INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Funções em Python

ALG – Algoritmos e Programação

Aula 14

Curso Técnico em Informática para Internet – Integrado ao Ensino Médio

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Objetivos da aula

- Conhecer o conceito de funções e seus benefícios para a programação
- Criar funções, com presença ou ausência de parâmetros

Dividir para conquistar

- Modularizar é quebrar um programa em vários “programas” menores
- Em geral, problemas complexos exigem algoritmos complexos.
  - Mas sempre pode ser possível dividir um problema grande em problemas menores. Desta forma, cada parte menor tem um algoritmo mais simples, mais fácil de ser elaborado
- Esse trecho menor é chamado de sub-rotina
  - No caso da linguagem Python, as sub-rotinas são chamadas de funções
- O uso de sub-rotinas permite tornar o código modular, onde estas podem ser executadas (chamadas) em diferentes partes do programa e tantas vezes quanto for necessário.

Dividir para conquistar

- Exemplo:

```
print("Início")
print("Ola, bem-vindo!!!")
print("Esse é o meu programa!")

print("Programa em execução...")

print("Espero que tenha gostado!")
print("Obrigado por usar o meu programa")
print("Fim!")
```

Apresenta o programa

Se despede

Dividir para conquistar

- Com o uso de funções, podemos dar nome a trechos de código
- Após nomear trechos de códigos, podemos chamá-los no programa
- O exemplo anterior ficaria:

```
print("Início")
apresentar()
print("Programa em execução...")
despedir()
print("Fim!")
```

Devemos criar os códigos "apresentar()" e "despedir()"

Criando funções sem argumentos e sem retorno

- Primeiro define-se o nome da função, criando seu cabeçalho
  - Utiliza-se palavra reservada "**`def`**"
- Após isso, define-se o que a função deve realizar, criando o corpo da função

```
def apresentar():
    print("Ola, bem-vindo!!!")
    print("Esse é o meu programa!")
```

Nome da função (cabeçalho)

Recuo é obrigatório

Tarefas a realizar (corpo da função)

Criando funções sem argumentos e sem retorno

- Exemplo completo

```
def apresentar():
    print("Ola, bem-vindo!!!")
    print("Esse é o meu programa!")

def despedir():
    print("Espero que tenha gostado!")
    print("Obrigado por usar o meu programa")
```

Funções

```
print("Início")
apresentar()
print("Programa em execução...")
despedir()
print("Fim!")
```

Programa principal

Funções e programas

- Um programa completo é dividido em um programa principal e diversas funções (a quantidade que for necessária)
- O programa principal é aquele em que inicia a execução do programa. Ele pode chamar as demais funções
- Durante a execução do programa, quando encontra-se a invocação (ou chamada) de uma função, a execução do programa principal é interrompida. A seguir, executa-se o corpo da função invocada. Ao terminar a função, o programa principal volta a ser executado no ponto em que foi interrompido.

Atividade 01

- Crie um programa que:
  1. Dê boas vindas ao usuário.
  2. Após isso, pergunte o nome ao usuário e escreva a mensagem: “Ola, Fulano”, onde Fulano é o nome digitado.
  3. Se despeça do usuário.
- O código referente ao item 2 deve ser colocado dentro de uma função. O restante deve ser implementado no programa principal.

Vantagens do uso de funções

- Divide o programa em partes, o que facilita o gerenciamento e entendimento
- Evita repetição de código
- Pode deixar o código menor

Parâmetros em funções

- Forma de enviar valores para serem trabalhados pelas funções.
- Os parâmetros são variáveis
  - Devem ter seu valor informado quando se chama a função.
- Uma função pode receber vários parâmetros, dos mais diversos tipos.

Funções que recebem parâmetros

```
def nome_da_funcao(<parâmetros>) {
   <comandos>
}
```

- nomeDaFuncao: identificador da função
- <parâmetros>: nomes dos parâmetros

Funções que recebem parâmetros

- Exemplo: Função para somar dois números

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

Função

```
print("Bem-vindo ao programa")
soma(2, 3)
```

Programa principal

Atividade 02

- Incremente o exemplo abaixo, para que o usuário informe os valores a serem somados. A leitura dos valores deve ser realizada no programa principal

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

Função

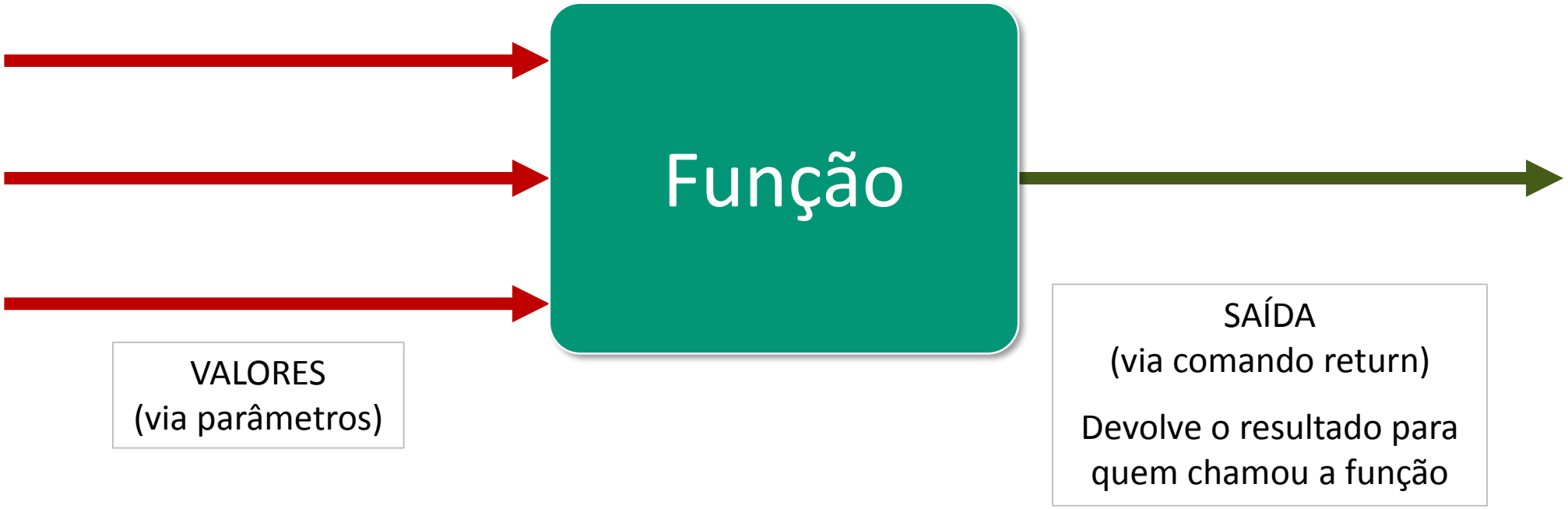
```
print("Bem-vindo ao programa")
soma(2, 3)
```

Programa principal

Retorno de funções

- As funções geralmente retornam um valor para quem chamou
- Entender as funções como “caixas-pretas”: você fornece alguns valores, e ela retorna um resultado de alguma operação sobre esses valores
- A função deve indicar o retorno com o comando return (valor a ser retornado)
- O comando return retorna o valor da função e sai dela

Retorno de funções



Retorno de funções

- Exemplo de criação da função:

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

- Exemplo de uso da função:

```
resultado = soma(2, 3)
print(“Resultado da soma:”, resultado)
```

Exercícios

Exercícios

1) Crie uma função que leia o nome de uma pessoa e escreva uma saudação para ela. A leitura deverá ser feita dentro da função (não haverá parâmetros)
2) Crie uma função que leia o nome de uma pessoa e sua idade. Após isso escreve se a pessoa é maior ou menor de idade. A leitura deverá ser feita dentro da função (não haverá parâmetros)
3) Crie uma função que **receba por parâmetro** o nome de uma pessoa. A função deve escrever uma saudação para ela.
4) Crie uma função que **receba por parâmetro** o nome e a idade de uma pessoa. A função deve escrever se a pessoa é maior ou menor de idade
5) Crie uma função que receba por parâmetros três números e calcule a soma deles. Imprima o valor da soma dentro da função.

Exercícios

5) Crie uma função que recebe como parâmetro dois números A e B. A função deve escrever o resultado de A elevado a B.
6) Crie uma função que recebe como parâmetro um número qualquer. A função deve escrever se o número recebido é positivo ou negativo
7) Crie uma função que recebe como parâmetro um número inteiro. A função deve escrever o mês correspondente aquele número. Por exemplo, se o número passado for 1, deve-se escrever “Janeiro”. Se for 10, deve-se escrever “Outubro”. Se for um valor inválido, deve-se escrever “Mês inválido”.
8) Crie uma função que escreva um valor aleatório (pesquise!!!)

Exercícios

9) Crie uma função que recebe como parâmetro um número Y e uma String S. A função deve escrever a string recebida Y vezes na tela.
10) Crie uma função que recebe dois números e escreva o maior entre eles
11) Crie uma função que recebe três números e escreva o maior entre eles.
12) Crie uma função que recebe o valor de um produto e o desconto a ser aplicado. A função deve escrever o valor a ser pago pelo cliente, o qual deve ser aplicado o desconto
13) Crie uma função que recebe a quantidade de km rodado por um taxista e o valor de cada km. A função deve escrever o valor que o taxista deve receber pela corrida.

Exercícios

Funções com retorno

Exercícios

1) Crie uma função que receba por parâmetros três números e retorna a soma deles.
2) Crie uma função que recebe como parâmetro um número qualquer. A função deve retornar um número inteiro, conforme a seguir:
   - Retornar 1 se o número recebido é positivo
   - Retornar -1 se o número recebido é negativo
   - Retornar 0 se o número recebido é zero
3) Crie uma função que recebe dois números e retorne o maior entre eles
4) Crie uma função que recebe o valor de um produto e o desconto a ser aplicado. A função deve retornar o valor a ser pago pelo cliente no produto

Exercícios

5) Crie uma função que recebe a quantidade de km rodado por um taxista e o valor de cada km. A função deve retornar o valor que o taxista deve receber pela corrida.
6) Faça uma função que receba quatro valores, referentes as notas que um aluno obteve nos bimestres. A função deve retornar a média final desse aluno. (Pesquise como arredondar a nota).
7) Faça uma função que receba quatro valores, referentes as notas que um aluno obteve nos bimestres. A função deve retornar Verdadeiro se o aluno foi aprovado e Falso caso contrário.
8) Faça uma função que receba uma lista como parâmetro e retorne sua soma

Escopo de Variáveis

- O escopo de uma variável indica sua visibilidade no código, ou seja, onde a variável é acessível.

- Temos dois escopos para variáveis em Python:
  - **global** e **local**

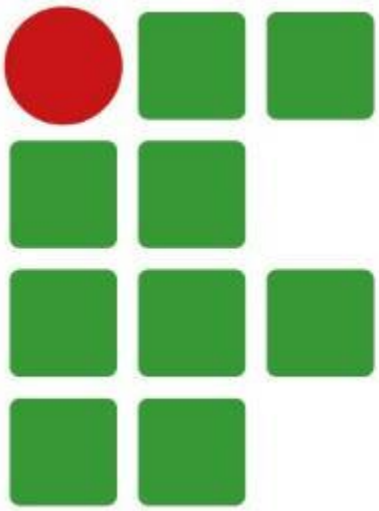
Variável global e local

- **Variáveis globais:**
  - São criadas fora das funções.
  - Podem ser acessadas por todas as funções presentes no programa onde estão definidas.
- **Variáveis locais:**
  - São criadas dentro de uma função.
  - Existem apenas dentro da função onde foi declarada.
  - As variáveis locais são inicializadas a cada nova chamada à função.
- **Obs**: em caso de conflito, ou seja, a variável local e a global terem nomes idênticos, vale a variável local.

Exemplo de variável local e global

```
VAR_GLOBAL="Variavel definida fora da função"
def imprimir_frase():
    VAR_LOCAL="Variavel definida na função imprimir_frase()"
    print("Variável global: ", VAR_GLOBAL)
    print("Variável local: ", VAR_LOCAL)
print("Executando a função escreve_texto:")

###########PROGRAMA PRINCIPAL#################
imprimir_frase()
print("Tentando acessar a variável global no programa principal:")
print("Variável global: ", VAR_GLOBAL)
print("Tentando acessar a variável local no programa principal:")
print("Variável local: ", VAR_LOCAL)
```



INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos