

# FUNDAMENTOS DA CIÊNCIA DOS DADOS E LÓGICA DE PROGRAMAÇÃO



sagah<sup>+</sup>

---

# Uso da linguagem Python

***Victor de Andrade Machado***

## OBJETIVOS DE APRENDIZAGEM

- > Descrever a linguagem Python e sua aplicabilidade na resolução de problemas.
  - > Implementar algoritmos estruturados simples em Python.
  - > Identificar bibliotecas e pacotes que podem ser utilizadas com Python.
- 

## Introdução

A Python é uma linguagem de programação de alto nível, interpretada, multiparadigma e de propósito geral, que ganhou popularidade em uma ampla variedade de áreas, desde desenvolvimento *web* até ciência de dados. A linguagem se destaca por sua sintaxe simples e sua legibilidade, o que a torna uma excelente escolha tanto para iniciantes quanto para programadores experientes.

Neste capítulo, você vai estudar a linguagem Python e sua aplicabilidade na resolução de problemas, destacando suas características e vantagens. Vai, também, acompanhar a implementação de algoritmos estruturados simples na linguagem Python, com exemplos para ilustrar como ela facilita a criação e a execução de algoritmos eficientes. Por fim, você vai conhecer as bibliotecas e os pacotes mais relevantes da linguagem Python, que podem ser utilizados para estender suas funcionalidades e solucionar problemas específicos em diferentes áreas, como análise de dados, visualização e aprendizado de máquina.

## Introdução à linguagem Python

A sintaxe da linguagem Python é projetada para ser simples e fácil de ler, com uma estrutura clara e organizada. O uso de indentação em vez de chaves para delimitar blocos de código a torna mais legível e menos propensa a erros. Por isso, é relativamente fácil de aprender, especialmente para iniciantes em programação, tendo uma vasta quantidade de recursos de aprendizado.

Por ser uma linguagem multiplataforma, um programa escrito em Python pode ser executado em diferentes sistemas operacionais, como Windows, macOS e Linux, sem a necessidade de modificação. Outra vantagem é sua biblioteca, que oferece uma variedade de módulos e funcionalidades prontas para uso, como módulos para manipulação de arquivos, acesso à internet, geração de números aleatórios, criptografia, entre outros. Além disso, a comunidade de desenvolvedores dessa linguagem é muito ativa e engajada, o que resulta em uma ampla gama de recursos disponíveis. Existem inúmeros pacotes e bibliotecas de terceiros desenvolvidos pela comunidade que podem ser facilmente integrados aos projetos em Python, permitindo a expansão das suas funcionalidades.

Confira a seguir um processo de instalação e configuração padrão, além das principais áreas de aplicação da linguagem Python.

## Instalação e configuração da linguagem Python

Há diferentes distribuições disponíveis, como a Python oficial (CPython), Anaconda, Miniconda e outras. Para instalação, deve-se acessar o site oficial da linguagem e baixar a versão mais recente, que inclui um instalador com orientações durante o processo. Além do interpretador Python, é recomendado usar um ambiente de desenvolvimento integrado (IDE) para escrever e executar os programas, como:

- PyCharm: IDE desenvolvido pela JetBrains, que oferece uma ampla gama de recursos e ferramentas para desenvolvimento Python;
- Visual Studio Code: editor de código leve e altamente personalizável, com suporte a extensões para Python e integração com ferramentas de depuração;
- Jupyter Notebook: aplicação web que permite criar e compartilhar documentos que contenham código Python, além de visualizações, gráficos e anotações.

Confira a seguir as etapas para instalação e sua extensão no Visual Code, considerando um sistema operacional Windows 11.

1. Acesse o site oficial da Python.
2. Na página inicial, clique no botão *Downloads* no canto superior direito.
3. Selecione a versão mais recente do Python para o seu sistema operacional (Windows, macOS, Linux).
4. Baixe o instalador apropriado e execute-o.
5. Siga as instruções do instalador para concluir a instalação.

Após a instalação, o Python estará pronto para uso. Para verificar se a instalação deu certo, deve-se abrir o terminal (no Windows, o *prompt* de comando) e digitar “python”. Se aparecer o *prompt* interativo do Python, significa que foi instalado corretamente.

## Principais áreas de aplicação

No desenvolvimento *web*, a Python é amplamente utilizada por sua disponibilidade de bons *frameworks*, como o Django, que fornece um conjunto completo de ferramentas para criação de aplicativos *web* robustos e escaláveis, e o Flask, um *microframework* leve e flexível, ideal para desenvolvimento de aplicativos *web* menores e mais simples. Também é usada para automatizar tarefas relacionadas à implantação, aos testes e ao gerenciamento de infraestrutura de servidores (PYTHON, 2023).

Além disso, é uma excelente escolha para automatizar tarefas repetitivas e mundanas. Com bibliotecas como Selenium, Beautiful Soup e Requests, é possível criar *scripts* para automatizar ações em *sites*, extrair dados da *web* e realizar testes automatizados. A automação de tarefas com a Python pode economizar tempo e esforço, aumentando a eficiência em diversas áreas.

A análise de dados e a ciência de dados pode utilizar a linguagem por sua rica biblioteca de ferramentas. Bibliotecas como NumPy, Pandas e Matplotlib são usadas para manipulação de dados, análise estatística, visualização de dados e criação de gráficos. Já bibliotecas como Scikit-learn e TensorFlow fornecem recursos para aprendizado de máquina e desenvolvimento de modelos preditivos.

Embora a Python não seja a linguagem mais comum para desenvolvimento de jogos de alto desempenho, ela é amplamente utilizada para prototipagem rápida e desenvolvimento de jogos mais simples. A biblioteca Pygame é uma

escolha popular para desenvolver jogos 2D em Python, oferecendo recursos para gráficos, som, entrada do usuário e física básica.

Em ambientes de *big data* e processamento de dados em escala, bibliotecas como PySpark, Dask e Apache Kafka permitem o processamento eficiente e distribuído de grandes volumes de dados. Com essas bibliotecas, é possível lidar com análise de dados em tempo real, processamento paralelo e integração com sistemas de armazenamento distribuídos.

A Python também é muito adotada para inteligência artificial e aprendizado de máquina. Com bibliotecas como TensorFlow, PyTorch, Scikit-learn e Keras, é possível desenvolver e implementar modelos de aprendizado de máquina, redes neurais, algoritmos de processamento de linguagem natural, entre outras possibilidades (PYTHON, 2023).

Essas são apenas algumas das áreas de aplicação da linguagem Python. Sua versatilidade permite que seja utilizada em muitos outros domínios, como automação de processos, desenvolvimento de aplicativos de *desktop*, análise financeira, entre outros. A ampla gama de bibliotecas disponíveis e a comunidade ativa de desenvolvedores contribuem para torná-la uma escolha comum para diversas aplicações.

## Implementações em Python

A implementação de algoritmos em Python é relativamente simples, em virtude de sua estrutura fácil e intuitiva, bem como a proximidade, ainda que em linguagem própria, dos algoritmos estruturados em português. Os comandos de atribuição são usados para atribuir valores a variáveis (MENEZES, 2019). Em Python, o símbolo "=" é usado para realizar atribuições, conforme mostrado a seguir:

```
x = 10  
nome = "João"
```

A Python oferece comandos simples para entrada e saída de dados. O comando `print()` é usado para exibir mensagens na saída padrão, enquanto o comando `input()` permite a entrada de dados pelo usuário (PYTHON, 2023). A Figura 1 demonstra esse tipo de utilização.

```
nome = input("Digite seu nome: ")  
print("Olá, ", nome)
```

Utiliza-se a indentação para delimitar blocos de código:

```
if idade >= 18:  
    print("Você é maior de idade")  
else:  
    print("Você é menor de idade")
```

A Python suporta vários paradigmas de programação, como programação procedural, orientada a objetos e funcional, o que dá flexibilidade aos desenvolvedores, permitindo que eles escolham o estilo de programação que melhor se adequa ao problema em questão. Sua tipagem é dinâmica, ou seja, as variáveis não precisam ser declaradas com um tipo específico. Os tipos das variáveis são inferidos automaticamente durante a execução do programa, tornando o código mais conciso e flexível, como o exemplo a seguir (LUTZ; ASCHER, 2007; MENEZES, 2019):

```
# Atribuição inicial de um valor inteiro à variável 'x'  
x = 10  
print(x) # Saída: 10  
  
# A variável 'x' agora recebe uma string  
x = "Hello, world!"  
print(x) # Saída: Hello, world!  
  
# A variável 'x' é reatribuída como uma lista  
x = [1, 2, 3]  
print(x) # Saída: [1, 2, 3]
```

Nesse exemplo, começamos atribuindo um valor inteiro à variável x. Em seguida, imprimimos o valor de x, que é 10. Reatribuímos à variável x uma string contendo Hello, world!. Novamente, imprimimos o valor de x, e agora ele é Hello, world!. Finalmente, reatribuímos à variável x uma lista contendo os números 1, 2 e 3. Ao imprimir x pela última vez, vemos que o valor é a lista [1, 2, 3]. Em cada reatribuição, a variável x muda de tipo sem a necessidade de declarar explicitamente o tipo, em virtude, como vimos, da tipagem dinâmica da Python (LUTZ; ASCHER, 2007). Vamos ver agora outras estruturas comuns de algoritmos utilizando a linguagem Python.

As funções permitem agrupar um bloco de código em uma unidade reutilizável. Em Python, as funções são definidas usando-se o comando `def` seguido pelo nome da função e seus parâmetros (LUTZ; ASCHER, 2007; PYTHON, 2023):

```
def somar(a, b):
    return a + b

resultado = somar(3, 5)
print(resultado)
```

Confira a seguir um passo a passo do código.

1. Definimos a função `somar` utilizando a palavra-chave `def` e o nome da função e dos parâmetros entre parênteses. Nesse caso, a função recebe dois parâmetros: `a` e `b`.
2. Dentro do corpo da função, temos uma única linha de código que realiza a soma dos parâmetros `a` e `b` utilizando o operador de adição `+`.
3. A instrução `return` é utilizada para retornar o resultado da soma para quem chamou a função. Nesse caso, estamos retornando a soma de `a` e `b`.
4. Fora da definição da função, criamos uma variável chamada `resultado` e atribuímos a ela o valor retornado pela função `somar` quando chamada com os argumentos `3` e `5`.
5. Finalmente, imprimimos o valor da variável `resultado` utilizando a função `print`.

Vamos ver, agora, um exemplo para calcular a média de notas digitadas pelo usuário:

```
# Solicitar três valores ao usuário
valor1 = float(input("Digite o primeiro valor: "))
valor2 = float(input("Digite o segundo valor: "))
valor3 = float(input("Digite o terceiro valor: "))

# Calcular a média simples
media = (valor1 + valor2 + valor3) / 3

# Exibir a média
print("A média dos três valores é:", media)
```

Neste exemplo, usamos a função `input()` para solicitar ao usuário que digite os três valores. Em seguida, convertemos os valores para o tipo `float` usando `float(input())`, pois assumimos que os valores podem ser decimais. Calculamos a média simples somando os três valores e dividindo por 3. O resultado é armazenado na variável `média`. Por fim, exibimos o resultado usando a função `print()`. A mensagem “A média dos três valores é:” é exibida junto com o valor da média calculada.

O próximo exemplo recebe dois valores do usuário e realiza todas as operações matemáticas básicas (adição, subtração, multiplicação, divisão e exponenciação) entre esses valores, exibindo os resultados no final:

```
# Solicitar dois valores ao usuário
valor1 = float(input("Digite o primeiro valor: "))
valor2 = float(input("Digite o segundo valor: "))

# Realizar as operações matemáticas
soma = valor1 + valor2
subtracao = valor1 - valor2
multiplicacao = valor1 * valor2
divisao = valor1 / valor2
exponenciacao = valor1 ** valor2

# Exibir os resultados
print("Resultado da adição:", soma)
print("Resultado da subtração:", subtracao)
print("Resultado da multiplicação:", multiplicacao)
print("Resultado da divisão:", divisao)
print("Resultado da exponenciação:", exponenciacao)
```

Nesse programa, solicitamos dois valores ao usuário usando a função `input()` e convertemos os valores para o tipo `float`. Em seguida, realizamos as operações matemáticas básicas entre os valores fornecidos. A adição é realizada com o operador `+`; a subtração com o operador `-`; a multiplicação com o operador `*`; a divisão com o operador `/`; e a exponenciação com o operador `**`. Finalmente, exibimos os resultados de cada operação usando a função `print()`, fornecendo uma mensagem descritiva seguida pelo valor correspondente.

## Bibliotecas e pacotes Python

A Python oferece uma grande variedade de bibliotecas e pacotes, o que aumenta suas funcionalidades básicas. Essas bibliotecas podem ser instaladas e importadas em um projeto Python para fornecer recursos extras (AMARAL, 2018).

A NumPy, por exemplo, é uma biblioteca fundamental para computação científica em Python. Ela fornece suporte para *arrays* multidimensionais, operações matemáticas de alto desempenho e funções para manipulação de dados (PYTHON, 2023):

```
import numpy as np

# Criando um array unidimensional
arr = np.array([1, 2, 3, 4, 5])

# Operações matemáticas com arrays
dobra = arr * 2
soma = np.sum(arr)

# Criando um array bidimensional
matriz = np.array([[1, 2, 3], [4, 5, 6]])

# Acessando elementos do array
elemento = matriz[1, 2]
```

O código fornecido demonstra o uso da biblioteca NumPy em Python para realizar operações com *arrays*. Confira a seguir um passo a passo do código.

- Importamos a biblioteca NumPy utilizando o comando `import numpy as np`. A convenção de renomear a biblioteca para `np` é comumente utilizada para facilitar a digitação de código.
- Criamos um *array* unidimensional chamado `arr` utilizando a função `np.array()`. O *array* é inicializado com os valores `[1, 2, 3, 4, 5]`.
- Realizamos uma operação matemática com o *array* `arr`, multiplicando todos os elementos por 2. O resultado é atribuído à variável `dobra`.
- Utilizamos a função `np.sum()` para calcular a soma de todos os elementos do *array* `arr`. O resultado é atribuído à variável `soma`.

- Criamos um *array* bidimensional chamado `matriz` utilizando a função `np.array()`. A matriz é inicializada com os valores `[[1, 2, 3], [4, 5, 6]]`.
- Acessamos um elemento específico da matriz utilizando a notação de indexação. No exemplo fornecido, acessamos o elemento na segunda linha e terceira coluna, que é o valor 6. O valor é atribuído à variável `elemento`.

No código fornecido, destacamos a criação de *arrays*, a realização de operações matemáticas com *arrays* e o acesso a elementos específicos em *arrays* bidimensionais.

A Pandas, por sua vez, é uma biblioteca utilizada para análise e manipulação de dados em Python, porque oferece estruturas de dados flexíveis, como o `DataFrame`, que permite a organização e a análise de conjuntos de dados (AMARAL, 2018; PYTHON, 2023). Confira este exemplo de utilização do Pandas:

```
import pandas as pd

# Criando um DataFrame
data = {'Nome': ['Alice', 'Bob', 'Charlie'],
        'Idade': [25, 30, 35]}
df = pd.DataFrame(data)

# Exibindo os primeiros registros do DataFrame
print(df.head())

# Filtrando os registros com idade acima de 30
df_filtrado = df[df['Idade'] > 30]
```

O código fornecido demonstra o uso da biblioteca Pandas em Python para criar um `DataFrame`, exibir os primeiros registros e filtrar os registros com base em uma condição. O passo a passo do código está listado a seguir.

1. Importamos a biblioteca Pandas utilizando o comando `import pandas as pd`. Convencionou-se renomear a biblioteca para `pd` para facilitar a digitação do código.
2. Criamos um dicionário chamado `data`, que contém as informações para construir o `DataFrame`. O dicionário tem duas chaves: 'Nome' e

- 'Idade'. Cada chave corresponde a uma lista de valores, representando o nome e a idade das pessoas.
3. Utilizamos a função `pd.DataFrame()` para criar o DataFrame a partir do dicionário `data`. O DataFrame é atribuído à variável `df`.
  4. Utilizamos o método `head()` do DataFrame para exibir os primeiros registros. O método `head()` retorna as primeiras cinco linhas do DataFrame. Essa função é útil para verificar a estrutura e os dados iniciais do DataFrame.
  5. Utilizamos a notação de indexação para filtrar os registros com idade acima de 30. Criamos uma nova variável, chamada `df_filtrado`, que armazena um novo DataFrame contendo apenas os registros em que a coluna 'Idade' apresenta valores maiores do que 30. Isso é feito utilizando-se uma condição booleana dentro dos parênteses, que retorna apenas as linhas que satisfazem a condição.

Já a Matplotlib é uma biblioteca para criação de gráficos em Python. Ela permite a geração de uma ampla variedade de gráficos, como gráficos de linha, barras, dispersão e histogramas (AMARAL, 2018):

```
import matplotlib.pyplot as plt

# Criando um gráfico de linha
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
plt.plot(x, y)
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Linha')
plt.show()
```

Importamos a biblioteca Matplotlib utilizando o comando `import matplotlib.pyplot as plt`. Como nos outros exemplos, a convenção de renomear a biblioteca para `plt` é para facilitar a digitação de código. Definimos duas listas, `x` e `y`, que representam os pontos do gráfico de linha. No exemplo fornecido, `x` contém os valores `[1, 2, 3, 4, 5]`, e `y` contém os valores `[1, 4, 9, 16, 25]`. Utilizamos a função `plt.plot(x, y)` para criar o gráfico de linha. A função `plot()` recebe as listas `x` e `y` como argumentos e traça uma linha conectando os pontos correspondentes. Utilizamos as funções `plt.xlabel()`, `plt.ylabel()` e `plt.title()` para adicionar rótulos aos eixos `x`

e `y` é um título ao gráfico, respectivamente. Essas funções são usadas para fornecer informações adicionais e descrições claras do gráfico. A função `plt.show()` exibe o gráfico na interface gráfica padrão do Python. Essa biblioteca é especialmente útil para representar dados de forma visual e compreensível, facilitando a análise e a comunicação dos resultados.



### Saiba mais

---

A Python tem uma das comunidades de desenvolvedores mais ativas e engajadas do mundo da programação, o que resulta em uma enorme quantidade de recursos disponíveis, desde bibliotecas e pacotes até tutoriais e fóruns de suporte. A comunidade visa a melhorar a linguagem e fornecer soluções para problemas.

---

A linguagem Python é frequentemente usada em conjunto com outras linguagens de programação. Sua capacidade de integração e interoperabilidade com linguagens como C, C++, Java e R permite que os desenvolvedores tirem proveito dos pontos fortes de cada linguagem, expandindo as capacidades de seus projetos. A capacidade de simplificar o processo de desenvolvimento de software a torna acessível a muitos usuários, permitindo avanços em áreas como ciência de dados, aprendizado de máquina e automação de tarefas. Com sua popularidade crescente e vasta gama de recursos disponíveis, a Python tem um papel importante no mundo da programação e da tecnologia.

## Referências

- AMARAL, F. *Introdução à ciência de dados: mineração de dados e big data*. Rio de Janeiro: Alta Books, 2018.
- LUTZ, M.; ASCHER, D. *Aprendendo Python*. 2. ed. Porto Alegre: Bookman, 2007.
- MENEZES, N. N. C. *Introdução à programação com Python: algoritmos e lógica de programação para iniciantes*. 3. ed. São Paulo: Novatec, 2019.
- PYTHON. Python Software Foundation. *Python*, 2023. Disponível em: <https://www.python.org/>. Acesso em: 20 jul. 2023.

## Leituras recomendadas

- ASCENCIO, A. F. G.; CAMPOS, E. A. V. *Fundamentos de programação de computadores: algoritmos, Pascal, C/C++ (Padrão ANSI) e Java*. 3. ed. São Paulo: Pearson, 2012.

GOODRICH, M. T.; TAMASSIA, R. *Estruturas de dados e algoritmos em Java*. 5. ed. Porto Alegre: Bookman, 2013.

RAMALHO, L. *Python fluente: programação clara, concisa e eficiente*. São Paulo: Novatec, 2015.



### ***Fique atento***

---

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declararam não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

---

Conteúdo:

**sagah**<sup>+</sup>