centro universitário FEI

CC1612
Fundamentos de Algoritmos

Prof. Danilo H. Perico

Modularização: Funções

Funções

- Funções **são blocos de código** que realizam **determinadas tarefas** que normalmente precisam ser executadas diversas vezes dentro da mesma aplicação

- Assim, **tarefas muito utilizadas costumam ser agrupadas em funções**, que, depois de definidas, podem ser utilizadas / chamadas em qualquer parte do código somente pelo seu nome

Funções

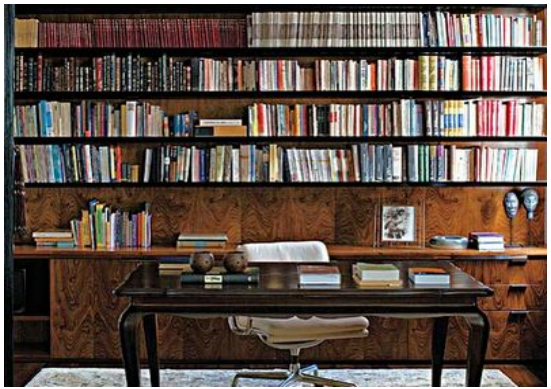
- Invocar/chamar uma função pode ser visto como uma contratação de uma pessoa para executar um trabalho específico.
- **Exemplo: função para organizar papéis e livros**

Funções

Contratante
(Programa Principal)

Trabalho a fazer
(entrada)

Contratado
(função)



Trabalho feito
(saída)

Slide: Prof. Guilherme Wachs

Funções

- Já utilizamos algumas funções no Python:
  - `print(“Olá!”);`
  - `input(“Digite o valor da entrada:”)`
  - `int(“3”)`
  - `range(0,100)`
  - `randint(0,10)`
  - `sleep(2)`
  - `sqrt(9)`

No Python: Funções - *def*

- Podemos criar / definir nossas próprias funções no Python utilizando a palavra-chave ***def*** seguido do ***nome da função***, parêntesis ***( )*** e ***:***
- Sintaxe:

```
def <nome da função>():
    # tarefas que serão realizadas dentro da função
```

- Exemplo:

```
def imprimeOla():
    print("Olá")
```

Funções **com** e **sem** parâmetros

- As funções podem ou não ter **parâmetros**, que **são valores enviados às funções** dentro dos parêntesis no momento em que elas são chamadas
- Exemplos:

Sem parâmetros:

```
def soma():
    a = 9
    b = 8
    print(a+b)

soma()
```

```
17
```

Chamada da função

Com parâmetros:

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

soma(3,4)
```

```
7
```

Chamada da função

Funções - *return*

- Além dos parâmetros, as funções podem ou não ter um **valor de retorno**
- O retorno é definido pela palavra-chave ***return***
- Exemplos:

Sem parâmetros:

```
def soma():
    a = 9
    b = 8
    return(a+b)

print(soma())
```

```
17
```

Com parâmetros:

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

print(soma(3,4))
```

```
7
```

Funções

- Exemplo: Fazer uma função que retorne ***True*** ou ***False*** para a verificação de números pares.
  - **Precisa de parâmetros? Sim ou Não?**
  - **É melhor usar ou não o *return*?**

Funções

- Exemplo: Fazer uma função que retorne ***True*** ou ***False*** para a verificação de números pares.

```
def par(num):
    return(num % 2 == 0)

print(par(3))
print(par(4))
print(par(67))
```

```
False
True
False
```

Funções

- Exemplo:
- Se precisarmos de uma função que retorne a string “*par*” ou “*ímpar*”, podemos reutilizar a função par e criar uma função nova:

```
def par(num):
    return(num % 2 == 0)

def parOuImpar(x):
    if par(x) == True:
        return "par!"
    else:
        return "ímpar!"

print(parOuImpar(4))
print(parOuImpar(3))
print(parOuImpar(56))
```

```
par!
ímpar!
par!
```

Funções - escopo das variáveis: **locais** *vs.* **globais**

- Quando usamos funções, trabalhamos com **variáveis internas**, que **pertencem somente à função**.
- Estas variáveis internas são chamadas ***variáveis locais***
- **Não** podemos acessar os valores das variáveis locais fora da função a que elas pertencem
- É por isso que passamos parâmetros e retornamos valores das funções. Os **parâmetros** e o ***return*** possibilitam a troca de dados no programa

Funções - escopo das variáveis: locais *vs.* globais

- Por sua vez, as ***variáveis globais*** são **definidas fora das funções** e podem ser vistas e acessadas por todas as funções e pelo "*código principal*" (que não está dentro de uma função específica)

Funções - escopo das variáveis: locais *vs.* globais

- Exemplo:

```
# variável global:
a = 5

def alteraValor():
    # variável local da função alteraValor():
    a = 7
    print("Dentro da função 'a' vale: ", a)

print("'a' antes da chamada da função: ", a)
alteraValor()
print("'a' depois da chamada da função", a)
```

```
'a' antes da chamada da função:  5
Dentro da função 'a' vale:  7
'a' depois da chamada da função 5
```

Funções - escopo das variáveis: locais *vs.* globais

- Exemplo:
- Se quisermos modificar a variável global dentro da função, devemos utilizar a palavra-chave ***global***

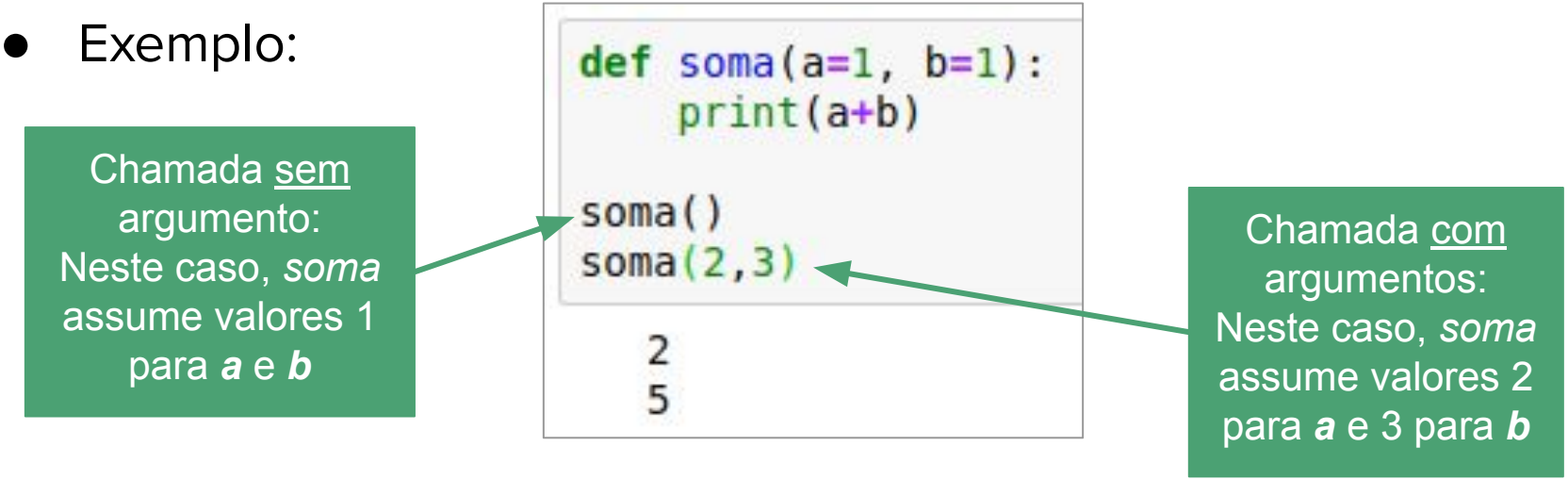
```
# variável global:
a = 5

def alteraValor():
    # dizemos para a função que a variável 'a' é global:
    global a
    a = 7
    print("Dentro da função 'a' vale: ", a)

print("'a' antes da chamada da função: ", a)
alteraValor()
print("'a' depois da chamada da função", a)
```

```
'a' antes da chamada da função:  5
Dentro da função 'a' vale:  7
'a' depois da chamada da função 7
```

Funções - parâmetros opcionais

- Podemos ainda criar funções que podem ou não receber argumentos.

- Exemplo:



Funções *lambda*

- No Python, podemos criar funções simples em somente uma linha
- Estas funções, ou expressões, são chamadas de ***lambda***
- Exemplo: função ***lambda*** que recebe um parâmetro ***x*** e retorna o quadrado deste número.
  - ***lambda*** cria uma função ***a***

```
a = lambda x: x**2
print(a(3))
```

```
9
```

Funções *lambda*

- Exemplo:

- Função ***lambda*** que calcula o aumento, dado o valor inicial e a porcentagem de aumento:

```
aumento = lambda a,b : a*b/100
aumento(100,5)
```

```
5.0
```

Modularização:
Funções - Exercícios

Exercícios

1. Escreva uma função com parâmetros que retorne o maior de dois números. A função deve se chamar ***maximo(x, y).***
2. Escreva uma função com parâmetros chamada ***multiplo(x, y)***. Esta função deve receber dois números e retornar ***True*** se o primeiro for múltiplo do segundo número; a função retorna ***False*** caso contrário.
3. Escreva uma função com parâmetros que receba a base e a altura de um triângulo e retorne sua área (A = base * altura / 2).

Exercícios

4. Uma data é considerada mágica quando o dia multiplicado pelo mês é igual ao ano de dois dígitos. Por exemplo, 10 de junho de 1960 é uma data mágica porque junho é o sexto mês e 6 vezes 10 é 60, o que equivale ao ano de dois dígitos. Escreva uma função que determine se uma data é ou não uma data mágica. Use sua função em um programa que deve encontrar e exibir todas as datas mágicas do século XX.

Exercícios

5. Faça uma função que receba três notas de um aluno como parâmetros e uma letra. Se a letra for A, a função deverá calcular a média aritmética das notas do aluno; se for P deverá calcular a média ponderada com pesos 5, 3 e 2. A média calculada deve ser devolvida à função principal para, então, ser mostrada.

Exercícios

6. Escreva uma função que calcula o quociente e o resto da divisão inteira entre dois números. Utilize apenas as operações de soma e subtração para calcular o resultado. Dica: utilize uma estrutura de repetição para isso.

   Faça um programa principal que recebe o dividendo e o divisor do usuário e, depois de chamar a função, exibe o quociente e o resto.

Exercícios

7. Existem restrições para que uma pessoa possa doar sangue. Uma delas é relativa ao peso. Mulheres tem que pesar no mínimo 50kg e homens no mínimo 60kg. Faça uma função para informar se uma pessoa está ou não apta a doar sangue sabendo seu sexo e seu peso.

   O programa principal deve ler as entradas, acionar a função e exibir a resposta.

Exercícios

8. Escreva uma função chamada ***exponencial*** que recebe um valor ***n*** passado por valor e dois inteiros ***b*** e ***k*** passados por “referência". Sua função deve retornar em ***b*** e ***k*** valores tal que $b^k = n$ e ***b*** seja o menor possível.