

Design de Software

Aula 7 – Python (funções)

2016 – Engenharia

Fábio Ayres <fabioja@insper.edu.br>

Raul Ikeda <rauligs@insper.edu.br>

Objetivos de Aprendizado

- Criar funções no Python, explorando as várias possibilidades de construção
- Organizar código em Módulos

Uso de funções

- Funções nativas já usadas

`input()`

`float()`

`print()`

- Funções que precisam ser importadas

```
[from random import randint  
randint()]
```

Declarando novas funções

- Função muito simples

```
def nome_da_funcao():  
    # operações
```

- Função com argumento

```
def nome_da_funcao(argumento1, argumento2):  
    # operações
```

- Função com retorno

```
def nome_da_funcao(argumento1, argumento2):  
    resultado = argumento1 * argumento2  
    return resultado
```

Chamando as funções

- Depois da função ser criada, esta é executada da mesma forma de qualquer outra função:

```
1 def hello():  
2     print("Hello, world!")  
3  
4 hello()
```

A função deve sempre ser declarada antes de ser usada.

Variáveis Locais

- Tem visibilidade (escopo) limitado.

```
1 a=5
2 def muda_e_imprime():
3     a=7
4     print("a dentro da função: %d" % a)
5 print("a antes de mudar: %d" % a)
6 muda_e_imprime()
7 print("a depois de mudar: %d" % a)
```

O que esse programa faz?

Parâmetros Opcionais

- É possível criar funções com parâmetros opcionais se definido valores padrões.

```
1 def frase(nome, curso="Design de Software", escola="Insper"):  
2     print("%s é aluno de %s no %s."%(nome,curso,escola))  
3  
4 frase("Você")  
5 frase("Você", "Modelagem e Simulação")
```

Retorno de múltiplos valores

- O Python permite que múltiplos valores sejam retornados, para isso separe eles por vírgula.

```
1 def localize(cidade):
2     if cidade == "Sao Paulo":
3         return("23º 32' 51\" S", "46º 38' 10\" W")
4     elif cidade == "Rio de Janeiro":
5         return("22º 54' 10\" S", "43º 12' 27\" W")
6
7
8 cidade="Sao Paulo"
9 latitude, longitute = localize("Sao Paulo")
10 print("%s está localizado a %s, %s."%(cidade, latitude, longitute))
```


Exercícios

Escreva uma função que:

1. receba dois números e retorne **True** se o primeiro número for múltiplo do segundo. (Nilo Menezes 8.2)
2. receba o lado (l) de um quadrado e retorne sua área ($A = lado^2$). (Nilo Menezes 8.3)
3. receba a base e a altura de um triângulo e retorne sua área ($A = base \times altura / 2$). (Nilo Menezes 8.4)
4. receba um texto e retorne **True** se a primeira letra for uma vogal.
5. receba três números e retorne o maior deles.
6. receba um numero que será usado para definir o tamanho de uma lista a ser retornada com a série de Fibonacci.
7. receba sim/Sim/SIM/não/nao/Não/Não/NAO/NÃO e converta para **True** ou **False**

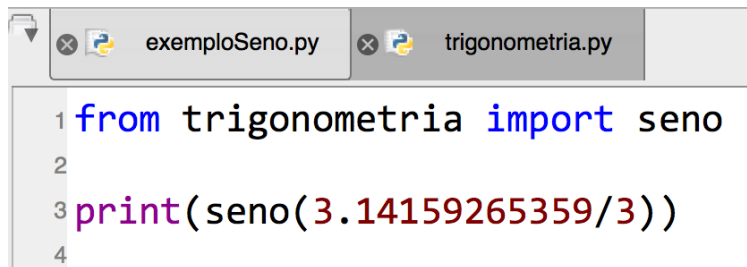
Parâmetros Nomeados

Os valores podem ser passados diretamente pelo nome dos argumentos, não precisam respeitar a ordem dos parâmetros neste caso.

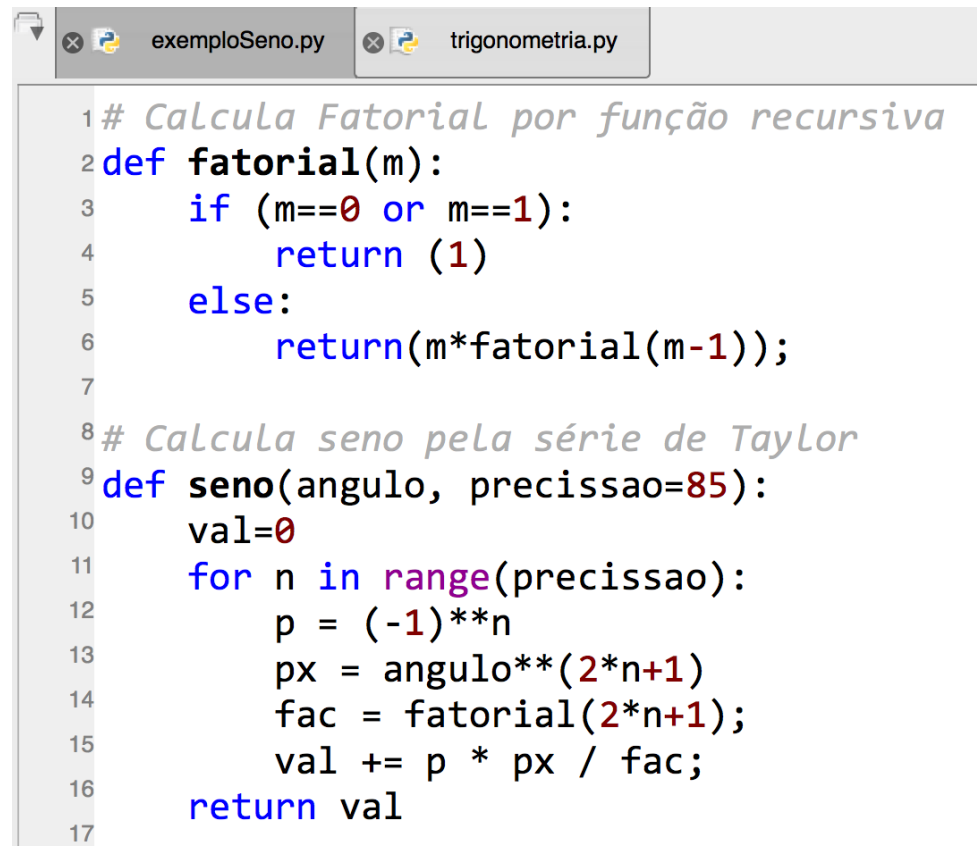
```
1 def força(massa, aceleração):  
2     return massa*aceleração  
3  
4 print(força(aceleração=9.8, massa=120))
```

Módulos


Arquivos .py são módulos naturalmente e podem ser importados diretamente com a chamada **import**.



```
1 from trigonometria import seno
2
3 print(seno(3.14159265359/3))
4
```



```
1 # Calcula Fatorial por função recursiva
2 def fatorial(m):
3     if (m==0 or m==1):
4         return (1)
5     else:
6         return(m*fatorial(m-1));
7
8 # Calcula seno pela série de Taylor
9 def seno(angulo, precisao=85):
10     val=0
11     for n in range(precisao):
12         p = (-1)**n
13         px = angulo**(2*n+1)
14         fac = fatorial(2*n+1);
15         val += p * px / fac;
16     return val
17
```



```
import aaa  
aaa.bbb
```

```
from aaa import bbb  
bbbb
```

```
from aaa import *  
bbbb
```

```
talvez como nota:  
import module as name
```

Exercícios

Escreva uma função que:

8. receba uma lista de palavras e retorne o comprimento da palavra mais longa.
9. receba uma lista e imprima a primeira metade da lista em uma linha e a segunda metade em outra linha
10. receba um texto e retorne **True** se for um palíndromo.
11. receba um texto com fórmulas matemáticas básicas e retorne o resultado. Ex: "2 + 2 * 7"
12. resolva um antigo quebra-cabeça chinês clássico:
Contamos 35 cabeças e 94 pernas entre as galinhas e coelhos em uma fazenda. Quantos coelhos e quantas galinhas que nós temos?

Para a próxima aula

- Arquivos e Matrizes

Capítulo 9 do livro

Introdução à Programação com Python,
de Nilo Ney Menezes.

(disponível na biblioteca)

Ler:

http://www.python-course.eu/matrix_arithmetic.php



Insper

www.insper.edu.br

Fun: pythonic

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language. Likewise, to say of an interface or language feature that it is pythonic is to say that it works well with Python idioms, that its use meshes well with the rest of the language.

wikipedia