

# Programação em Python – UC1

---

## Aula 03 – Parte 1



# GIT e GITHUB

---



# GIT E GITHUB

---



O código fonte do nosso software está em constante evolução.

Erros podem surgir, novas propostas aparecem, e de repente... é necessário reverter alterações.

Mas e se você pudesse ter uma "ferramenta de controle do tempo" para seu código?



# GIT E GITHUB: ATIVIDADE 0

---



Realize uma pesquisa na web sobre "ferramenta de controle de versão" para código fonte de programas. Identifique as alternativas disponíveis.



5 min...



# GITHUB: ATIVIDADE 0

Sistema	Distribuído?	Fácil de Usar?	Popularidade
Git	<input checked="" type="checkbox"/> Sim	★ ★ ★	★ ★ ★ ★ ★
SVN	<input type="checkbox"/> Não	★ ★	★ ★
Mercurial	<input checked="" type="checkbox"/> Sim	★ ★ ★	★ ★
Perforce	<input type="checkbox"/> Não	★	★ ★ ★
ClearCase	<input type="checkbox"/> Não	★	★
Fossil	<input checked="" type="checkbox"/> Sim	★ ★ ★	★



# GIT E GITHUB

---

O Git é um sistema de controle de versão distribuído. Mas calma!

- ◆ Controle de versão → Ele guarda um histórico de todas as alterações feitas no código. Você pode voltar atrás sempre que precisar.
  - ◆ Distribuído → Diferente de sistemas antigos que centralizavam tudo em um só lugar, o Git permite que cada programador tenha sua própria cópia completa do repositório.
    - ◆ Colaborativo → Vários programadores podem trabalhar no mesmo projeto ao mesmo tempo, sem perder código ou sobreescriver mudanças.



# GIT: NA PRÁTICA

---

Agora que sabemos por que usar o Git,  
vamos ver o que ele pode fazer! .

- Criar uma pasta para o projeto
- Criar um repositório
- Adicionando arquivos
- Salvando uma nova versão
- Visualizando histórico
- Criando e trocando de ramificação
- Juntando modificações
- Enviando código



# GIT: ATIVIDADE 1

---



Agora que sabemos por que usar o Git,  
vamos ver o que ele pode fazer! 

URL da Atividade 1

<https://github.com/luisrodrigonet/Python-UC01/blob/main/Aula03/Aula03-Tarefa01.md>



# GIT: ATIVIDADE 2

---



Agora que sabemos por que usar o Git,  
vamos ver o que ele pode fazer! 🚀

URL da Atividade 2

<https://github.com/luisrodrigonet/Python-UC01/blob/main/Aula03/Aula03-Tarefa02.md>



# GITHUB: ATIVIDADE 3

---



Realize uma pesquisa na WEB e verifique:

- 1 Quem fez o GitHub e foi vendido pra quem, por quem e por quanto.



5 min...



# GITHUB: ATIVIDADE 3

---



1 Quem fez o GitHub e foi vendido pra quem, por quem e por quanto.

# O GitHub foi criado por Chris Wanstrath, PJ Hyett, Tom Preston-Werner e Scott Chacon em 2008.

# Em 2018, a Microsoft comprou o GitHub por US\$ 7,5 bilhões.



# GIT E GITHUB: ATIVIDADE 4



Leia-me!

🚀 GIT e Github o par perfeito ! 🤝

URL da Atividade 4

<https://github.com/luisrodrigonet/Python-UC01/blob/main/Aula03/Aula03-Tarefa04.md>



# GIT E GITHUB: ATIVIDADE 5



GIT e Github o par perfeito ! 🤝

URL da Atividade 5

<https://github.com/luisrodrigonet/Python-UC01/blob/main/Aula03/Aula03-Tarefa05.md>





# Tipos de Dados

## Nativos do Python



# TIPOS DE DADOS

---



Em Python, tudo é um objeto, e cada objeto tem um tipo.

Saber quais são os tipos de dados e como usá-los corretamente vai te ajudar a escrever códigos mais eficientes e sem bugs.

Vamos nessa?

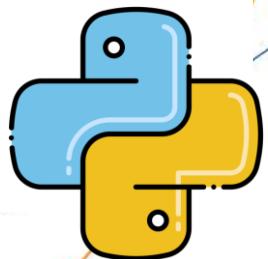


# TIPOS DE DADOS

---

Nesta parte do curso focaremos nos Python tem tipos de dados embutidos (built-in) que podem ser divididos em

- 🎯 Números Inteiros
- 🎯 Números ponto flutuantes
- 🎯 Números Completos
- 🎯 Texto
- 🎯 Booleamos
- 🎯 Sequências
- 🎯 Conjuntos
- 🎯 Mapeamentos
- 🎯 Especiais



# TIPOS DE DADOS

---

## 1 Números (int, float, complex)

### Inteiros (int):

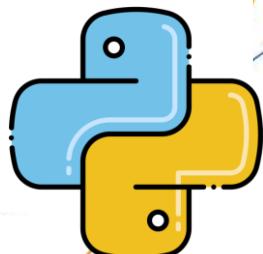
Números sem parte decimal.

### Pontos flutuantes (float):

Números com casas decimais.

### Complexos (complex):

Números com parte real e imaginária.



# TIPOS DE DADOS

---

## 1 Números (int, float, complex)



```
inteiro = 10
```

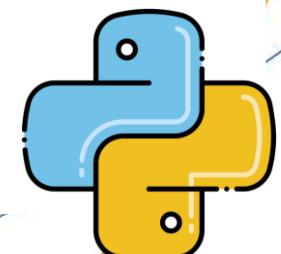
```
ponto_flutuante = 3.14
```

```
numero_complexo = 2 + 3j
```

```
print(type(inteiro))
```

```
print(type(ponto_flutuante))
```

```
print(type(numero_complexo))
```



# TIPOS DE DADOS

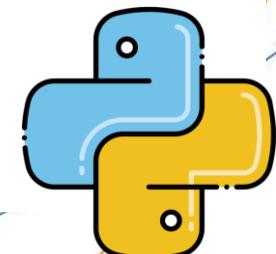
---

## 1 Números (int, float, complex)



💡 **Curiosidade:**  
Python aceita **notação científica** para floats!

```
grande = 1.5e6    # 1.5 * 10^6 = 1500000.0  
pequeno = 2.3e-4  # 2.3 * 10^-4 = 0.00023
```



# TIPOS DE DADOS

---



## 2 Texto (str)

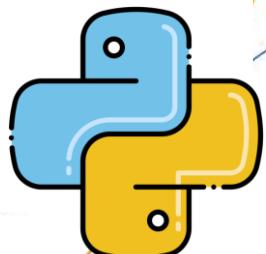
Strings (str) são sequências de caracteres usadas para representar textos.

```
nome = "Python"
```

```
frase = 'Isso é uma string!'
```

```
multilinha = """Isso é uma string de múltiplas  
linhas!"""
```

```
print(type(nome))
```



# TIPOS DE DADOS

---



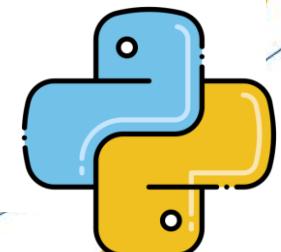
## 3 Booleanos (bool)

Os valores booleanos representam Verdadeiro (True) ou Falso (False).

```
verdadeiro = True
```

```
falso = False
```

```
print(type(verdadeiro))
```

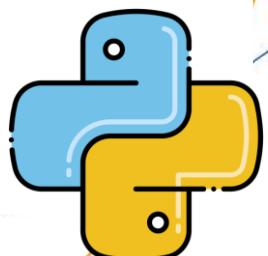


# TIPOS DE DADOS

---

## 4 Sequências (list, tuple, range)

- 🎯 **Listas (list)** – mutáveis
- 🎯 **Tuplas (tuple)** – imutáveis
- 🎯 **Intervalos (range)** – gerador de sequências



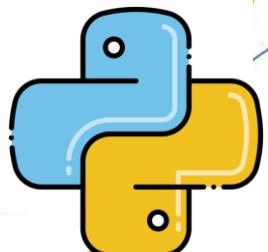
# TIPOS DE DADOS

---

## 5 Conjuntos (set, frozenset)

-  Armazemam elementos únicos (sem repetições).
-  Sem ordem específica.

-  Conjuntos (set)
-  Conjuntos congelados (frozenset)



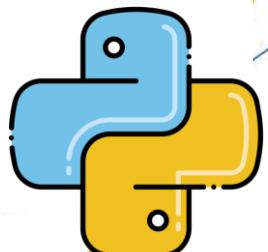
# TIPOS DE DADOS

---



## 6 Mapeamentos (dict) 🔑

- 🎯 Estruturas chave-valor.
- 🎯 Semelhantes a dicionários da vida real!

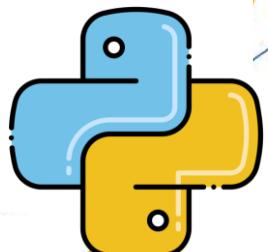


# TIPOS DE DADOS

---

## 7 Tipos Especiais

- 🎯 **NoneType** - valor nulo
- 🎯 **bytes** - dados binários
- 🎯 **bytearray** - dados binários
- 🎯 **memoryview** - dados binários





# Tipos de Expressões suportadas pelo Python



# EXPRESSÕES

---



Uma expressão em Python é qualquer código que seja avaliado para produzir um valor.

Isso inclui operações matemáticas, comparações, chamadas de funções e até atribuições de variáveis!

Essas expressões são fundamentais para a construção de lógica e manipulação de dados em Python

# EXPRESSÕES

---



Os principais tipos de expressões suportadas são:

- ◆ Expressões Aritméticas
- ◆ Expressões de Comparação
- ◆ Expressões Lógicas
- ◆ Expressões de Atribuição
- ◆ Expressões Condicionais (Ternárias)
- ◆ Expressões de Membros
- ◆ Expressões de Identidade
- ◆ Expressões de Indexação e Fatiamento



# Expressões Aritméticas

---

# EXPRESSÕES ARITMÉTICAS

---



Uma expressão aritmética é uma combinação de números, operadores e parênteses, que Python avalia e resolve.

As expressões aritméticas são fundamentais para qualquer linguagem de programação, pois nos permitem realizar cálculos matemáticos

# EXPRESSÕES ARITMÉTICAS

---



Uma expressão aritmética é uma combinação de números, operadores e parênteses, que Python avalia e resolve.

As expressões aritméticas são fundamentais para qualquer linguagem de programação, pois nos permitem realizar cálculos matemáticos

# EXPRESSÕES ARITMÉTICAS

Op	Nome	Exemplo ( $a = 10$ , $b = 3$ )			
+	Adição	$a + b$	$\rightarrow$	$10 + 3$	$= 13$
-	Subtração	$a - b$	$\rightarrow$	$10 - 3$	$= 7$
*	Multiplicação	$a * b$	$\rightarrow$	$10 * 3$	$= 30$
/	Divisão real	$a / b$	$\rightarrow$	$10 / 3$	$= 3.3333$
//	Divisão inteira	$a // b$	$\rightarrow$	$10 // 3$	$= 3$
%	Módulo (resto)	$a \% b$	$\rightarrow$	$10 \% 3$	$= 1$
**	Exponenciação	$a ** b$	$\rightarrow$	$10 ** 3$	$= 1000$

# EXPRESSÕES ARITMÉTICAS

---



## Ordem de Precedência dos Operadores

1. Parênteses → ()
2. Exponenciação → \*\*
3. Multiplicação → \*
4. Divisão → /
5. Divisão Inteira → //
6. Módulo → %
7. Adição → +
8. Subtração → -



# Desafio 3:

---

# 🔔 DESAFIO 3 🔔

---

⚠️ O recrutador entregou uma lista de atividades para você executar, relacionadas a variáveis e operadores.

📢 Realize as atividades contidas no arquivo “Aula03-Desafio03.md” que está na pasta “Aula 3” .