



PROFº JANSEN LEITE.

► O que é uma função?

Função é um **bloco de códigos** com uma **finalidade específica** e que permite sua **reutilização** em todo o projeto.

EXEMPLO:

```
exemplo.py > ...  
1  def saudacao():  
2      print("Olá, mundo!")  
3
```

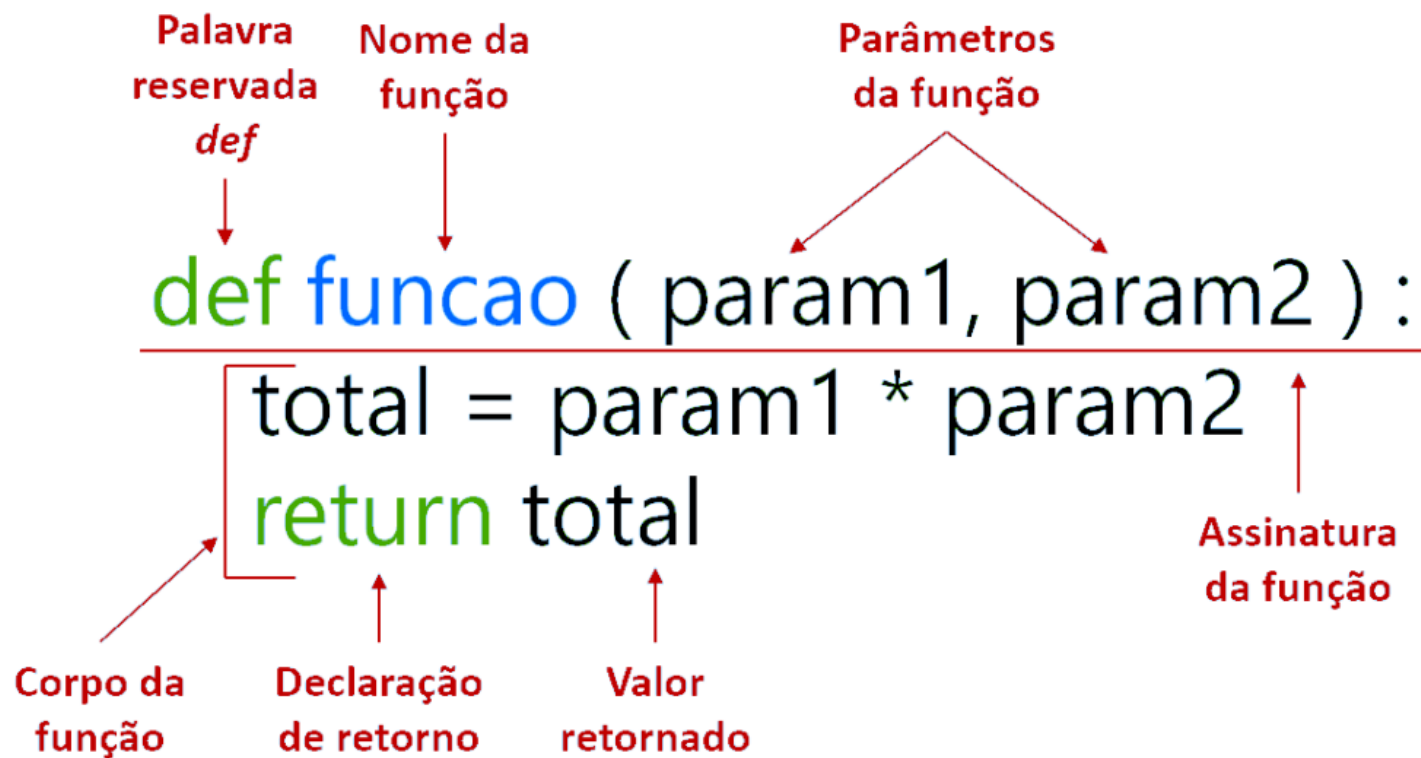
Python

Vantagens de se utilizar funções?

- ▶ Reutilização do código em qualquer modulo de seu projeto;
- ▶ Padronização do código seguindo as boas práticas de desenvolvimento;
- ▶ Permite implementação de Teste Unitários automatizados com a biblioteca unittest;
- ▶ Manutenções eficientes;
- ▶ Legibilidade;
- ▶ Redução de Bugs (Códigos mais coesos);

```
teste_automatizado.py > TestUnit1
1  import unittest
2  from programa_recebe_dados import programa_recebe_dados
3
4  class TestUnit1(unittest.TestCase):
5
6      def test_soma(self):
7          self.assertEqual(somar(2, 3), 5)
8          self.assertEqual(somar(-1, 1), 0)
9
10     def test_subtracao(self):
11         self.assertEqual(subtrair(5, 3), 2)
12
13     def test_multiplicacao(self):
14         self.assertEqual(multiplicar(4, 2), 8)
15
```

Como Declarar uma Função em Python



```
def funcao(param1, param2):  
    total = param1 * param2  
    return total
```

► O que é módulo?

Módulo é um arquivo que armazena escopos de códigos como classes, funções, variáveis e bibliotecas.

calculadora.py X

calculadora.py > ...

```
1  def somar(a, b):
2      return a + b
3
4  def subtrair(a, b):
5      return a - b
6
7  def multiplicar(a, b):
8      return a * b
9
10 def dividir(a, b):
11     if b == 0:
12         raise ValueError("Divisão por zero não permitida")
13     return a / b
14
```

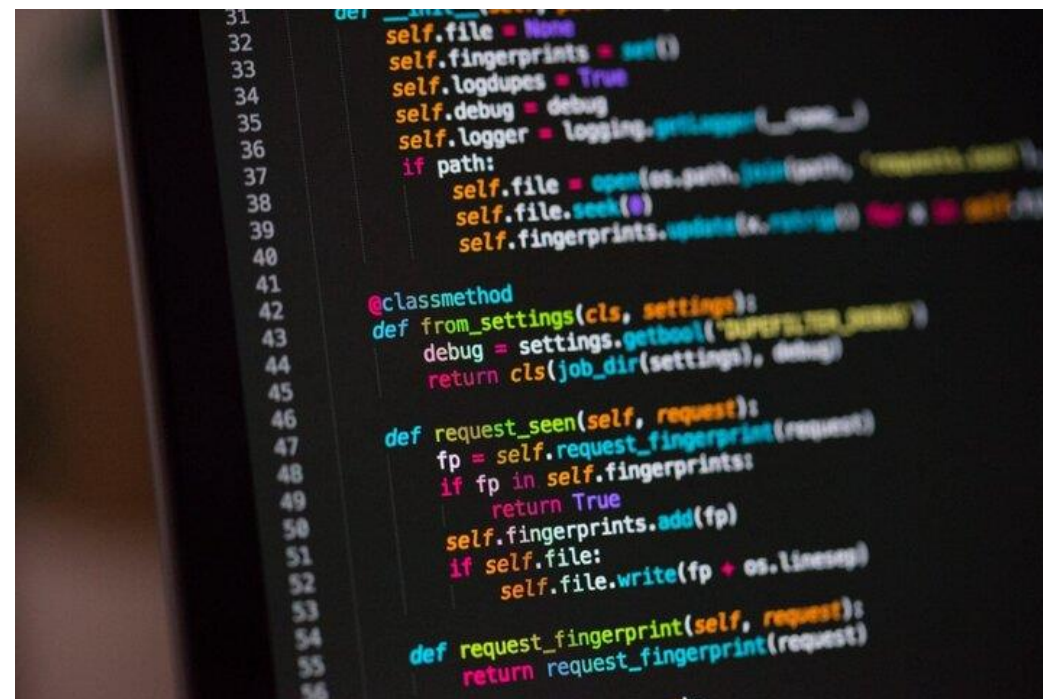
VANTAGENS DE SE UTILIZAR MÓDULOS

- ▶ Importação e reutilização;
- ▶ Colaboração;
- ▶ Reaproveitamento de Bibliotecas;
- ▶ Legibilidade;
- ▶ Manutenção fatorada;

Python

► O que são Bibliotecas?

É uma coleção de módulos e funções pré-programadas que oferecem funcionalidades prontas para uso em outros projetos, evitando a necessidade de escrever o código do zero.



```

31
32 self.file = None
33 self.fingerprints = set()
34 self.logdups = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'),
39                     'a')
40     self.file.seek(0)
41     self.fingerprints.update(self._get_fingerprints())
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('SUPERMAN_DEBUG')
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)

```

► Como importar uma biblioteca Nativa?

No início de seu script basta inserir o comando **import** seguido do nome do **módulo**.

Exemplo:

```
1  import math
2
3  def funcao(param1, param2):
4      total = param1 * param2 * math.pi
5      return total
6
```


Python

C:\Windows\system32\cmd.exe

```
C:\Users>pip install pandas
Collecting pandas
  Downloading pandas-2.3.3-cp313-cp313-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.26.0 in c:\users\oreonadmin\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.3.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\oreonadmin\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in c:\users\oreonadmin\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Downloading pandas-2.3.3-cp313-cp313-win_amd64.whl (11.0 MB)
----- 11.0/11.0 MB 11.3 MB/s 0:00:00
Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.3.3 pytz-2025.2 tzdata-2025.2
```

Par
est
inst
bas
cor
nor

► Pronto! Instalado...

Agora você já pode usar a biblioteca em seus projetos Python.

```
python Copy code

import pandas as pd

# Carregando dados CSV
df = pd.read_csv('caminho_para_o_arquivo.csv')

# Carregando dados Excel
df = pd.read_excel('caminho_para_o_arquivo.xlsx')

# Carregando dados de uma base de dados SQL
import sqlalchemy
engine = sqlalchemy.create_engine('sqlite:///caminho_para_o_banco_de_dados.db')
df = pd.read_sql_table('nome_da_tabela', engine)
```

► O que são alias?

Alias são “apelidos” ou nomes alternativos dados para os módulos da biblioteca. Isso permite com que você personalize seus códigos.

```
1  import math as m
2
3  def funcao(param1, param2):
4      total = param1 * param2 * m.pi
5      return total
```

► Como declarar os Alias?

```
import math
```

Invoca a Forma Nativa

```
math.pi
```

```
import math as m
```

```
m.pi
```

Invoca modificando por um ALIAS (m)

Invoca somente o método

```
from math import *
```

```
pi
```

```
from math import pi
```

```
pi
```

Invoca nativamente e somente o método declarado.

► O que são classes no Python?

Classes são modelos criados para manipular objetos (instanciar) características e atributos aos métodos.

```
class programa_recebe_dados:

    def __init__(self):
        print("\n##### SEJA BEM VINDO AO SISTEMA #####\n")
```

Manipulando Classes no Python

```
1 class Aluno:
2     def __init__(self, nome, idade):
3         self.nome = nome
4         self.idade = idade
5         self.notas = []
6
7     def adicionar_nota(self, nota):
8         self.notas.append(nota)
9
10    def calcular_media(self):
11        if self.notas:
12            return sum(self.notas) / len(self.notas)
13        else:
14            return 0
```

Nome da Classe

Atributos da Classe

Método da Classe

Método da Classe

► O que são Construtores?

São métodos invocados automaticamente que ao criar um objeto de uma classe, inicializam seus atributos.

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

pessoa1 = Pessoa("Maria", 25)

print(f"Nome: {pessoa1.nome}, Idade: {pessoa1.idade}")
```

► O que são objetos?

Objeto é uma entidade fundamental que encapsula dados (atributos) e comportamentos (métodos).

```
18 # Métodos do Objeto
19 aluno1 = Aluno('João', 15)
20 aluno1.adicionar_nota(8)
21 aluno1.adicionar_nota(7)
22 print(f'Média: {aluno1.calcular_media()}')
23
24 aluno2 = Aluno('Maria', 16)
25 aluno2.adicionar_nota(9)
26 print(f'Média: {aluno2.calcular_media()}')
```

Métodos do objeto aluno1

Métodos do objeto aluno2

Python



**Agora vamos
praticar!!!**