ACH5531

Introdução à Computação

Estruturas de repetição

Prof. Dr. Grzegorz Kowal

grzegorz.kowal@usp.br

https://sites.google.com/usp.br/ach5531

1° sem 2019 – sexta-feira, 14h00-15h45 – CB, Bloco 3, 2° andar, Lab. 6

Estrutura de repetição

Em algumas situações, é comum que uma mesma instrução (ou conjunto delas) precise ser executada várias vezes seguidas. Nesses casos, normalmente utilizamos um *loop* (ou laço de repetição), que permite executar um bloco de código repetidas vezes, enquanto uma dada condição é atendida.

Em Python, os loops são codificados por meio dos comandos **for** e **while**. O primeiro nos permite percorrer os itens de uma coleção e, para cada um deles, executar um bloco de código. Já o **while**, executa um conjunto de instruções várias vezes enquanto uma condição é atendida.

Estrutura de repetição **for**

A sintaxe da estrutura **for** é seguinte:

```
for <item> in <lista de itens>:
  <bloco de comandos 1>
else:
  <bloco de comandos 2>
```

Exemplo 1:

```
nomes = ['Pedro', 'Joao', 'Leticia' ]
for nome in nomes:
  print(nome)
```

A variável definida na primeira linha é uma lista inicializada com uma sequência de valores do tipo *string*. A instrução **for** percorre todos esses elementos, um por vez e, em cada caso, atribui o valor do item à variável *nome*, que é impressa em seguida. O resultado, então, é a impressão de todos os nomes contidos na lista.

Estrutura de repetição **for**

A estrutura de repetição **for** executa um ciclo para cada elemento do objeto que está sendo iterado. Nas vezes em que precisamos que determinada variável seja incrementado ou decrementada a cada ciclo, a forma mais simples, é gerando uma lista com a função **range()**.

```
for <variavel> in range(start, stop, step):
  <bloco de comandos>
```

Os argumentos *start* e *step* são opcionais. O primeiro indica o valor com qual a variável é iniciada. O argumento *stop* indica antes de qual valor parar a iteração. O último argumento *step* define o incremento de elemento, ou seja, o passo, intervalo numérico entre cada elemento.

Exemplo 2:

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

No código a seguir, temos um exemplo onde a variável *item* recebera um número, iniciando em zero, finalizando em 9 e, a cada ciclo, o número seguinte da sequência é atribuído à variável *item*.

Exemplo 3

Escreva um programa que imprime 10 primeiros números inteiros, iniciando de zero, e quadrados deles.

```
# Imprime o objetivo do programa.
print("Programa imprime tabela de números inteiros e quadrados deles.\n")

# Imprime o cabeçalho e tabela.
print("n\tn²\n")
for n in range(10):
  print(n, "\t", n**2)
```

O resultado de execução do programa:

```
Programa imprime tabela de números inteiros e quadrados deles.

n       n²

0        0
1        1
2        4
3        9
4        16
5        25
6        36
7        49
8        64
9        81
```

Estrutura de repetição **while**

O comando **while**, por sua vez, faz com que um conjunto de instruções seja executado enquanto uma condição for atendida. Quando o resultado passa a ser falso, a execução é interrompida, saindo do *loop*, e passa para o próximo bloco. A sintaxe da estrutura **while** é seguinte:

```
while <expressão lógica>:
  <bloco de comandos>
```

Exemplo 4:

```
contador = 0
while contador < 5:
  print(contador)
  contador = contador + 1
```

No código a seguir, vemos um exemplo de uso do laço **while**, onde definimos a variável contador, iniciando com 0, e enquanto seu valor for menor que 5, executamos as instruções das linhas 3 e 4.

Observe que na linha 4 incrementamos a variável contador, de forma que em algum momento seu valor ultrapasse 5. Quando isso for verificado na linha 2, o laço será interrompido. Caso a condição de parada nunca seja atingida, o loop será executado infinitamente, gerando problemas no programa.

Estrutura de repetição **while** infinita

O comando **while** pode ter versão infinita, onde a expressão lógica é substituída por True. Neste caso, para interromper a repetição precisamos usar estrutura **if** com comando **break**. A sintaxe da versão infinita de **while** é seguinte:

```
while True:
  <bloco de comandos>
  if <expressão lógica>:
    break
```

Exemplo 5:

```
contador = 0
while True:
  print(contador)
  contador = contador + 1
  if contador > 5:
    break
```

Neste exemplo a repetição **while** nunca ia terminar pois o argumento dela é sempre verdadeiro. Utilizando if e podemos substituir a verificação da condição de repetição dentro do bloco de comandos, e interromper a repetição com comando **break**.

Exemplo 6

Escreva um programa que gera um número inteiro entre 0 e 10 e solicita o usuário adivinhar o valor do número, até achar o valor correto, dando somente dicas que o número é menor ou maior. Imprime o número de tentativas no final. Use seguente bloco de comandos para gerar o número aleatório entre 0 e 10:

```
from random import randomint
n = randint(0, 10)
```

```
from random import randint
# gera um número aleatório entre 0 e 10
n = randint(0, 10)
# zera número de tentativas
i = 0
# repete até o usuário achar o número
while True:
  m = int(input("Digite um número inteiro entre 0 e 10: "))
  i = i + 1
  if m < n:
    print("O número é maior do que ", m)
  elif m > n:
    print("O número é menor do que ", m)
  else:
    print("Bingo! O número é ", n)
    print("Número de tentativas: ", i)
    break
```

Exercícios

8. Escreva um programa que calcula média de um arbitrário número de valores entre 0 e 10, ou seja, o programa repete a solicitação de valores até o usuário fornecer espaço. Após isso, o programa imprime o valor médio dos números fornecidos. Use a estrutura **while**.

9. Escreva um programa usando estrutura de repetição **for** que usando uma palavra ou texto fornecido pelo usuário, imprime este texto na ordem invertida. Para obter o número de caráteres do texto fornecido use a função **len**(texto). Os caráteres individuais podem ser obtidos usando a sintaxe **texto[6]**, onde 6 indica o setimo caráter da variável **texto**. Para juntar duas cadeias de texto use simbolo '+' como mostrado no exemplo:

```
nome = 'Paulo'
subrenome = 'Coelho'
nome_completo = nome + ' ' + sobrenome
```