Programação Python

Aula 02: Funções, lambda e Módulo ‘math’

Prof. Eduardo Corrêa Gonçalves

23/03/2021

Sumário

Funções (1/12)

- **Criando Funções**
  - A palavra reservada **def** é utilizada para a definição de funções.
  - Dentro do corpo usamos **return** para retornar o resultado.

```
def faixa_etaria(idade):
  if (idade < 18): return '<18'
  elif (idade >= 18 and idade < 30): return '18-29'
  elif (idade >= 30 and idade < 40): return '30-39'
  else: return '>=40'

#chamando a função com diferentes valores
a = faixa_etaria(15)
b = faixa_etaria(50)
c = faixa_etaria(35)
print(a); print(b); print(c)
```

```
<18
>=40
30-39
```

Funções (2/12)

- **Criando Funções**
  - No corpo do programa, as funções precisam ser definidas antes de serem chamadas.
  - Você não pode, por exemplo, colocar o comando “faixa_etaria(35)” antes do código que define esta função.
  - Veja que a própria IDE Spyder reclama quando você tenta fazer isso.


```
 8
 9 #chamando a função com diferentes valores para o parâmetro "idade"
10 a = faixa_etaria(18)
11 b = faixa_etaria(50)
12 c = faixa_etaria(35)
13
14 o "faixa_etaria()"
15 ade):
16  :
17 '
18     elif (idade >= 20 and idade < 30) :
19         return '20-29'
20     elif (idade >= 30 and idade < 40) :
21         return '30-39'
22     else :
23         return '>=40'
```

Funções (3/12)

- **Modulos**
  - Há 2 formas de utilizar uma função:
    1. Criá-la e executá-la em um mesmo programa
    2. Defini-la em um **arquivo separado** e importá-la com o comando **import**.

  - **Exemplo – import**:
    - Se você salvar um módulo “minhas_funcoes.py” com o código da função “faixa_etaria()”, poderá importa-lo dessa forma:

```
import minhas_funções

a = minhas_funcoes.faixa_etaria(15)
b = minhas_funcoes.faixa_etaria(50)
c = minhas_funcoes.faixa_etaria(35)
```

Funções (4/12)

- **Parâmetros Opcionais**
  - A função “soma_numeros()” possui três parâmetros, mas o terceiro recebe o valor **None** como default.
    - Com isto, ele se torna opcional.

```
def soma_numeros(x,y,z=None):
  if (z is None):
    return x+y
  else:
    return x+y+z

print(soma_numeros(1, 2))
print(soma_numeros(1, 2, 3))
```

```
>>>
3
6
```

Funções (5/7)

- **Parâmetros com Valor Default**
  - Na função “f_calcula()”, o terceiro parâmetro (“operacao”) possui o valor “+” como default.
    - Se a função for chamada sem a especificação deste 3º parâmetro, o valor “+” será automaticamente adotado.

```
def f_calcula(x,y,operacao='+'):
  if (operacao=='+'): return x+y
  elif (operacao=='-'): return x-y
  elif (operacao=='*'): return x*y
  elif (operacao=='/'): return x/y
  else: return 'operação inválida!'

print(f_calcula(1, 2)) #retorna 1+2 = 3
print(f_calcula(1, 2, '+')) #retorna 1+2 = 3
print(f_calcula(1, 2, '-')) #retorna 1-2 = -1
print(f_calcula(1, 2, '*')) #retorna 1*2 = 2
print(f_calcula(1, 2, '/')) #retorna 1/2 = 0.5
print(f_calcula(1, 2, '.')) #retorna 'operação inválida
```

Funções (6/12)

- **Procedimento**
  - É possível criar uma função que não retorna valor.
    - Basta não usar return.
    - Nesse caso, a função atuará como uma procedure.

```
def cumprimenta(nome):
  print('olá', nome)

cumprimenta('alunos da ENCE!')
```

```
>>> olá alunos da ENCE!
```

Funções (7/12)

- **Valores de parâmetros**
  - Considere uma variável *x* passada como argumento de uma função:
    - Se *x* for de um **tipo primitivo** (int, float, string ou boolean) qualquer alteração em seu conteúdo não será refletida para o programa principal.
    - Se *x* **não** for de um tipo primitivo (ex.: lista, dicionário etc.), a alteração será refletida.

```
def soma_um(numero):
  numero=numero+1
  print("somei um dentro da função: ", numero)

k=100
print('valor original de k:', k)
soma_um(k)
print('Terminou a função e k, na verdade, não mudou:', k)
```

```
valor original de k: 100
somei um dentro da função:  101
Terminou a função e k, na verdade, não mudou: 100
```

Funções (8/12)

- **Valores de parâmetros**
  - Como corrigir?
    - É simples... faça a função retornar um valor!!!

```
def soma_um(numero):
  numero=numero+1
  print("somei um dentro da função: ", numero)
  return numero

k=100
print('valor original de k:', k)
k = soma_um(k)
print('Terminou a função e agora k mudou:', k)
```

```
valor original de k: 100
somei um dentro da função:  101
Terminou a função e agora k mudou: 101
```

Funções (9/12)

- ***args:** recurso que nos permite passar um **número arbitrário de parâmetros** para uma função

```
def pessoa(nome, *args):
  print("- nome (primeiro parâmetro): ", nome)
  print("- características (outros parâmetros): ")
  for arg in args:
    print("\t",arg)

pessoa('Jane','escritora','sagitariana','romântica')
pessoa('John','músico')
```

```
- nome (primeiro parâmetro):  Jane
- características (outros parâmetros):
        escritora
        sagitariana
        romântica

- nome (primeiro parâmetro):  John
- características (outros parâmetros):
        músico
```

Funções (10/12)

- **Funções pré-definidas:** Fazem parte da própria linguagem Python

Numéricas

Função	Descrição
abs(*x*)	retorna o valor absoluto de x;
pow(*x*,*y*)	retorna x elevado a y
round(*x*,*d*)	retorna x arredondado para d casas decimais (neste caso, x deve ser um float)

Strings

Função	Descrição
len(*s*)	retorna o comprimento da string (número de caracteres).
max(*s*)	retorna o maior caractere de *s*, considerando a ordem lexicográfica.
min(*s*)	retorna o menor caractere de *s*, considerando a ordem lexicográfica.

Funções (11/12)

- **Funções pré-definidas:** Fazem parte da própria linguagem Python

Funções de Conversão

Função	Descrição
float(*s*)	Converte a string *s* para um float. A variável *s* deve conter um número válido (inteiro ou real), caso contrário ocorrerá erro.
int(*s*)	Converte a string *s* para um float. A variável *s* deve conter um número inteiro válido, caso contrário ocorrerá erro.
str(*n*)	Convere o número *n* para uma string.

Funções (12/12)

- **Funções pré-definidas – exemplo:**

```
#funções numéricas
n1=100; n2=3.141592653; n3=9.99
print(abs(1000), abs(-500)) #1000 500
print(round(n2,2)) #3.14
print(round(n2), round(n3)) #3 10

#funções de string
s1='python'
print(len(s1)) #6
print(max(s1)) #'y'
print(min(s1)) #'h'

#funções de conversão
s1='5'; s2='9.99'
print(int(s1)) #converteu '5' -> 5
print(float(s2)) #converteu '9.99' -> 9.99
print('PI com 10 digitos é: ' + str(n2))
print('PI com 2 digitos é: ' + str(round(n2,2)))
```

Funções lambda

- **lambda**
  - Notação abreviada que pode ser empregada para definir funções simples (uma expressão, uma linha)
  - Muito popular entre os *pythonistas*!
  - Sintaxe: lambda parâmetros:expressão.

```
soma = lambda x,y: x + y

print(soma(1,2))   #3
print(soma(5,10))  #15
```

Módulo ‘math’ (1/3)

- É um dos (muitos) que fazem parte da ***standard library***.
  - Biblioteca padrão do Python, automaticamente instalada por qualquer distribuição do Python (CPython, Anaconda, WinPython etc.).

- O módulo ‘math’ fornece uma série de funções e constantes matemáticas úteis, como funções de arredondamento, trigonométricas, logarítmicas, etc.
  - Antes de utilizá-lo, você precisa realizar a sua importação da seguinte maneira: **import** math

- *Consulte o “Meu Primeiro Livro de Python”, pag. 50-52 para uma relação com algumas das principais funções do módulo ‘math’*
  - Ou acesse a documentação do Python: https://docs.python.org/3/library/math.html

Módulo ‘math’ (2/3)

- **Módulo math – exemplo 1:**

```
import math

#constante PI
print('PI=',math.pi) #3.141592653589793

#funções de arredondamento
x1 = 5.9;
print('ceil',math.ceil(x1))   #6   (teto)
print('floor',math.floor(x1)) #5   (piso)
print('trunc',math.trunc(x1)) #5   (truncamento)

#logaritmo
x2 = 1024
print('log de',x2,'na base 2: ', math.log2(x2)) #10
```

Módulo ‘math’ (3/3)

- **Funções pré-definidas – exemplo 2:**

```
import math

#imprime tabela com seno, cosseno e tangente
#de 30, 45 e 60
#note que é preciso converter os ângulos para radianos
print('\n')
for angulo_graus in range(30,61,15):
  angulo_radianos = math.radians(angulo_graus)
  print('\n* * * Angulo=',angulo_graus, ' graus')
  print('SENO=',round(math.sin(angulo_radianos),2))
  print('COSSENO=',round(math.cos(angulo_radianos),2))
  print('TANGENTE=',round(math.tan(angulo_radianos),2))
```

Tarefas

(1) – Fazer um programa para calcular y = f(x) + g(x), onde:

$h(x) = x^2 - 16$

f(x) { h(x), se h(x) ≥ 0

{ 1, se h(x) < 0

g(x) { $x^2 + 16$ se f(x) = 0

{ 0, caso contrário

Em seu programa, defina y e h(x) como lambda e f(x) e g(x) como funções.

(2) – Dado um inteiro positivo *n*, criar uma função para determinar o número harmônico H_n definido por:

$$H_n = \sum_{k=1}^{n} \frac{1}{k}$$

Tarefas

(3) –Escreva uma função chamada “mensagem_alien” com 2 parâmetros

- pessoa: nome de uma pessoa para a qual o alienígena dirá uma frase
- tipo_alien: ‘B’ = alienígena bonzinho, ‘M’ = alienígena malvado

A função deverá retornar uma das duas seguintes frases de acordo com o tipo do alien:

- Para o tipo ‘B’, retornar “*pessoa*, eu vim em missão de paz!”
- Para o tipo ‘M’, retornar “*pessoa*, eu vou te abduzir e escravizar!”

Obs.: o símbolo “+” serve como operador de concatenação de strings.

(4) – Crie uma versão recursiva do programa que calcula o número harmônico. Utilize a seguinte regra de recorrência:

$H_0 = 0$

$H_n = H_{n-1} + 1/n$

Referências

- Corrêa, E. (2020). “Meu Primeiro Livro de Python”. V 2.0.0, edubd, 2020. (*capítulo 2*).
  - Disponível em: https://github.com/edubd/meu_primeiro_livro_de_python