N_ALG PROG_A5 - Texto de apoio

Site: EAD Mackenzie
Tema: ALGORITMOS E PROGRAMAÇÃO I {TURMA 01B} 2021/2
Livro: N_ALG PROG_A5 - Texto de apoio

Impresso por: ANDRE SOUZA OCLECIANO .
Data: terça, 28 set 2021, 20:10

Índice

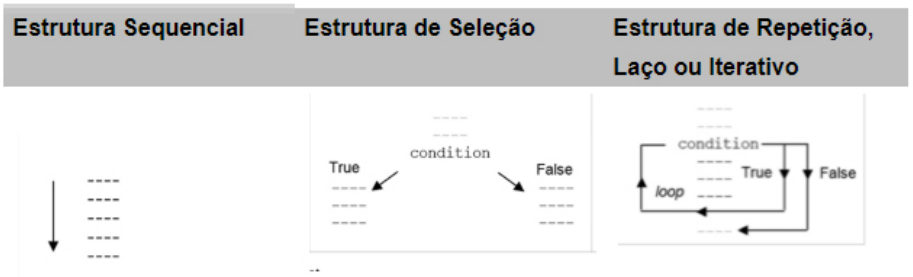
1. ESTRUTURA DE REPETIÇÃO – WHILE

2. REFERÊNCIAS

1. ESTRUTURA DE REPETIÇÃO – WHILE

Já vimos duas formas de controlar o fluxo de um programa: estrutura de controle sequencial e estrutura de controle de seleção ou condicional.

Figura 1 – Formas de controle do fluxo de um programa



Fonte: DIERBACH (2012)

Iniciaremos as **estruturas de repetição**, também conhecidas por **loop (laço)** e **iterativo**, que, junto com as demais estruturas vistas até agora, propiciarão a solução da uma gama muito maior de problemas.

As estruturas de repetição permitem que uma ação seja executada mais de uma vez sem que tenhamos que executar novamente o programa.

Uma estrutura de **controle iterativo, estrutura de repetição ou simplesmente laço** é uma estrutura de controle de fluxo formada por um conjunto de instruções que são executadas um **número determinado de vezes**, ou até que determinada **condição** se torne verdadeira ou falsa.

A linguagem Python classifica as estruturas de controle iterativo em:

- **loop definido** ou **loop for** quando se sabe previamente o número de vezes que um bloco de instruções será executado.
- **Loop indefinido** quando o bloco de instruções será executado até que uma condição se torne verdadeira ou falsa.

Estudaremos, inicialmente, a estrutura de loop indefinido com a declaração while.

Loop Indefinido – while

Uma declaração while é uma estrutura de controle iterativo que executa repetidamente um bloco de instruções baseado em uma expressão booleana ou condição. Ou ainda, o loop indefinido ou condicional mantém a iteração até que certas condições sejam atingidas.

Sintaxe:

while <condição>:

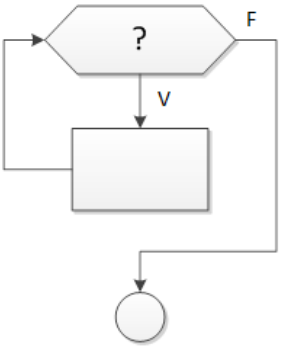
<bloco de instruções ou bloco verdade>

Esta estrutura permite executar diversas vezes um bloco de instruções, sempre verificando **antes** se a <**condição**> é verdade.

A condição é uma expressão booleana, semelhante às expressões que usamos nas estruturas condicionais com if e elif.

A <**condição**> é primeiro avaliada. Enquanto a <**condição**> for verdadeira, o <**bloco verdade**> é executado. Quando a <**condição**> for falsa, a iteração termina e a execução continua com a instrução após o laço do while.

O fluxograma ilustra esse loop:



Exemplo: Considere o programa que exibe os números inteiros de 1 a 3.

```
x=1 #inicialização
while x<=3:   #condição
    print(x)  #instrução
    x=x+1     #incremento
```

Teste de Execução

x	x <= 3	print(x)
1	True	1
2	True	2
3	True	3
4	False (pára)	

A variável x é inicializada com o valor igual a 1, e a condição é avaliada pela primeira vez.

Como a condição x<=3 é verdadeira, a instrução dentro do laço é executada, exibindo o valor de x e atualizando-o para 2.

O controle de execução retorna para o topo do laço na avaliação da condição novamente. Como 2 é menor ou igual a 3, o bloco dentro do laço é executado, exibindo o valor de x e atualizando-o para 3.

Retoma-se ao topo da estrutura com o teste da condição, como é verdade, o bloco de instruções é executado novamente, pela terceira vez. Nesse ponto, a variável x assumiu o valor 4 e a condição x<=3 resulta em falso (False), terminando, assim, a repetição do bloco.

Vejamos alguns exemplos:

Exemplo 1 – Escreva um programa em Python para exibir os números de 50 a 100.

File Edit Format Run Options Window Help

```
ct = 50
while ct<=100:
    print(ct, end = ' ')
    ct+=1
```

Saída:

```
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
>>> |
```

Exemplo 2 – Elabore um programa em Python para escrever na tela a contagem regressiva do lançamento de um foguete. O programa deve exibir 10, 9, 8, ..., 1,0 e Fogo!

```
File  Edit  Format  Run  Options  Window  Help
'''O programa deve exibir 10, 9, 8, ..., 1,0 e Fogo!'''
ct = 10
while ct>0:
    print(ct, end = ', ')
    ct-=1
print('0 e Fogo!')
```

Saída:

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 e Fogo!
>>>
```

Variável Contador

A variável contador é utilizada para controlar a contagem do número de execuções do bloco de instruções.

Recebe um valor inicial (geralmente, 0 ou 1) e é incrementada em algum ponto do algoritmo de um valor constante (geralmente, 1); poderá, no entanto, contar com incremento ou decremento de qualquer valor, dependendo do problema a ser resolvido.

Incrementar uma variável é o mesmo que somar um valor constante a ela. O valor da variável também pode ser decrementado, que é o mesmo que subtrair um valor constante a essa variável.

Exemplo 3 – Escreva um programa em Python para exibir todos os números pares de 0 até o número informado pelo usuário.

Faça também uma versão que mostre os pares do número digitado até 0.

Versão que apresenta os pares de 0 até o número digitado:

```
File  Edit  Format  Run  Options  Window  Help
n = int (input('Digite um número:'))
ct =0
while ct<=n:
    print(ct, end = ' ')
    ct+=2
```

Saída:

```
Digite um número:10
0 2 4 6 8 10
>>>
```

Versão que apresenta os pares do número digitado até 0:

File Edit Format Run Options Window Help

```
n = int (input('Digite um número:'))
ct =n
while ct>=0:
    print(ct, end = ' ')
    ct-=2
```

Saída:

```
Digite um número:10
10 8 6 4 2 0
>>> |
```

Variável Acumulador

A Variável acumulador é utilizada para controlar a contagem; recebe um valor inicial (geralmente, 0) e é incrementada em algum ponto do algoritmo de um valor variável.

Exemplo 4 – Calcular a soma de cinco números digitados pelo usuário.

```
soma=0   #inicialização da variável acumulador
cont = 1 #inicialização da variável contador

while cont <= 5:  #condição
    num = int(input("Digite um número: "))
    soma = soma + num #atualização da variável acumulador
    cont = cont + 1   #atualização da variável contador

print("A soma dos números = %d" %soma)
```

Teste de Execução

soma	cont	cont <= 5	num
0	1	True	3
3	2	True	4
7	3	True	2
9	4	True	5
14	5	True	8
22	6	False (pára)	

A soma dos números = 22

Teste de Execução do http://pythontutor.com/

Print output (drag lower right corner to resize)

```
Digite um número: 3
Digite um número: 4
Digite um número: 2
Digite um número: 5
Digite um número: 8
A soma dos números = 22
```

Frames Objects

Global frame

soma	22
cont	6
num	8

Loop Infinito

O Loop infinito é uma estrutura de controle iterativo que nunca termina (ou eventualmente termina com um erro de sistema).

Existe a possibilidade de criar um laço infinito e, dentro da repetição, definir a condição de parada, com uso da instrução **break**.

Exemplo 5 – Faça um programa em Python que leia uma sequência de números e encerre quando o usuário digitar o número zero. Ao final da entrada de dados, apresente a soma dos números digitados.

Solução definindo a condição no while:

```
File  Edit  Format  Run  Options  Window  Help
soma = 0
print('Digite um conjunto de números (0 encerra): ')
n = int (input())
while n!=0:
    soma+=n
    n = int (input())
print('Soma = ', soma)
```

Entrada do primeiro número

Entrada dos próximos números

Observe, no código acima, que é feita a entrada de dados fora do laço do primeiro número. Isto é necessário para que, na condição do while (n!=0), a variável n tenha um valor a ser testado (primeiro número digitado).

Será feito o teste no *while* e, se o primeiro número digitado for diferente de zero, seu valor será acumulado na variável **soma** e o próximo número de entrada (n) será solicitado.

A condição do while (n!=0) será testada e, enquanto o número digitado for diferente de zero, ele será acumulado na variável soma e o próximo número de entrada será solicitado.

Quando o usuário digitar **0** na variável **n** de entrada, a condição do *while* dará **False** e as instruções fora do laço serão executadas. Nesse caso, será apresentada a mensagem **Soma =**, e o valor armazenado na variável **soma** (acumulador).

Solução com loop infinito:

```
File  Edit  Format  Run  Options  Window  Help
soma = 0
print('Digite um conjunto de números (0 encerra): ')

while True:
    n = int (input())
    if n==0:
        break
    soma+=n
print('Soma = ', soma)
```

O comando break encerra a repetição

Observe, nessa versão, que o *while* **True** provoca um loop infinito. O número será sempre solicitado dentro do laço e será feito o teste da estrutura condicional. Se o número de entrada for igual a **0**, o comando break sairá da repetição. Observe que, ao sair da repetição, o número 0 não será acumulado na variável **soma**.

Nas duas versões, o programa terá o mesmo comportamento de saída:

```
Digite um conjunto de números (0 encerra):
2
3
-1
4
0
Soma =  8
>>>
```

Validação de dados

Consiste em verificar se o valor informado pelo usuário está correto ou não. Anteriormente, usamos a estrutura condicional para checar informações de entrada e, em caso de erro, encerrávamos o programa.

Usando uma estrutura de repetição, podemos repetir a entrada de dados até que o usuário digite um valor correto.

Exemplo 6:

Se o usuário precisa digitar um número no intervalo 1 ≤ n ≤ 50, podemos usar a estrutura de repetição que recebe o número e, enquanto este número estiver fora do intervalo permitido, pede novamente a digitação.

```
num = int(input('Digite um número de 1 a 50: '))
while num < 1 or num > 50:
    num = int(input('Número fora do intervalo! Digite novamente: '))
print('Número aceito!')
```

Execução:

num	num < 1 or num > 50	Tela
		Digite um número de 1 a 50:
0	True or False = True	Digite um número de 1 a 50:
51	False or True = True	Digite um número de 1 a 50:
10	False or False = False	Número aceito!

2. 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.