Programação II

FUNÇÕES – PARTE II

Argumentos default

- É possível dar valores default a argumentos
- Se o chamador não especificar valores para esses argumentos, os defaults são usados
- Formato: def nome (arg1=default1, ..., argN=defaultN)

- Se apenas alguns argumentos têm default, esses devem ser os últimos

Exemplo

```
1  def f(nome,saudacao="Oi",pontuacao="!!"):
2      return saudacao+","+nome+pontuacao
3  print (f("Joao"))
4  #Oi,Joao!!
5  print (f("Joao","Parabens"))
6  #Parabens,Joao!!
7  print(f("Joao","Ah","..."))
8  #Ah,Joao...
```

Passando argumentos com nomes

- É possível passar os argumentos sem empregar a ordem de definição desde que se no meie cada valor passado com o nome do argumento correspondente

- Ex.:

```
1  def f(nome,saudacao="Oi",pontuacao="!!"): return saudacao+","+nome+pontuacao
2
3  print(f(saudacao="Valeu",nome="Joao"))
4  #Valeu,Joao!!
```

Funções que aceitam qualquer quantidade de argumentos

- Funções que aceitam qualquer quantidade de argumentos
- Utilize *
- Por exemplo:

```
1  def media(primeiro, *outros):
2      return (primeiro + sum(outros)) / (1 + len(outros))
3
4  # Exemplos de uso
5  print(media(1, 2))
6  print(media(1, 2, 3, 4))
```

Funções que aceitam somente argumentos nomeados

- Para aceitar somente argumentos de forma nomeada coloque os argumentos nomeados após um argumento *
- Exemplo:

```
1  def recebe(tamanhomax, *, bloco):
2  # Recebe uma mensagem
3      print(tamanhomax)
4  #recebe(1024, True)        #TypeError - ERRADO
5  recebe(1024, bloco=True)  #OK - CORRETO
```

Funções que retornam múltiplos valores

- Para retornar múltiplos valores de uma função, basta retornar uma tupla.
- Por exemplo:

```
>>> def myfun():
...     return  1, 2, 3
...
>>> a, b, c = myfun()
>>>a
1
>>>b
2
>>>c
3
>>> x = myfun()
>>> x
(1, 2, 3)
```

Documentando funções

- Ao invés de usar comentários para descrever o que uma função, é mais vantajoso usar *docstrings*
Uma constante string escrita logo após o cabeçalho da função (comando def)
Permite o acesso à documentação a partir do interpretador, usando a notação função
- .__doc__

- def recebe(tamanhomax, *, bloco):
- "Recebe uma mensagem."
- print(tamanhomax)
- #recebe(1024, True) #TypeError - ERRADO
- recebe(1024, bloco=True) #OK - CORRETO
- print(recebe.__doc__)

Exemplo 4

Funções anônimas ou inline

- Funções simples podem ser substituídas por uma expressão *lambda*

```
1  # INLINE:
2  add = lambda x, y: x + y
3  print(add(2,3))
4  # TRADICIONAL:
5  def add(x, y):
6      return x+y
7  print(add(2,3))
```

Exercício

Faça o exercício abaixo em uma folha com seu nome e data:

1) Faça uma função que receba qualquer quantidade de valores e efetue a soma. Documente e imprima usando *docstrings*.
2) Faça uma função inline que divide dois valores
3) Faça uma função de multiplicação de dois valores, base e altura, que somente aceite argumentos nomeados
4) Faça uma função que soma dois valores em que se nenhum argumento for definido, o primeiro vale 1 e o segundo vale 4.

Obrigado!

Alguma dúvida?
rafael.brinhosa@ifc-araquari.edu.br