

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Computational Thinking

PROF. EDUARDO GONDO

Funções

Em linguagens orientada à objetos chamamos de **métodos**, linguagens estruturadas chamamos de **procedimentos** e **funções**. Mesmo com denominações diferentes o objetivo é o mesmo:

- ▶ executar as mesmas instruções periodicamente
- ▶ simplificação do programa
- ▶ possibilidade de reaproveitamento
- ▶ facilidade de manutenção
- ▶ testes mais modulares
- ▶ organização melhor do conhecimento

Exemplos

Já temos usado algumas funções na linguagem Python:

- ▶ `print`
- ▶ `input`
- ▶ `int` e `float`
- ▶ entre outras

Agora, aprenderemos como criar as nossas próprias funções.

I Funções - Elementos

Alguns elementos para criarmos uma função:

- ▶ o que a função deve fazer
- ▶ um nome significativo para a função
- ▶ parâmetros que ela deve receber
- ▶ qual ou quais informações ela deve retornar
- ▶ também podemos ter funções que não retornam nada

Funções - Sintaxe

Vamos criar uma função que aplica um reajuste percentual sobre um valor monetário:

```
1 def aumento(valor, percentual):  
2     novoValor = (1 + percentual / 100) * valor  
3     print(novoValor)  
4  
5 aumento(350.00, 15)
```

- ▶ para definir uma função em Python, usamos a palavra `def`
- ▶ `valor` e `percentual` são os parâmetros da função
- ▶ esta função imprime o resultado do aumento na tela
- ▶ uma função só pode ser usada após sua implementação
- ▶ na linha 05 temos um exemplo de chamada da função

I Boas práticas

Algumas dicas para construir boas funções

- ▶ uma função faz apenas uma coisa e bem
- ▶ escreva pouco erre pouco: funções com poucas linhas são fáceis de corrigir erros ou eles não possuem erros
- ▶ use nomes de funções, parâmetros e variáveis significativos
- ▶ em geral, crie funções que retornem valores
- ▶ vamos reescrever a função anterior para retornar o valor do aumento

Funções - Aumento

```
1 def aumento(valor, percentual):
2     novoValor = (1 + percentual / 100) * valor
3     return novoValor
4
5 resultado = aumento(350.00, 15)
6 print(resultado)
```

- ▶ diferente de outras linguagens de programação, funções em Python podem retornar mais de um valor
- ▶ veja abaixo um exemplo:

```
1 def divisao(a, b):
2     return a // b, a % b
3
4 res = divisao(5, 3)
5 print(res)
```

- ▶ o retorno neste caso é uma **tupla**
- ▶ uma tupla representa um conjunto de dados
- ▶ ainda vamos aprender a manipular tuplas, fiquem tranquilos

Passagem de parâmetros

- ▶ podemos passar parâmetros para as funções de duas formas: **valor** e **referência**
- ▶ quando o parâmetro é por valor, podemos alterar o valor da variável dentro da função que nada acontece ao sair dela
- ▶ quando o parâmetro é passado por referência, qualquer alteração da variável dentro da função reflete no restante do programa
- ▶ na linguagem C ou VB podemos escolher o modo de passar os parâmetros
- ▶ já na linguagem Java há uma regra bem definida, se o parâmetro for um objeto, então a passagem é por referência
- ▶ caso contrário, para tipos básicos, a passagem é por valor

Passagem de parâmetros

- ▶ na linguagem Python, tudo são objetos: números inteiros, reais, string, etc
- ▶ logo a passagem de parâmetros sempre é por referência
- ▶ contudo, a modificação do parâmetro dependerá se o objeto é **imutável** ou **mutável**
- ▶ objetos imutáveis não são alterados: *int*, *float*, *bool*, *string*, *tuples*
- ▶ objetos mutáveis podem ser alterados: *list*, *dict*, *set*
- ▶ fique **atento** à esta informação pois isso pode ocasionar resultados inesperados no seu programa

Funções - Exemplos

Escreva um algoritmo que recebe um inteiro positivo n e calcula $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$. Por exemplo, se $n = 6$, então $6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$.

```
1  n = int(input("Informe n: "))
2  prod = 1
3  while n >= 1:
4      prod = prod * n
5      n = n - 1
6
7  print(prod)
```

```
1  def fatoria(n):
2      prod = 1
3      while n >= 1:
4          prod = prod * n
5          n = n - 1
6
7      return prod
8
9  fat = fatoria(6)
10 print(fat)
```

Funções - Exemplos

Dizemos que um inteiro positivo n é perfeito se for igual a soma de seus divisores positivos diferentes de n . Escreva um algoritmo que dado n inteiro positivo, verifica se ele é perfeito.

```
1  n = int(input("Informe n: "))
2  div = 1
3  soma = 0
4
5  while div < n:
6      if n % div == 0:
7          soma = soma + div
8      div = div + 1
9
10 if n == soma:
11     print("É perfeito")
12 else:
13     print("Não é perfeito")

1  def perfeito(n):
2      div = 1
3      soma = 0
4
5      while div < n:
6          if n % div == 0:
7              soma = soma +
                  div
8              div = div + 1
9
10     if n == soma:
11         return True
12     else:
13         return False
14
15 resp = perfeito(6)
16 print(resp)
```

Funções - Exemplos

Dado um número inteiro na base decimal, converta para sua representação binária.

```
1  n = int(input("Informe n: "))
2  pot = 1
3  soma = 0
4
5  while n != 0:
6      resto = n % 2
7      soma = soma +
          resto * pot
8      pot = pot * 10
9      n = n // 2
10
11 print(soma)
```

```
1  def decToBin(n):
2      pot = 1
3      soma = 0
4
5      while n != 0:
6          resto = n % 2
7          soma = soma +
              resto * pot
8          pot = pot * 10
9          n = n // 2
10
11     return soma
12
13
14 resp = decToBin(6)
15 print(resp)
```

Funções - Exemplos

- ▶ Dados dois números inteiro positivos a e b , escreva um algoritmo e uma função que encontra o menor número inteiro que é múltiplo do número a e do número b .
- ▶ Dados dois números inteiro positivos a e b , escreva um algoritmo e uma função que encontra o maior número inteiro que divide o número a e o número b .