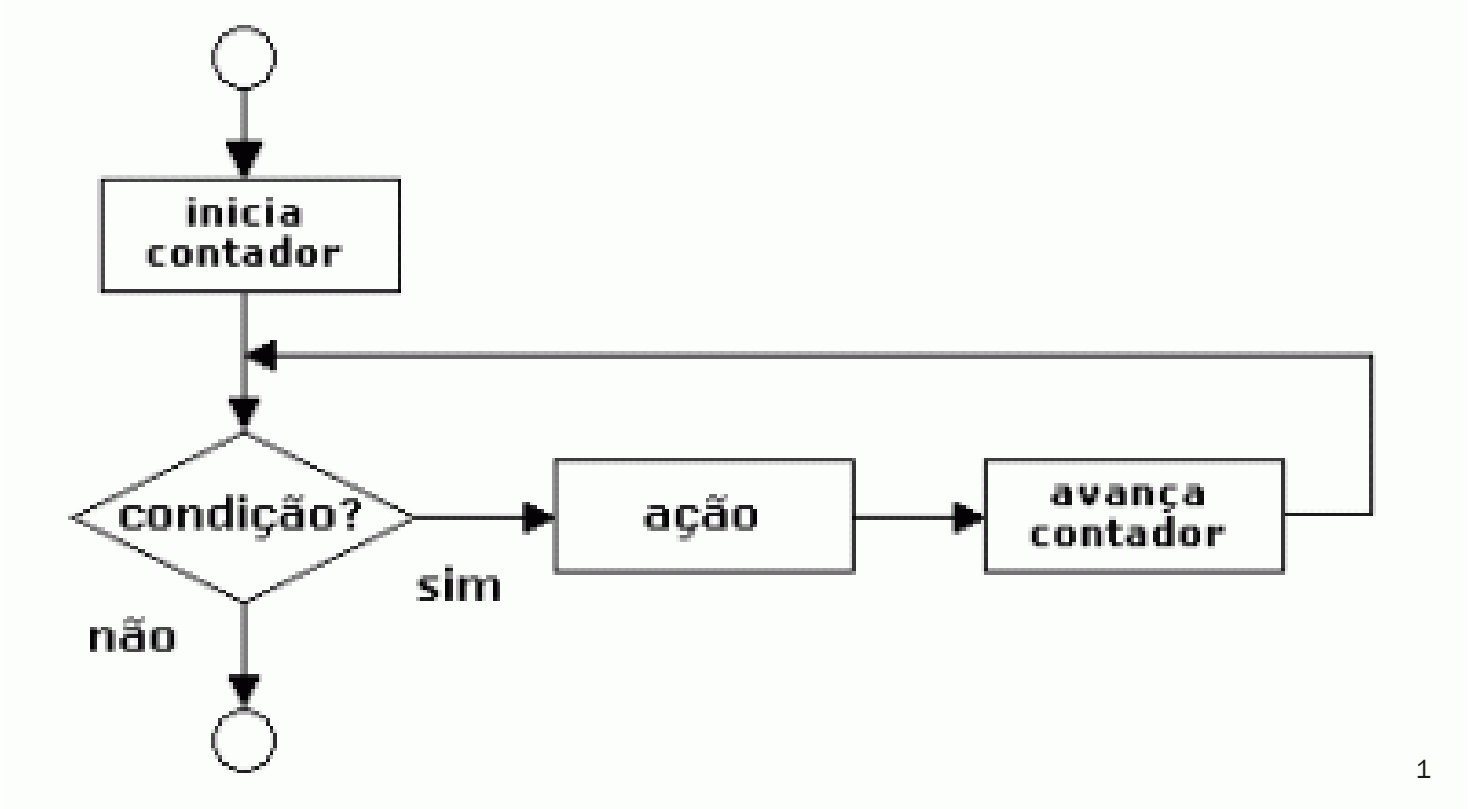
Estrutura de Repetição

Algoritmos e Programação em Python

Prof. Fabio Fernando Kobs, Dr.



Agenda

- Estruturas de Repetição
  - Conceitos iniciais
  - **Estrutura de repetição *while***
    - Contadores
    - Acumuladores
    - Interrompendo a repetição
    - Função *range()*
  - **Estrutura de repetição *for***
- Exercícios

Conceitos Iniciais

Estruturas de Repetição:

- É uma estrutura que permite que uma sequência de comandos seja executada repetidamente até que uma determinada condição de interrupção seja satisfeita.

Conceitos Iniciais

Exemplo 1

Imprimindo de 1 a 7:

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
```

Conceitos Iniciais

Exemplo 2

Imprimindo de 1 a 3 incrementando:

```
x = 1
print(x)
x = x + 1        # ou x += 1
print(x)
x = x + 1
print(x)
```

E se o objetivo fosse imprimir 100 números?

Estrutura de Repetição *while* (enquanto)

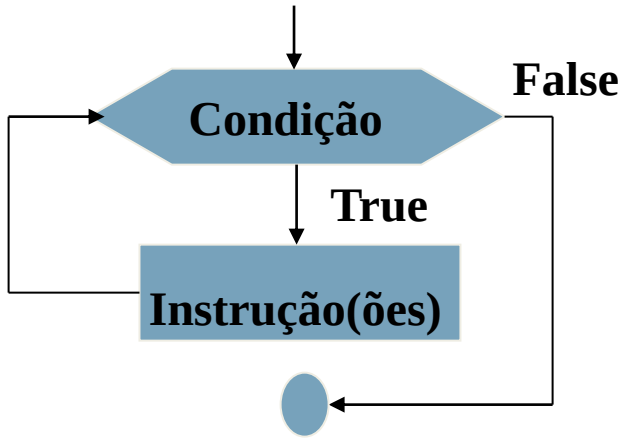
- Repete um bloco enquanto a condição for verdadeira.
- É uma estrutura que testa a condição no início, e só repete se a condição for verdadeira. Sintaxe em Python:

```
while condição:
    bloco
```

- Condição é uma expressão lógica, e bloco representa a(s) linha(s) de programa a repetir enquanto o resultado da condição for verdadeiro.

Estrutura de Repetição *while*

Fluxograma



Uma única condição (expressão lógica) é avaliada.

ENQUANTO o resultado for verdadeiro (**True**), um determinado conjunto de instruções é executado (bloco). E a condição é testada novamente.

Ou seja, repete até o resultado da condição for igual a Falso.

Estrutura de Repetição *while* (enquanto)

Exemplo 3

Imprimindo de 1 a 5 com *while*:

```
x = 1
while x <= 5:
    print(x)
    x += 1
```

ou

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

Exemplo 4

Imprimindo de 100 a 3000 com *while*:

```
x = 100
while x <= 3000:
    print(x)
    x += 1
```

Estrutura de Repetição *while*

Exercícios

1. Faça um programa para exibir os números de 50 a 100.
2. Faça um programa para exibir os números de 500 a 480.
3. Faça um programa para escrever a contagem regressiva do lançamento de um foguete. O programa deve imprimir 10, 9, 8, ..., 1, 0 e Fogo! na tela.

Estrutura de Repetição *while*

Contadores

- São variáveis que atuam **contando** os valores a cada vez que o código é executado.

Exemplo da impressão de 1 até um número digitado pelo usuário:

```
fim = int(input("Digite o último número a imprimir: "))
x = 1
while x <= fim:
    print(x)
    x += 1
```

Estrutura de Repetição *while*

Contadores

Exemplo da impressão de números pares de 2 até um número digitado pelo usuário:

```
fim = int(input("Digite o último número a imprimir: "))
x = 2
while x <= fim:
    if x % 2 == 0:
        print(x)
    x = x + 1  # x += 1
```

ou

```
while x <= fim:
    print(x)
    x = x + 2   # x+=2
```

Estrutura de Repetição *while*

Exercícios

4. Faça um programa para imprimir os números ímpares de 1 até o número digitado pelo usuário.
5. Faça um programa para escrever os 10 primeiros múltiplos de 3 (por exemplo: 3, 6, 9 ...).

Estrutura de Repetição *while*

Contadores

Exemplo para imprimir a tabuada (de 1 a 10) **de adição** de um número digitado pelo usuário.

```
n = int(input("Tabuada de adição de: "))
x = 1
while x <= 10:
    print(f"{n}+{x}={n+x}")
    x += 1
```

Estrutura de Repetição *while*

Exercícios

6. Faça um programa para exibir os resultados de uma tabuada de um número digitado pelo usuário:

   Por exemplo: Tabuada de 2

   2 x 1 = 2

   2 x 2 = 4 ...

7. Faça um programa de forma que o usuário também digite o início e o fim da tabuada, em vez de começar de 1 até 10, além do número que se deseja a tabuada.

Estrutura de Repetição *while*

Acumuladores

- São variáveis que atuam **acumulando** os valores a cada vez que o código é executado.
- Utilizado em programas para calcular o total de uma soma, por exemplo.
- A diferença entre um **contador** e um **acumulador** é que nos contadores o valor adicionado é **constante** e, nos acumuladores, **variável**.

Estrutura de Repetição *while*

Acumuladores

Exemplo de um programa que calcula a soma de 5 números digitados pelo usuário.

```
soma = 0      # acumulador
n = 1         # contador
while n <= 5:
    x = int(input("Digite um número: "))
    soma = soma + x       # ou   soma += x
    n = n + 1             # ou   n += 1
print(f"Soma: {soma}")
```

Estrutura de Repetição *while*

Interrompendo a repetição

- Dependendo do problema, a habilidade de terminar o *while* dentro do bloco a repetir pode ser necessário.
- A instrução ***break*** é utilizada para interromper a execução de *while* independentemente do valor atual de sua condição.

Estrutura de Repetição *while*

Interrompendo a repetição: Exemplo para somar vários números até que o usuário digite **0 (zero) para parar a leitura**.

Este procedimento é conhecido na programação como FLAG (“bandeira” em inglês), ou seja, exerce a função de sinalizar alguma coisa. Nesse caso, ocorre o término da leitura ao digitar **0 (zero)**.

```
soma = 0
while True:
    v = int(input(“Digite um número ou 0 para sair: ”))
    if v == 0:
        break
    soma = soma + v        # soma += v
print(f“Soma: {soma}”)
```

Estrutura de Repetição *while*

Exercícios

8. Faça um programa para calcular e imprimir a média de 5 números digitados pelo usuário.
9. Faça um programa que pergunte o depósito inicial e a taxa de juros de uma poupança. Exiba os valores mês a mês para os 12 primeiros meses. Escreva o total ganho com juros no período.
10. Faça um programa que pergunte o valor inicial de uma dívida e o juro mensal. Pergunte quantos meses para pagar a dívida. Imprima o novo valor da dívida incluindo os juros.
11. Calcular a idade média de uma turma de alunos onde não é conhecido a priori o número de alunos. Neste exercício utilizar uma FLAG que indicará o término do número de alunos. Assumiremos que esta FLAG é um número negativo.

Estrutura de Repetição *while*

Exercícios

12. Faça um programa que leia números inteiros do teclado, até que o usuário digite 0 (zero). No final da execução, exiba a quantidade de números digitados, assim como a soma e a média aritmética.
13. Supondo que a população de um país A seja da ordem de 90.000.000 de habitantes com uma taxa anual de crescimento de 3% e que a população de um país B seja, aproximadamente, de 200.000.000 de habitantes com uma taxa anual de crescimento de 1.5%, fazer um programa que calcule e escreva o número de anos necessários para que a população do país A ultrapasse ou se iguale à população do país B, mantidas essas taxas de crescimento.
14. Escreva um programa que leia um número e verifique se é ou não um número primo. Para fazer essa verificação, calcule o resto da divisão do número por 2 e depois por todos os números ímpares até o número lido. Se o resto de uma dessas divisões for igual a zero, o número não é primo. Observe que 0 e 1 não são primos e que 2 é o único primo que é par.

Estrutura de Repetição *while*

Repetições aninhadas

- Quando se tem um bloco de código que deseja executar x número de vezes, então um bloco de código dentro desse código que se deseja executar y número de vezes, usa-se o que é conhecido como "*loop* aninhado" ou "repetições aninhadas".
- Dependendo do problema, pode ser necessário combinar vários *while* com incremento de duas variáveis.
- Considere, por exemplo, imprimir as tabuadas de multiplicação de 1 a 10.

```
tab = 1
while tab <= 10:
    núm = 1
    while núm <= 10:
        print(f"{tab} x {núm} = {tab*núm}")
        núm += 1
    tab += 1
```

Função *range*

- A função *range* (“intervalo”) é utilizada para gerar listas simples, retornando uma sequência seguindo um padrão específico (na maioria das vezes, inteiros sequenciais), o que, portanto, atende ao requisito de fornecer uma sequência para a instrução *for* repetir (iterar).

  range(stop)

  range([start], stop[, step])

  intervalo aberto] x [Quando os colchetes se opõem ao número, temos um intervalo aberto. Assim, todos os números no intervalo estarão contidos na sequência numérica, menos os extremos.

- Como *for* pode operar diretamente em sequências, muitas vezes não há necessidade de contar. Esta é uma construção comum para programadores que vierem de outra linguagem com sintaxe de estrutura de repetição diferente.
- Exemplo para imprimir de 0 a 9 na tela:

```
for v in range(10):        # range(10) gera a lista [0,1,2,3,4,5,6,7,8,9]
    print(v)
```

Função *range*

- Com a mesma função pode-se indicar qual é o primeiro número a gerar. Para isso, deve-se utilizar dois parâmetros: início e fim.
- Exemplo:

```
for v in range(5, 8):        # range(5,8) gera a lista [5,6,7]
    print(v)
```

- Com a mesma função pode-se indicar qual é o primeiro número a gerar e o intervalo do incremento. Para isso, deve-se utilizar três parâmetros: início, fim e intervalo.
- Exemplo:

```
for v in range(2, 15, 3):    # range(2,15,3) gera a lista [2,5,8,11,14]
    print(v)
```

Estrutura de Repetição *for*

- A instrução ***for*** é usada quando se tem um bloco de código que deseja **repetir um número fixo de vezes**, sem a necessidade de testar uma condição a cada repetição.
- É usado em combinação com um objeto iterável, como uma lista ou um intervalo (itera sobre os membros de uma sequência em ordem, executando o bloco a cada vez).
- Funciona de forma parecida a ***while***, mas a cada repetição utiliza o próximo elemento da lista, que se repete até o fim.
- Exemplo da impressão dos elementos de 0 a 2:

```
for e in range(0, 3):
    print(e)
```

Estrutura de Repetição *for*

Saídas antecipadas

- Como o laço *while*, o laço *for* pode sair antes que o objeto dado (lista gerada pelo *range*, por exemplo) seja finalizado. Isso é feito usando a instrução ***break***, que sairá imediatamente do bloco de repetição e continuará a execução na primeira instrução após o bloco.
- Também se pode ter uma cláusula ***else*** opcional, que será executada se a estrutura *for* sair de forma limpa, ou seja, sem quebrar.
- Exemplos:

```
for x in range(3):
    if x == 1:
        break
```

```
for x in range(3):
    print(x)
else:
    print('Final x =', x)
```

Estrutura de Repetição *for*

Repetições aninhadas

- Quando se tem um bloco de código que deseja executar x número de vezes, então um bloco de código dentro desse código que se deseja executar y número de vezes, usa-se o que é conhecido como "loop aninhado".
- Em Python são usados sempre que alguém tem uma lista de listas - um objeto iterável dentro de um objeto iterável.
- Exemplo:

```
for x in range(1, 11):
    for y in range(1, 11):
        print(f‘{x} * {y} = {x*y}’)
```

Estrutura de Repetição *for*

Exercício

15. Tem-se um conjunto de dados contendo a altura e o sexo (masculino=1, feminino=2) de N pessoas. Fazer um programa que calcule e escreva:
- a maior e a menor altura do grupo;
- a média de altura das mulheres;
- o número de homens;
- % de mulheres.

Adotar como FLAG altura <= 0.

16. Uma pesquisa sobre algumas características físicas da população de uma determinada região coletou os seguintes dados, referentes a cada habitante, para serem analisados:
- sexo (masculino, feminino);
- cor dos olhos (azuis, verdes, castanhos);
- cor dos cabelos (louros, castanhos, pretos);
- idade em anos.

Para cada habitante, foi perfurado um cartão com esses dados, e o último cartão, que não corresponde a ninguém, conterá o valor da idade igual a –1. Fazer um programa que determine e escreva:
- a maior idade dos habitantes;
- porcentagem de indivíduos do sexo feminino cuja idade está entre 18 e 35 anos, inclusive, e que tenham olhos verdes e cabelos louros.

Estrutura de Repetição *for*

Exercício

17. Escreva um programa que leia uma *string* (palavra ou frase), conte quantos caracteres desta *string* são iguais a ‘a’ e substitua os que forem iguais à ‘a’ por ‘b’. O programa deve imprimir o número de caracteres modificados e a *string* modificada.

18. Faça um programa que inverta uma *string*: leia a *string* e armazene-a invertida numa outra *string*. Ao final, imprima as duas *strings*.