Aula 4

Lógica de Programação e Algoritmos

Prof. Vinicius Pozzobon Borin

1

Conversa Inicial

2

- **O objetivo desta aula é construir algoritmos com estruturas de repetição**

3

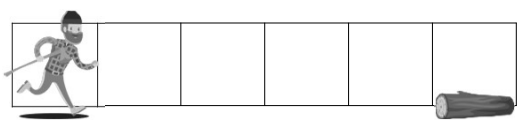
- **Aprenderemos:**
  - ***while* (enquanto)**
  - ***for* (para)**
  - **Laços de repetição aninhados**

4

Estrutura de repetição

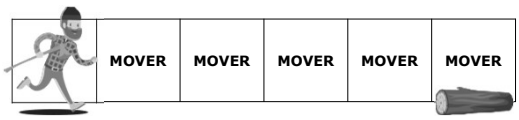
5

Exemplo lúdico



6

Exemplo lúdico



ivector/shutterstock - Incomible/shutterstock

7

Descrição narrativa do movimento

- **Início**
  1. **Mover**
  2. **Mover**
  3. **Mover**
  4. **Mover**
  5. **Mover**
- **Fim**

8

Motivação

- **Escrever *Mover* cinco vezes não parece trabalhoso, certo?**
- **Imagine escrever 100 ou 1000 vezes. Não é prático**
- **Para resolver problemas assim, linguagens de programação utilizam estruturas de repetição**

9

Estrutura de repetição

- **Estrutura no programa em que todas as instruções contidas nela se repetem de maneira indefinida, até que uma condição seja satisfeita**
- **Sinônimos: estrutura iterativa, laço de repetição ou *loop* de repetição**

10

- **Vejamos no Python um exemplo**

11

Estrutura de repetição *while*

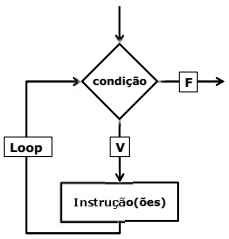
12

while (enquanto)

- **Repete um bloco de instruções enquanto determinada condição se mantiver verdadeira**
- **Caso contrário, ocorre o desvio para a primeira linha de código após este bloco de repetição**

13

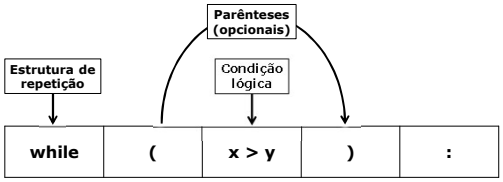
while (enquanto)



Fonte: Borin, 2020.

14

Python



Fonte: Borin, 2020.

15

- **Vamos praticar no Python**
- **Atenção com a indentação**

16

Variável de controle

- **Define a condição de parada com que o laço é executado**
- **Chamamos de *iterador* a variável de controle que realiza a contagem do número de vezes que o laço está sendo executado**
- **Voltamos ao Python**

17

Variáveis contadoras e acumuladoras

- **Contadoras**
  - **Acrescentam valores constantes em uma variável**
- **Acumuladoras**
  - **Acumulam totais, como um somatório**

18

Exercício com contador

- Escreva um algoritmo que imprima na tela somente valores dentro de um intervalo especificado pelo usuário e que sejam número pares

19

Exercício com acumulador

- Escreva um algoritmo que calcule a sua média de notas em determinada disciplina
- Vamos assumir que a média final é dada pela média aritmética de cinco notas digitadas

20

Tópicos importantes com laços em Python

21

Operadores especiais de atribuição

Operador	Exemplo	Equivalente
+=	x += 1	x = x + 1
-=	x -= 1	x = x - 1
*=	x *= 2	x = x * 2
/=	x /= 2	x = x / 2
**=	x **= 2	x = x ** 2
//=	x //= 4	x = x // 4

22

- Vamos praticar operadores especiais de atribuição no Python

23

Validando dados de entrada

- Exemplo
  - Crie um algoritmo que receba um valor do tipo inteiro via teclado
  - No entanto, o programa só deve aceitar, obrigatoriamente, valores inteiros e positivos
  - Qualquer valor negativo, ou igual a zero, deve ser rejeitado pelo programa e um novo valor deve ser solicitado

24

- **Vejamos no Python**

25

Instrução *break*

- **A instrução *break* serve para encerrar um laço de repetição abruptamente, independentemente do estado da variável de controle do laço**

26

Exercício

- **Escreva um algoritmo que fique recebendo frases ou palavras digitadas pelo usuário**
- **Encerre o algoritmo quando a palavra "sair" for digitada**

27

- **Voltamos ao Python**

28

Instrução continue

- **O comando *continue* serve para retornar o laço ao início a qualquer momento, independentemente do estado da variável de controle da condicional do laço**

29

Exercício

- **Escreva um algoritmo que realize um *login* em um sistema**
- **O usuário deve informar seu nome e senha**

30

- Voltamos ao Python

31

Valores *Truthy* e *Falsey*

- Dados não booleanos também podem ser tratados como *True* ou *False* em uma condição, seja esta de uma estrutura condicionada ou de um laço
- *Falsey*/*False*
  - Número zero (*int* ou *float*) e *string* vazia
- *Truthy*/*True*
  - Qualquer outro dado

32

- Voltamos ao Python

33

Estrutura de repetição *for*

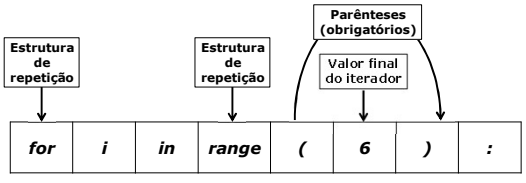
34

***for* (para)**

- Assim como o *while*, essa estrutura repete um bloco de instruções enquanto uma condição se mantiver verdadeira
- No entanto, diferentemente do *while*, o *for* é empregado em situações em que o número de vezes que o laço irá executar é finito e bem definido

35

Python



Fonte: Borin, 2020.

36

- Vamos praticar no Python
- Atenção com a indentação

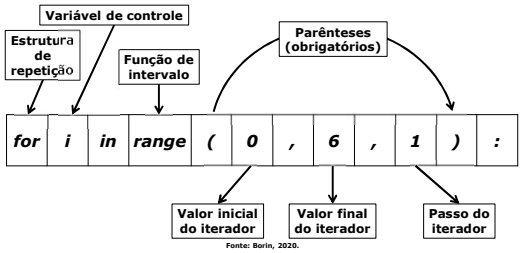
37

for (para)

- Podemos definir o valor inicial do iterador?
  Sim
- Podemos definir o passo?
  Sim

38

Python: com três parâmetros



Fonte: Borin, 2020.

39

- Vamos praticar no Python

40

Varredura de string

- Podemos passar por todos os caracteres de uma *string* usando um laço *for*

41

- Vamos praticar no Python

42

Comparativo *while* e *for*

```
x = 1
while (x < 6):
    print(x)
    x = x + 1
```

```
for i in range(1,6,1):
    print(i)
```

Valor inicial do iterador
Valor final do iterador
Passo do iterador

43

Exercício

- **Escreva um algoritmo que calcule a média dos números pares de 1 até 100 (1 e 100 inclusos). Implemente o laço usando *for***

44

Estruturas de repetição aninhadas

45

- **Podemos inserir laços dentro de outro laço**
- **Não existe limite para quantos laços podemos colocar dentro de outro**
- **Podemos misturar *while* e *for***

46

Exercício

- **Escreva um algoritmo em Python que calcule a tabuada de todos os números de 1 até 10, e, para cada número, vamos calcular a tabuada multiplicando-o pelo intervalo de 1 até 10**

47

- **Vamos resolver no Python diferentes implementações**
  - ***2-while***
  - ***2-for***
  - ***while+for***

48