Introdução às Funções em Python

Fundamentos de Programação em Python

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizagem da Aula

- Compreender o que são funções e qual é sua importância na organização de programas em Python.
- Definir funções utilizando a palavra-chave def.
- Entender como utilizar parâmetros e valores de retorno em funções.
- Aplicar boas práticas de documentação, utilizando docstrings.

O que são funções?

- Uma **função** é um bloco de código reutilizável que realiza uma tarefa específica.
- Funções ajudam a **organizar melhor o código**, **evitar repetições** e facilitar a **leitura e manutenção**.
- Python já possui várias **funções embutidas**, como `print()`, `len()`, `input()`, entre outras.

Exemplos:

```
print('Hello World')
```

```
type(32)
```

Por que usar funções?

- Evita **duplicação** de código.
- Facilita **testes** e depuração.
- Permite **reutilização** em diferentes partes do programa.
- Organiza o raciocínio lógico em **blocos** bem definidos.

Estrutura de uma função

```
def nome_da_funcao(lista de parametros):
    instruções
    return valores
```

- `def` → palavra-chave para definir uma função
- `nome_da_funcao` → nome escolhido pelo programador
- `parametros` → dados de entrada (opcional)
- `return` → valor de saída (opcional)

Exemplo de função simples

```
def saudacao():
    print("Olá! Seja bem-vindo(a).")
```

Chamando a função:

```
saudacao()
```

Saída:

```
Olá! Seja bem-vindo(a).
```

Funções com parâmetros

```
def cumprimentar(nome):
    print("Olá,", nome)
```

Chamando a função:

```
cumprimentar("Ada")
```

Saída:

```
Olá, Ada
```

Funções com retorno

```
def somar(a, b):
    resultado = a + b
    return resultado
```

Uso da função:

```
resultado = somar(3, 5)
print(resultado)
```

Saída:

```
8
```

Parâmetros opcionais e valores padrão

```
def apresentar(nome, saudacao="Olá"):
    print(saudacao, nome)
```

Chamadas possíveis:

```
apresentar("Maria")
apresentar("João", "Oi")
```

Saídas:

```
Olá Maria
Oi João
```

Documentação de funções

- É uma boa prática escrever uma **docstring** no início da função:

```
def somar(a, b):
    """Retorna a soma de dois números inteiros."""
    return a + b
```

Dica: Use `help(somar)` para visualizar a documentação da função.

Docstring no Formato Google

```python
def multiplicar(a, b):
    """
    Multiplica dois números e retorna o resultado.

    Args:
        a (int or float): O primeiro número.
        b (int or float): O segundo número.

    Returns:
        int or float: O resultado da multiplicação entre a e b.

    Example:
        >>> multiplicar(2, 3)
        6
    """
    return a * b
```

- O bloco `Args` descreve os parâmetros de entrada.
- O bloco `Returns` descreve o tipo e o significado do valor retornado.
- O bloco `Example` mostra como a função pode ser utilizada na prática.

Escopo de Variáveis

O **escopo** define **onde uma variável pode ser acessada** dentro do código. Variáveis podem ter **escopo local** ou **global**, dependendo de onde são declaradas.

Escopo Local

- Uma **variável local** é declarada **dentro de uma função**.
- Só pode ser usada **dentro dessa função**.
- Ela **não existe fora** do bloco em que foi criada.

```
def saudacao():
    nome = "Maria"  # variável local
    print("Olá,", nome)

saudacao()
print(nome)  # Erro! 'nome' não está definida fora da função
```

Escopo de Variáveis (cont.)

Escopo Global

- Uma variável global é declarada fora de qualquer função.
- Pode ser acessada dentro e fora das funções, mas não pode ser modificada dentro de uma função sem a palavra-chave global.

```
mensagem = "Bem-vindo!"  # variável global

def mostrar_mensagem():
    print(mensagem)  # acessa a variável global

mostrar_mensagem()
```

Modificando Variáveis Globais

Para modificar uma variável global dentro de uma função, é necessário usar a palavra-chave `global`. Veja o exemplo abaixo:

```
contador = 0

def incrementar():
    global contador
    contador += 1

incrementar()
print(contador)  # Resultado: 1
```

Conclusão e Prática

O que aprendemos hoje:

- O que são funções e para que servem
- Como definir funções com e sem parâmetros
- Como utilizar `return` para obter valores de saída
- Como usar valores padrão e documentação

Próximo passo:

- Praticar criando funções para resolver pequenos problemas
- Entender escopo de variáveis e funções recursivas

Exercícios

1. Crie uma função chamada `calcular_media` que receba três notas como parâmetros e retorne a média aritmética das notas.
2. Escreva uma função chamada `eh_primo` que receba um número inteiro entre 0 e 20 como parâmetro e retorne `True` se ele for primo e `False` caso contrário.
3. Crie uma função chamada `converter_celsius_para_fahrenheit` que receba uma temperatura em graus Celsius e retorne o valor convertido para Fahrenheit. A fórmula de conversão é $F = C \times \frac{5}{9} + 32$.
4. Crie uma função chamada `fib` que recebe um número inteiro `n` como parâmetro e retorna os `n` primeiros termos da sequência de Fibonacci. Exemplos:
   - Para `n = 3`, temos `0, 1, 1`
   - Para `n = 5`, temos `0, 1, 1, 2, 3`

Dúvidas e Discussão