N_ALG PROG_A6 - Texto de apoio

Site: EAD Mackenzie
Tema: ALGORITMOS E PROGRAMAÇÃO I {TURMA 01B} 2021/2
Livro: N_ALG PROG_A6 - Texto de apoio

Impresso por: ANDRE SOUZA OCLECIANO .
Data: terça, 28 set 2021, 20:11

Índice

1. ESTRUTURA DE REPETIÇÃO – FOR

Vimos anteriormente, a estrutura de repetição **while**, uma estrutura de controle iterativo que executa repetidamente um bloco de instruções baseado em uma expressão booleana ou condição.

Nesta aula, estudaremos a estrutura de repetição **for**, uma estrutura de controle **iterativo** que repete a instrução ou o bloco de instruções para cada valor de uma sequência de valores.

Usamos o **loop definido** ou **loop for** quando sabemos previamente o número de vezes que um bloco de instruções será executado.

SINTAXE:

for <variavel> in range(inicial, final, passo):

    <bloco de instruções>

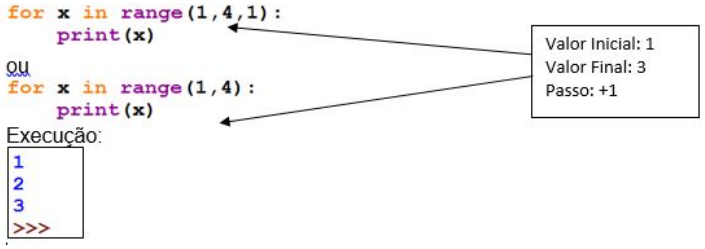
A função **range** retorna uma lista de valores consecutivos, começando com o valor **inicial**, menores que o valor final, sendo incrementada em um determinado **passo**.

Detalhes importantes:

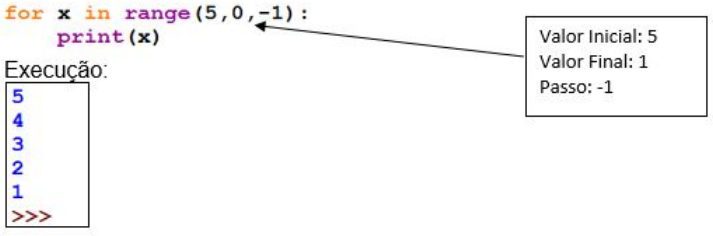
O valor inicial pode ser omitido e, neste caso, assume o valor 0.

O passo também pode ser omitido, assumindo, por padrão, o valor 1.

O valor final é obrigatório



Exemplo 2: um programa que exibe os números inteiros de 5 até 1.



Abaixo, há algumas aplicações dessas variações que podemos usar no intervalo de valor do comando for. Veja os valores que serão exibidos para cada um dos seguintes itens:

a)
```
for k in range(1,11):
    print(k)
```

```
1
2
3
4
5
6
7
8
9
10
>>>
```

Observe que, no Código, o valor de k terá valor inicial igual a 1 e valor final igual a 11, uma vez que o laço for irá se repetir até que o valor de k seja <11. Ou seja, quando o valor de k for 10 o comando dentro do laço será executado e quando o k atingir o valor 11, o laço será finalizado.

b)
```
for k in range(10):
    print(k)
```

```
0
1
2
3
4
5
6
7
8
9
>>>
```

Neste caso, o valor de k inicia por zero uma vez que esta informação foi omitida dentro dos parênteses. Haverá repetição até que k seja menor que 10, por isso a saída de dados mostra os valores de 0 a 9. Quando k tiver o valor 10, o laço será terminado. Observe também que, devido à omissão do passo dentro dos parênteses, o programa soma 1 ao valor de k a cada iteração.

c)
```
for k in range(2,12,2):
    print(k)
```

```
2
4
6
8
10
>>>
```

Neste caso, o valor de k começa por 2 e vai, a cada iteração, somar 2, devido ao valor colocado no terceiro parâmetro dentro dos parênteses. Observe que, se k for igual a 10, o laço será executado, porém, na próxima iteração, k recebe o valor 12 e, desta forma, o laço é interrompido.

d)
```
for k in range(20,0,-2):
    print(k)
```
```
20
18
16
14
12
10
8
6
4
2
>>>
```

Neste último caso, o valor que k começa com 20 e irá, a cada iteração, decrementar a variável k em 2, ou seja, realizará a operação k=k-2. O laço será interrompido quando a condição k>0 for falsa, pois o valor 0 é o valor final do laço.

2. INTERRUPÇÃO DA REPETIÇÃO

É possível utilizar os comandos continue e break com a seguinte intenção:

Use o comando **continue** para iniciar imediatamente a próxima volta do loop, ignorando os comandos que possam existir abaixo do **continue**, dentro do laço. A instrução continue interrompe a execução do ciclo sem suspender a execução do laço de repetição.

Exemplo 3: Faça um programa em Python que mostre os valores ímpares entre 1 e 11.

```
File  Edit  Format  Run  Options  Window  Help
x = 0
while(x <= 10):
    x+=1
    if(x % 2==0):
        continue
    print(x)
```

Observe que os valores gerados na variável x vão de 0 até 10, com passo 1, porém apenas os valores ímpares serão impressos.

Execução:

```
1
3
5
7
9
11
>>>
```

O código abaixo terá o mesmo comportamento, a única diferença é que ele foi escrito utilizando a estrutura de repetição *for*.

```
File  Edit  Format  Run  Options  Window  Help
for x in range(12):
    if(x % 2==0):
        continue
    print(x)
```

Porém, usando o comando break, encerrará imediatamente o loop. Ao encontrar o comando break, será executado o próximo comando após o término do laço.

A instrução break finaliza a iteração e as instruções após o laço (fora do laço) serão executadas normalmente. O objetivo dessa instrução é forçar a interrupção da iteração.

Exemplo 4:

```
File  Edit  Format  Run  Options  Window  Help
x = 0
while(x<10):
    x+=1
    if(x==5):
        print("Interrompendo a execução da repetição.")
        break
    print(x)
print('Instrução após a iteração')
```

Observe que, nesse código, a repetição testa, a cada iteração, se x é menor do que 10. Sendo verdadeira esta expressão, os comandos dentro do laço serão executados, porém, se x for igual a 5, será dada a mensagem e o laço interrompido.

Execução:

```
1
2
3
4
Interrompendo a execução da repetição.
Instrução após a iteração
>>>
```

3. REPETIÇÃO USANDO ELSE

Em Python, podemos colocar um else no laço de repetição que será executado no final da iteração. O propósito disso é executar alguma instrução ou bloco de código ao final do loop.

Veja os exemplos abaixo.

Exemplo 5:

```
File  Edit  Format  Run  Options  Window  Help
count = 0
while count <= 5:
    print(count)
    count += 1
else:
    print('após iteração')
```

Execução:

```
0
1
2
3
4
5
após iteração
>>>
```

loop *while*, a expressão é testada enquanto for verdadeira. A partir do momento em que ela se torna falsa, o código da cláusula *else* será executado, se estiver presente.

Veja, usando o *else* no *for*:

Exemplo 6:

```
File  Edit  Format  Run  Options  Window  Help
for i in range(5):
    print(i)
else:
    print('após iteração')
```

Execução:

```
0
1
2
3
4
após iteração
>>>
```

Se dentro da repetição for executado um **break**, o loop será encerrado sem executar o conjunto da cláusula *else*. Veja um exemplo.

Exemplo 7:

```
File  Edit  Format  Run  Options  Window  Help
x = 0
while x < 10:
    print(x)
    x += 1
    if x == 6:
        print("x é igual a 6")
        break
else:
    print("fim while")
```

Execução:

```
0
1
2
3
4
5
x é igual a 6
>>>
```

Observe que o contador repetirá até que x fosse menor que 10, ou seja, até 9, porém como foi colocada uma condição que, se verdadeira, seria executado o comando break, quando x ficou com valor igual a 6, o laço foi interrompido (com a instrução break). Desta forma, veja que a instrução dentro do else não foi executada.

Mesmo que este código fosse implementado usando for, como no código abaixo, o comportamento seria o mesmo, ou seja, a instrução dentro do else não seria executada.

```
File  Edit  Format  Run  Options  Window  Help
for x in range(0,10,1):
    if x == 6:
        print("x é igual a 6")
        break
    print(x)
else:
    print("fim do for")
```

Execução:

```
0
1
2
3
4
5
x é igual a 6
>>>
```

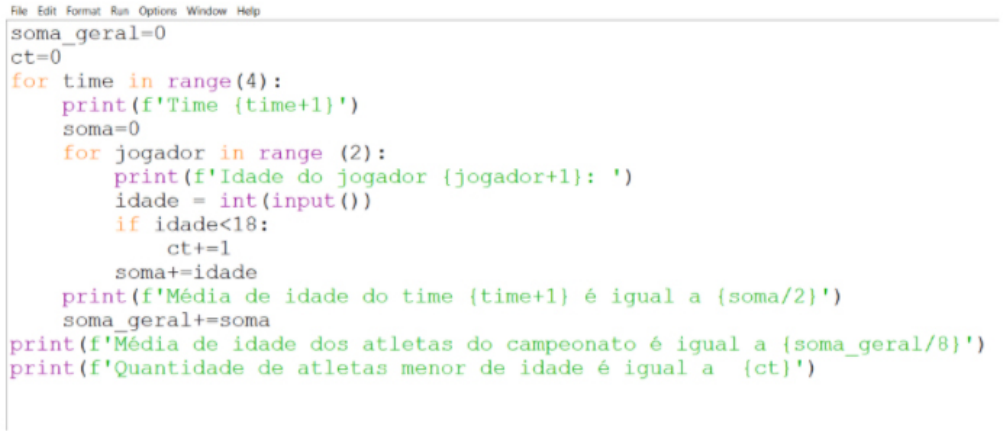
4. REPETIÇÕES ANINHADAS

Podemos combinar vários laços para obter resultados mais complexos, como a repetição com incrementos de duas ou mais variáveis.

Exemplo 8: Faça um programa em Python para resolver o seguinte problema:

Em um campeonato de tênis de mesa, existem quatro times e cada um possui dois jogadores. Faça um programa que receba a idade de cada um dos jogadores, calcule e mostre a quantidade de jogadores menores de idade do campeonato, a média das idades de cada time e a média de idade do campeonato.

Existem algumas formas de resolver este problema, veja uma possível solução:

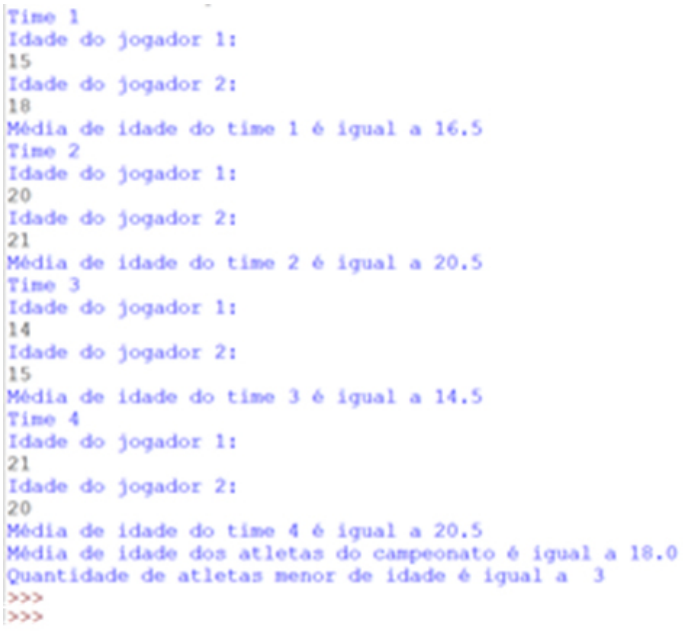


Observe que a variável time conta a quantidade de times, e a variável jogador conta os jogadores por time. Quando a variável time recebe o valor 0, a variável soma será zerada e o contador jogador repetirá a contagem de 0 e 1, ou seja, será feito o processamento do primeiro time.

Ao finalizar o for do jogador, a variável time valerá 1 e, mais uma vez, a variável soma será zerada e o contador jogador repetirá a contagem de 0 e 1, ou seja, será feito o processamento do segundo time. E, assim, até completar quatro times.

Ao sair do laço do time, será mostrada a média das idades de todos os jogadores e a quantidade de atletas com idade menor que 18.

Execução:



5. REFERÊNCIAS

DIERBACH, C. *Introduction to Computer Science Using Python*: A Computational Problem Solving Focus. New York: Wiley, 2012.

MENEZES, N. N. C. *Introdução à Programação com Python*: algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2014.