Aula 2 – Estruturas de Seleção

Objetivo da Aula

Compreender o conceito e os funcionamentos das estruturas de seleção. Aplicar corretamente a Estrutura de Seleção Simples, a Estrutura de Seleção Composta e a Estrutura de Seleção Encadeada.

Apresentação

Quando utilizamos uma linguagem de programação para resolver um problema, escrevemos uma sequência de instruções, de forma lógica, que será executada para obter a solução desejada. Na sua forma mais básica, essas instruções envolvem a declaração de variáveis e o uso de comandos para entrada, processamento e saída.

No entanto, essa forma de programar (chamada de "estrutura sequencial") nem sempre será suficiente para solucionar um problema, como comparar valores para cumprir uma tarefa. Em situações como essa, são usadas estruturas de seleção.

Vamos aprender como utilizar as estruturas de seleção para tratar diferentes tipos de comparação na linguagem Python e analisar alguns exemplos práticos.

1. Estrutura de Seleção Simples

A estrutura de seleção simples é a forma mais básica de estrutura de seleção. Ela permite que o programa execute um bloco de código se uma condição especificada for verdadeira. Caso contrário, o bloco de código é ignorado, e o programa segue para a próxima linha de código.

Em Python, a estrutura de seleção simples é implementada usando a palavra-chave "if". A sintaxe básica é a seguinte:

```
if condição:
bloco de código
```

O bloco de código só será executado se a condição especificada for verdadeira. A condição pode ser qualquer expressão que possa ser avaliada como True ou False. Por exemplo:

```
x = 10
if x > 5:
    print("x é maior que 5")
```

Nesse exemplo, a condição é "x > 5", que é verdadeira, porque x tem o valor 10. Portanto, o bloco de código é executado e a mensagem "x é maior que 5" é exibida.

Na linguagem Python, os operadores relacionais são usados em estruturas de seleção para comparar valores e avaliar se uma condição é verdadeira ou falsa. Existem seis operadores relacionais básicos: "<", ">", "<=", ">=", "==" e "!=".

O operador "<" retorna True se o valor da esquerda for menor que o valor da direita. Por exemplo:

```
a = 10
b = 5
if b < a:
    print("b é menor que a")
```

Nesse caso, a condição é verdadeira, pois o valor da variável "b" é menor que o valor da variável "a".

O operador ">" retorna True se o valor da esquerda for maior que o valor da direita. Por exemplo:

```
a = 10
b = 5
if a > b:
    print("a é maior que b")
```

Nesse caso, a condição é verdadeira, visto que o valor da variável "a" é maior que o valor da variável "b".

O operador "<=" retorna True se o valor da esquerda for menor ou igual ao valor da direita. Por exemplo:

```
a = 10
b = 5
if b <= a:
    print("b é menor ou igual a a"
```

Nesse caso, a condição é verdadeira, pois o valor da variável “b” é menor ou igual ao valor da variável “a”.

O operador “>=“ retorna True se o valor da esquerda for maior ou igual ao valor da direita. Por exemplo:

```
a = 10
b = 5
if a >= b:
    print("a é maior ou igual a b")
```

Nesse caso, a condição é verdadeira, uma vez que o valor da variável “a” é maior ou igual ao valor da variável “b”.

O operador “==“ retorna True se o valor da esquerda for igual ao valor da direita. Por exemplo:

```
a = 10
b = 5
if a == 10:
    print("a é igual a 10")
```

Nesse caso, a condição é verdadeira, pois o valor da variável “a” é igual a 10.

O operador “!=“ retorna True se o valor da esquerda for diferente do valor da direita. Por exemplo:

```
a = 10
b = 5
if a != b:
    print("a é diferente de b")
```

Nesse caso, a condição é verdadeira, pois o valor da variável “a” é diferente do valor da variável “b”.

Na linguagem Python, os operadores lógicos são usados em estruturas de seleção para avaliar expressões booleanas e determinar se uma condição é verdadeira ou falsa. Existem três operadores lógicos básicos: “and”, “or” e “not”.

O operador “and” retorna True se ambas as expressões avaliadas forem verdadeiras. Por exemplo:

```
a = 10
b = 5
if a > 5 and b < 10:
    print("Ambas as expressões são verdadeiras")
```

Nesse caso, a condição é verdadeira, pois a variável “a” é maior que 5, e a variável “b” é menor que 10. O operador “or” retorna True se pelo menos uma das expressões avaliadas for verdadeira. Por exemplo:

```
a = 10
b = 5
if a > 5 or b > 10:
    print("Pelo menos uma das expressões é verdadeira")
```

Nesse caso, a condição é verdadeira, já que a variável “a” é maior que 5. O operador “not” inverte o valor de uma expressão booleana. Por exemplo:

```
a = 10
if not a == 5:
    print("A condição é verdadeira")
```

Nesse caso, a condição é verdadeira, pois a variável “a” não é igual a 5.

É possível, também, combinar os operadores lógicos em expressões mais complexas. Por exemplo:

```
a = 10
b = 5
if (a > 5 and b < 10) or not (a == 10):
    print("A condição é verdadeira")
```

Nesse caso, a condição é verdadeira, uma vez que a primeira expressão é verdadeira e a segunda expressão é falsa.

2. Estrutura de Seleção Composta

A estrutura de seleção composta permite que o programa execute um bloco de código, se uma condição especificada for verdadeira, e execute outro bloco de código se a condição for falsa. Em Python, a estrutura de seleção composta é implementada usando a palavra-chave “if” para a primeira condição e a palavra-chave “else” para a condição contrária. A sintaxe básica é a seguinte:

```
if condição:
bloco de código 1
else:
bloco de código 2
```

O bloco de código 1 é executado se a condição for verdadeira, e o bloco de código 2 é executado se a condição for falsa. Por exemplo:

```
x = 10
if x > 5:
    print("x é maior que 5")
else:
    print("x é menor ou igual a 5")
```

Nesse exemplo, a condição é "x > 5", que é verdadeira, porque x tem o valor 10. Portanto, o bloco de código 1 é executado, e a mensagem "x é maior que 5" é exibida.

3. Estrutura de Seleção Encadeada

A estrutura de seleção encadeada permite que o programa execute diferentes blocos de código, dependendo de várias condições. Em Python, a estrutura de seleção encadeada é implementada usando a palavra-chave "if" para a primeira condição, a palavra-chave "elif" (abreviação de "else if") para as condições intermediárias, e a palavra-chave "else" para a condição final. A sintaxe básica é a seguinte:

```
if condição 1:
bloco de código 1
elif condição 2:
bloco de código 2
elif condição 3:
bloco de código 3
else:
bloco de código 4
```

Veja um exemplo dessa estrutura na linguagem Python:

```
idade = 20
if idade >= 18:
    print("Você é maior de idade!")
elif idade >= 16:
    print("Você é adolescente!")
else:
    print("Você é menor de idade!")
```

Nesse exemplo, o programa verifica se a idade é maior ou igual a 18. Se for, o bloco de código dentro do if será executado, e "Você é maior de idade!" será impresso. Caso contrário, o programa verifica se a idade é maior ou igual a 16. Se for, o bloco de código dentro do elif será executado, e "Você é adolescente!" será impresso. Caso contrário, o bloco de código dentro do else será executado, e "Você é menor de idade!" será impresso.

Ao programar em Python, é importante lembrar que a indentação é usada para indicar os blocos de código associados às estruturas de controle, como as estruturas de seleção. Cada bloco de código deve estar indentado em relação ao bloco de código anterior. Por exemplo:

```
idade = 20
if idade >= 18:
    print("Você é maior de idade!")
    print("Parabéns!")
else:
    print("Você é menor de idade!")
```

Nesse caso, o segundo bloco de código dentro do if está indentado em relação ao primeiro bloco de código dentro do if. Isso indica que o segundo bloco de código está associado ao primeiro bloco de código dentro do if.

4. Comandos Adicionais e Exemplos

Se você está comparando o mesmo valor com várias constantes ou verificando por tipos ou atributos específicos, você também pode achar a instrução match útil. Ela foi lançada na versão 3.10 do Python.

Uma instrução match pega uma expressão e compara seu valor com padrões sucessivos, fornecidos como um ou mais blocos de case. Isso é superficialmente semelhante a uma instrução switch em C, Java ou JavaScript (e muitas outras linguagens). Observe o exemplo a seguir:

```
sigla = 'SP'
match sigla:
    case 'RJ':
        print('Rio de Janeiro')
    case 'SP':
        print('São Paulo')
    case 'MG':
        print('Minas Gerais')
    case 'ES':
        print('Espírito Santo')
    case other:
        print('A sigla não foi identificada como parte da região sudeste')
```

No exemplo, o valor armazenado na variável "sigla" é comparado com cada possibilidade indicada pela instrução case, e, em seguida, será exibido um texto por meio do comando print. Note que, no último bloco, a instrução other atua como um curinga para o caso de nenhuma outra possibilidade corresponder. Essa instrução poderia ser substituída pelo símbolo de sublinhado, conforme essa outra versão:

```
sigla = 'SP'
match sigla:
    case 'RJ':
        print('Rio de Janeiro')
    case 'SP':
        print('São Paulo')
    case 'MG':
        print('Minas Gerais')
    case 'ES':
        print('Espírito Santo')
    case _:
        print('A sigla não foi identificada como parte da região sudeste')
```

Agora que já conhecemos as estruturas de seleção e o seu funcionamento na linguagem Python, vamos trabalhar com exemplos práticos em alguns cenários.

- **Cenário 1**: imagine que um programador tenha que construir um programa que leia o salário bruto de um empregado, calcule e informe sua gratificação (com base em seu salário bruto) e seu salário líquido com base na tabela a seguir.

Faixa salarial (R$)	Gratificação (R$)
Menor que 2000	5%
De 2000 a 2500	10%
Maior que 2500 e menor ou igual a 3000	15%
Maior que 3000	20%

```
salario_bruto = float(input('digite o seu salario:'))
if salario_bruto < 2000:
    gratificaçao = salario_bruto * 5/100
elif salario_bruto >= 2000 and salario_bruto <= 2500:
    gratificaçao = salario_bruto * 10/100
elif salario_bruto > 2500 and salario_bruto <= 3000:
    gratificaçao = salario_bruto * 15/100
else:
    gratificaçao = salario_bruto * 20/100
salario_liquido = salario_bruto + gratificaçao
print(f'seu salario liquido é de R${salario_liquido:.2f}')
print(f'Sua gratificação foi de R${gratificaçao:.2f}')
```

Esse código é um exemplo de como um programa em Python pode calcular o salário líquido de um funcionário com base em seu salário bruto, considerando diferentes faixas de gratificação, de acordo com o valor do salário bruto.

Vamos analisar linha por linha:

```
1   salario_bruto = float(input('digite o seu salario:'))
```

Nessa linha, o programa solicita ao usuário que digite o valor do salário bruto. O valor é lido como uma string e, em seguida, convertido em um número de ponto flutuante (float), usando a função float(), e atribuído à variável salario_bruto.

```
2   if salario_bruto < 2000:
3       gratificaçao = salario_bruto * 5/100
```

Nessa linha, o programa verifica se o salário bruto é menor que 2000. Se sim, o programa calcula a gratificação como 5% do salário bruto e armazena o resultado na variável gratificação.

```
4   elif salario_bruto >= 2000 and salario_bruto <= 2500:
5       gratificaçao = salario_bruto * 10/100
```

Se a primeira condição não for verdadeira, o programa verifica se o salário bruto está entre 2000 e 2500. Se sim, o programa calcula a gratificação como 10% do salário bruto e armazena o resultado na variável gratificação.

```
6   elif salario_bruto > 2500 and salario_bruto <= 3000:
7       gratificaçao = salario_bruto * 15/100
```

Se a segunda condição não for verdadeira, o programa verifica se o salário bruto está entre 2500 e 3000. Se sim, o programa calcula a gratificação como 15% do salário bruto e armazena o resultado na variável gratificação.

```
8   else:
9       gratificaçao = salario_bruto * 20/100
```

Se nenhuma das condições anteriores for verdadeira, o programa assume que o salário bruto é maior que 3000, calcula a gratificação como 20% do salário bruto e armazena o resultado na variável gratificação.

```
10  salario_liquido = salario_bruto + gratificaçao
```

Nessa linha, o programa calcula o salário líquido, somando o salário bruto e a gratificação, e armazena o resultado na variável salario_liquido.

```
11    print(f'seu salario liquido é de R${salario_liquido:.2f}')
```

Nessa linha, o programa imprime na tela uma mensagem com o valor do salário líquido, formatando-o como um número de ponto flutuante, com duas casas decimais, e adicionando um prefixo que indica a moeda (R$).

```
12    print(f'Sua gratificação foi de R${gratificaçao:.2f}')
```

Nessa linha, o programa imprime na tela uma mensagem com o valor da gratificação, formatando-o como um número de ponto flutuante, com duas casas decimais, e adicionando um prefixo que indica a moeda (R$).

Com esse código, o programa consegue calcular o salário líquido de um funcionário, com base em seu salário bruto e na faixa de gratificação correspondente ao valor do salário bruto.

- **Cenário 2**: agora o programador precisa escrever um código que seja capaz de armazenar um valor inteiro de 1 a 7, para representar o dia da semana, e, ao final, mostre na tela o dia da semana correspondente ao valor armazenado.

Vamos à primeira solução para o problema:

```
1   semana = int(input('digite um numero:'))
2   if semana == 1:
3       print('domingo')
4   elif semana == 2:
5       print('segunda')
6   elif semana == 3:
7       print('terça')
8   elif semana == 4:
9       print('quarta')
10  elif semana == 5:
11      print('quinta')
12  elif semana == 6:
13      print('sexta')
14  elif semana == 7:
15      print('sabado')
16  else:
17      print('Dia invalido')
```

A primeira linha do código solicita que o usuário digite um número, que é armazenado na variável semana como um inteiro (usando a função int).

Em seguida, começa uma estrutura if-elif-else, que é uma maneira comum em Python de testar várias condições.

A primeira condição testada é se semana é igual a 1. Se essa condição for verdadeira, o código imprime “domingo”.

Caso a primeira condição seja falsa, a próxima condição é testada: se semana é igual a 2. Se essa condição for verdadeira, o código imprime “segunda”.

Se a segunda condição for falsa, a terceira condição é testada, e assim por diante, até que todas as condições sejam testadas.

Se semana não corresponder a nenhum dos valores testados nas condições if e elif, a condição else é executada, e o código imprime “Dia inválido”.

O código termina após a execução da última linha de código.

```
1   semana = int(input('digite um numero:'))
2   match semana:
3       case 1:
4           print('domingo')
5       case 2:
6           print('segunda')
7       case 3:
8           print('terça')
9       case 4:
10          print('quarta')
11      case 5:
12          print('quinta')
13      case 6:
14          print('sexta')
15      case 7:
16          print('sabado')
17      case other:
18          print('Dia invalido')
```

A primeira linha do código solicita ao usuário que digite um número, que será armazenado na variável semana.

Em seguida, a sintaxe de match case é utilizada para testar o valor da variável semana. Essa sintaxe é nova no Python 3.10 e permite fazer múltiplos testes em uma expressão, tornando o código mais legível e fácil de entender.

Cada case representa um dia da semana, com os valores 1 a 7 correspondendo de domingo a sábado, respectivamente.

Se o valor de semana corresponder a um dos valores dos casos declarados, o código imprime o nome do dia correspondente.

Caso semana não corresponda a nenhum dos valores dos casos declarados, o código imprime “Dia inválido”.

A sintaxe other é utilizada para tratar casos em que o valor da variável semana não corresponde a nenhum dos casos especificados. Se other não fosse especificado, o código geraria uma exceção ValueError quando o valor da variável semana não correspondesse a nenhum dos casos.

Note que a sintaxe de match case é um recurso relativamente novo no Python e só está disponível a partir da versão 3.10. Caso você esteja usando uma versão anterior do Python, a sintaxe de if-elif-else seria mais apropriada para testar o valor da variável semana.

Considerações Finais da Aula

As estruturas de seleção são fundamentais para resolver problemas de programação que envolvam comparações lógicas. Assim como em outras linguagens de programação, a linguagem Python oferece diferentes maneiras de comparar valores e executar uma ou mais instruções, de acordo com o resultado da comparação lógica especificada na estrutura de seleção utilizada pelo programador.

Para que uma estrutura de seleção auxilie na obtenção da solução, é necessário estar atento às regras de indentação do código, à sintaxe adequada para a estrutura escolhida e, também, ao uso adequado dos operadores lógicos e relacionais que forem utilizados na estrutura.

Além disso, é importante verificar se o código utilizado na estrutura de seleção foi escrito de forma a verificar todas as possibilidades possíveis para o problema a ser tratado.

Materiais Complementares

Livro:

Introdução à programação com Python
2010, Nilo Ney Coutinho Menezes. São Paulo, Novatec.

Esse livro aborda os conceitos básicos da linguagem Python, os quais vimos nesta aula, bem como a sua sintaxe e os seus comandos básicos.

Site

AlexandreLouzada/Pyquest: Pyquest/envExemplo/Lista02
2023, Alexandre N. Louzada. Github.
O link indicado permite o acesso a um repositório com várias listas de exemplo de programas em Python para auxiliar estudantes de programação.
Link para acesso: https://github.com/AlexandreLouzada/Pyquest/tree/master/envExemplo/Lista02 (acesso em 24 maio 2023.)

Referências

ALVES, William P. ***Programação Python***: aprenda de forma rápida. [s.l.]: Editora Saraiva, 2021.

PYTHON SOFTWARE FOUNDATION. ***Python Language Site***. Documentation, 2023. Página de documentação. Disponível em: https://docs.python.org/pt-br/3/tutorial/index.html. Acesso em: 8 mar. 2023.