

Programação em Python – UC1

Aula 04 – Parte 1



GIT e GITHUB

Como trabalhar com um código fonte de um
programa de forma simultânea com a equipe
em máquinas diferentes ?





Desafio 4.1





DESAFIO 4.1 – GIT + GITHUB



- 1** Criar um repositório online no GitHub.
- 2** Criar o arquivo README.md diretamente no GitHub.
- 3** Clonar o repositório para a máquina local.
- 4** Criar um novo arquivo localmente.
- 5** Enviar o arquivo para o repositório no GitHub.
- 6** Criar um novo arquivo diretamente no GitHub.
- 7** Baixar o arquivo do repositório para a máquina local.





Desafio 4.2





DESAFIO 4.2 – GIT + GITHUB



- 1** Excluir todos os arquivos locais.
- 2** Recriar a pasta do repositório local.
- 3** Recuperar os arquivos clonando novamente o repositório do GitHub.



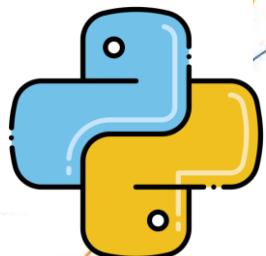
Expressões de atribuição

EXPRESSÕES DE ATRIBUIÇÃO



No Python, uma expressão de atribuição é usada para armazenar **um valor em uma variável**.

Parece algo simples, mas há **várias formas** de fazer isso, e entender bem como funciona pode deixar seu código mais limpo, eficiente.



Op	Descrição	Exemplo	Equivalente a
=	Atribuição simples	x = 10	x = 10
+=	Atribuição com adição	x += 5	x = x + 5
-=	Atribuição com subtração	x -= 3	x = x - 3
*=	Atribuição com multiplicação	x *= 2	x = x * 2
/=	Atribuição com divisão	x /= 4	x = x / 4
//=	Atribuição com divisão inteira	x //= 3	x = x // 3
%=	Atribuição com módulo (resto da divisão)	x %= 2	x = x % 2
**=	Atribuição com exponenciação	x **= 3	x = x ** 3



Desafio 4.3





DESAFIO 4.3 – ATRIBUIÇÃO



- 1 Desenvolva um programa que **solicite** um número **inteiro** ao usuário. Em seguida, aplique **todas** as **expressões** de atribuição disponíveis na linguagem. Para cada operação realizada, **exiba** o nome da expressão e o resultado obtido.
- 2 Após testar o código, **envie-o** para o seu repositório no **GitHub**.





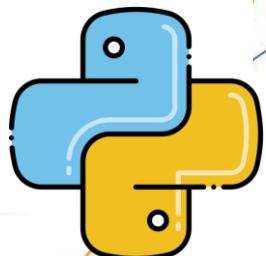
Operadores de Comparação

OPERADORES DE COMPARAÇÃO



As **expressões comparativas** em Python servem para comparar dois valores e **retornam** sempre um valor booleano: **True** (verdadeiro) ou **False** (falso).

Elas são muito utilizadas em **estruturas condicionais** e **laços de repetição**, sendo essenciais para a lógica de programação!



Operador	Descrição	Exemplo	Resultado
<code>==</code>	Igualdade	<code>5 == 5</code>	True
<code>!=</code>	Diferente	<code>5 != 3</code>	True
<code>></code>	Maior que	<code>7 > 3</code>	True
<code><</code>	Menor que	<code>2 < 5</code>	True
<code>>=</code>	Maior ou igual	<code>10 >= 10</code>	True
<code><=</code>	Menor ou igual	<code>3 <= 2</code>	False



Observações:



O operador `==` é diferente de `=!`



Atribuição `[=]`
→ guarda um valor



Comparação `[==]`
→ verifica se dois valores são iguais)

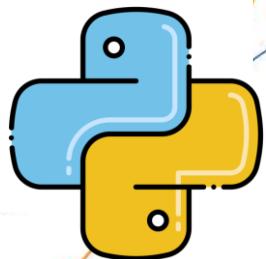
Estruturas Condicionais

ESTRUTURAS CONDICIONAIS



As **estruturas condicionais** permitem que um programa tome **decisões** com base em condições.

Elas são essenciais para **criar fluxos dinâmicos**, onde o código pode seguir **diferentes caminhos** dependendo dos valores de variáveis ou da entrada do usuário.



ESTRUTURAS CONDICIONAIS



Estrutura if

```
if condição:  
    # bloco de código
```

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

ESTRUTURAS CONDICIONAIS



Estrutura elif

```
if condição1:  
    # bloco de código  
elif condição2:  
    # bloco de código
```

```
idade = 16  
if idade >= 18:  
    print("Você é maior de idade.")  
elif idade >= 16:  
    print("Você é adolescente.")
```

ESTRUTURAS CONDICIONAIS



Estrutura else

```
if condição1:  
    # bloco de código  
elif condição2:  
    # bloco de código  
else:  
    # bloco de código
```

```
idade = 12  
if idade >= 18:  
    print("Você é maior de idade.")  
elif idade >= 16:  
    print("Você é adolescente.")  
else:  
    print("Você é criança.")
```



Desafio 4.4





DESAFIO 4.4 – CONDICIONAL



O recrutador entregou uma lista de atividades para você executar, relacionadas às condicionais.