RACIOCÍNIO ALGORÍTMICO

FUNÇÕES

FUNÇÕES

DEFINIÇÃO

Funções são fragmentos de códigos que permitem organizar algoritmos e reduzir sua repetição. A partir do uso correto de funções um projeto fica mais fácil de manter, bem como torna-se mais legível, uma vez que funcionalidades específicas podem ser colocadas em poucas linhas.

```
def nomedafuncao(param1, param2, ...):
    return resultado
```

```
def soma(num1, num2):
    s = num1 + num2
    return s
```

FUNÇÕES

Exemplo de aplicação 1: *Elaborar um programa que cadastra contatos de agenda telefônica. A função de cadastro deve ser realizada dentro de uma função chamada inserir, que recebe como parâmetros o nome e o telefone do contato, bem como a agenda de contatos.*

```
01. agenda_telefonica = {}
02.
03. def inserir(nome, telefone, agenda):
04.     agenda[nome] = telefone
05.
06. while True:
07.     nome = input("Digite o nome do contato: ")
08.     telefone = input("Digite o telefone do contato: ")
09.     inserir(nome, telefone, agenda_telefonica)
10.     if input("Gostaria de adicionar um novo contato? (s/n) ") == "n":
11.         break
12.
13. print(agenda_telefonica)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic01.py
Digite o nome do contato: Maria
Digite o telefone do contato: (41) 98765-4321
Gostaria de adicionar um novo contato? (s/n) s
Digite o nome do contato: João
Digite o telefone do contato: (11) 98877-6655
Gostaria de adicionar um novo contato? (s/n) n
{'Maria', '(41) 98765-4321'), ('João', '(11) 98877-6655'}

Process finished with exit code 0
```

FUNÇÕES

Exemplo de aplicação 2: *Elaborar um programa que cadastra contatos de agenda telefônica. A função de cadastro deve ser realizada dentro de uma função chamada inserir, que recebe como parâmetros o nome e o telefone do contato, bem como a agenda de contatos. A função deve verificar se o contato já existe e, em caso positivo, perguntar se o telefone deve ser modificado, retornando true ou false de acordo com a inclusão/moficação ter sido executada ou não na agenda.*

```
01. agenda_telefonica = {}
02.
03. def inserir(nome, telefone, agenda):
04.     if nome in agenda:
05.         if input("Contato já cadastrado. Deseja alterar o telefone? (s/n) ") == "n":
06.             return False
07.     agenda[nome] = telefone
08.     return True
09.
10. while True:
11.     nome = input("Digite o nome do contato: ")
12.     telefone = input("Digite o telefone do contato: ")
13.     if inserir(nome, telefone, agenda_telefonica):
14.         print("Contato adicionado ou atualizado com sucesso!")
15.     else:
16.         print("Falha ao tentar adicionar o contato!")
17.     if input("Gostaria de adicionar um novo contato? (s/n) ") == "n":
18.         break
19.
20. print(agenda_telefonica)
```

FUNÇÕES

ESCOPO DE FUNÇÃO

```
01. def minha_funcao():
02.     x = 10
03.     print(f"Valor de x dentro da função: {x}")
04.
05. x = 20
06. minha_funcao()
07. print(f"Valor de x fora da função: {x}")
```

este exemplo vai gerar a seguinte saída:

```
Valor de x dentro da função: 10
Valor de x fora da função: 20
```

FUNÇÕES

PARÂMETROS COM VALOR PADRÃO

Funções em Python permitem que sejam passados parâmetros com valores padrão, os quais podem ser omitidos no momento de chamar a função.

```
def nomedafuncao(param1 = valor_padrao, param2 = valor_padrao, ...):
```

Para exemplificar, a função inserir() do exemplos anteriores poderia receber como um valor padrão para o telefone o valor “Sem telefone”.

```
03. def inserir(nome, agenda, telefone = “Sem telefone”):
```

FUNÇÕES

PARÂMETROS ARBITRÁRIOS

Exemplo de aplicação 3: *Elaborar um programa que usa uma função chamada* ***somar()*** *que efetua a soma de um número aleatório de números informados, retornando o resultado da operação.*

```
01. def somar(*numeros):
02.     soma = 0
03.     for i in range(len(numeros)):
04.         soma += numeros[i]
05.     return soma
06.
07. resultado = somar(2, 3, 4, 5, 8)
08. print(f"A soma dos números é {resultado}")
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic03.py
A soma dos números é 22

Process finished with exit code 0
```

FUNÇÕES

RETORNOS MÚLTIPLOS (TUPLAS)

Exemplo de aplicação 4: *Elaborar um aplicativo que faz uso de uma função que recebe diversos valores numéricos e retorna qual são os valores maior e menor da lista.*

```
01. def maior_menor(*numeros):
02.     maior = -1000000
03.     menor = 1000000
04.     for numero in numeros:
05.         if numero > maior:
06.             maior = numero
07.         if numero < menor:
08.             menor = numero
09.     return maior, menor
10.
11. resultado = maior_menor(7, 15, 3, 22, 1, 8)
12. print(f"O maior número é {resultado[0]}, e o menor número é {resultado[1]}")
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic04.py
O maior número é 22, e o menor número é 1

Process finished with exit code 0
```

FUNÇÕES

ENCADEAMENTO DE FUNÇÕES

Exemplo de aplicação 5: *Elaborar um programa que, aplicando a fórmula de bháskara por funções, encontre as raízes de um polinômio do segundo grau. A saber:*

$$x = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{e} \quad \Delta = b^2 - 4ac$$

```
01. def delta(a, b, c):
02.     return b*b - 4*a*c
03.
04. def bhaskara(a, b, c):
05.     d = delta(a, b, c)
06.     if d < 0:
07.         print("As raízes são imaginárias")
08.         return 0, 0, False
09.     else:
10.         x1 = (-b + d ** 0.5) / 2 * a
11.         x2 = (-b - d ** 0.5) / 2 * a
12.         return x1, x2, True
13.
14. result1 = bhaskara(1, 3, 1)
15. if result1[2]:
16.     print(f"As raízes são {result1[0]} e {result1[1]}")
17.
18. result2 = bhaskara(1, 2, 1)
19. if result2[2]:
20.     print(f"As raízes são {result2[0]} e {result2[1]}")
21.
22. result3 = bhaskara(1, 1, 1)
23. if result3[2]:
24.     print(f"As raízes são {result3[0]} e {result3[1]}")
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic05.py
As raízes são -0.3819660112501051 e -2.618033988749895
As raízes são -1.0 e -1.0
As raízes são imaginárias

Process finished with exit code 0
```

FUNÇÕES

EXERCÍCIOS

Funções

1. Criar um programa que calcula a partir de uma função o fatorial de um número. Exemplo: Fatorial de 5 => 5! = 5.4.3.2.1. Obs.: Por propriedade, 0! = 1.

2. Criar um programa que calcula o fatorial de um número, mas de forma recursiva, ou seja, chamando a própria função fatorial de dentro dela mesma.

3. Criar um programa que receba uma lista de números e retorne a média dos mesmos.

FUNÇÕES

EXERCÍCIOS

Desafio

4. Criar um programa que, fazendo uso de funções, cadastra contatos em uma agenda telefônica, podendo excluir estes contatos. Deve ser exibido um menu com as opções: inserir, remover e sair.