Ao explorarmos as instruções if else percebemos que elas são uma poderosa e fundamental construção da linguagem, que nos permite controlar o fluxo do código com base em condições lógicas que determinamos de acordo com a necessidade de nosso projeto.

Com o surgimento de novas versões do Python, a linguagem de programação continua a evoluir e se aprimorar, trazendo consigo uma variedade de novas funcionalidades que enriquecem a experiência de desenvolvimento. Essas inovações não apenas aprimoram a eficiência e a legibilidade do código, mas também oferecem aos desenvolvedores ferramentas mais poderosas e expressivas para lidar com desafios diversos.

Dentre as funcionalidades mais recentes, destaca-se a introdução da instrução match no Python 3.10, oferecendo uma abordagem mais elegante para a correspondência de padrões em dados. Essa adição não apenas simplifica a lógica do código, mas também proporciona uma alternativa expressiva e legível às construções tradicionais de controle de fluxo, como if elif else, que são necessários para adaptar o comportamento do programa, como vimos anteriormente.

A principal finalidade da instrução match é simplificar a lógica de código ao facilitar o trabalho com diferentes padrões de dados. Isso pode tornar o código mais legível e conciso em situações onde uma cadeia de if elif else pode se tornar complicada e difícil de manter.

**Sintaxe do Match**

Durante o projeto, fizemos uma cadeia de condicionais em que cada uma indicava uma opção do nosso aplicativo, como adicionar restaurante, listar restaurantes, ativar restaurante e finalizar o app, além de colocarmos uma condicional para caso a opção informada não fosse válida (através do uso do else).

Utilizando as instruções if elif else tivemos esse resultado:

opcao\_escolhida = int(input('Escolha uma opção: '))

**if** opcao\_escolhida == 1:

print('Adicionar restaurante')

**elif** opcao\_escolhida == 2:

print('Listar restaurantes')

**elif** opcao\_escolhida == 3:

print('Ativar restaurante')

**elif** opcao\_escolhida == 4:

print('Finalizar app')

**else**:

print('Opção inválida!')

Copiar código

Se decidirmos utilizar a instrução match nesse caso, obteríamos o seguinte resultado:

opcao\_escolhida = int(input('Escolha uma opção: '))

**match** opcao\_escolhida:

**case** 1:

print('Adicionar restaurante')

**case** 2:

print('Listar restaurantes')

**case** 3:

print('Ativar restaurante')

**case** 4:

print('Finalizar app')

**case** \_:

print('Opção inválida!')

Copiar código

Perceba que recebemos a variável opção\_escolhida como parâmetro da instrução match e será feito um comparativo com todos os valores determinados pelos blocos de case, e no final temos uma instrução case \_, que é um padrão curinga, que corresponde a qualquer valor que não tenha sido correspondido pelos casos anteriores, ou seja, equivalente ao else da condicional anterior.

Tendo esse exemplo, podemos entender a sintaxe padrão do match. Dentro de um bloco match, você pode utilizar a instrução case para definir padrões específicos que serão comparados com a expressão que está sendo correspondida.

A estrutura básica é a seguinte:

**match** expressão:

**case** padrão\_1:

# Código a ser executado se expressão corresponder a padrão\_1

**case** padrão\_2:

# Código a ser executado se expressão corresponder a padrão\_2

# ... outros casos ...

**case** \_:

# Código a ser executado se nenhum dos padrões anteriores corresponder. Isso é útil para tratar casos não específicos.

Copiar código

**Qual instrução devo usar?**

O if nos proporciona uma maneira eficaz de tomar decisões simples ou complexas em nosso código, adaptando o comportamento do programa de acordo com as circunstâncias determinadas. Ao usar match, podemos simplificar a lógica do código em situações que envolvem múltiplos padrões complexos. Ela oferece uma estrutura mais legível, especialmente quando temos diversos casos a serem tratados.

Podemos ver na tabela a seguir quando podemos utilizar cada uma dessas instruções, de acordo com suas vantagens:

| **Vantagens da Instrução match** | **Vantagens da Estrutura if** |
| --- | --- |
| Lidar com condições complexas e múltiplos padrões de maneira mais intuitiva. | Implementação clássica e amplamente conhecida. |
| Sintaxe concisa melhora a legibilidade do código, especialmente em casos complexos. | Amplamente suportada em todas as versões do Python. |
| Permite desestruturação direta, evitando repetição excessiva de variáveis. | Estrutura simples e direta para lógica condicional básica. |
| Adiciona expressividade ao código, especialmente em situações de correspondência de padrões. | Pode ser mais intuitiva para devs familiarizados com estruturas de controle convencionais. |

No geral, tanto o if quanto o match são ferramentas poderosas à disposição de pessoas desenvolvedoras de Python. A escolha entre elas depende das necessidades específicas do seu código e das preferências da pessoa que está a desenvolver o código. A instrução match oferece uma nova abordagem elegante para a correspondência de padrões, abrindo novas possibilidades e aprimorando a expressividade do código.

Experimente ambas as opções e escolha aquela que melhor se alinha com o seu estilo de programação e os requisitos do projeto.

Para saber mais sobre a instrução match, acesse a documentação do python que aborda esse tema:

* [Instrução Match](https://docs.python.org/pt-br/3/tutorial/controlflow.html#match-statements)