

Subprogramação

Alex Sandro Costa e Everton Freitas Godinho

Exemplo motivacional

```
a = [4, 10, 2, 8]
maior = 0
pos = 0
for i in range(len(a)):
    if maior < a[i]:
        maior = a[i]
        pos = i
print("O maior número é " + str(maior) + " na posição " + str(pos + 1))

b = [9, 3, 5, 7]
maior = 0
pos = 0
for i in range(len(b)):
    if maior < b[i]:
        maior = b[i]
        pos = i
print("O maior número é " + str(maior) + " na posição " + str(pos + 1))
```

Subprogramação

- A subprogramação é o ato de transformar um trecho de código que se repete em uma função que pode ser usadas ao longo do programa quantas vezes forem necessárias
- Sem a utilização da subprogramação, a repetição de trechos de código se torna um problema por crescer o código desnecessariamente, tornando-o mais difícil de ser mantido no futuro

Exemplo motivacional como função

```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos
```

DEFINIÇÃO DA FUNÇÃO

```
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a) } CHAMADA DA FUNÇÃO  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))
```

```
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b) } CHAMADA DA FUNÇÃO  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```

Vantagens da subprogramação

- Toda vez que o programa precisar utilizar aquele trecho de código, basta chamar a função
- Se mudanças forem necessárias ou erros forem encontrados no trecho de código, basta alterar em apenas um local, na função
- Facilita na leitura do código do fonte e no entendimento do trecho de código pois podemos dar nomes significativos as funções

Fluxo de execução

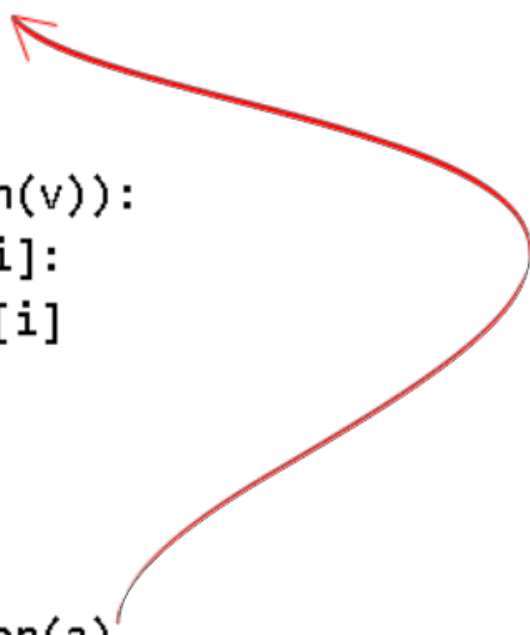
```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos
```

↓


```
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))  
  
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```

Fluxo de execução

```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos  
  
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))  
  
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```



Fluxo de execução



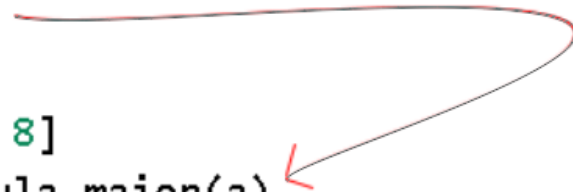
```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos
```

```
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))
```

```
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```


Fluxo de execução


```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos  
  
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))  
  
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```



Fluxo de execução

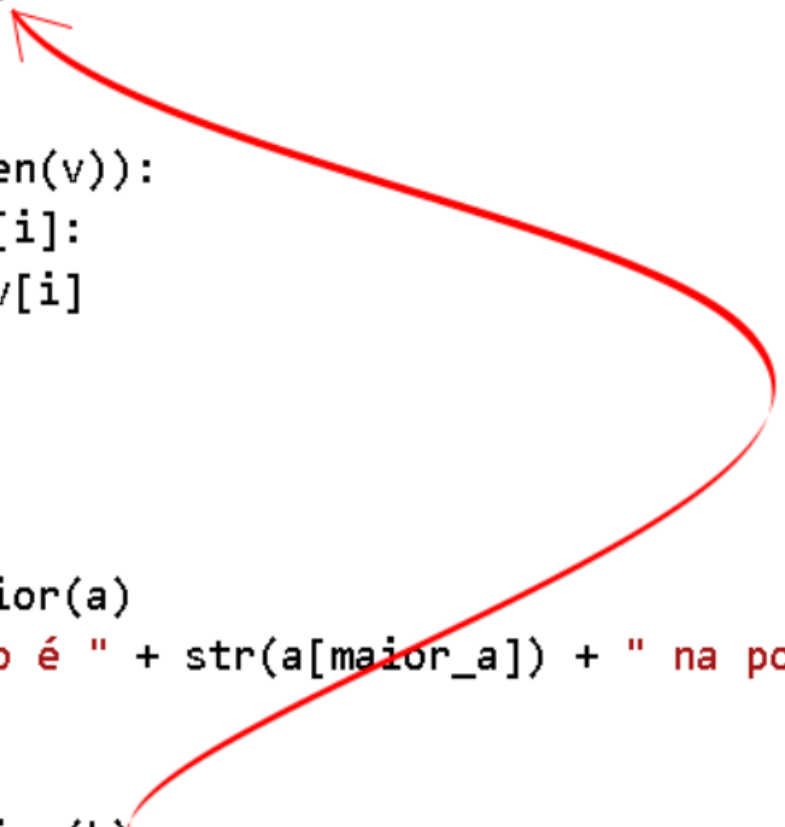
```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos
```

```
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))  
  
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```



Fluxo de execução

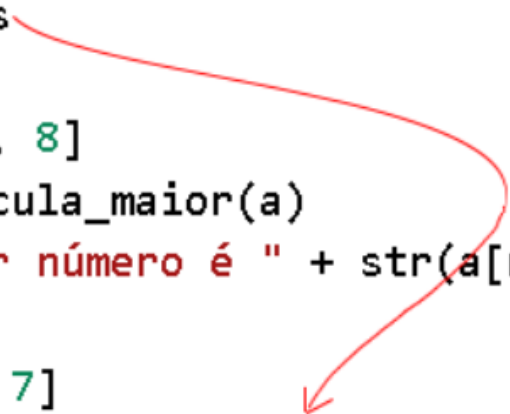
```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos
```



```
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))  
  
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```

Fluxo de execução

```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos  
  
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))  
  
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```



Fluxo de execução

```
def calcula_maior(v):  
    maior = 0  
    pos = 0  
    for i in range(len(v)):  
        if maior < v[i]:  
            maior = v[i]  
            pos = i  
    return pos
```

```
a = [4, 10, 2, 8]  
maior_a = calcula_maior(a)  
print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))
```

```
b = [9, 3, 5, 7]  
maior_b = calcula_maior(b)  
↓ print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```

Mais exemplos

```
def converte(escala, temperatura):  
    convertido = 0  
    if escala[0].lower() == "c":  
        convertido = temperatura * (9 / 5) + 32  
    if escala[0].lower() == "f":  
        convertido = (temperatura - 32) * (5 / 9)  
    return convertido
```

PARÂMETROS

COMANDOS

RETORNO

```
c = converte("Celsius", 20)  
print(c)  
f = converte("Fahrenheit", 120)  
print(f)
```

Pontos principais

- As funções são definidas antes do programa principal
- A função só será executada quando ela for chamada
- Programas podem possuir nenhuma ou muitas funções e essas funções podem ser chamadas nenhuma ou muitas vezes
- Funções podem ter nenhum ou vários parâmetros
- Funções podem retornar um valor ou não retornar valor nenhum. Se retornarem um valor a execução da função acaba quando atingirem o comando return, se não possuírem um valor a execução da função acaba na sua última linha de código

Escopo de variáveis

- Ao introduzirmos funções, podemos perceber que existem as variáveis declaradas dentro e fora das funções
- Então precisamos entender o escopo das variáveis, ou seja, onde a variável atua
- Temos as variáveis globais, que são declaradas no programa principal, fora das funções, que podem ser vistas tanto por funções quanto pelo programa
- Temos as variáveis locais, que são declaradas dentro da função, essas variáveis apenas podem ser vistas pela função, e são descartadas após a execução da função

Ilustrando esses conceitos

```
def calcula_maior(v):  
    LOCAL  
    maior = 0  
    LOCALS {  
        pos = 0  
        for i in range(len(v)):  
            if maior < v[i]:  
                maior = v[i]  
                pos = i  
    return pos
```

```
GLOBALS {  
    a = [4, 10, 2, 8]  
    maior_a = calcula_maior(a)  
    print("O maior número é " + str(a[maior_a]) + " na posição " + str(maior_a + 1))
```

```
GLOBALS {  
    b = [9, 3, 5, 7]  
    maior_b = calcula_maior(b)  
    print("O maior número é " + str(b[maior_b]) + " na posição " + str(maior_b + 1))
```

Ilustrando esses conceitos

```
                                LOCAIS
                                {
def converte(escala, temperatura):
LOCAL {convertido = 0
      if escala[0].lower() == "c":
          convertido = temperatura * (9 / 5) + 32
      if escala[0].lower() == "f":
          convertido = (temperatura - 32) * (5 / 9)
      return convertido
      }

GLOBAIS { c = converte("Celsius", 20)
        print(c)
        f = converte("Fahrenheit", 120)
        print(f)
        }
```

Ilustrando esses conceitos

- As variáveis globais podem ser acessadas na função
- Apesar de possível, o uso de parâmetros é mais adequado que o acesso direto a variável
- As variáveis locais não podem ser acessadas no programa principal

```
def funcao():  
    print(a)  
    print(b)  
    LOCAL { m = 56  
    return m
```

```
GLOBALS { a = 12  
          b = 30
```

```
print(funcao())
```

```
print(m)
```

Mais exemplos

Função sem parâmetro e sem retorno

```
def menu():  
    print("1. Soma.");  
    print("2. Subtração.");  
    print("3. Multiplicação.");  
    print("4. Divisão.");  
    print("0. Sair");  
  
menu()  
opcao = int(input("Digite a opção desejada: "))
```

Agora é sua vez!

1. Escreva uma função que converta graus para radianos e uma que converta radianos para graus.
2. Escreva uma função que calcule o fatorial de um número.
3. Escreva uma função que verifique se um número é primo.
4. Escreva uma função que verifique quantas vezes um número é divisível por outro.