Match-Case

KARINA CASOLA

SENAC - FUTURO PROGRAMADOR

Contextualização

O ‘match case‘ foi introduzido no Python 3.10 como uma forma de implementar pattern matching, permitindo criar códigos mais limpos e organizados. Isso é especialmente útil em operações que envolvem múltiplas condições.

Definição Teórica

Pattern matching é uma técnica usada para verificar uma estrutura específica em dados e executar ações baseadas nessa estrutura. O ‘match case‘ permite combinar expressões e executar códigos dependendo do valor ou padrão.

Situação Problema

Imagine um sistema que identifica o tipo de arquivo enviado para um servidor e executa uma ação apropriada, como abrir, arquivar ou deletar.

Código para o Problema

```
def file_action(file_type):
    match file_type:
        case "txt":
            return "Abrir arquivo texto."
        case "zip":
            return "Descompactar arquivo."
        case "jpg" | "png":
            return "Abrir imagem."
        case _:
            return "Tipo desconhecido."
```

Exemplo Lúdico

Analogamente, o ‘match case‘ funciona como um sistema de classificação de frutas: - Uma maçã? Descascar! - Uma laranja? Espremer! - Uma banana? Cortar!

Código Python Completo

```python
def classificar_fruta(fruta):
    match fruta:
        case "maçã":
            print("Descascar a maçã!")
        case "laranja":
            print("Espremer a laranja!")
        case "banana":
            print("Cortar a banana!")
        case _:
            print("Ação desconhecida para essa fruta.")

# Entrada do usuário
fruta = input("Digite o nome de uma fruta: ")
classificar_fruta(fruta)
```

CRUD com Match Case

Requisitos: - Criar, ler, atualizar e deletar dados de usuários.

Código: Funções CRUD

```
def crud_operation(action, data):
    match action:
        case "create":
            return f"Usuário {data} criado."
        case "read":
            return f"Lendo dados de {data}."
        case "update":
            return f"Atualizando {data}."
        case "delete":
            return f"Deletando {data}."
        case _:
            return "Ação desconhecida."
```

Introdução

Utilizamos o recurso `match case` do Python para classificar frutas com base na entrada do usuário. Veja o código e sua funcionalidade abaixo.

Código: Recebendo Entrada do Usuário

```
Digite o nome de uma fruta:
```

Código: Classificação

```
variavel = input("Digite o nome de uma fruta: ")

match variavel:
    case "maçã":
        print("É uma maçã!")
    case "banana":
        print("É uma banana!")
    case _:
        print("Fruta desconhecida")
```

Exemplo 1 - Número para texto

Enunciado:
O usuário insere um número de 1 a 3, e o programa retorna o número em texto. Caso não seja reconhecido, exibe uma mensagem padrão.

Exemplo 1 - Código

```
numero = int(input("Digite um número (1-3): "))

match numero:
    case 1:
        print("Um")
    case 2:
        print("Dois")
    case 3:
        print("Três")
    case _:
        print("Número desconhecido")
```

Exemplo 2 - Combinar múltiplos valores

Enunciado:
O usuário insere uma letra, e o programa verifica se é uma vogal ou consoante utilizando o operador ‘—‘.

Exemplo 2 - Código

```
letra = input("Digite uma letra: ").lower()

match letra:
    case "a" | "e" | "i" | "o" | "u":
        print("É uma vogal")
    case _:
        print("É uma consoante")
```

Exemplo 3 - Trabalhando com tipos diferentes

Enunciado:
O usuário insere um valor (lista vazia, lista com elementos ou outro tipo de dado). O programa identifica e exibe a classificação.

Exemplo 3 - Código

```
valor = eval(input("Digite um valor (ex: [], [1, 2, 3], 10): "))

match valor:
    case []:
        print("Lista vazia")
    case [primeiro, *resto]:
        print(f"Primeiro elemento: {primeiro}")
    case _:
        print("Outro tipo de dado")
```

Exercício 1 - Número para texto

Enunciado:
Receba um número e informe se ele é 1, 2 ou outro número. Caso o número não seja 1 ou 2, exiba uma mensagem padrão.

Exercício 1 - Código

```
numero = int(input("Digite um número: "))

match numero:
    case 1:
        print("Você digitou UM")
    case 2:
        print("Você digitou DOIS")
    case _:
        print("Outro número")
```

Exercício 2 - Verificar vogal ou consoante

Enunciado:

Receba uma letra do usuário e informe se ela é uma vogal ou consoante.

Exercício 2 - Código

```
letra = input("Digite uma letra: ").lower()

match letra:
    case "a" | "e" | "i" | "o" | "u":
        print("Vogal")
    case _:
        print("Consoante")
```

Exercício 3 - Dia da semana

Enunciado:
Receba o número de um dia (1-7) e informe o nome do dia correspondente. Caso o número seja inválido, exiba uma mensagem de erro.

Exercício 3 - Código

```
dia = int(input("Digite o número do dia (1-7): "))

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

Exercício 4 - Classificar nota escolar

Enunciado:
Receba uma nota (0 a 10) do usuário e classifique como: Excelente, Bom, Regular ou Reprovado.

Exercício 4 - Código

```
nota = float(input("Digite sua nota (0 a 10): "))

match nota:
    case _ if 9 <= nota <= 10:
        print("Excelente")
    case _ if 7 <= nota < 9:
        print("Bom")
    case _ if 5 <= nota < 7:
        print("Regular")
    case _:
        print("Reprovado")
```

Exercício 5 - Identificar tipo de triângulo

Enunciado:
Receba os lados de um triângulo e classifique como: Equilátero, Isósceles ou Escaleno.

Exercício 5 - Código

```
l1 = float(input("Lado 1: "))
l2 = float(input("Lado 2: "))
l3 = float(input("Lado 3: "))

match (l1, l2, l3):
    case _ if l1 == l2 == l3:
        print("Equilátero")
    case _ if l1 == l2 or l1 == l3 or l2 == l3:
        print("Isósceles")
    case _:
        print("Escaleno")
```

Exercício 6 - Detectar estação do ano

Enunciado:

Receba o número de um mês (1-12) e informe a estação correspondente.

Caso o número seja inválido, exiba uma mensagem de erro.

Exercício 6 - Código

```
mes = int(input("Digite o número do mês (1-12): "))

match mes:
    case 12 | 1 | 2:
        print("Verão")
    case 3 | 4 | 5:
        print("Outono")
    case 6 | 7 | 8:
        print("Inverno")
    case 9 | 10 | 11:
        print("Primavera")
    case _:
        print("Mês inválido")
```

Exercício 7 - Simular um menu

Enunciado:
Simule um menu com as opções de Sacar, Depositar, Ver saldo e Sair.
Informe a ação correspondente ao usuário.

Exercício 7 - Código

```
print("""
1 - Sacar
2 - Depositar
3 - Ver saldo
4 - Sair
""")
opcao = int(input("Escolha uma opção: "))

match opcao:
    case 1:
        print("Saque realizado")
    case 2:
        print("Depósito realizado")
    case 3:
        print("Seu saldo é R$ 1000,00")
    case 4:
        print("Saindo...")
```

Exercício 8 - Verificar tipo de animal

Enunciado:
Receba o nome de um animal do usuário e informe se é doméstico, selvagem ou desconhecido.
Exemplos: Cachorro, Gato, Hamster (domésticos); Leão, Tigre, Elefante (selvagens).

Exercício 8 - Código

```
animal = input("Digite um animal: ").lower()

match animal:
    case "cachorro" | "gato" | "hamster":
        print("É um animal doméstico")
    case "leão" | "tigre" | "elefante":
        print("É um animal selvagem")
    case _:
        print("Animal desconhecido")
```

Comparação Match Case x Switch Case

Vamos comparar o ‘match case‘ do Python com o ‘switch case‘ presente em C, JavaScript e Java. Cada abordagem tem suas peculiaridades que atendem diferentes necessidades de programação.

Aspecto: Início

Python: `match variavel:`
C/JavaScript/Java: `switch(variavel) {`

Aspecto: Caso

Python: `case valor:`
C/JavaScript/Java: `case valor:`

Aspecto: Padrão

Python: `case _:`
C/JavaScript/Java: `default:`

Aspecto: Quebra Automática?

Python: Sim (não precisa `break`).
C/JavaScript/Java: Não (precisa `break`).

Aspecto: Padrões Complexos?

Python: Sim! (como tuplas e classes).
C/JavaScript/Java: Não (apenas valores simples).

Conclusão

O `match case` do Python oferece maior flexibilidade em comparação ao `switch case` de outras linguagens, permitindo padrões mais complexos e eliminando a necessidade de `break`.