Python - Instrução Match-Case

Antes de sua versão 3.10, o Python não possuía um recurso semelhante ao switch-case em C ou C++ . No Python 3.10, uma técnica de correspondência de padrões chamada **match-case** foi introduzida, que é semelhante à construção **switch-case** disponível em C/C++/Java etc.

Declaração de caso de correspondência Python

Uma instrução **match-case** do Python pega uma expressão e compara seu valor com padrões sucessivos dados como um ou mais blocos case. O uso é mais semelhante à correspondência de padrões em linguagens como Rust ou Haskell do que uma instrução switch em C ou C++. Somente o primeiro padrão correspondente é executado. Também é possível extrair componentes (elementos de sequência ou atributos de objetos) do valor em variáveis .

O uso básico de **match-case** é comparar uma variável com um ou mais valores.

Sintaxe

A seguir está a sintaxe da instrução match-case em Python -

```
match variable_name:
   case 'pattern 1' : statement 1
   case 'pattern 2' : statement 2
   ...
   case 'pattern n' : statement n
```

Exemplo

O código a seguir possui uma função chamada weekday(). Ele recebe um argumento inteiro, combina-o com todos os valores possíveis de números de dias da semana e retorna o nome do dia correspondente.

```
def weekday(n):
   match n:
      case 0: return "Monday"
      case 1: return "Tuesday"
      case 2: return "Wednesday"
      case 3: return "Thursday"
      case 4: return "Friday"
      case 5: return "Saturday"
      case 6: return "Sunday"
      case _: return "Invalid day number"
```

```
print (weekday(3))
print (weekday(6))
print (weekday(7))
```

Saída

Ao ser executado, este código produzirá a seguinte saída -

```
Thursday
Sunday
Invalid day number
```

A última instrução case na função tem "_" como valor a ser comparado. Ele serve como caso curinga e será executado se todos os outros casos não forem verdadeiros.

Casos combinados na declaração de correspondência

Às vezes, pode haver uma situação em que, em mais de um caso, uma ação semelhante tenha que ser tomada. Para isso, você pode combinar os casos com o operador OR representado por "|" símbolo.

Exemplo

```
def access(user):
   match user:
      case "admin" | "manager": return "Full access"
      case "Guest": return "Limited access"
      case _: return "No access"
print (access("manager"))
print (access("Guest"))
print (access("Ravi"))
```

Saída

O código acima define uma função chamada access() e possui um argumento de string, representando o nome do usuário. Para usuário administrador ou gerente, o sistema concede acesso total; para Hóspedes o acesso é limitado; e para o resto, não há acesso.

```
Full access
Limited access
No access
```

Listar como o argumento na declaração Match Case

Como o Python pode comparar a expressão com qualquer literal , você pode usar uma lista como valor de caso. Além disso, para um número variável de itens na lista, eles podem ser analisados em uma sequência com o operador "*".

Exemplo

```
def greeting(details):
   match details:
      case [time, name]:
         return f'Good {time} {name}!'
      case [time, *names]:
         msg=''
         for name in names:
            msg+=f'Good {time} {name}!\n'
         return msg

print (greeting(["Morning", "Ravi"]))
print (greeting(["Afternoon","Guest"]))
print (greeting(["Evening", "Kajal", "Praveen", "Lata"]))
```

Saída

Ao ser executado, este código produzirá a seguinte saída -

```
Good Morning Ravi!
Good Afternoon Guest!
Good Evening Kajal!
Good Evening Praveen!
Good Evening Lata!
```

Usando "if" na cláusula "Case"

Normalmente o Python compara uma expressão com casos literais. No entanto, permite incluir a instrução if na cláusula case para cálculo condicional da variável de correspondência.

Exemplo

No exemplo a seguir, o argumento da função é uma lista de valor e duração, e o conjunto de juros deve ser calculado para um valor menor ou superior a 10.000. A condição está incluída na cláusula **case** .

```
def intr(details):
   match details:
      case [amt, duration] if amt<10000:
         return amt*10*duration/100
      case [amt, duration] if amt>=10000:
         return amt*15*duration/100
print ("Interest = ", intr([5000,5]))
print ("Interest = ", intr([15000,3]))
```

Saída

Ao ser executado, este código produzirá a seguinte saída -

```
Interest = 2500.0
Interest = 6750.0
```