Fundamentos de Programação

Estruturas de REPETIÇÃO

O Comando *For*

Aula 11

Post It da Aula Passada

Variações do *If*

- Estruturas *if-else*
  - Teste condicional falso
- Estruturas *if-elif-else*
  - Múltiplos testes condicionais
  - No máximo um é aceito

Condicionais Aninhados

- “Múltiplas perguntas”
- Uma estrutura condicional dentro de outra
- Indentação

A Intuição de Repetição

Tipicamente, existem algumas tarefas que devemos realizar que se repetem temporalmente:

- Tomar um copo d'água (**cinco vezes por dia**)
- Almoçar (**uma vez por dia**)
- Aula de fundamentos de programação (**uma vez por semana**)

De forma bastante geral, podemos compreender essas tarefas repetidas a cada intervalo de tempo como as formadoras da nossa **rotina**

A Intuição de Repetição

Quando estamos falando sobre tarefas repetitivas, algumas delas são bastante simples, sendo feitas por um **certo período ou número de vezes**:

- Misture realizando movimentos circulares por 5 minutos
- Corte a carne em 10 pedaços de mesmo tamanho
- Faça 10 repetições do exercício

Esse tipo de tarefa tem um momento de parada bem definido e, em geral, a quantidade de processos e o jeito de executar os mesmos são bem determinados

Os Comandos de Repetição

É a partir desses processos bem determinados que começaremos a estudar as estruturas de repetição

Para linguagens de programação em geral, temos dois grandes expoentes para criar estruturas de repetição, são eles os comandos:

for
while

Tanto o comando *for*, quanto o comando *while* são providos em Python

Os Comandos de Repetição

Se considerarmos uma ordem relacionada ao quão genérico um comando de repetição é, primeiramente precisaríamos estudar o ***while* (mais genérico)**, para só então estudarmos o ***for* (mais específico)**

Porém, considerando que estamos primeiramente tratando de problemas resolvidos em um número pré-determinado de passos, se torna mais conveniente estudarmos primeiro o *for*, para depois estudarmos o *while*

O Comando *For*

O comando *for* nos permite definir um conjunto de operações que será repetido por um número definido de vezes. Vamos começar com um problema super simples:

Exiba na tela **cinco vezes** a *string* **“Fundamentos de Programação”**

Analisando nosso problema, **(i)** a palavra “exiba” nos remete à função *print*, **(ii)** cinco vezes nos remete a uma quantidade constante de repetições dessa exibição, e **(iii)** “Fundamentos de Programação” indica o que deve ser exibido

O Comando *For*

A **solução mais direta e ingênua** para esse problema seria:

```
print(“Fundamentos de Programação”)
print(“Fundamentos de Programação”)
print(“Fundamentos de Programação”)
print(“Fundamentos de Programação”)
print(“Fundamentos de Programação”)
```

Essa solução, de fato, resolve o problema, porém é **demasiadamente verbosa e muito pouco genérica**

O Comando *For*

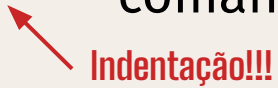
Para solucionar os problemas elencados, vamos remodelar a nossa solução para utilizar a estrutura de repetição *for*. O "jeitão" dessa estrutura em Python é como segue:

for **variável de controle de iteração** **in** **espaço de iteração**:
    comandos executados a cada iteração

Lembrando que uma **iteração nada mais é do que um ato de repetição de algo**, no nosso caso, de comandos e operações

O Comando *For*

for **variável de controle de iteração** **in** **espaço de iteração**:
    comandos executados a cada iteração



****variável de controle de iteração****: uma variável qualquer que irá armazenar a iteração atual considerando o espaço de iteração
****espaço de iteração****: consiste no conjunto de valores que a variável irá assumir a cada iteração da estrutura definida
****comandos executados a cada iteração****: bloco que comandos que será executado para cada iteração

O Comando *For*

for **variável de controle de iteração** **in** **espaço de iteração**:
    comandos executados a cada iteração

A nossa variável de iteração pode ser uma já existente (terá seu valor substituído) ou uma declarada no ato da definição da estrutura de repetição (muito comum se utilizar i — de índice ou *index*)

for i **in** **espaço de iteração**:
    comandos executados a cada iteração

O Comando *For*

for i **in** **espaço de iteração****:
    comandos executados a cada iteração

O espaço de iteração pode ser definido por meio de uma lista (veremos no futuro) ou qualquer outro elemento iterador. Por enquanto, vamos focar na função *range*

A **função *range* define um intervalo numérico**, ela recebe de um a três argumentos, mas, por enquanto, vamos utilizar a mesma com apenas um argumento: o tamanho do intervalo numérico inteiro desejado com passo um

O Comando *For*

range(tamanho do intervalo**)**

Considerando nosso problema inicial (exiba na tela cinco vezes a *string* “Fundamentos de Programação”), sabemos que nosso intervalo é 5

range(5)

Como já vimos em outras ocasiões, intervalos numéricos em Python começam no zero, portanto range(5) gera um intervalo de 0 a 4 -- [0, n) ou [0, n-1]

O Comando *For*

Sendo assim, até então temos o seguinte:

for i **in** range(5)**:**
    comandos executados a cada iteração

O que nos falta é **definir os comandos que serão executados a cada iteração de i variando no intervalo [0,4]**. Segundo o nosso enunciado, devemos exibir “Fundamentos de Programação”:

print(“Fundamentos de Programação”)

O Comando *For*

Com isso temos a solução do nosso problema inicial: exiba na tela cinco vezes a *string* “Fundamentos de Programação”

```
for i in range(5):
    print(“Fundamentos de Programação”)
```

Mas, como essa estrutura de repetição é de fato executada?

...

O Comando *For*

```
for i in range(5):
    print(“Fundamentos de Programação”)
```

“Para a variável i assumindo os valores em [0,1,2,3,4]”

- **i = 0**
  - print(“Fundamentos de Programação”)
- **i = 1**
  - print(“Fundamentos de Programação”)

O Comando *For*

- **i = 2**
  - print(“Fundamentos de Programação”)
- **i = 3**
  - print(“Fundamentos de Programação”)
- **i = 4**
  - print(“Fundamentos de Programação”)

Todos os valores possíveis para i foram atribuídos, logo não há mais valores previstos para a repetição, então pare a mesma!

O Comando *For*

E se, ao invés de exibirmos a *string* “Fundamentos de Programação”, nós exibirmos a nossa variável de iteração i?

```
for i in range(5):
    print(i)
```

Qual seria a saída do nosso algoritmo?

O Comando *For*

Uma possibilidade interessante da estrutura de repetição *for* é a **construção do nosso espaço de iteração a partir de parâmetros diferentes**, por exemplo:

```
var_str = input(“Digite o seu nome: ”)
for i in range(len(var_str)):
    print(“A letra”, i, “do seu nome é:”, var_str[i])
```

O que acontece se o usuário digitar “José”?

O Comando *For*

Além disso, outra possibilidade, agora mais relacionada à linguagem de programação Python, é a de **iteração sob uma lista não numérica**

Por exemplo, podemos considerar uma ***string* como uma lista de caracteres (cadeia de caracteres)**, então...

```
var_str = input(“Digite o seu nome: ”)
for letra in var_str:
    print(letra)
```

O Comando *For*

E, claro, vale a pena lembrar que nosso **bloco de comandos e operações pode conter várias linhas de código**, desde que se observe a correta indentação

```
for i in range(5):
    res_exp = 2 ** i
    print("O resultado de 2 elevado ao índice", i, "é:", res_exp)
```

Exercício #11

Faça um programa que receba uma *string* do usuário pela **entrada padrão**. Declare uma **variável** que servirá como um contador. **Para cada carácter da *string*** recebida, verifique se o mesmo é uma **vogal, se sim, adicione uma unidade ao contador**. Ao final, **exiba a quantidade de vogais** existente na *string* fornecida pelo usuário

Fundamentos de Programação

Aula 11

Obrigado e ATÉ A PRÓXIMA!