

Fundamentos da Programação LEIC/LETI

Aula 5

Funções

Definição de funções. Aplicação de funções. Abstração procedimental. Exemplos simples.

Alberto Abad, Tagus Park, IST, 2021-22

Na semana passada aprendemos...

- O que é e quais são as fases da atividade de programação?
- A descrever sintaxe de uma linguagem de programação --> **BNF**
- Sobre o Python e como interagir com ele: interpretador vs. script
- Alguns elementos básicos de programação:
 - Tipos, nomes, expressões, condições, entrada/saída, funções embebidas, etc.
- Sobre execução sequencial e sobre instruções para alterar o fluxo de execução:
 - Seleção --> **IF**
 - Repetição --> **WHILE**

Funções

Funções

- Conjunto pares ordenado entrada (domínio) e saída (contra-domínio)
 - Definição por extensão ou por abstração
 - Como utilizar funções? Definição e aplicação:

$f(x, y) = x + y$ Definições equivalentes?
 $f(3,5) = 8$ Aplicação!

- Igual que na Matemática, a utilização de funções em programação compreende a **definição** da função (nome, argumentos e algoritmo) e **aplicação de função** (execução do algoritmo sobre valores passados como argumentos).
- Exemplo funções Python já conhecidas: `print(...)`, `input(...)`, `eval(...)`

Funções

Definição de Funções (BNF)

```
<definição de função> ::=
```

```
def <nome> (<parâmetros formais>): NEWLINE
```

```
INDENT <corpo> DEDENT
```

```
<parâmetros formais> ::= <nada> | <nomes>
```

```
<nomes> ::= <nome> | <nome>, <nomes>
```

```
<nada> ::=
```

```
<corpo> ::= <definição de função>* <instruções em função>
```

```
<instruções em função> ::=
```

```
<instrução em função> NEWLINE |
```

```
<instrução em função> NEWLINE <instruções em função>
```

```
<instrução em função> ::= <instrução> | <expressão> | <instrução return>
```

```
<instrução return> ::= return | return <expressão>
```

Funções

Aplicação Funções (BNF)

<aplicação de função> ::= <nome>(<parâmetros concretos>)

<parâmetros concretos> ::= <nada> | <expressões>

<expressões> ::= <expressão> |
 <expressão>, <expressões>

Funções

Definição e Aplicação de Funções, Exemplo 1:

```
def soma(a,b):  
    return a + b
```

Exemplos:

- Aplicação antes e após definição.
- Aplicação: soma(2) , soma(2,5 , soma(7,5) , soma(3*2, 6+4) , soma

```
a, b = 2, 5  
print("soma(a,b) =", soma(a,b))  
print("soma(b,a) = ", soma(b,a))  
print("a =", a, "b =", b)
```

```
In [69]: f(2,5)
```

```
Out[69]: 7
```

Funções

Definição de Aplicação Funções, Exemplo 2:

```python def soma\_progressao\_aritmetica(n): pass # complete ``` - Aplicação: ```python soma = 4 + 6  
print(soma\_progressao\_aritmetica(100), soma) ``` - Que acontece com a variável \*soma\*!?!? - Conseguem pensar numa solução não iterativa?

```
In [88]: def soma_progressao_aritmetica(n):
 return n*(n+1)//2

res = soma_progressao_aritmetica(99)
```

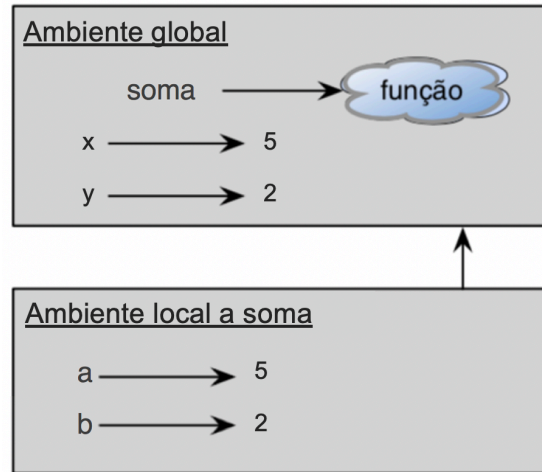
## Funções

### Ambientes e quadros (*frames*)

- Ambientes: Global vs. Local
- Passos seguidos pelo Python quando uma função é invocada:
  - Os parâmetros concretos são avaliados (ordem arbitrária)
  - Os parâmetros formais da função são associados com os valores concretos no ambiente local (em ordem)
  - O corpo da função é executado no ambiente local (os ambientes locais existem só até a função terminar) e o valor de *return* é retornado ao ambiente global

# Funções

## Ambientes e quadros (*frames*)



- Gostavam ver? <http://pythontutor.com/visualize.html> (<http://pythontutor.com/visualize.html>)

## Funções

### Abstração procedimental

- As funções permitem aos programadores pensar no **que** (faz a função) e não no **como** (a função é implementada).

#### Exemplo #1

```
def soma_prog_arit(n):
 iter = 1
 soma = 0
 while iter <= n:
 soma = soma + iter
 iter = iter + 1
 return soma

def soma_prog_arit(n):
 if n < 1:
 return 0
 else:
 return n*(n+1)//2
```

## Funções

### Abstração procedimental

- As funções permitem aos programadores pensar no **que** (faz a função) e não no **como** (a função é implementada).

#### Exemplo #2

```
def quadrado(x):
 return x * x

def quadrado(n):
 pass # complete
```

```
In [1]: def quadrado_v1(x):
 return x*x

 def quadrado_v2(x): # **
 pass

 def quadrado_v3(x): # pow
 pass

 def quadrado_v3(x): # iter
 pass
```

## A treinar!!!!

### Funções

## Exemplo 1, Tabela conversão temperaturas

- Escreva uma função que converta temperaturas de Fahrenheit para Celsius  $C = 5 * (F - 32)/9$
- Escreva uma função que recebe uma temperatura mínima e máxima (inteiros) em Fahrenheit e imprime a tabela de conversão para Celsius

```
In [45]: def fahr_para_cent(fahr):
 pass

 def tabela(min_f, max_f):
 pass

tabela(40, 50)

F: 40 ---> C: 4.4444444444444445
F: 41 ---> C: 5.0
F: 42 ---> C: 5.5555555555555555
F: 43 ---> C: 6.1111111111111111
F: 44 ---> C: 6.6666666666666667
F: 45 ---> C: 7.2222222222222222
F: 46 ---> C: 7.7777777777777778
F: 47 ---> C: 8.3333333333333334
F: 48 ---> C: 8.888888888888889
F: 49 ---> C: 9.4444444444444445
F: 50 ---> C: 10.0
```

## Funções

### Exemplo 2, Potência de dois números inteiros

- Solução iterativa para k positivos
- E para k negativos?
- E para qualquer k ?

```
In [106]: def potencia(x, n):
 pass

 # Power of two numbers inteiros
 x = eval(input("Escreva a base da potencia: "))
 n = eval(input("Escreva a potencia: "))

 print(potencia(x, n))
```

```
Escreva a base da potencia: 2
Escreva a potencia: -4
```

```


ValueError Traceback (most recent c
all last)
<ipython-input-106-d384ce75a56c> in <module>
 20
 21
----> 22 print(potencia(x, n))

<ipython-input-106-d384ce75a56c> in potencia(x, n)
 5 raise ValueError('exponente deve ser inteiro')
 6 elif n < 0:
----> 7 raise ValueError('n deve ser positivo')
 8
 9 pot = 1

ValueError: n deve ser positivo
```



## Funções

### Exemplo 3, Factorial

- Para números inteiros não negativos (  $\text{fact}(0) = 1$  ) iterativo

```
In [75]: def factorial(x):

 fact = 1
 while x != 0:
 fact = fact * x
 x = x - 1

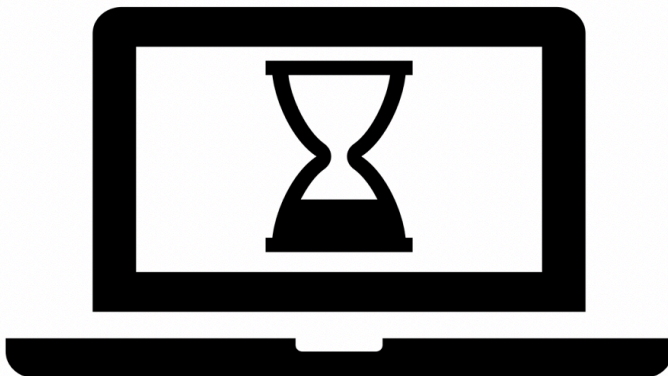
 return fact

x = eval(input('Inteiro: '))
f = factorial(x)
print(f)
```

```
Inteiro: 35
10333147966386144929666651337523200000000
```

### Funções - Tarefas próxima aula

- Trabalhar matéria apresentada hoje
- Fazer todos os exercícios/programas



In [ ]: