Programação de Aplicativos

Professor: Euclides Paim
euclidespaim@gmail.com

Lógica de Programação

Estruturas de Controle

Professor: Euclides Paim

euclidespaim@gmail.com

Programação de Aplicativos

Estruturas de Controle

Sumário

- Revisão
- Instrução *Match*
- Sintaxe
- Valor padrão
- Valores Combinados

A Instrução `match` em Python

Em vez de escrever muitos comandos `if...else`, você pode usar a instrução `match` para tornar o código mais limpo e organizado.

O que é `match`?

A instrução `match` seleciona um entre vários blocos de código para serem executados, de acordo com o valor de uma expressão.

Sintaxe:

```
match expressão:
  case valor1:
    bloco_de_código
  case valor2:
    bloco_de_código
  case valor3:
    bloco_de_código
```

Programação de Aplicativos

Sintaxe, Variáveis e Tipos de Dados

Como funciona?

1. A **expressão** é avaliada **uma única vez**.
2. O valor obtido é comparado com os valores definidos em cada `case`.
3. Se houver **correspondência** (match), o **bloco de código** associado ao `case` é executado.

```
dia = 4
match dia:
  case 1:
    print("Segunda")
  case 2:
    print("Terça")
  case 3:
    print("Quarta")
  case 4:
    print("Quinta")
  case 5:
    print("Sexta")
  case 6:
    print("Sábado")
  case 7:
    print("Domingo")
```

Valor Padrão no `match`

Podemos usar o **caractere sublinhado** `_` como um **caso padrão**, para executar um bloco de código quando **nenhum outro** `case` **for correspondente**.

Como funciona o `_`?

- O underline `_` funciona como **"qualquer outro valor"**.
- Deve ser colocado como o **último** `case`.
- Isso garante que, se nenhum valor corresponder, ainda assim algo será executado.

Valor Padrão no `match`

Exemplo:

```
day = 4
match day:
  case 6:
    print("Hoje é Sábado")
  case 7:
    print("Hoje é Domingo")
  case _:
    print("Contando os dias para o final de semana!")
```

O valor **`_` sempre corresponde (match)** a qualquer valor.

Por isso, é **essencial colocá-lo como o último `case`**, para que ele funcione como um **caso padrão**, semelhante a um `else`.

Combinando Valores no `match`

Podemos usar o caractere **pipe** `|` como um operador **"ou" (`or`)** dentro de um `case`.

Isso permite que um único bloco de código seja executado se **qualquer um dos valores especificados corresponder.**

Exemplo:

```
day = 4
match day:
  case 1 | 2 | 3 | 4 | 5:
    print("Hoje é dia de trabalho")
  case 6 | 7:
    print("Eu amo final de semana!")
```

Use `|` para combinar múltiplos valores em um único `case`

Isso reduz repetições e deixa o código mais limpo

`if` como Condição Extra nos Casos (`Guards`)

Podemos adicionar uma instrução `if` após um `case` para verificar **condições adicionais**.
Essas condições são chamadas de **guards**.

Como funciona?

- Primeiro, o valor precisa **corresponder** ao `case`.
- Em seguida, a **condição `if` é avaliada**.
- O bloco de código só é executado se **ambas as condições forem verdadeiras**.

`if` como Condição Extra nos Casos (`Guards`)

Exemplo:

```
mes = 5
dia = 4
match dia:
  case 1 | 2 | 3 | 4 | 5 if mes == 4:
    print("Um dia da semana em Abril")
  case 1 | 2 | 3 | 4 | 5 if mes == 5:
    print("Um dia da semana em Maio")
  case _:
    print("Sem correspondência")
```

Guards adicionam **flexibilidade** e **precisão**

Úteis quando o mesmo valor pode ter significados diferentes dependendo do contexto

Programação de Aplicativos

Sintaxe, Variáveis e Tipos de Dados

Exercícios:

Exercício 1: Solicite ao usuário o número de um mês (1 a 12).

- Use `match` para imprimir o nome do mês.
- Use `_` como valor padrão caso o número não seja válido.

Programação de Aplicativos

Sintaxe, Variáveis e Tipos de Dados

Exercícios:

Exercício 2 – Previsão do tempo

Enunciado:

Solicite ao usuário o tipo de tempo (1: Sol, 2: Chuva, 3: Nublado, 4: Neve).
Use `match` com `|` para combinar e mostrar mensagens:

- Sol ou Nublado → “Leve um óculos escuro.”
- Chuva ou Neve → “Não esqueça o guarda-chuva ou casaco.”

Programação de Aplicativos

Sintaxe, Variáveis e Tipos de Dados

Exercícios:

Exercício 3: Solicite ao usuário o tipo de tempo (1: Sol, 2: Chuva, 3: Nublado, 4: Neve).
Use `match` com `|` para combinar e mostrar mensagens:

- Sol ou Nublado → “Leve um óculos escuro.”
- Chuva ou Neve → “Não esqueça o guarda-chuva ou casaco.”

Referências

Referências Básicas

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. ***Algoritmos: teoria e prática*****.** 3. ed. Rio de Janeiro: Elsevier, 2013.

SEBESTA, Robert W. ***Conceitos de linguagens de programação***. 10. ed. São Paulo: Pearson, 2018.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. ***Algoritmos: lógica para desenvolvimento de programação de computadores*****.** 27. ed. São Paulo: Érica, 2016.

DOWNEY, Allen B. ***Pense em Python: como pensar como um cientista da computação***. Tradução de Cássio de Souza Costa. 2. ed. São Paulo: Novatec, 2016.

Referências Complementares

IEPSEN, Edécio Fernando. ***Lógica de programação e algoritmos com JavaScript***. 2. ed. São Paulo: Novatec, 2022.

Referências na Internet

https://www.w3schools.com/python/