```
formatado = "{:0.2f} €".format( float(valor) )
```

1.3 Exercícios iniciais

Para cada exercício, comece por definir os parâmetros de entrada e de saída. Se isso não for feito com cuidado, não conseguirá chegar a uma boa solução.

1. Calcule a área de um círculo, sabendo o raio.
2. Escreva uma função que calcule o máximo de três números inteiros.
3. Escreva uma função que calcule o total de segundos utilizados por um atleta numa prova, sabendo apenas o número de horas e minutos gastos.

[1]:
```
import math
```

1.4 Âmbito das variáveis em Python

Dentro das funções pode-se (e deve-se) usar variáveis que só têm existem dentro dessa função. Fora da função não podem ser usadas.

Na função `euro()`usamos uma variável `formatado` que fica com o valor da string que representa o valor monetário. Se tentarmos usar o valor fora da função, não vamos ter acesso ao valor da variável dentro da função.

O seguinte código funciona como esperado:

[2]:
```
phones = 25

def euro(valor):
    formatado = "{:0.2f} €".format( float(valor) )
    return formatado

print( euro(phones) )
```

```
25.00 €
```

Se quisermos usar o valor da variavél `formatado` o que acontece?

[4]:
```
phones = 25

def euro(valor):
    formatado = "{:0.2f} €".format( float(valor) )
    return formatado

print( euro(phones), formatado )
```

```
      ␣
↪---------------------------------------------------------------------------

       NameError                                 Traceback (most recent call␣
↪last)

       <ipython-input-4-79f1d95150eb> in <module>
         5     return formatado
         6
   ----> 7 print( euro(phones), formatado )


       NameError: name 'formatado' is not defined
```

Este mecanismo serve para conter o código das funções independente de umas das outras e independente do código escrito fora das funções.

Pelo contrário, as variáveis que são declaradas fora das funções estão definidas dentro e fora das funções, funcionado como variáveis globais. Veja o seguinte exemplo:

[16]:
```
import emoji

profissão = 'professor'

def viva():
    print('Viva o nosso ' + profissão + emoji.emojize(' :thumbs_up:'))

viva()
```

```
Viva o nosso professor
```

Como estas **variáveis declaradas fora das funções são globais**, podem ser alteradas dentro das funções.

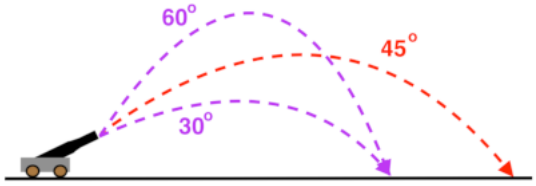
[17]:
```
profissão = 'professor'

def viva():
    profissão = 'médico'
    print('Viva o nosso ' + profissão + emoji.emojize(' :thumbs_up:'))
```

```
viva()
```

```
Viva o nosso médico
```

É um mecanismo bastante flexível, mas é muito perigoso! É perigoso porque depois torna-se difícil saber ao certo onde a variável foi alterada e deixa-se de perceber o que está a acontecer no programa. Regra de ouro: **ao escrever programas, evite a utilização de variáveis globais**.

1.5 Exercício de física



Escreva uma função que determine a distância alcançada por um projéctil que é lançado com uma determinada velocidade inicial e um determinado ângulo em relação ao solo. Pode consultar as fórmulas na Internet ou mesmo ver um vídeo detalhado na Khan Academy.

Radianos e graus As funções trigonométricas estão à espera de valores em radianos. Use a função `math.radians()` para converter o ângulo inicial em radianos (se a função receber um ângulo em graus).

1.5.1 Fórmulas

Dada uma velocidade inicial v_i e um ângulo inicial θ_i, as fórmulas que nos dão o alcance R, o tempo de voo T e a altura máxima h são:

$$R = \frac{v_i^2 \cdot \sin(2 \cdot \theta_i)}{g} \tag{1}$$

$$h = \frac{v_i^2 \cdot \sin(\theta_i)^2}{2g} \tag{2}$$

$$T = \frac{2 \cdot v_i \cdot \sin(\theta_i)}{g} \tag{3}$$

Considera-se a aceleração da gravidade $g = 9.8\ m/s^2$.

Resolução

[2]:
```
import math
```

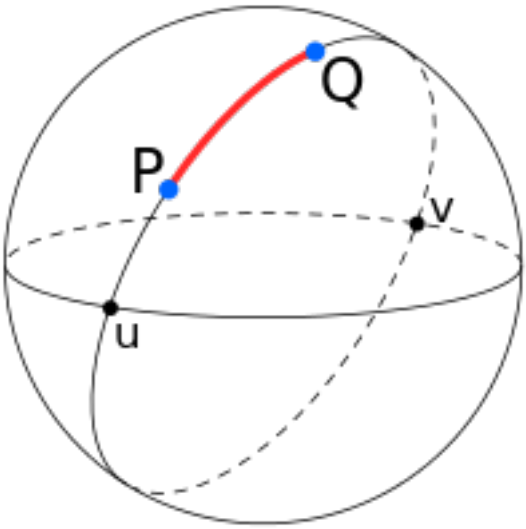
Exemplo de utilização

[11]:
```
vel = input()
angulo = input()
m = alcance( float(vel), float(angulo) )
print( "O seu projéctil alcançou {:0.2f} metros.".format( m ) )
```

```
10
45
O seu projéctil alcançou 10.20 metros.
```

1.6 Exercício complementar

Escreva uma função que calcule a **distância ortodrómica** aproximada entre dois pontos da terra



`P` e `Q`. Os pontos `P` e `Q` são dados pelas respetivas coordenadas referentes à longitude (representada por *lambda*, λ) e latitude (representada por *phi*, ϕ). Vamos usar a fórmula para o este cálculo apresentada na Wikipédia.

Assim sendo, as coordenadas de `P` e `Q` são dadas por:

$$P = \lambda_1, \phi_1$$

$$Q = \lambda_2, \phi_2$$

A distância d é dada por:

$$r = 6378137 \tag{4}$$

$$\Delta\lambda = \lambda_1 - \lambda_2 \tag{5}$$

$$\Delta\sigma = \arccos\big(\sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos(\Delta\lambda)\big) \tag{6}$$

$$d = r \cdot \Delta\sigma \tag{7}$$

Valores para teste Considere para testar a sua função os seguintes valores (apenas para exemplo):

$$P_{\lambda_1,\phi_1} = -8.4542053, 40.5832343$$

$$Q_{\lambda_2,\phi_2} = -8.6533827, 40.6393801$$

Para obter outros valores para `P` e `Q` pode consultar as coordenadas de um ponto no mapa usando o OpenStreetMap ou o Google Maps, entre outros.

Parâmetros de entrada e de saída Comece por definir muito bem os parâmetros de entrada e de saída. Só depois se preocupe com o corpo da função.

Radianos e graus As funções trigonométricas estão à espera de valores em radianos. Use a função `math.radians()` para converter os valores das coordenadas em graus em radianos.

1.6.1 Resolução

[3]:
```
import math
```

1.6.2 Exemplo de utilização

[10]:
```
braga = [41.55054, -8.42646]
guimaraes = [41.44249, -8.29383]

print( "A distância é de {:0.1f} metros".format(dortodromica(braga, guimaraes)␣
 ↪))
```

```
A distância é de 16339.0 metros
```