Funções em Python

Funções

Na programação, funções são blocos de código que realizam determinadas tarefas que normalmente precisam ser executadas diversas vezes dentro de uma aplicação.

Quando surge essa necessidade, para que várias instruções não precisem ser repetidas, elas são agrupadas em uma função, à qual é dado um nome e que poderá ser chamada/executada em diferentes partes do programa.

Como declarar?

A palavra `def` define o início da função

Juntamente com a palavra `pass` podemos criar uma função que não faz absolutamente nada.

```
def foo():
    pass
```

O comando `return` interrompe a execução da função e define o retorno

Exemplo de funções.

```
def foo():
    return "retorno da função"
```

Exemplo

```
input_name = “alexandre”

input_idade = 25


def mostra_idade(nome, idade):

    return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)


frase = mostra_idade(input_name, input_idade)

print(frase)
```

Exemplo

```
input_name = “alexandre”

input_idade = 25
```

```
def mostra_idade(nome, idade):

    return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)


frase = mostra_idade(input_name, input_idade)

print(frase)
```

Exemplo

```
input_name = “alexandre”

input_idade = 25
```

```
def mostra_idade(nome, idade):

    return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)
```

```
frase = mostra_idade(input_name, input_idade)

print(frase)
```

Exemplo

```
input_name = “alexandre”

input_idade = 25
```

```
def mostra_idade(nome, idade):

    return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)
```

```
frase = mostra_idade(input_name, input_idade)

print(frase)
```

Argumentos de funções

Argumentos (ou parâmetros) são como variáveis que recebem seus valores iniciais do chamador

Essas variáveis, assim como outras definidas dentro da função são ditas locais, isto é, só existem no lugar onde foram definidas

Ao retornar ao ponto de chamada, as variáveis locais são descartadas

Os argumentos podem ser opcionais ou obrigatórios

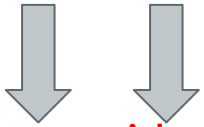
Os argumentos podem receber funções

Os argumentos podem ser nomeados ou posicionais

Os argumentos podem ser empacotados

Exemplo

Argumentos ou Parâmetros

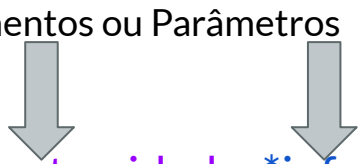


```
def mostra_idade(nome, idade):

    return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)
```

Exemplo

```
def mostra_idade(nome, idade):
    return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)
```




```
def print_resultado(mostra_idade, *info):
    print(mostra_idade(*info))
```

Argumentos default

É possível dar valores default a argumentos Se o chamador não especificar valores para esses argumentos, os defaults são usados

```
def mostra_idade(nome=`alexandre`, idade=25):
        return “meu nome é: “ + nome + “ minha idade é: “ + str(idade)
```

Passando argumentos com nomes

É possível passar os argumentos sem empregar a ordem de definição desde que se nomeie cada valor passado com o nome do argumento correspondente

- Ex.:

```
>>> def f(nome,saudacao="Oi",pontuacao="!!"):
        return saudacao+","+nome+pontuacao

>>> print f(saudacao="Valeu",nome="Joao")
Valeu,Joao!!
```

Documentando Funções

Ao invés de usar comentários para descrever o que uma função, é mais vantajoso usar docstrings

Uma constante string escrita logo após o cabeçalho da função (comando def)

Permite o acesso à documentação a partir do interpretador, usando a notação função . __doc__

```
>>> def fat(n):
...     "Retorna o fatorial de n."
...     for i in range(n-1,1,-1): n*=i
...     return n
...
>>> fat(4)
24
>>> print fat.__doc__
Retorna o fatorial de n.
```

Lista de parâmetros variável

Se o último argumento de uma definição de função começa com *, todos os valores passados, a partir daquele, são postos numa tupla

Ex.:

```
>>> def imprime(nome,*atributos):
...     print nome,atributos
...
>>> imprime ('a',1,2,'b')
a (1, 2, 'b')
>>> def media(*valores):
...   total=0.0
...   for x in valores: total+=x
...   return total/len(valores)
...
>>> media(1,2,3,4)
2.5
```

Funções definidas em funções

Funções podem ser definidas dentro de funções

Ex.:

```
>>> def f(x):
        def g(y): return x*y
        return g(2)

>>> print f(4)
8
```

Exercícios

Crie uma função que receba um nome e um sobrenome e retorne o nome completo.

Crie uma função que receba o nome e a idade e retorno um text “meu nome é xxxx e minha idade é xx”.

Crie uma lista e uma função que receba um valor e armazene na lista criada.

Cria uma função que receba como parâmetro 2 números e retorne a soma desses números.

Crie um dicionário com nome e telefone e crie uma função que receba o nome e retorne o telefone desse dicionário.

Cria uma lista de dicionários com nome e telefone e cria uma função que receba como argumento o nome e retorne o telefone referente a esse nome

Cria uma função que receba como argumento uma lista de numeros retorne uma tupla com o maior e o menor valor desta lista.

Exercícios

Crie uma função que receba um número e retorne True se o número for par ou False se for ímpar

Crie uma função que receba o nome e a idade e retorne um dicionário com os valores de entrada.

Crie uma lista de textos e uma função que receba um texto e retorna True caso esse texto esteja na lista caso contrário retorna False.

Crie uma lista e uma função que receba um valor e remova da lista caso esse valor esteja na lista.

Crie uma função de faça a transposição de uma matriz, transformando cada linha da matriz em coluna.