



# FUNÇÕES

Autoria de [Carolina Soares](#)

# O QUE É UMA FUNÇÃO

É um bloco de código que realiza certa funcionalidade, a fim de simplificar o código.

```
def funcao():  
    # Bloco de código
```

# CRIANDO UMA FUNÇÃO

```
def imprimirMsg():  
    print("Criando uma função com Python!")
```

# CHAMANDO UMA FUNÇÃO

```
def funcao():  
    # Bloco de código  
funcao()
```

# CRIANDO E CHAMANDO UMA FUNÇÃO

```
def imprimirMsg():
```

```
    print("Uma função Python!")
```

```
imprimirMsg()
```

Saída do código:

Uma função Python!

# PALAVRA RESERVADA NA FUNÇÃO

```
def funcao():  
    pass # Nada vai acontecer  
funcao()
```

# PARÂMETROS NA FUNÇÃO

```
def funcao(valor1, valor2):
```

```
    # Bloco de código
```

```
funcao(2, 3)
```



As funções podem ter parâmetros de diversos tipos: `float`, `string`, `boolean`, `list`, `map`...

# PASSANDO PARÂMETROS NA FUNÇÃO

```
# Recebendo dados
```

Saída do código:

```
def somarNumeros(num1, num2): 7
```

```
    soma = num1 + num2
```

```
    print(soma)
```

```
somarNumeros(5, 2)
```



# PASSANDO PARÂMETROS NA FUNÇÃO

```
pessoas = ["Maria", "Pedro", "José", "Eduarda"]
```

```
def addValorNaLista(lista, valor):
```

```
    lista.append(valor)
```

```
def removerValorNaLista():
```

```
    lista.remove(valor)
```

```
def contarTamanhoDaLista(lista):
```

```
    print(len(lista))
```

```
addValorNaLista(pessoas, "Letícia")
```

```
removerValorNaLista(pessoas, "Pedro")
```

```
contarTamanhoDaLista(pessoas)
```

```
verificarSeExisteNaLista(pessoas, "Ricardo")
```

# PARÂMETROS PADRÕES NA FUNÇÃO

```
def somarNumeros(num1=0, num2=0):  
    soma = num1 + num2  
    print(soma)  
somarNumeros()
```

Saída do código:  
0

# PARÂMETROS ARGS E KWARGS

# Número variável de parâmetros

```
def mostrarNumeros(*args):
```

```
    print(args)
```

```
    print(type(args))
```

```
mostrarNumeros(10, 20, 30, 40)
```

Saída:

```
(10, 20, 30, 40)
```

```
<class 'tuple'>
```

# Número variável de parâmetros nomeados

```
def guardarPessoas(**kwargs):
```

```
    print(kwargs)
```

```
    print(type(kwargs))
```

```
guardarPessoas(nome='João', idade=35)
```

Saída:

```
{'nome': 'João', 'idade': 35}
```

```
<class 'dict'>
```

# RETORNANDO VALORES NA FUNÇÃO

```
def funcao(valor1, valor2):  
    return valor1 + valor2  
  
somaValores = funcao(2, 3)
```

# RETORNANDO UM VALOR NA FUNÇÃO

```
def somarNumeros(num1, num2):  
    return num1 + num2  
  
soma = somarNumeros(5, 2)  
print(soma)
```

Saída do código:  
7

# RETORNANDO UM VALOR NA FUNÇÃO

```
def contarTamanhoDaLista(lista):  
    return len(lista)
```

Saída do código:

4

```
pessoas = ["Maria", "Pedro", "José", "Eduarda"]  
tamanhoLista = contarTamanhoDaLista(pessoas)  
print(f"A lista tem tamanho: {tamanhoLista}")
```

# RETORNANDO MÚLTIPLOS VALORES

```
def somarNumerosEFazerMedia(num1, num2):  
    soma = num1 + num2  
    media = soma / 2  
    return soma, media  
print(somarNumerosEFazerMedia(10, 2))
```

Saída do código:  
(12, 6.0)

# CRIANDO BOAS FUNÇÕES



# MELHORANDO FUNÇÕES

Função 1:

```
def somarNumeros():  
    print(5 + 2)
```

Função 2:

```
def somarNumeros(num1, num2):  
    print(num1 + num2)  
  
somarNumeros(5, 2)
```

Função 3:

```
def somarNumeros(num1, num2):  
    return num1 + num2  
  
print(somarNumeros(5, 2))
```

# MELHORANDO FUNÇÕES

## Exemplo 1:

```
def somarNumerosEFazerMedia(num1, num2):  
    soma = num1 + num2  
    media = soma / 2  
    return soma, media  
  
print(somarNumerosEFazerMedia(10, 2))
```

## Exemplo 2:

```
def somarNumeros(num1, num2):  
    return num1 + num2  
  
def fazerMedia(soma_valores):  
    return soma_valores / 2  
  
soma = somarNumeros(10, 2)  
media = fazerMedia(soma)  
print(soma, media)
```

**MAIS FUNÇÕES**

# FUNÇÕES LAMBDA

Uma função lambda também chamada de função anônima, é uma função simples que não precisa ser nomeada e são úteis para resolver problemas simples, onde não precisa ser reutilizada no código.

```
f = lambda {argumentos}: {expressão}
```

# FUNÇÕES LAMBDA

```
doblo = lambda x: x * 2
```

Saída do código:

```
print(doblo(3))
```

6

# FUNÇÕES LAMBDA

Função tradicional:

```
def dobrarNumero(numero):  
    return numero * 2  
  
dobrarNumero(3)
```

Função anônima (lambda):

```
doblo = lambda x: x * 2  
  
doblo(3)
```

# FUNÇÕES LAMBDA

```
soma = lambda x, y: x + y
```

Saída do código:

```
print(soma(2, 3))
```

5

# FUNÇÕES EMBUTIDAS

## Funções comuns:

- ❑ `print()`
- ❑ `len()`
- ❑ `type()`
- ❑ `range()`
- ❑ `input()`

## Funções para tipo:

- ❑ `int()`
- ❑ `float()`
- ❑ `str()`
- ❑ `list()`
- ❑ `tuple()`
- ❑ `dict()`

## Funções de iteração:

- ❑ `map()`
- ❑ `max()`
- ❑ `min()`
- ❑ `sum()`

## Funções matemáticas:

- ❑ `abs()`
- ❑ `pow()`



# FUNÇÕES EMBUTIDAS

Funções embutidas			
<b>A</b> abs() aiter() all() any() anext() ascii()	<b>E</b> enumerate() eval() exec()	<b>L</b> len() list() locals()	<b>R</b> range() repr() reversed() round()
<b>B</b> bin() bool() breakpoint() bytearray() bytes()	<b>F</b> filter() float() format() frozenset()	<b>M</b> map() max() memoryview() min()	<b>S</b> set() setattr() slice() sorted() staticmethod() str() sum() super()
<b>C</b> callable() chr() classmethod() compile() complex()	<b>G</b> getattr() globals()	<b>N</b> next()	<b>T</b> tuple() type()
<b>D</b> delattr() dict() dir() divmod()	<b>H</b> hasattr() hash() help() hex()	<b>O</b> object() oct() open() ord()	<b>V</b> vars()
	<b>I</b> id() input() int() isinstance() issubclass() iter()	<b>P</b> pow() print() property()	<b>Z</b> zip()  -__import__()

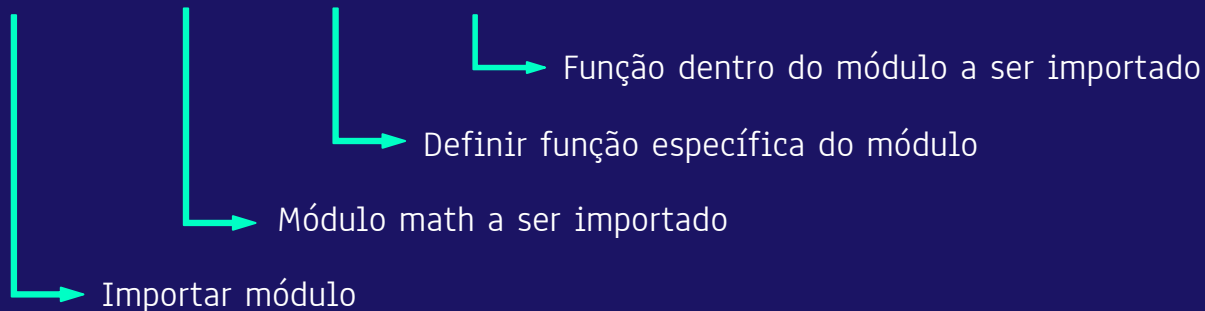
<https://docs.python.org/pt-br/3/library/functions.html>

# MÓDULOS E FUNÇÕES

# MÓDULOS

Um módulo é um arquivo contendo definições e instruções Python, onde pode conter classes, funções, variáveis ou qualquer outra coisa que você possa escrever em um script Python normal.

```
import math from sqrt
```



# IMPORTANDO MÓDULOS

`import os` → Funções de sistemas operacional

`import math` → Funções matemáticas básicas

`import random` → Gerar números aleatórios

`import datetime` → Datas e horas

`import time` → Medições de tempo e espera

`import csv` → Arquivos CSV

`import re` → Expressões regulares (Regex)

`import json` → Dados em formato JSON

`import requests` → Requisições HTTP

# IMPORTANDO FUNÇÕES DO MÓDULO MATH

```
import math  
  
raiz = math.sqrt(25) # Saída: 5.0  
  
sen = math.sin(45) # Saída: 0.8509035245341184  
  
cos = math.cos(35) # Saída: -0.9036922050915067  
  
pi = math.pi # 3.141592653589793
```

# IMPORTANDO FUNÇÕES ESPECÍFICAS

```
from math import sqrt  
raiz = sqrt(25) # Saída: 5.0
```

# DEFININDO APELIDO DO MÓDULO

```
import math as m  
raiz = m.sqrt(25) # Saída: 5.0
```

# IMPORTANDO FUNÇÕES DO MÓDULO RANDOM

```
import random
```

```
# Gerar um número aleatório entre 1 e 100
```

```
random_number = random.randint(1, 100)
```

```
# Gerar um número aleatório entre 0 e 1
```

```
random_float = random.random()
```

```
# Gerar uma sequência de 5 números aleatórios entre 1 e 100
```

```
random_sequence = random.sample(range(1, 101), 5)
```



# FUNÇÕES DO MÓDULO RANDOM PARA LISTAS

```
import random

frutas = ['maçã', 'banana', 'cereja', 'laranja']

# Escolher um item aleatório da lista

print(random.choice(frutas)) # Saída: banana

# Embaralhar a lista

random.shuffle(frutas)

print(frutas) # Saída: ['maçã', 'banana', 'cereja', 'laranja']
```

# BIBLIOTECAS

Uma biblioteca é um conjunto de módulos com funcionalidades específicas para serem reutilizadas no código. Algumas bibliotecas são padrões e já vem na instaladas como o `math`, `random`, `json` e outras.

## Bibliotecas externas (requerem instalação):

- ❑ NumPy
- ❑ Pandas
- ❑ Matplotlib
- ❑ Unittest
- ❑ Tkinter
- ❑ Django

# EXERCÍCIOS

1. Crie uma função que receba três números inteiros e retorne o dobro da soma dos três números.
2. Faça uma função que retorne se o número é par ou ímpar.
3. Crie uma lista de produtos qualquer de uma loja de roupas e desenvolva 4 funções que irão operar a lista, são elas: adicionar, remover, editar e exibir um produto. Pense em como irá criar cada função e faça verificações para saber por exemplo se um produto primeiro existe na lista para poder ser removido. Ao final, mostre a lista após as alterações realizadas nas funções.
4. Crie 4 funções lambdas que irão realizar operações matemáticas simples de soma, subtração, multiplicação e divisão entre dois números fornecidos pelo usuário.
5. Faça uma função que sorteie um número de 1 a 6 de um dado 10 vezes e a cada chamada na função, mostre o valor sorteado. Também, contabilize o número de vezes que cada número foi sorteado e exiba as quantidades ao final do programa.
6. Crie três funções para calcular o seno, cosseno e tangente de um número, converta os resultados para graus, utilizando a função `degrees` do `math` e retorne os resultados arredondados em duas casas decimais. Ao final utilize as funções `max` e `min` para saber qual é o valor máximo e mínimo dos três resultados. Exiba todos os resultados das funções.

# EXERCÍCIOS

7. A sequência de Fibonacci é uma sequência infinita de números que começa com 0 e 1 e cada número subsequente é a soma dos dois números anteriores. Um exemplo de sequência: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34... Note que o resultado da soma dos primeiros números da sequência  $0 + 1$  é igual a 1 que significa o próximo número da sequência, a mesma lógica se aplica para os próximos números como  $1 + 1$  é igual a 2. Crie uma função que tenha como parâmetro o número máximo da sequência de Fibonacci e retorne uma lista com todos os números que foram somados na essa sequência até o número máximo.