

ESTRITAMENTE CONFIDENCIAL

**ELABORATA**  
INFORMÁTICA

# Python com Lógica de Programação





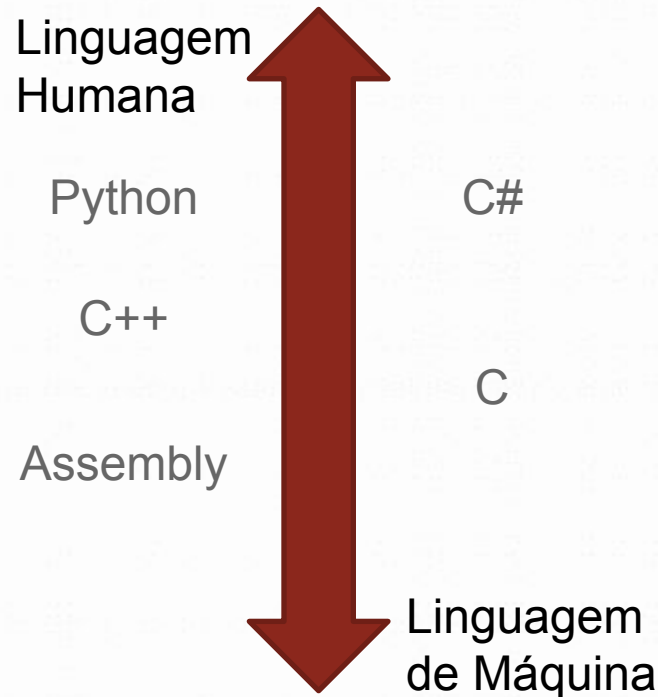
# AULA 01

Introdução ao Python,  
ambiente de trabalho,  
entrada, saída, variáveis e  
operações básicas

# O que é o Python?

Python é uma linguagem de programação de **alto nível**, fácil de ler e escrever.

Possui uma **sintaxe simples** e **flexível**, o que a torna ideal para **iniciantes**.

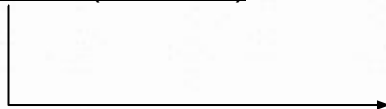


# Por que aprender Python?

- **Python** é uma das **linguagens** de programação mais utilizadas e **mais populares**, com aplicações em áreas como:
  - **Inteligência artificial**
    - PyTorch
    - TensorFlow
    - Scikit-learn
  - **Análise de dados**
    - NumPy
    - Pandas
    - Matplotlib
    - SciPy
  - **Desenvolvimento web**
    - Django
    - Flask
    - CherryPy
    - FastAPI
  - **Automação**
    - Selenium
  - **Internet das Coisas**
    - MicroPython
    - Home Assistant

# Configurando o ambiente de trabalho

1. **Linux**, Windows ou Mac;
2. Python **3.10** ou superior;
3. Visual Studio Code;
  - Extensões sugeridas para Python:
    - Python (Microsoft)
    - indent-rainbow (oderwat)



```
class AsyncExitStackMiddleware:
    def __init__(
        self, app: ASGIApp, context_name: str = "fastapi_astack"
    ) -> None:
        self.app = app
        self.context_name = context_name

    async def __call__(
        self, scope: Scope, receive: Receive, send: Send
    ) -> None:
        if AsyncExitStack:
            dependency_exception: Optional[Exception] = None
            async with AsyncExitStack() as stack:
                scope[self.context_name] = stack
                try:
                    await self.app(scope, receive, send)
                except Exception as e:
                    dependency_exception = e
                    raise e
```

# Instalando o Python

- **Linux**

Para instalar a **última** versão em distribuições baseadas no Debian (Ubuntu, Linux Mint, Debian):

1. Abra o terminal;
2. `sudo apt update`
3. `sudo apt install python3`

# Instalando o Python

- **Mac**
  - [python.org/downloads/macos](https://python.org/downloads/macos)

# Instalando o Python

- **Windows**

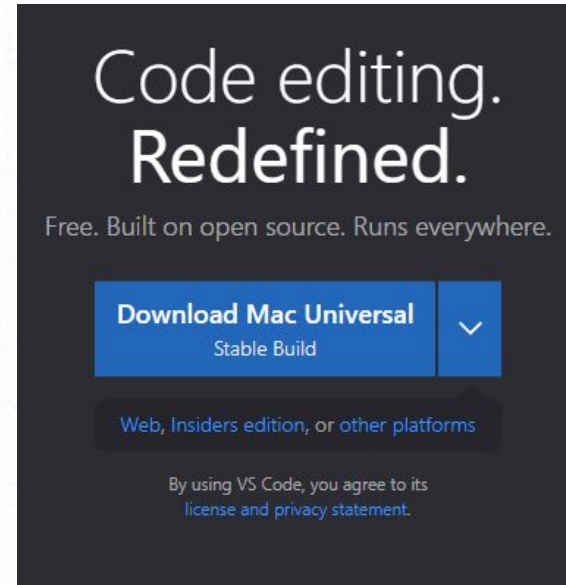
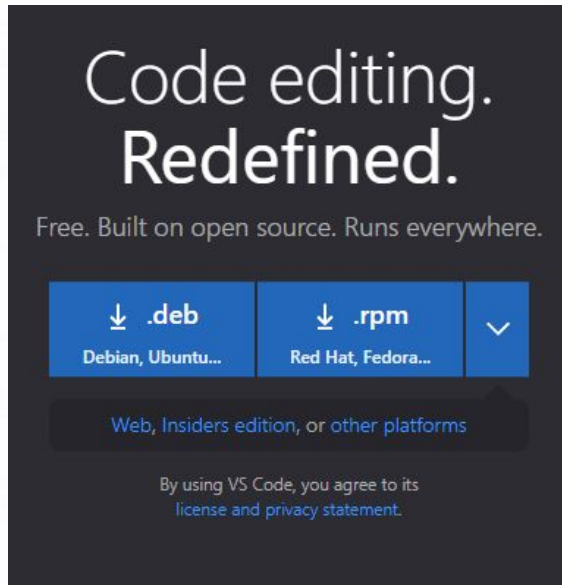
1. [Microsoft Store](#)
2. [python.org/downloads/windows](https://python.org/downloads/windows)

1. Marque a opção “Add Python 3.X to PATH” durante a instalação.



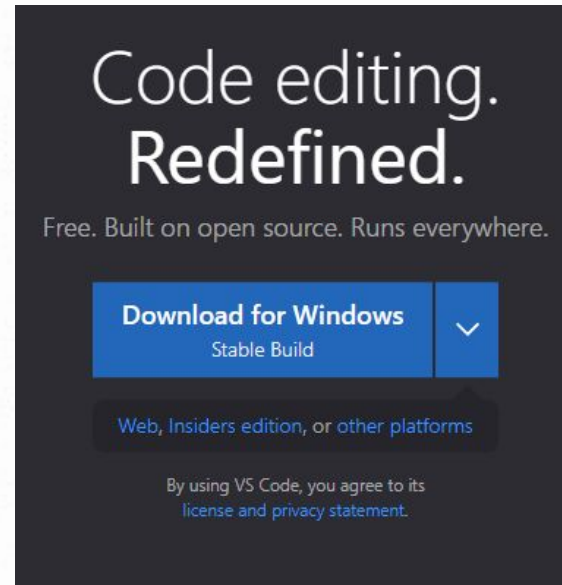
# Instalando o Visual Studio Code

- **Linux ou Mac**
  - [code.visualstudio.com](https://code.visualstudio.com)



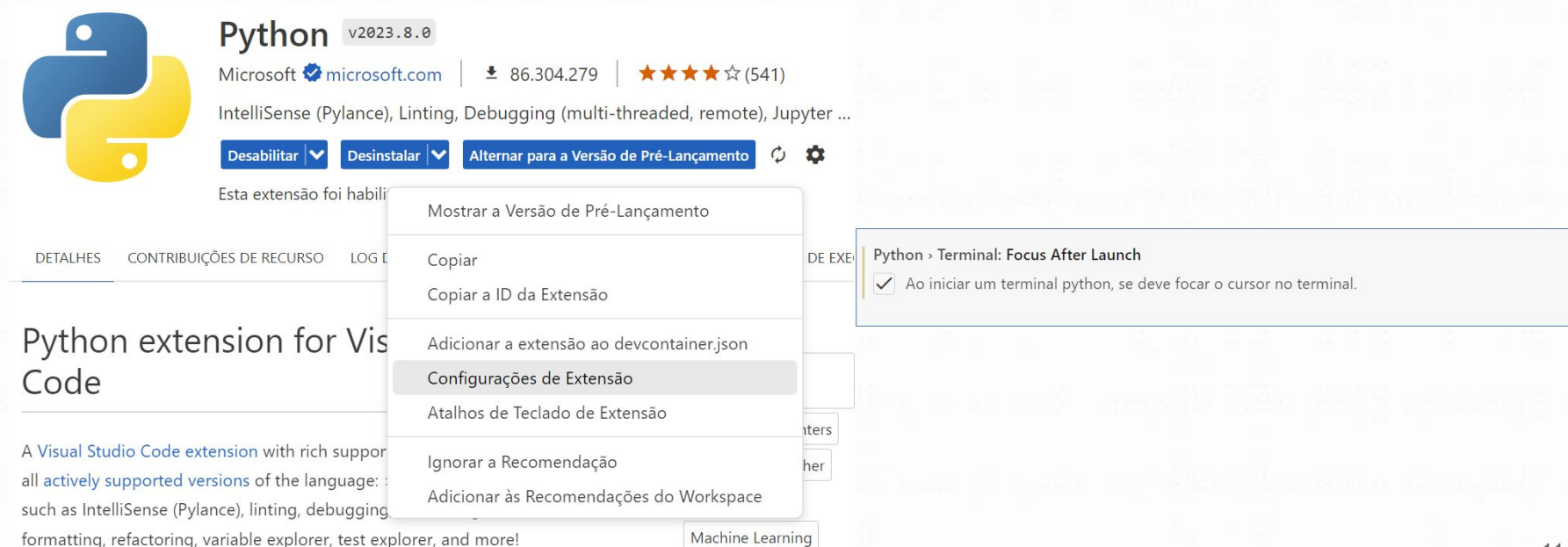
# Instalando o Visual Studio Code

- **Windows**
  1. [Microsoft Store](#)
  2. [code.visualstudio.com](https://code.visualstudio.com)



# Configurando o Visual Studio Code

- Alterar a configuração “Terminal: Focus After Launch”



The screenshot shows the Visual Studio Code extension marketplace page for the Python extension (v2023.8.0) by Microsoft. A context menu is open over the extension, with 'Configurações de Extensão' (Extension Settings) selected. To the right, a callout box displays the configuration 'Python > Terminal: Focus After Launch' with a checked checkbox and the description: 'Ao iniciar um terminal python, se deve focar o cursor no terminal.'

**Python** v2023.8.0  
Microsoft [microsoft.com](https://microsoft.com) | 86.304.279 | ★★★★★ (541)  
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter ...

Desabilitar Desinstalar Alternar para a Versão de Pré-Lançamento

Esta extensão foi habilitada

DE EXE

**Python extension for Visual Studio Code**

A Visual Studio Code extension with rich support for all actively supported versions of the language: Python, such as IntelliSense (Pylance), linting, debugging, formatting, refactoring, variable explorer, test explorer, and more!

Machine Learning

Mostrar a Versão de Pré-Lançamento

Copiar

Copiar a ID da Extensão

Adicionar a extensão ao devcontainer.json

**Configurações de Extensão**

Atalhos de Teclado de Extensão

Ignorar a Recomendação

Adicionar às Recomendações do Workspace

**Python > Terminal: Focus After Launch**

☒ Ao iniciar um terminal python, se deve focar o cursor no terminal.

# Python – Variáveis

- O que são variáveis?

**Variáveis** são **identificadores** que **representam valores** na memória durante a **execução** de um programa. Elas são usadas para **guardar e manipular dados**, permitindo que esses valores sejam **utilizados e alterados** conforme necessário ao longo do programa.

Você pode querer imaginar variáveis como **caixas** nas quais seu programa **armazena temporariamente dados** enquanto está em execução.

# Python – Variáveis

- **Como criar variáveis?**

Você **atribui** um **valor** a uma **variável** utilizando o sinal de igual (=).

Por exemplo:

```
a = 10
frase = "Vovó foi à feira"
valor = 12.99
c, d = 15, 20
```

# Python – Variáveis

- **Convenções de nomenclatura de variáveis**
- **Nomes de variáveis: Devem ser escritos em letras minúsculas.** Se o nome consiste em várias palavras, elas devem ser **separadas** por sublinhados (\_\_) para melhorar a legibilidade. Isso é conhecido como **snake\_case**. Exemplos: `minha_variavel`, `calcular_media`.
- **Não use símbolos especiais:** Símbolos como `!`, `@`, `#`, `$`, `%`, emojis, etc. **não são permitidos** em nomes de variáveis em Python.
- **Nomes de variáveis devem ser descritivos:** Eles devem ser o mais descritivos possível para tornar o código mais legível e compreensível. Por exemplo, em vez de usar `a` ou `b`, use nomes como `altura`, `largura`, etc.

# Python – Variáveis

- **Convenções de nomenclatura de variáveis**
- **Evite usar palavras-chave do Python como nomes de variáveis:** Palavras como `list`, `str`, `dict` são palavras-chave do Python e devem ser evitadas como nomes de variáveis para evitar confusão.
- **Nomes de variáveis não devem começar com números:** Em Python, os nomes das variáveis não podem começar com um número. Por exemplo, `1variavel` é inválido, enquanto `variavel1` é válido.
- **Nomes de variáveis podem conter números e sublinhados:** Além das letras, os nomes das variáveis podem conter **números** e **sublinhados**. Por exemplo, `variavel_1` é um nome de variável válido.

# Python – Variáveis

- **Declaração e atribuição**

- Declarar uma variável é criar um **nome simbólico** para um **endereço de memória**, que armazenará um valor;
- **Atribuição** é **definir** um **valor** para a **variável** declarada;
- Em Python, ao **declarar** uma variável, é **obrigatório** fazer uma **atribuição**.



# Python – Variáveis

- **Declaração e atribuição**

Uma variável **não pode ser utilizada** em uma expressão **sem ter sido inicializada**:

```
bruto = 1000  
liquido = bruto - imposto
```

Ambas as **variáveis precisam ser** previamente **declaradas**, caso contrário o seguinte erro acontecerá:

```
bruto = 1000  
liquido = bruto - imposto  
  
Traceback (most recent call last):  
File "c:\projects\aulas\01\variaveis.py", line 2, in <module>  
    liquido = bruto - imposto  
              ^^^^^^^  
NameError: name 'imposto' is not defined
```

# Python – Tipos de variáveis

- Tipos de dados

<b>x = 1</b>	<b>Int</b>	Números inteiros, positivos ou negativos
<b>x = 1.0</b>	<b>Float</b>	Números com casas decimais
<b>x = "1"</b>	<b>String</b>	Texto, sequência de caracteres
<b>x = True</b>	<b>Bool</b>	Verdadeiro ou falso
<b>x = None</b>	<b>Null</b>	Nulo, ausência de valor

# Python – Tipos de variáveis

- **Inferência de tipo**

A **inferência de tipo** é a capacidade de **deduzir** automaticamente o **tipo de uma variável com base no valor que está sendo atribuído a ela**.

No Python, a inferência de tipo permite que os programadores **não declarem explicitamente o tipo de uma variável** ao definir ou atribuir um valor a ela.

Por exemplo, ao atribuir um valor a uma variável, o Python **automaticamente** deduz o tipo da variável com base no tipo do valor atribuído.

# Python – Tipos de variáveis

- **Tipagem forte**

O Python possui **tipagem forte**, **evitado** assim várias **operações** que no geral **não fariam sentido**.

Se tentarmos fazer uma operação matemática entre duas variáveis com tipos **incompatíveis**, o Python gerará um **erro de tipo**:

```
a = 10      # Inteiro
b = "10"    # String
c = a + b    # Soma de Inteiro com String
print(c)    # Mostra o resultado da soma
```

```
Traceback (most recent call last):
```

```
File "c:\projects\aulas\01\variaveis.py", line 3, in <module>
```

```
    c = a + b
```

```
~~~~~
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

# Python – Tipos de variáveis

- **Conversão de tipo explícita**

Para realizar a soma no exemplo anterior, é necessário primeiro **converter explicitamente** o valor da variável **b** para **int**, usando a função **int()**.

```
a = 10          # Inteiro
b = "10"        # String
c = a + int(b)  # Soma de Inteiro com Inteiro
print(c)        # Mostra o resultado da soma -> 20
```

Ou para somar strings, **converter explicitamente** o valor da variável **a** para **string**, usando a função **str()**.

```
a = 10          # Inteiro
b = "10"        # String
c = str(a) + b  # Soma de String com String
print(c)        # Mostra o resultado da soma -> 1010
```

# Python – Tipos de variáveis

- **Conversão de tipo explícita**

É importante observar que conversões explícitas **nem sempre são possíveis**:

```
a = "Vovó foi à feira"  
int(a)
```

Traceback (most recent call last):

File "c:\projects\aulas\01\variaveis.py", line 2, in <module>

```
    int(a)
```

ValueError: invalid literal for int() with base 10: 'Vovó foi à feira'

# Python – Tipos de variáveis

- **Conversão de tipo implícita**

A **conversão** de tipo **implícita** refere-se à conversão **automática** de **um tipo de dado em outro**, sem a necessidade de intervenção explícita do programador.

Essa conversão ocorre automaticamente durante certas operações ou expressões, quando o Python é capaz de deduzir a conversão necessária para produzir o resultado desejado.

# Python – Tipos de variáveis

- **Como descobrir o tipo de uma variável?**

Para **descobrir** o **tipo** de uma **variável**, basta utilizar a função **type()**.

Por exemplo:

```
nome = "Carlos"
idade = 31
altura, doenca = 1.76, False
endereco = None

print(type(nome))      # <class 'str'>
print(type(idade))     # <class 'int'>
print(type(altura))    # <class 'float'>
print(type(doenca))    # <class 'bool'>
print(type(endereco))  # <class 'NoneType'>
```



# Python – Entrada, saída e cálculos simples

- **Input, processamento e output**

Quase todos os programas de computador são escritos para **receber (input)**, **processar e exibir (output)** dados.

Por exemplo:

- Uma calculadora recebe números e instruções de um teclado (input), realiza o cálculo solicitado (processamento) e exibe a resposta em sua tela (output).

# Python – Entrada, saída e cálculos simples

- **Output**

No Python, para exibir (**output**) texto ou números no terminal do usuário, utilizamos a **função print()**.

Cada função **print()** exibirá sua **saída** em uma **nova linha**.

```
print("Olá, Mundo")
print("Eu tenho")
print(31)
print("anos de idade.")
# Olá, Mundo
# Eu tenho
# 31
# anos de idade.
```

# Python – Entrada, saída e cálculos simples

- **Output**

Uma função **print()** pode ser composta por **várias partes**, com cada parte **separada** por uma **vírgula**.

Note que o Python **substitui** automaticamente a vírgula por um **espaço** quando a saída é exibida.

```
print("Olá, Mundo")  
print("Eu tenho", 31, "anos de idade.")  
# Olá, Mundo  
# Eu tenho 31 anos de idade.
```

# Python – Entrada, saída e cálculos simples

- **Input**

Para solicitar (**input**) ao usuário que digite texto ou números no terminal, utilizamos a função **input()**.

Quando a função **input()** é usada, o programa deve **armazenar** os **dados** que o usuário insere em uma **variável**. O retorno da função **input()** **sempre** é uma **string**.

```
print("Por favor, digite seu nome.")  
nome_do_usuario = input()
```

# Python – Operações básicas

- **Operadores**

- **Aritméticos;**
- **Relacionais;**
- **Lógicos;**

# Python – Operações básicas

- **Operadores aritméticos**

São utilizados para realizar **operações matemáticas** com variáveis ou valores.

# Python – Operações básicas

- Operadores aritméticos

Operador	Descrição	Exemplo
+	Adição	$1 + 2 = 3$
-	Subtração	$5 - 2 = 3$
*	Multiplicação	$5 * 6 = 30$
**	Exponenciação	$2 ** 4 = 16$
/	Divisão	$17 / 4 = 4.25$
//	Divisão inteira	$17 // 4 = 4$
%	Resto da divisão	$17 \% 4 = 1$

# Python – Operações básicas

- **Operadores relacionais**

São utilizados para **comparações** entre variáveis ou valores, **sempre retornando** um valor **booleano** (**verdadeiro ou falso**).



# Python – Operações básicas

- Operadores relacionais

Operador	Descrição	Exemplo
>	Maior que	6 > 8 = False
<	Menor que	8 < 9 = True
>=	Maior ou igual a	12 >= 15 = False
<=	Menor ou igual a	4 <= 4 = True
!=	Diferente de	"Abacaxi" != "Tomate" = True
==	Igual a	False == False = True

# Python – Operações básicas

- **Operadores lógicos**

São utilizados para **combinar** valores **booleanos**, retornando um novo valor **booleano**.

# Python – Operações básicas

- Operadores lógicos

Operador	Descrição	Exemplo	Exemplo
and	E	(a > b) <b>and</b> (c == d)	True <b>and</b> False = False
or	Ou	(e <= 5) <b>or</b> (f == False)	True <b>or</b> False = True
not	Não	<b>not</b> True	<b>not</b> True = False

# Python – Operações básicas

- **Precedência de operações matemáticas**

Em Python, como em outras linguagens de programação, existe uma **ordem de precedência** em **operações matemáticas**, que define a **ordem em que elas são avaliadas**.

Se várias operações estiverem presentes na mesma expressão, algumas são executadas antes de outras.

# Python – Operações básicas

- **Precedência de operações matemáticas**

A ordem de precedência é a seguinte:

1. **Parênteses ( )**;
2. **Exponenciação (\*\*)** , da **direita para a esquerda**;
3. **Multiplicação (\*)**, **divisão (/)**, **divisão inteira (//)** e **módulo (%)**, da **esquerda para a direita**;
4. **Adição (+)** e **subtração (-)**, da **esquerda para a direita**;

```
print(2 - 3 / ( 4 + 6 ) * 2**3 % 2) # 1.6
# 2 - 3 / (4 + 6) * 2**3 % 2
# 2 - 3 / 10 * 2**3 % 2
# 2 - 3 / 10 * 8 % 2
# 2 - 0.3 * 8 % 2
# 2 - 2.4 % 2
# 2 - 0.4
# 1.6
```

# OBRIGADO!

ESTRITAMENTE CONFIDENCIAL





[www.elaborata.com.br](http://www.elaborata.com.br)

### **Horário de Atendimento Comercial**

Segunda à sexta – das 9:00h às 19:30h e

Sábado - das 8:00h às 15:00h.

Rua Monsenhor Celso, 256 - 1º Andar  
Centro - Curitiba - PR

**41.3324.0015**

 **41.99828.2468**

[cursos@elaborata.com.br](mailto:cursos@elaborata.com.br)

