Algoritmos Computacionais

Conteúdo: Funções em Python

Prof. Dsc. Giomar Sequeiros
giomar@eng.uerj.br

Funções

Funções: sintaxe básica

- Em Python, uma **função** é definida **usando** a palavra-chave "**def**" seguida pelo **nome** da função e seus **parâmetros**, entre parênteses.
- O bloco de código que define a função deve estar **indentado**.
- A sintaxe **básica** é a seguinte:

```
def nome_da_funcao(parametro1, parametro2, ...):
    # corpo da função
    return valor_de_retorno # opcional
```

- Os **parâmetros** são **variáveis** que a função recebe como **entrada**. Eles são **opcionais** e podem ser de **qualquer tipo** de dados.
- Dentro da função, esses **parâmetros** se comportam como **variáveis locais**.
- A instrução "**return**" é **opcional** e é usada para retornar um valor da função.

Regras de para nomear funções

- Os nomes devem começar com uma **letra** (maiúscula ou minúscula) ou um **sublinhado** (_).
- Os nomes podem conter **letras** (maiúsculas e minúsculas), dígitos e sublinhados (_).
- Os nomes são **sensíveis** a maiúsculas e minúsculas, o que significa que as variáveis numero e Numero são diferentes.
- Não é possível usar palavras reservadas do Python como nomes. Exemplos: if, else, for, while, def, class, etc.
- Usar nomes descritivos que indiquem o seu propósito ou função no código.
- Evite usar nomes de uma única letra, a menos que sejam usados como variáveis contadores em loops curtos.
- Evite usar caracteres especiais, como @, $, %, *.

Funções - exemplo 1

- Função simples que recebe dois parâmetros numéricos e retorna a soma deles:

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

# Testando a função
resultado = soma(2, 3)
print(resultado) # imprime 5
```

Separamos **funcionalidade** do **teste**.

Funções - exemplo 2

- Função que calcula a média de dois números

```
def media(a,b):
    '''Função que calcula a média de dois números
    a e b'''
    return (a+b)/2 # retorna a soma de a e b entre 2
```

Funções - exemplo 3

- Função que calcula a área de um retângulo

```
def arearetangulo(b, h):
    '''calcula a área do retangulo de base b e altura h'''
    return b*h
```

Funções que invocam outras funções

- É possível criar uma **função** que **chama** outra **função**.
- Isso pode ser útil para dividir a lógica em partes menores e mais gerenciáveis. Aqui está um exemplo simples:

```
def quadrado(n):
    return n ** 2

def cubo(n):
    return n ** 3

def soma_quadrado_cubo(n):
    return quadrado(n) + cubo(n)

#Teste
resultado = soma_quadrado_cubo(3)
print(resultado) # Saída: 36
```

Funções que invocam outras funções - exemplo 1

- Considere o exemplo a seguir:

```
def pi():
    '''retorna o valor de pi com duas casas decimais'''
    return 3.14

def areaCirculo(r):
    '''calcula a area de um círculo de raio r'''
    return pi()*r**2

def areaCoroa(r1,r2):
    '''calcula a area de uma cora circular dados os raios r1 e
    r2 de dois círculos concentricos, sendo r1>r2
    '''
    return areaCirculo(r1) - areaCirculo(r2)
```

- Chamamos a função areaCirculo duas vezes evitando reescrever funcionalidades já existentes

Funções que invocam outras funções - exemplo 2

- Consideremos a funçao **media**, podemos invocar ela desde outra função

```
def media(a,b):
    '''Função que calcula a média de dois números
    a e b'''
    return (a+b)/2 # retorna a soma de a e b entre 2
```

```
def principal():
    # entrada de dados
    n1 = float(input('Insira a nota 1: '))
    n2 = float(input('Insira a nota 2: '))
    proj = float(input('Insira a nota projeto: '))

    # processamento
    notaFinal = media(media(n1, n2), proj)

    #saida
    print('A nota final é',notaFinal)

#chamada principal
principal()
```

Separamos **funcionalidade** da **interação** do usuário com o objetivo de **reaproveitar** código.

Função principal com **entrada**, **processamento** e **saída** de dados.

O resultado de uma função pode ser argumento de outra.