



Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



INF 100 – Introdução à Programação

Funções
(parte 1)

Motivação

- Exemplo de cálculo de \sqrt{x} usando o algoritmo proposto por Heron de Alexandria (Método de Newton):

1. Leia x
2. Faça $r = x/2$ # chute inicial para a raiz
3. Faça $r = (r + x/r) / 2$
4. Se $|r^2 - x| > \varepsilon$, retorne ao passo 3
5. Escreva r

onde ε = um erro estabelecido qualquer, por exemplo, 10^{-10} .



Motivação

- Possível implementação em Python:

```
x = float( input('Entre com o valor de x: '))  
r = x/2  # chute inicial para a raiz  
while abs( r*r - x ) > 1e-10:  
    r = (r + x/r) / 2  
print('Raiz de', x, '=', r )
```



Motivação

- Imagine se tivéssemos que usar o Método de Newton sempre que fosse preciso calcular uma raiz quadrada!
 - Código confuso.
 - Difícil reusar em outros programas.
 - Fácil cometer erros.
 - Por que preciso saber como calcular raiz (ou seno, cosseno, logaritmo etc.) sempre que precisar disso?



Funções em Python

- Por isso usamos funções:

```
import math
```

```
x = float( input('Entre com o valor de x: '))
```

```
r = math.sqrt( x )
```

```
print('Raiz de', x, '=', r )
```



Funções em Python

- Outros exemplos de funções da biblioteca `math`:
 - `log(x)`
 - `log10(x)`
 - `exp(x)`
 - `sqrt(x)`
 - `tan(x)`
 - `sin(x)`
 - `cos(x)`
 - ...



...

```
while True:
    n = int( input('Entre com a quantidade de alunos: '))
    if n < 2 or n > 50:
        print('Valor deve estar entre 2 e 50')
    else:
        break

soma = 0
for i in range(0, n):
    while True:
        x = int( input('Entre com a nota do próximo aluno: '))
        if x < 0 or n > 100:
            print('Valor deve estar entre 0 e 100')
        else:
            break
    soma = soma + x
media = soma / n
```



...

Identificando oportunidades
para reuso de código...

```
while True:
    n = int( input('Entre com a quantidade de alunos: '))
    if n < 2 or n > 50:
        print('Valor deve estar entre 2 e 50')
    else:
        break
```

```
soma = 0
for i in range(0, n):
    while True:
        x = int( input('Entre com a nota do próximo aluno: '))
        if x < 0 or n > 100:
            print('Valor deve estar entre 0 e 100')
        else:
            break
    soma = soma + x
media = soma / n
```



Funções em Python

...

```
n = leiaValor('Entre com a quantidade de alunos: ', 2, 50)
soma = 0
for i in range(0, n):
    x = leiaValor('Entre com a nota do próximo aluno: ', 0, 100)
    soma = soma + x
media = soma / n
```

O código para a função leiaValor será apresentado mais adiante...



Funções em Python

- Vantagens do uso de funções:
 - Modularidade e clareza do código fonte.
 - Reuso de software.
 - Menores chances de erros.
 - Separar o uso da função de seus detalhes de implementação.
 - Permitem pensar no algoritmo em mais alto nível primeiro, para depois ir refinando ao nível de comandos da linguagem de programação.



Funções em Python

- Para criar uma função, devemos conhecer bem as suas necessidades:
 - A função necessita de dados de entrada (parâmetros)?
 - Se sim, quais dados?
 - Espera-se que a função responda com alguma informação?
 - Se sim, qual ou quais?
- Essas perguntas devem ser respondidas enquanto se projeta a função, e elas estarão especificadas na declaração da mesma.



Definição de uma função em Python

```
def nome( lista_de_parâmetros ):  
    <comandos>  
    return <valor(es) de retorno>
```

- **nome**: nome usado para chamar (usar) a função.
- **lista_de_parâmetros**: dados (constantes, variáveis) passados para a função.



Funções em Python

- Para se trabalhar com funções deve-se
 - Declarar a função;
 - Programar a função;
 - Evocar (chamar, usar) a função.
- Exemplo: definição da função **leiaValor...**



```
def leiaValor( msg, min, max ):  
    while True:  
        v = int( input( msg ))  
        if v < min or v > max:  
            print('Valor deve estar entre', min, 'e', max )  
        else:  
            break  
    return v  
  
n = leiaValor('Entre com a quantidade de alunos: ', 2, 50)  
soma = 0  
for i in range(0, n):  
    x = leiaValor('Entre com a nota do próximo aluno: ', 0, 100)  
    soma = soma + x  
media = soma / n  
...
```



Exercício

- Defina uma função chamada **abs()** que aceita como parâmetro um valor x qualquer e retorna o módulo ou valor absoluto de x .



Exercício

```
def abs( x ):  
    if x < 0:  
        return -x  
    else:  
        return x
```



Exercício – programa completo

```
def abs( x ):
    3 → if x < 0:
        return -x
    else:
        4 (return x) 2
x1 = float( input('Entre com o x1: ') )
x2 = float( input('Entre com o x2: ') )
print(' |x1 - x2| =', abs( (x1-x2) ))
print(' |x1| =', abs( x1 )) 1
print(' |x2| =', abs( x2 ))
```



Passagem de parâmetro – por valor

1. A expressão é avaliada (uma constante, uma variável ou uma expressão / fórmula);
2. O valor-resultado é copiado para dentro do parâmetro da função, obedecendo a mesma ordem de chamada e de declaração;
3. A execução do programa é desviada para o início da função;
4. Ao encontrar um comando **return**, o programa retorna a execução para o local de onde a função foi chamada, juntamente com o valor de retorno, se houver.



Funções em Python

- No nosso exemplo, a função `abs()` retorna um resultado.
- Podemos ter funções que executam alguma tarefa específica mas não retornam nenhum valor como resultado. Ex.:

`sound(frequencia, duracao_ms)`

Propósito: emitir um som com determinada frequência em hertz e duração em ms pelo autofalante do computador.



Funções em Python

```
def sound( frequencia, duracao_ms ):
```

```
...
```

```
...
```

```
...
```

Nesse caso não precisa ter o comando *return*. A execução da função terminará quando o último comando da função for executado.

```
def sound( frequencia, duracao_ms ):
```

```
...
```

```
...
```

```
return
```

Ou pode ter o comando *return*, mas sem o valor de retorno.



Funções em Python

- Também podemos ter funções que não recebem nenhum parâmetro:

```
def beep():
```

```
...
```

```
def pi():
```

```
    return 3.1415926535897
```



Funções em Python

- Exemplo de uso:

```
beep()  
sound( 200, 1000 )  
print( pi() )
```



Mais exemplos

...

```
def media( a, b ):  
    m = (a + b) / 2  
    return m
```

```
m = media( 5.5, 7.8 )  
print( 'Média =', m )
```



Mais exemplos

```
...  
def media( a, b ):  
    return (a + b) / 2  
  
m = media( 5.5, 7.8 )  
print( 'Média =', m )
```



Mais exemplos

```
...  
def maior( a, b ):  
    if a > b:  
        return a  
    else:  
        return b  
  
x = float( input('x = ') )  
y = float( input('y = ') )  
print('Maior valor =', maior( x, y ))
```



Funções em Python

- Erros comuns:

```
def f( x, y ):
    r = x*x + y    (esqueceu do return)
```

```
def funcao( x, y ):
    return x*x + y
```

```
def pi():
    return 3.1415926535897
```

```
print( pi ) ← pi()
print( funcao( 12 ) ) ←
```

(faltou um parâmetro)



Exercício 1

- Implemente uma função que recebe como parâmetros três valores reais a , b e c , e retorna o maior deles.



Exercício 1

```
def maior( a, b, c ):
    if a > b and a > c:
        return a
    else:
        if b > c:
            return b
        else:
            return c
```



Exercício 2

- Implemente uma função que recebe como parâmetro um valor inteiro x e retorna o valor de $x!$. A função pode supor que x será sempre inteiro e não negativo.



Exercício 2

versão 1

```
def fatorial( x ):
    fat = 1
    while x > 1:
        fat = fat * x
        x = x - 1
    return fat
```



Exercício 2

versão 2

```
def fatorial( x ):
    fat = 1
    for i in range(2, n+1):
        fat = fat * i
    return fat
```

