

Introdução a Ciência da Computação

Funções

Parte 2

Maurício M. Neto

Universidade Federal do Ceará

24 de Setembro de 2020

Objetivos

- 1 Revisando Funções
- 2 Funções em Python
- 3 Função para Validação
- 4 Parâmetros Opcionais
- 5 Nomeando Parâmetros
- 6 Funções como Parâmetro
- 7 Empacotamento e Desempacotamento de Parâmetros
- 8 Funções Lambda
- 9 Módulos

Revisando Funções em Python

- Funções são trechos de códigos que executam uma determinada tarefa bem definida e que necessita ser executada diversas vezes ao longo do programa
- As funções permitem realizar desvios na execução dos programas
- Principais características da Função:
 - ▶ Reduz a complexidade do algoritmo
 - ▶ Permite focar em um problema específico
 - ▶ Melhora a compreensão do código
 - ▶ Facilita a correção de erros
 - ▶ Reutilização do código
 - ▶ ...

Escopo de Variáveis

- Existem dois tipos de escopos: **Local** e **Global**
- **Escopo Local**
 - ▶ São variáveis declaradas dentro de uma função ou sub-rotina
 - ▶ São visíveis apenas dentro da sub-rotina
 - ▶ Quando a sub-rotina chega ao fim, as variáveis locais são destruídas
- **Escopo Global**
 - ▶ São declaradas fora de qualquer sub-rotina
 - ▶ São visíveis em qualquer ponto do programa, inclusive dentro das sub-rotinas

Funções em Pseudocódigo

Exemplo de Função em Pseudocódigo

SUB-ROTINA calculaAumento(sal NUMÉRICO)

 DECLARE perc, valor NUMÉRICO

 LEIA perc

 valor = sal * perc / 100

 RETORNE valor

FIM-SUB-ROTINA calculaAumento

Sintaxe de uma Função em Python

Sintaxe de uma Função em Python

```
def nomeFuncao(parametro1, parametro2,..., parametro N):  
    comando 1  
    comando 2  
    ...  
    comando N  
    return valor
```

Características de Função em Python

- As funções podem ter diversos parâmetros ou nenhum
- A função pode ou não retornar algo
 - ▶ O comando **return** não é obrigatório
 - ▶ Quando uma função não retorna nada chamamos de *procedimento*
- Também é possível retornar uma expressão
- **Em Python:** a função pode retornar mais um valor

Exemplo de função que retorna mais de um valor

```
def cadastro():  
    nome = input('Qual o seu nome: ')  
    idade = int(input('Qual a sua idade: '))  
    return nome, idade  
  
print('Vamos aos dados Cadastrais')  
nome, idade = cadastro()
```

Criando Funções de Validação

- Funções são úteis para validar entrada de dados
- Assim, podemos adicionar um grau de melhoria em nossos códigos
- A validação é muito importante para evitar erros difíceis de detectar depois de termos escrito os programas

Exemplo

Crie uma função que valide a entrada dada pelo o usuário.

A Função deve receber a pergunta, o valor máximo e o valor mínimo a ser digitado.

Criando Funções de Validação

```
def faixaValores(pergunta, min, max):  
    while True:  
        val = int(input(pergunta))  
        if (val < min or val > max):  
            print('Entrada Inválida! Digite valor entre %d e %d' % (max, min))  
        else:  
            return val
```

Figura: Exemplo de função que valida uma entrada.

Parâmetros Opcionais

- Nem sempre precisamos passar todos os parâmetros de função
- É possível tornar alguns parâmetros opcionais, deixando um valor padrão caso o usuário não queira modificá-los

```
def barra():  
    print('-' * 40)
```

- A função barra não recebe nenhum parâmetro
- A quantidade de traços exibidos na tela pode ser alterado

```
def barra(n = 40, caractere='*'):  
    print(caractere * n)
```

Parâmetros Opcionais

- Os dois parâmetros da função barra são opcionais
- A função anteriormente implementada pode ser chamada de diversas formas:
 - ▶ barra()
 - ▶ barra(10)
 - ▶ barra(10, '=')
 - ▶ ...
- Os parâmetros opcionais são úteis para evitar a passagem desnecessária dos mesmos valores

Parâmetros Opcionais e obrigatórios

- É possível criar funções que possuem parâmetros opcionais e obrigatórios
- Os parâmetros obrigatórios não possuem valores pré-definidos na função

Exemplo de função com parâmetros obrigatórios e opcionais

```
def somar(num1, num2, imprimir=False):  
    res = num1 + num2  
    if imprimir:  
        print(res)  
    else:  
        return res
```

Nomeando Parâmetros

- Python suporta a chamada de uma função com vários parâmetros
- Porém, temos que fornecer os parâmetros na ordem em que elas foram definidas
- Quando se especifica o nome de uma função, é possível passá-las em qualquer ordem

```
def retangulo(largura, altura, caractere='*'):  
    linha = caractere * largura  
    for i in range(altura):  
        print(linha)
```

```
retangulo(3,4)
```

```
retangulo(largura=3, altura=4)
```

```
retangulo(caractere='-', altura=4, largura=3)
```

Funções como Parâmetro

- Um recurso poderoso do Python é permitir a passagem de funções como parâmetro
- Assim é possível combinar várias funções para realizar uma tarefa

```
def somar(a,b):  
    return a + b  
  
def subtracao(a,b):  
    return a-b  
  
def imprimir(a,b, operacao):  
    print(operacao(a,b))  
  
imprimir(5, 4,somar)  
imprimir(7, 3, subtracao)
```

Empacotamento e Desempacotamento de Parâmetros

- O Python possibilita passar parâmetros empacotados em uma lista

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

```
listaParametros = [3,4]  
somar(*listaParametros)
```

- Utiliza-se asteriscos para indicar que queremos descompactar a lista '*listaParametros*' utilizando os valores como parâmetros para a função **somar**
- No caso do exemplo anterior: `listaParametros[0]` e `listaParametros[1]` são usados como parâmetros da função `somar`

Empacotamento e Desempacotamento de Parâmetros

- Pode-se criar funções que recebem um número indeterminado de parâmetros utilizando lista de parâmetros

Função que soma com número indeterminado de parâmetros

```
def somar(*args):  
    res = 0  
    for i in args:  
        res += i  
    return res
```

```
somar(1,2)  
somar(1,2,5,7)  
...
```

Funções Lambda

- Funções lambda são funções sem nome (anônimas)
- Essas funções são parecidas com expressões, por isso que comumente são chamadas de expressões lambda

Exemplo de funções lambda

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

- A função acima recebe um valor como parâmetro e retorna o dobro desse valor
- Não é possível utilizar a palavra reservada **return** em um função lambda

Sintaxe da função lambda

lambda parâmetro1, parâmetro2, ..., parâmetroN : expressão

Funções Lambda

- As funções lambdas podem receber diversos parâmetros

Exemplo de função lambda para cálculo do aumento de um salário

```
aumento = lambda salario, perc: (salario*perc/100)  
aumento(2500, 15)
```

Criando Módulos

- Quando um programa fica muito grande, necessitamos armazenar as funções em outros arquivos
- Essa abordagem facilita a manutenção e a reutilização do código
- O Python facilita a criação de módulos → Todo arquivo **.py** é um módulo que pode ser importado por meio de um comando **import**

Criando Módulos

módulo verificarFaixa.py

```
def verificarEntrada(texto, min, max):  
    while True:  
        try:  
            val = int(input(texto))  
            if (val >= min and v <= max):  
                return val  
        else:  
            print('Digite um valor entre %d e %d' % (min, max))  
    except:  
        print('A entrada deve ser um valor inteiro!')
```

Criando Módulos

módulo soma.py

```
import entrada
```

```
lista = [ ]
```

```
for i in range(10):
```

```
    lista.append(entrada.verificarEntrada('Digite um número: ', 0, 20))
```

```
print('Soma = %d' % (sum(lista)))
```