LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Funções em Python

Todo programador fala sobre funções, e como ela funciona, é importante todo programador pensar na solução utilizando funções. Veja a explicação mais simples que você poderá usar agora.

As funções fazem três coisas:

1. Nomeiam partes do código.
2. Recebem argumentos
3. Usando 1 e 2 permitem que crie seus ***"miniscripts"*** ou "pequenos comandos"

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

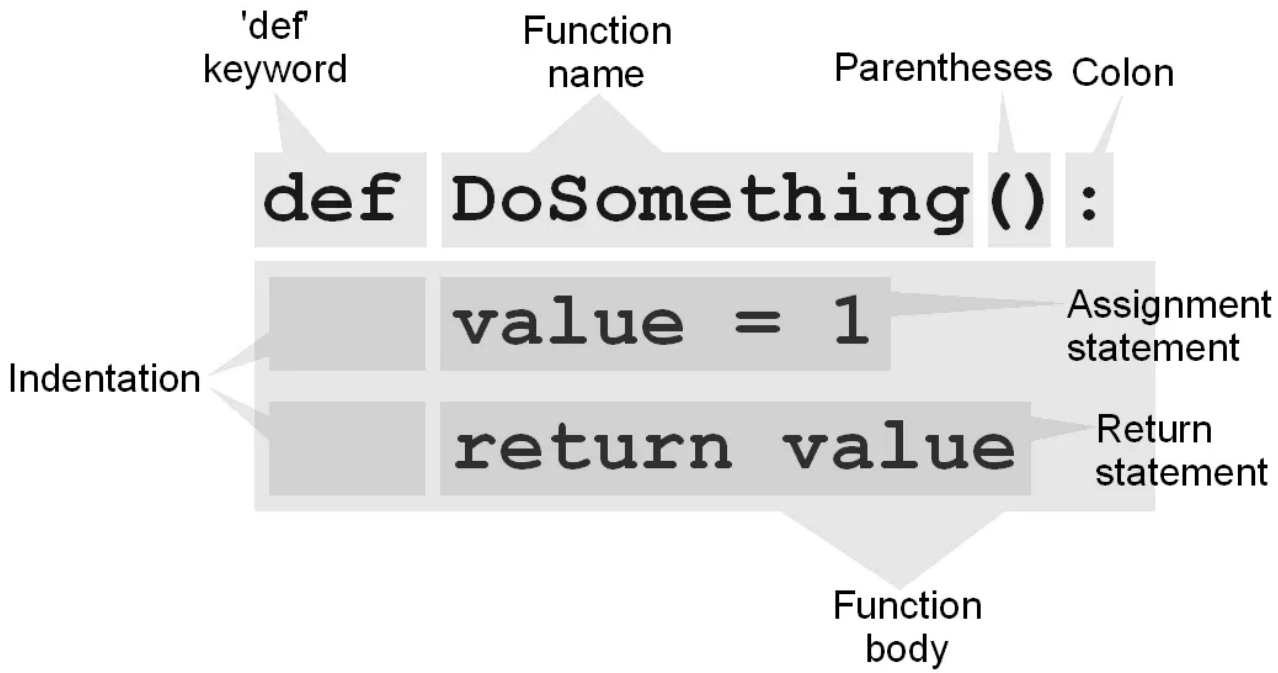
Por que utilizar funções em Python?

Funções são especialmente interessantes para **isolar uma tarefa específica** em um trecho de programa, isso significa que a solução de um problema **seja reutilizada em outras partes do programa**, sem precisar **repetir código**.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Como declarar uma função em Python.



Entre os ***parênteses*** é possível inserir ***argumentos*** nos ***parâmetros*** para serem utilizados ***dentro do escopo da função***. ***(Funções com parâmetros)***

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Como declarar uma função em Python.

```
def nome_da_função (parametro1, parametro2)
    # Código a ser executado
    return resultado
```

Parâmetros: São os valores que a função aceita como entrada. Eles são colocados entre parênteses e separados por vírgulas. Parâmetros são opcionais, e uma função pode ter zero ou mais parâmetros.

return: A palavra-chave return é usada para enviar um valor de volta quando a função é chamada. O return é opcional e pode ser omitido se a função não precisar retornar nenhum valor.

Corpo da Função: É o bloco de código indentado que contém as instruções que a função executa. Este é o local onde a tarefa específica da função é realizada.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Variáveis locais e globais

Quando usamos funções, começamos a trabalhar com **variáveis internas ou locais** e com **variáveis externas ou globais**. A diferença entre elas é a **visibilidade ou escopo**.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Variáveis locais

Uma variável local a função **existe apenas dentro dela**, sendo normalmente inicializada a cada chamada. **Não podemos acessar o valor de uma variável local fora da função** que a criou.

```
#função com variável local

def valor():
    a = 7 #variável local
    print("Variável local: ", a)



#chamada da função
valor()
```

Uma variável local a uma função existe somente dentro dela, sendo inicializada a cada chamada da função.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Variáveis globais

Uma variável global é definida **fora de uma função, podendo ser vista por todas as funções**.

```
#função com variável global

VALOR_DE_PI = 3.14

def valor_de_pi():
    print("Variável global: ", VALOR_DE_PI)

#chamada da função
valor_de_pi()
```

Uma bom uso de variáveis globais é **guardar valores constantes e que devem ser acessíveis por todas as funções** do programa.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Variáveis globais

Exemplo de um programa que calcula o volume de uma lata de refrigerante, utilizando a variável global **VALOR_DE_PI**.

```
#variável global usada dentro da função
VALOR_DE_PI = 3.14

def calcula_volume(raio, altura):
    volume = VALOR_DE_PI * raio ** 2 * altura
    print("Volume Calculado:", volume)

#chamar função em Python
calcula_volume(5, 15)
```

Podemos observar que a função recebe dois **parâmetros:** raio e altura.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Variáveis locais e globais

Uma variável local a função **existe apenas dentro dela**, sendo normalmente inicializada a cada chamada. **Não podemos acessar o valor de uma variável local fora da função** que a criou.

```
#função com variáveis locais e globais
a = 5 #variável global
def muda_valor():
    a = 7 #variável local
    print("Dentro da função: ", a)

print("Antes de mudar", a)
#chamada da função
muda_valor()
print("Depois de mudar", a)
```

Uma variável local a uma função existe somente dentro dela, sendo inicializada a cada chamada da função.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Variáveis locais e globais

Para modificar o valor de uma variável global dentro de uma função, **devemos informar que estamos usando variável global antes de inicializá-la**.

```
#modificando a variável global
a = 5 #variável global
def muda_valor():
    global a #variável global
    a = 7 #variável global
    print("Dentro da função: ", a)

print("Antes de mudar", a)
muda_valor()
print("Depois de mudar", a)
```

Tente definir variáveis globais de forma a facilitar a leitura de seu código fonte, pode utilizar um prefixo. Exemplo **G_PI = 3.14 G = global**

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Função sem argumento e função com argumento.

```python
#função sem argumento em Python
def hello_world():
    print("Olá mundo!")
```

```python
#função com argumento obrigatório em Python
def hello_world(msg):
    print(msg)
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Exemplo de funções utilizando o *return* e o *print*

```
#função calcular média em Python
def calc_media(n1, n2):
    media = float(n1+n2)/2
    print(media)
```

```
#função calcular média em Python
def calc_media(n1, n2):
    media = float(n1+n2)/2
    return media
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Função com parâmetro opcional

Caso não seja informado, será considerado o valor opcional.

```python
#função com parâmetro opcional em Python
import math


def potencia(base, expoente=2):
    #eleva número base à uma potência!
    if expoente == 0:
        return 1
    return math.pow(base, expoente)
```

A importação de uma biblioteca em Python, utiliza a palavra reservada ***import***

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Funções em Python

O empacotamento (packing) em funções Python refere-se à capacidade de uma função aceitar um número variável de argumentos.

Isso é feito usando os conceitos de argumentos posicionais, argumentos padrão e empacotamento de argumentos

```
#funções em Python - empacotamento
def contador(*num):
        print(num)

contador(1,2,3)
contador(1,3)

def soma_todos(*args):
        return sum(args)

print(soma_todos(1, 2, 3))
print(soma_todos(10, 20, 30, 40))
```

VAMOS PRATICAR.

EXEMPLOS E ATIVIDADES DE FUNÇÕES