Aula 3 – Estruturas de Repetição

Objetivo da Aula

Compreender o conceito e os funcionamentos das estruturas de repetição. Aplicar corretamente os contadores e acumuladores, bem como o controle de fluxo de estruturas de repetição.

Apresentação

Quando utilizamos uma linguagem de programação, pode ser necessário utilizar uma instrução ou até uma sequência de instruções diversas vezes. Isso torna a escrita do código trabalhosa e mais suscetível a erros.

Para tratar situações como essa, são usadas estruturas de repetição. Elas permitem que, em uma linguagem de programação, possamos usar uma instrução ou um grupo de instruções repetidas vezes em um bloco de código. Isso reduz o tamanho do código e torna sua leitura mais compreensível, além de minimizar a possibilidade de erros na escrita do código.

Em Python, existem dois tipos principais de estruturas de repetição: “for” e “while”. Vamos aprender como utilizar as estruturas de repetição para tratar diferentes tipos de problemas, na linguagem Python, e analisar alguns exemplos práticos.

1. Estrutura de Seleção “For”

A estrutura for é usada para iterar sobre uma sequência de elementos, como uma lista, uma tupla, um dicionário, um conjunto, uma string ou um objeto iterável. A sintaxe básica é a seguinte:

```
for variavel in sequencia:
# bloco de código
```

Aqui, variavel é a variável que recebe cada elemento da sequência um por vez, e sequencia é a sequência que estamos iterando. Na linguagem Python, uma das formas de trabalhar com uma sequência de elementos é por meio da função range().

A função range() é uma função integrada do Python que cria uma sequência de números inteiros. Ela é comumente usada em conjunto com a estrutura de repetição for, para executar um bloco de código em determinado número de vezes.

A sintaxe básica da função range() é a seguinte:

```
range(inicio, fim, passo)
```

Em que:

- inicio é o primeiro número da sequência. O valor padrão é 0;
- fim é o último número da sequência (não incluso). Esse é um parâmetro obrigatório;
- passo é a diferença entre cada número na sequência. O valor padrão é 1.

Por exemplo, o código a seguir criará uma sequência de números de 0 a 9:

```
for numero in range(10):
    print(numero)
```

Isso imprimirá os números de 0 a 9 na tela.

Agora, veja esse outro exemplo de uso da estrutura de repetição for com a função range() para imprimir números pares:

```
for numero in range(0, 11, 2):
    print(numero)
```

Isso imprimirá os números pares de 0 a 10 na tela.

1.1. Contadores e Acumuladores

Em programação, um contador é uma variável que é usada para contar quantas vezes determinado trecho de código é executado para iterar em uma estrutura de dados.

Por exemplo, em uma estrutura for, um contador pode ser usado para manter o controle do número de vezes que a repetição foi executada. Em um programa de contagem de votos, um contador pode ser usado para manter o controle do número total de votos registrados.

Já um acumulador é uma variável que é usada para armazenar valores, à medida que são processados em uma repetição. O acumulador geralmente começa com um valor inicial e é atualizado a cada iteração do loop, adicionando o valor atual a ele. Por exemplo, em um programa que calcula a média de um conjunto de números, um acumulador pode ser usado para manter a soma dos números.

Esses conceitos são importantes em programação, pois permitem que você controle o fluxo do seu programa e processe grandes quantidades de dados de forma eficiente.

Os conceitos de contadores e acumuladores são comuns no uso da estrutura de repetição for. Vamos a um exemplo:

```
1   # Definindo o valor inicial do contador e do acumulador
2   contador = 0
3   acumulador = 0
4
5   # Usando a repetição para percorrer um intervalo de números de 1 a 5
6   for i in range(1, 6):
7       # Incrementando o contador a cada iteração
8       contador += 1
9
10      # Acumulando o valor da variável i no acumulador a cada iteração
11      acumulador += i
12
13  # Imprimindo o resultado do contador e do acumulador
14  print("Contador:", contador)
15  print("Acumulador:", acumulador)
```

Na primeira linha, estamos definindo o valor inicial de duas variáveis: contador e acumulador. Essas variáveis serão usadas para exemplificar os conceitos de contadores e acumuladores no for.

Na segunda linha, estamos usando a estrutura de repetição for para percorrer um intervalo de números de 1 a 5 (ou seja, [1, 2, 3, 4, 5]). A cada iteração, a variável i assume um valor desse intervalo.

Na terceira linha, estamos incrementando o valor da variável contador em 1 unidade a cada iteração, para contar quantas vezes o loop foi executado.

Na quarta linha, estamos acumulando o valor da variável i na variável acumulador a cada iteração, para somar todos os valores do intervalo de números.

Por fim, na quinta e na sexta linha, estamos imprimindo o valor final do contador e do acumulador.

Agora que já conhecemos as estruturas de repetição for e seu funcionamento na linguagem Python, vamos analisar exemplos práticos, trabalhando alguns cenários.

- **Cenário 1**: um programador precisa desenvolver um código em Python capaz de contabilizar a quantidade de votos de dois candidatos a representante de turma de uma instituição de ensino. A turma é composta de 30 alunos, que devem escolher entre o candidato 1 (identificado pelo valor inteiro 1) ou o candidato 2 (identificado pelo valor inteiro 2). Caso o aluno insira um valor diferente dos valores indicados, para identifi-

car os candidatos 1 e 2, o voto será considerado nulo. O programa deve apresentar: a quantidade de votos de cada candidato e a quantidade de votos nulos.

```python
candidato_1 = 0
candidato_2 = 0
votos_nulos = 0

for i in range(30):
    voto = int(input(f"Digite o voto do aluno {i+1}: "))
    if voto == 1:
        candidato_1 += 1
    elif voto == 2:
        candidato_2 += 1
    else:
        votos_nulos += 1

print(f"Total de votos para o candidato 1: {candidato_1}")
print(f"Total de votos para o candidato 2: {candidato_2}")
print(f"Total de votos nulos: {votos_nulos}")
```

Nesse exemplo, nós inicializamos três variáveis: candidato_1, candidato_2 e votos_nulos, todas com o valor 0. Em seguida, utilizamos um loop for para pedir o voto de cada aluno. O loop é executado 30 vezes (número total de alunos na turma), e a variável i é usada para indicar o número do aluno.

Dentro do loop, utilizamos um if para verificar qual candidato foi escolhido ou se o voto é nulo. Se o voto for para o candidato 1, incrementamos o contador candidato_1. Se o voto for para o candidato 2, incrementamos o contador candidato_2. Caso contrário, incrementamos o contador votos_nulos.

Ao final, utilizamos o comando print para mostrar a quantidade de votos para cada candidato e a quantidade de votos nulos.

- **Cenário 2**: um programador precisa desenvolver um programa que calcula a média de um conjunto de 10 números inteiros.

```python
soma = 0
for i in range(10):
    numero = int(input(f"Digite o {i+1}º número: "))
    soma += numero

media = soma / 10
print(f"A média dos números é: {media}")
```

Nesse exemplo, nós inicializamos uma variável soma com o valor 0. Em seguida, utilizamos um loop for para pedir ao usuário que digite 10 números inteiros. A variável i é usada para indicar o número do número digitado.

Dentro do loop, armazenamos cada número digitado na variável numero e adicionamos esse valor à variável soma.

Após o loop, calculamos a média, dividindo a soma dos números por 10, e armazenamos esse valor na variável media. Por fim, utilizamos o comando print para mostrar o resultado da média.

2. Estrutura de Seleção “While”

A estrutura while é usada para repetir um bloco de código enquanto determinada condição for verdadeira. A sintaxe básica é a seguinte:

```
while condicao:
# bloco de código
```

Aqui, condicao é a condição que deve ser avaliada a cada iteração do loop. Enquanto a condição for verdadeira, o bloco de código será executado. É importante ter cuidado ao usar a estrutura while, pois, se a condição nunca se tornar falsa, o loop continuará indefinidamente, e seu programa poderá ficar preso em um loop infinito.

Abaixo estão alguns exemplos simples para ajudar a entender melhor essas estruturas de repetição em Python:

```
1  # Definindo o número inicial
2  numero = 1
3
4  # Usando while para imprimir os números de 1 a 5
5  while numero <= 5:
6      print(numero)
7      numero += 1
```

Na primeira linha, estamos definindo a variável numero com o valor inicial de 1, que será usado como exemplo para a estrutura de repetição while.

Na segunda linha, estamos usando a estrutura de repetição while para imprimir os números de 1 a 5. A condição numero <= 5 verifica se a variável numero ainda é menor ou igual a 5. Enquanto essa condição for verdadeira, o bloco de código dentro do while será executado.

Na terceira linha, estamos imprimindo o valor da variável numero naquela iteração do while.

Na quarta linha, estamos incrementando o valor da variável numero em 1 unidade, para que a próxima iteração imprima o próximo número.

2.1. Uso de Break e Continue

Tanto break quanto continue são palavras-chave em Python que são usadas para controlar o fluxo de execução de um loop (seja um loop for ou while).

O break é usado para encerrar a execução de um loop prematuramente, ou seja, interromper o loop antes que ele termine de executar todas as iterações previstas. Quando o break é encontrado dentro do loop, o código imediatamente sai do loop e continua a execução da próxima instrução após o loop.

Já o continue é usado para pular uma iteração do loop e ir para a próxima iteração. Quando o continue é encontrado dentro do loop, o código ignora o restante do bloco de código, dentro da iteração atual, e passa imediatamente para a próxima iteração do loop.

A seguir, veja exemplos de como utilizar o break e o continue em loops.

- Exemplo de uso de break:

```
1  i = 0
2  while i < 10:
3      i += 1
4      if i == 5:
5          break
6      print(i)
7  print("fim")
```

Nesse exemplo, o loop while vai executar enquanto a variável i for menor que 10. A cada iteração, a variável i é incrementada em 1. Se a variável i for igual a 5, a instrução break é executada, interrompendo o loop prematuramente. Como resultado, o programa exibirá apenas os números de 1 a 4 e, em seguida, exibirá a mensagem “fim”.

- Exemplo de uso de continue:

```
1  i = 0
2  while i < 10:
3      i += 1
4      if i == 5:
5          continue
6      print(i)
7  print("fim")
```

Nesse exemplo, o loop while vai executar enquanto a variável i for menor que 10. A cada iteração, a variável i é incrementada em 1. Se a variável i for igual a 5, a instrução continue é executada, pulando a iteração atual e indo diretamente para a próxima iteração. Como resultado, o programa exibirá os números de 1 a 4 e de 6 a 10; em seguida, exibirá a mensagem "fim".

3. Exemplos Práticos

Agora que já conhecemos a estrutura de repetição while e seu funcionamento na linguagem Python, vamos analisar exemplos práticos, trabalhando alguns cenários semelhantes aos utilizados para a estrutura for.

- **Cenário 1**: um programador precisa desenvolver um código em Python capaz de contabilizar a quantidade de votos de dois candidatos a representante de turma de uma instituição de ensino. A turma é composta de uma quantidade indeterminada de alunos, que devem escolher entre o candidato 1 (identificado pelo valor inteiro 1) ou o candidato 2 (identificado pelo valor inteiro 2). Caso o aluno insira um valor diferente dos valores indicados, para identificar os candidatos 1 e 2, o voto será considerado nulo. O programa deve apresentar: a quantidade de votos de cada candidato, a quantidade de votos nulos e a quantidade total de votos.

```python
candidato1 = 0
candidato2 = 0
nulos = 0
votos = 0

while True:
    voto = int(input("Digite o número do candidato (1 ou 2): "))
    if voto == 1:
        candidato1 += 1
    elif voto == 2:
        candidato2 += 1
    else:
        nulos += 1

    votos += 1
    continuar = input("Deseja continuar? (s/n) ")
    if continuar == "n":
        break

print(f"Total de votos: {votos}")
print(f"Candidato 1: {candidato1} votos")
print(f"Candidato 2: {candidato2} votos")
print(f"Votos nulos: {nulos}")
```

Nesse exemplo, inicializamos quatro variáveis: candidato1, candidato2, nulos e votos, todas com o valor 0.

Em seguida, utilizamos um loop while True para continuar recebendo votos enquanto o usuário desejar. Dentro do loop, pedimos ao usuário para digitar o número do candidato (1 ou 2) e verificamos se o voto é válido (se é igual a 1 ou 2). Se for igual a 1, incrementamos a variável candidato1 em 1. Se for igual a 2, incrementamos a variável candidato2 em 1. Caso contrário, incrementamos a variável nulos em 1. Também incrementamos a variável votos em 1 a cada voto recebido.

Por fim, perguntamos ao usuário se ele deseja continuar a votação. Se a resposta for "n", saímos do loop while.

Após o loop, mostramos o resultado da contagem dos votos, utilizando o comando print.

- **Cenário 2**: um programador precisa desenvolver um programa que calcula a média de um conjunto de 10 números inteiros.

```
1   soma = 0
2   contador = 0
3
4   while contador < 10:
5       numero = int(input("Digite um número inteiro: "))
6       soma += numero
7       contador += 1
8
9   media = soma / 10
10  print("A média dos números é:", media)
```

Nesse exemplo, temos duas variáveis importantes: soma e contador. A variável soma é responsável por acumular a soma dos números digitados pelo usuário, enquanto a variável contador é responsável por contar quantos números foram digitados até o momento.

Dentro do laço while, o programa solicita que o usuário digite um número inteiro, que é armazenado na variável numero. Em seguida, o programa soma o valor de numero à variável soma e incrementa o valor de contador em 1.

Após o laço while, o programa calcula a média dos números, dividindo a variável soma por 10 (que é a quantidade de números que deve ser digitada).

Por fim, o programa imprime na tela a média calculada.

Considerações Finais da Aula

As estruturas de repetição são importantes para executar uma ou mais instruções que precisam ser executadas várias vezes. A linguagem Python oferece estruturas que podem ser escolhidas de acordo com o tipo de problema que está sendo tratado.

É possível estabelecer a quantidade de vezes que uma repetição será executada de acordo com um número ou uma condição. Também podemos fazer o uso de variáveis para determinar a quantidade de uma ocorrência, para acumular valores dentro de uma repetição e para usar comando para controlar o fluxo de uma repetição.

É importante salientar a importância de estar atento às regras de indentação e sintaxe e à lógica aplicada para resolver o problema, com o apoio da estrutura de repetição.

Materiais Complementares

Livro:

Curso Intensivo de Python: Uma introdução prática e baseada em projetos à programação
2016, Eric Matthes. Novatec Editora.
Esse livro apresenta mais informações sobre os conceitos básicos trabalhados nesta aula, relativos às características da linguagem Python, à sua sintaxe e aos 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 utilizando estruturas de repetição.
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.