Aula 2 - Condicionais com Python

Desvendando as estruturas condicionais

Condicionais: Decisões Inteligentes em Código

Nesta aula, vamos mergulhar nas **estruturas condicionais** do Python. Elas são a chave para que seus programas possam **tomar decisões** e reagir a diferentes situações, tornando-os muito mais dinâmicos.

1 **Entender if, elif e else**

Aprender como estas instruções funcionam para direcionar o fluxo do programa.

2 **Lógica Booleana**

Dominar a aplicação da lógica True/False em suas condições.

3 **Programas mais inteligentes**

Utilizar condições para criar softwares que pensam e reagem.

Introdução às Estruturas Condicionais

As estruturas condicionais são como "caminhos" que seu programa pode seguir. Imagine um GPS que escolhe a rota dependendo do trânsito: seu código fará algo similar, executando diferentes partes com base em uma condição ser **verdadeira** ou **falsa**.

```
if condição:

    # código executado se a condição for verdadeira
```

```
if condição:
    ação a ser realizada
elif condição:
    ação a ser realizada
elif condição:
    ação a ser realizada
elif condição:
    ação a ser realizada
else:
    ação a ser realizada
```

If, Elif e Else: O Trio da Decisão

if

O **if** é a base: ele testa uma condição principal.

elif

O **elif** (else if) permite testar **múltiplas condições** em sequência.

else

Se a condição do **if** e **elif** não for verdadeira, a condição **else** será usada.

Classificando um Número

```
if x > 0:
    print("Positivo")
elif x == 0:
    print("Zero")
else:
    print("Negativo")
```

Este código simples classifica um número em uma de três categorias, mostrando a versatilidade dos condicionais.

Operadores Relacionais e Lógicos

Para formular as condições, usamos **operadores** específicos:

Operadores Relacionais

- **==**: Igual a
- **!=**: Diferente de
- **<**: Menor que
- **>**: Maior que
- **<=**: Menor ou igual a
- **>=**: Maior ou igual a

Eles comparam valores e retornam True ou False.

Operadores Lógicos

- **and**: Ambas as condições devem ser verdadeiras.
- **or**: Pelo menos uma condição deve ser verdadeira.
- **not**: Inverte o resultado da condição.

Eles combinam condições para cenários mais complexos.

Exemplo com **and**:

```
if idade >= 18 and idade < 60:
    print("Adulto")
```

Aqui, ambas as condições de idade precisam ser verdadeiras para imprimir "Adulto".

A Indentação é Crucial em Python!

Em Python, a **indentação** (os espaços no início da linha) não é apenas um estilo, é **obrigatória**! Ela define os blocos de código que pertencem a uma condição, laço, função, etc.

Atenção: Erros de indentação levam a erros de sintaxe (IndentatioError) e impedem que seu programa funcione!

Exemplo Correto ✅

```
if x > 10:
    print("Maior que 10")
```

O espaço antes do `print` indica que ele faz parte do bloco `if`.

Exemplo Incorreto ❌

```
if x > 10:
print("Maior que 10")
```

Sem a indentação correta, o Python não reconhece o `print` como parte do `if`.

Aninhamento de Condições: Decisões Dentro de Decisões

Podemos ter estruturas condicionais **dentro de outras**. Isso é chamado de **aninhamento**.

É uma ferramenta poderosa para verificar **múltiplos critérios** em uma sequência lógica, afinando as condições para resultados mais específicos.

Exemplo Prático:

```
if x > 0:
    if x % 2 == 0:
        print("Número positivo e par")
    else:
        print("Número positivo e ímpar")
```

Neste exemplo, primeiro verificamos se `x` é positivo. **Só então**, dentro desse bloco, verificamos se ele é par ou ímpar.

O aninhamento permite criar lógicas complexas e detalhadas para o seu programa.

Lógica Booleana: A Base das Decisões

Em Python, as condições são avaliadas para um de dois valores booleanos:

> **True**
>
> Representa o valor **verdadeiro**.

> **False**
>
> Representa o valor **falso**.

Qualquer expressão que envolva uma comparação (**operadores relacionais**) ou combinação (**operadores lógicos**) sempre resultará em True ou False. Estes valores são fundamentais para o **fluxo de decisão** em seus programas.

Exemplos de Avaliação:

```
print(5 > 2)    # Saída: True
print(3 == 4)   # Saída: False
print(not True) # Saída: False
```

Compreender a lógica booleana é essencial para escrever condições eficientes e eficazes.

Hora de Praticar!

Agora, vamos colocar a mão na massa e aplicar o que aprendemos. O melhor jeito de aprender é codificando!

1 Classificador de Números

Crie um programa que peça um número ao usuário e informe se ele é **positivo**, **negativo** ou **zero**.

Sistema de Aprovação Escolar

Desenvolva um programa que receba a nota de um aluno. Se a nota for **maior ou igual a 7**, o programa deve exibir "Aprovado". Caso contrário, deve exibir "Reprovado".

Não tenha medo de errar! A prática leva à perfeição. Boa atividade!