



# Programação I

---

LUCAS SAMPAIO LEITE

# Agenda

---

- Estruturas de repetição:

  - **While**

  - For

# Estruturas de repetição

---

- ❑ Imagine um programa que calcula a média de um aluno;
- ❑ Este programa é bastante simples, bastaria:
  - ❑ Ler as notas do teclado;
  - ❑ Calcular a média; e
  - ❑ Imprimir o resultado.

# Estruturas de repetição

---

- ❑ Imagine um programa que calcula a média de um aluno.
- ❑ Este programa é bastante simples, bastaria:
  - ❑ Ler as notas do teclado;
  - ❑ Calcular a média; e
  - ❑ Imprimir o resultado.

```
main.py > ...  
1  nota1 = float(input('Digite a primeira nota do aluno: '))  
2  nota2 = float(input('Digite a segunda nota do aluno: '))  
3  nota3 = float(input('Digite a terceira nota do aluno: '))  
4  
5  media = (nota1+nota2+nota3)/3  
6  
7  print('A média do aluno é: ',media)
```

# Estruturas de repetição

---

- ❑ Imagine um programa que calcula a média de um aluno.
- ❑ Este programa é bastante simples, bastaria:
  - ❑ Ler as notas do teclado;
  - ❑ Calcular a média; e
  - ❑ Imprimir o resultado.

**Como poderíamos fazer para calcular a média de todos os alunos de uma turma em um mesmo programa?**

main.py > ...

```
1 nota1 = float(input('Digite a primeira nota do aluno: '))  
no: '))  
uno: '))
```

```
print('A média do aluno é: ', media)
```

# Estruturas de repetição

---

- ❑ Observe que precisamos repetir os mesmos comandos, várias vezes, até que a média de todos os alunos tenha sido calculada e impressa.
- ❑ As linguagens de programação oferecem mecanismos para repetir comandos varias vezes, que são conhecidos como laços (em inglês: loops).
- ❑ Em Python temos duas opções de estruturas de repetição:
  - ❑ while;
  - ❑ for.

# Estruturas de repetição

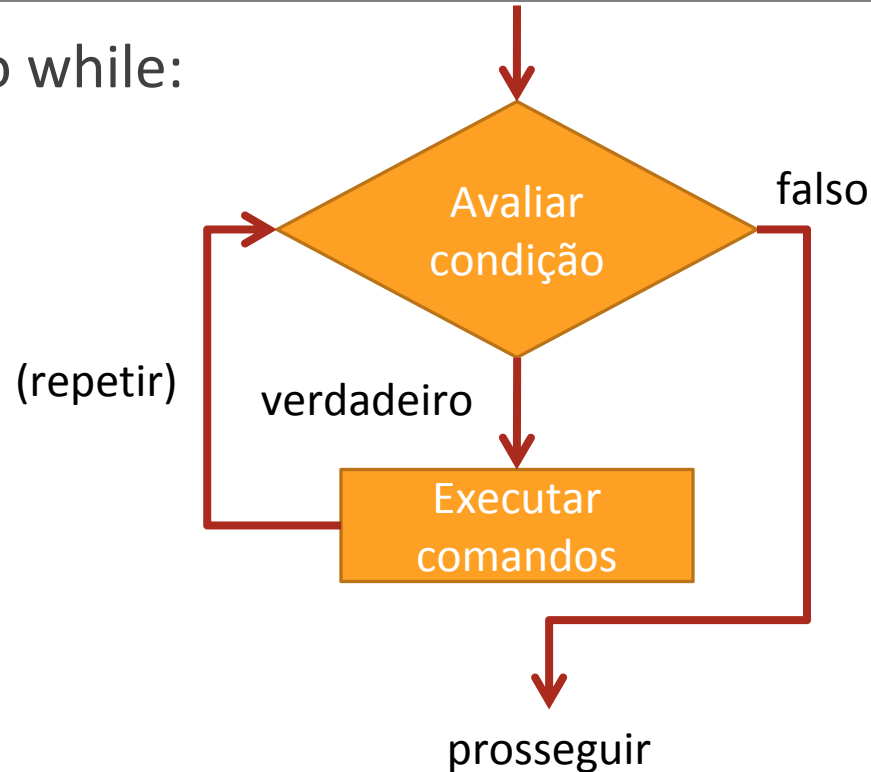
---

- ❑ Dois tipos de Repetição:
  - ❑ Repetição condicional: executa um bloco de código enquanto uma condição lógica for verdadeira (while);
  - ❑ Repetição contável: executa um bloco de código um número determinado de vezes (for).



# Estruturas de repetição (while)

❑ Fluxograma do while:



❑ Observe que há a possibilidade de nunca se executar os comandos caso a primeira avaliação da condição já resulte em falso.



# Estruturas de repetição (while)

---

- ❑ A estrutura while possui a seguinte sintaxe:

```
while <condição>:  
    bloco verdadeiro
```

- ❑ A palavra “while” significa “enquanto” em português, portanto, lê-se:
- ❑ “Enquanto a expressão booleana for verdadeira, execute os comandos do bloco abaixo”.
- ❑ Ou seja, o bloco de comandos será repetido enquanto a expressão booleana for verdadeira.
- ❑ Algo dentro do laço deve ser capaz de modificar o resultado da expressão booleana, caso contrário o laço nunca terminará, e o programa entrará em “loop infinito”.

# Estruturas de repetição (while)

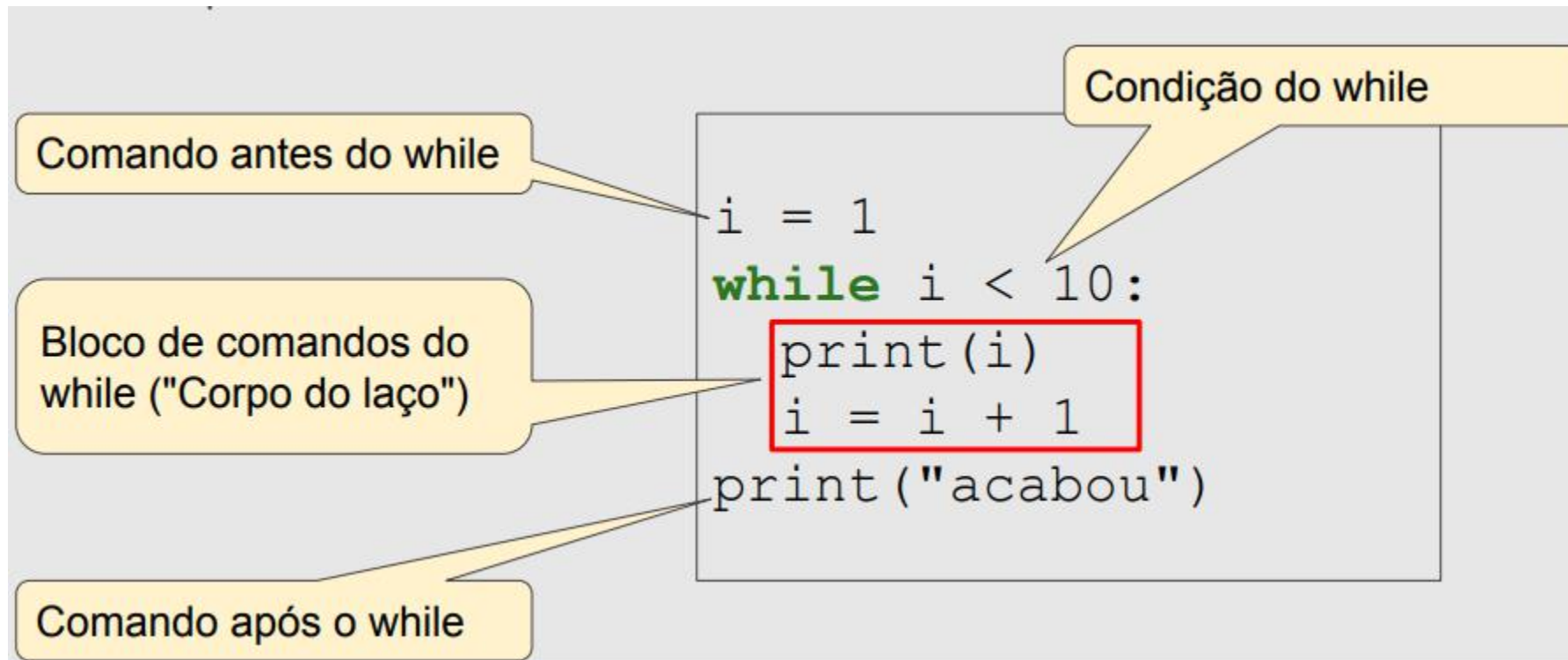
---

Executa um bloco de comando(s) enquanto a condição é verdadeira (True).

```
while condicao:  
    comando(s)
```

# Estruturas de repetição (while)

---



# Estruturas de repetição (while)

---

Passo 1: teste da condição de parada

```
i = 1  
while i < 10:  
    print(i)  
    i = i + 1  
print("acabou")
```

# Estruturas de repetição (while)

---

Passo 1: teste da condição de parada

Passo 2: Caso a condição for verdadeira, execute os comandos do bloco do while e volte para o Passo 1

```
i = 1  
while i < 10:  
    print(i)  
    i = i + 1  
print("acabou")
```

# Estruturas de repetição (while)

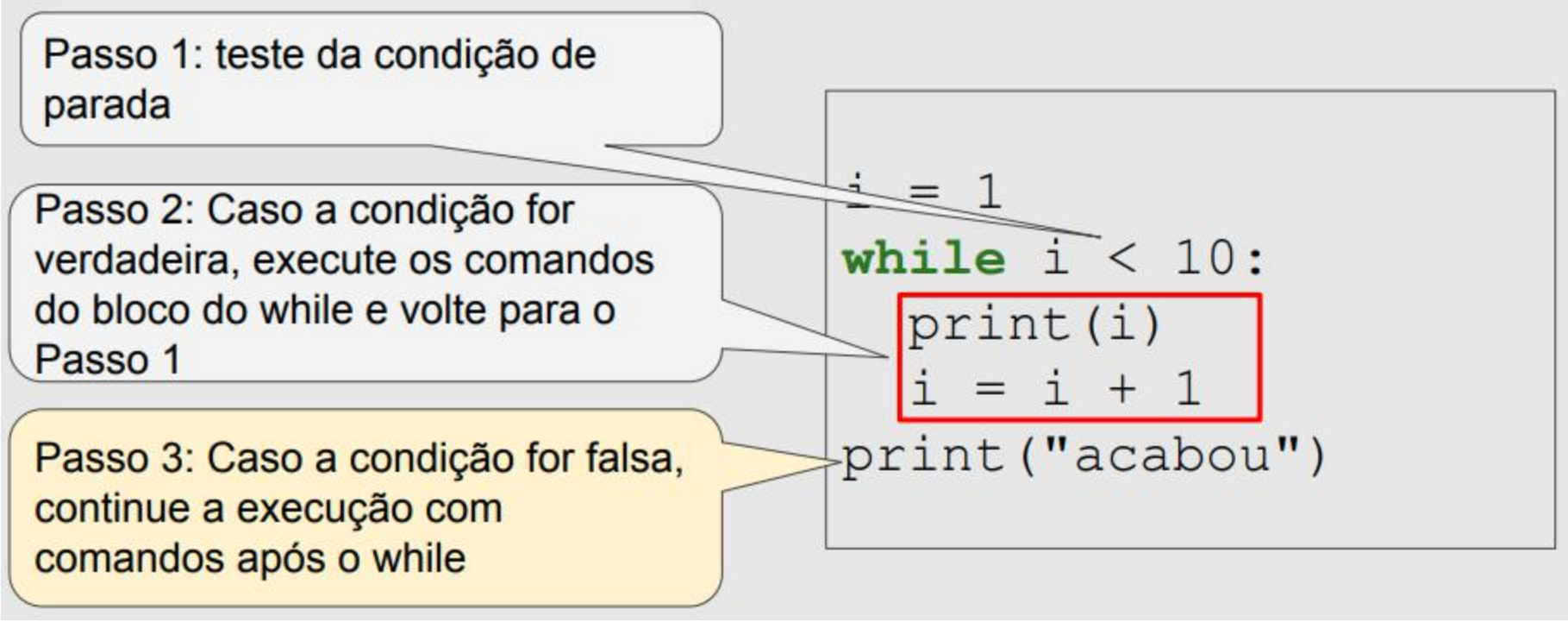
---

Passo 1: teste da condição de parada

Passo 2: Caso a condição for verdadeira, execute os comandos do bloco do while e volte para o Passo 1

Passo 3: Caso a condição for falsa, continue a execução com comandos após o while

```
i = 1  
while i < 10:  
    print(i)  
    i = i + 1  
print("acabou")
```

The diagram illustrates the execution of a while loop. It features a code block on the right and three callout boxes on the left. The first callout box, labeled 'Passo 1: teste da condição de parada', has a line pointing to the condition 'i < 10' in the while statement. The second callout box, labeled 'Passo 2: Caso a condição for verdadeira, execute os comandos do bloco do while e volte para o Passo 1', has a line pointing to the indented block of code inside the while loop: 'print(i)' and 'i = i + 1'. The third callout box, labeled 'Passo 3: Caso a condição for falsa, continue a execução com comandos após o while', has a line pointing to the final line of code, 'print("acabou")'. The code block is enclosed in a light gray box, and the callout boxes are also light gray, with the third one being yellow.

# Estruturas de repetição (while)

---

❑ Qual será a saída produzida por este programa?

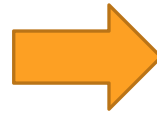
```
1  i = 1
2  while i < 10:
3      print(i)
4      i = i + 1
5  print("acabou")
```

# Estruturas de repetição (while)

---

- ❑ Qual será a saída produzida por este programa?

```
1  i = 1
2  while i < 10:
3      print(i)
4      i = i + 1
5  print("acabou")
```



```
lucas@lucas-Inspiron-5548:~/
_24_02.py
1
2
3
4
5
6
7
8
9
acabou
```



# Estruturas de repetição (while)

---

❑ Qual será a saída produzida por este programa?

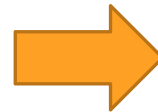
```
1  i = 1
2  while i != i:
3      i = i + 1
```

# Estruturas de repetição (while)

---

❑ Qual será a saída produzida por este programa?

```
1  i = 1
2  while i != i:
3      i = i + 1
```



Ele nunca vai entrar na repetição  
(no laço).

Condição será sempre falsa!!!

# Estruturas de repetição (while)

---

❑ Qual será a saída produzida por este programa?

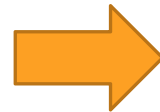
```
1  i = 1
2  while i == i:
3      i = i + 1
```

# Estruturas de repetição (while)

---

❑ Qual será a saída produzida por este programa?

```
1  i = 1
2  while i == i:
3      i = i + 1
```



Ele entra na repetição e nunca sai (laço infinito).

Condição será sempre verdadeira!!!

# Estruturas de repetição (while)

- ❑ Resolução do problema das médias com a estrutura de repetição while:

```
main.py > ...  
1  repetir = True  
2  
3  while(repetir):  
4      nota1 = float(input('Digite a primeira nota do aluno: '))  
5      nota2 = float(input('Digite a segunda nota do aluno: '))  
6      nota3 = float(input('Digite a terceira nota do aluno: '))  
7  
8      media = (nota1+nota2+nota3)/3  
9  
10     print('A média do aluno é: ', media)  
11  
12     digito = input('Deseja continuar? (s/n):')  
13     if digito == 'n':  
14         repetir = False
```

# Estruturas de repetição (while)

---

- ❑ Outros exemplos do uso de while. O que eles fazem?

```
main.py > ...  
1   i = 0  
2   while(i < 10):  
3       print(i)  
4       i = i+1
```

# Estruturas de repetição (while)

---

- ❑ Outros exemplos do uso de while. O que eles fazem?

```
main.py > ...  
1  i = 0  
2  while(i < 10):  
3      print(i)  
4      i = i+1
```

```
main.py > ...  
1  i = 10  
2  while(i >= 1):  
3      print(i)  
4      i = i-1
```

# Estruturas de repetição (while)

- Outros exemplos do uso de while. O que eles fazem?

```
main.py > ...  
1 i = 0  
2 while(i < 10):  
3     print(i)  
4     i = i+1
```

```
main.py > ...  
1 a = 0  
2 b = 2  
3 while a <= b:  
4     print( a, ' <= ', b, ' ' )  
5     a += 1
```

```
main.py > ...  
1 i = 10  
2 while(i >= 1):  
3     print(i)  
4     i = i-1
```



# Estruturas de repetição (while)

---

- ❑ Em alguns casos pode ser útil finalizar o laço no meio de uma repetição:
- ❑ Para isso, podemos utilizar o comando **break**

```
main.py > ...
1  while(True):
2      nota1 = float(input('Digite a primeira nota do aluno: '))
3      nota2 = float(input('Digite a segunda nota do aluno: '))
4      nota3 = float(input('Digite a terceira nota do aluno: '))
5
6      media = (nota1+nota2+nota3)/3
7
8      print('A média do aluno é: ', media)
9
10     digito = input('Deseja continuar? (s/n):')
11     if digito == 'n':
12         break
```

# Estruturas de repetição (while)

---

- ❑ Em alguns casos pode ser útil pular apenas uma iteração do laço.
- ❑ Para isso, podemos utilizar o comando **continue**.
- ❑ Imprime todos os números de 1 a 100, exceto 5 e seus múltiplos.

```
main.py > ...  
1   i = 0  
2   while i < 100:  
3       i = i + 1  
4       if i % 5 == 0:  
5           continue  
6       print(i)
```

# Exercícios

---

1. Escreva um programa que imprime todos os numeros de 0 até 50, incluindo-os.
2. Modifique o programa anterior de forma que este imprima apenas os números que são pares.
3. Escreva um programa para contar a quantidade de números pares entre dois números quaisquer fornecidos pelo usuário?
4. Escreva um programa para calcular o fatorial de um número fornecido pelo usuário.

# Exercícios

---

5. Faça um programa que peça dois números, base e expoente, calcule e mostre o primeiro número elevado ao segundo número. Não utilize a função de potência da linguagem ou o operador de exponenciação.
6. Desenvolva um gerador de tabuada, capaz de gerar a tabuada de qualquer número inteiro entre 1 a 10. O usuário deve informar de qual numero ele deseja ver a tabuada. A saída deve ser conforme o exemplo abaixo:

```
Tabuada de 5:  
5 x 1 = 5  
5 x 2 = 10  
...  
5 x 10 = 50
```

# Mais exercícios

---

7. A prefeitura de uma cidade deseja fazer uma pesquisa entre seus habitantes. Faça um algoritmo para coletar e armazenar dados sobre o salário e número de filhos de cada habitante e após as leituras, escrever:
- a) Média de salário da população
  - b) Média do número de filhos
  - c) Maior salário dos habitantes
  - d) Percentual de pessoas com salário menor que R\$ 150,00

Obs.: O final da leituras dos dados se dará com a entrada de um “salário negativo”.

# Dúvidas???

---



Fonte: <https://institutoseculoxxi.com.br/duvidas-entramos-em-contato-com-voce/>

# Agenda

---

- Estruturas de repetição
  - While
  - **For**

# Estruturas de repetição (for)

---

❑ Laço de repetição, assim como o while.

❑ A sintaxe da utilização do for com a função range():

```
for i in range <n>:  
    comandos
```

❑ Neste caso, o primeiro valor de i é 0 e segue sendo incrementado de um em um até o número n-1.

❑ Para utilizar o for é sempre necessário indicar uma variável iteradora que irá assumir um valor diferente para cada iteração do laço, e é sempre necessário indicar os limites de iteração.



# Estruturas de repetição (for)

---

❑ Laço de repetição, assim como o while.

❑ A sintaxe da utilização do for com a função range():

```
for i in range <n>:  
    comandos
```

❑ Exemplo:

```
main.py  
1  for i in range(5):  
2      print(i)  
3
```

Neste caso, o laço for imprime na tela o respectivo valor de i para cada iteração.

O primeiro valor de i é 0 e segue sendo incrementado de um em um até o número 4.

# Estruturas de repetição (for)

---

- ❑ É possível incluir um valor de início da contagem diferente de 0;
- ❑ Para isso devemos usar a função com dois parâmetros:
- ❑ Exemplo:

for i in range (<início>,<fim>):  
comandos

```
main.py > ...  
1  for i in range(2,6):  
2      print(i)  
3
```



Neste caso, o laço for imprime na tela o respectivo valor de i para cada iteração.  
O primeiro valor de i é 2 e segue sendo incrementado de um em um até o número 5.

# Estruturas de repetição (for)

---

- ❑ A função range também pode ser utilizada para limitar a execução do laço de repetição for, como a função range(m, n, p), que cria uma lista de inteiros começando em m e terminando em n-1 sendo incrementada de p em p.

```
main.py > ...  
1 m=1  
2 n=100  
3 p=2  
4 ✓ for i in range (m,n,p):  
5 |     print(i)
```

```
main.py > ...  
1 for i in range(1,100,2):  
2 |     print(i)  
3
```

O que seria impresso?

# Estruturas de repetição (for)

---

```
for x in range(0, 5, 1):  
    print(x)
```

**início (opcional)** –  
quando omitido,  
**início = 0**


**fim (obrigatório)**

**incremento (opcional)** –  
quando omitido,  
**incremento = 1**

# Estruturas de repetição (for)

---

- ❑ A variável iteradora pode assumir o valor de uma string ou elemento de uma lista:

```
 main.py > ...  
1   x = 'Lógica de programação'  
2   for i in x:  
3       print (i)
```

# Estruturas de repetição (for)

---

- ❑ O laço for também é utilizado para percorrer listas.
- ❑ Escrever **for i in [0, 1, 2, 3]:** é o mesmo que representar:
  - ❑ “Para cada valor de “i” dentro dos valores (0, 1, 2 e 3), faça:”.

```
main.py > ...  
1  ∨ for i in [0, 1, 2, 3]:  
2    |     print(i)  
3
```

# Estruturas de repetição (for)

---

- ❑ A função `len()` é utilizada quando se quer percorrer uma lista, de tal forma que o valor da variável iteradora não assuma cada valor da lista e sim cada posição da lista.
- ❑ Ou seja, não importa o conteúdo de cada posição da lista, a variável iteradora irá assumir o valor de cada posição.

```
main.py > ...
1  x = ['Lógica de programação', 'é', 'nota', 10.0, '\n']
2
3  for i in range(len(x)):
4      print (i)
5
6  for i in x:
7      print (i)
```

# Estruturas de repetição (for)

- ❑ O laço for pode ser utilizado através das instruções break e continue.
- ❑ A instrução break interrompe o laço (terminando-o por completo) e a instrução continue pula para a próxima iteração imediatamente (não termina o laço, apenas passa à próxima iteração).

```
main.py > ...
1  for i in range(5):
2      if i == 0:
3          print('\ni = 0, Então: ', i)
4      elif i == 1:
5          print('\ni = 1, Então: continue')
6          continue
7      elif 1 < i < 3:
8          print('\nA variável i, é: ', i)
9      elif i == 3:
10         print('\ni = 3, Então: break')
11         break
12     else:
13         print('\ni > 3, Então: ', i)
```

O que será impresso quando i=4?



# Exercícios (utilizando for)

---

1. Escreva um programa que imprime todos os numeros de 0 até 50, incluindo-os.
2. Modifique o programa anterior de forma que este imprima apenas os números que são pares.
3. Escreva um programa para contar a quantidade de números pares entre dois números quaisquer fornecidos pelo usuário?
4. Escreva um programa para calcular o fatorial de um número fornecido pelo usuário.

# Mais exercícios... (utilizando for)

---

5. Faça um programa que peça dois números, base e expoente, calcule e mostre o primeiro número elevado ao segundo número. Não utilize a função de potência da linguagem ou o operador de exponenciação.
6. Desenvolva um gerador de tabuada, capaz de gerar a tabuada de qualquer número inteiro entre 1 a 10. O usuário deve informar de qual numero ele deseja ver a tabuada. A saída deve ser conforme o exemplo abaixo:

```
Tabuada de 5:  
5 x 1 = 5  
5 x 2 = 10  
...  
5 x 10 = 50
```

# Mais exercícios (usando for ou while)

---

7. Escreva um programa que leia um número inteiro e calcule a soma de todos os divisores desse número, com exceção dele próprio. Ex: a soma dos divisores do número 66 é  $1 + 2 + 3 + 6 + 11 + 22 + 33 = 78$
8. Em Matemática, o número harmônico designado por  $H(n)$  define-se como sendo a soma da série harmónica:

$$H(n) = 1 + 1/2 + 1/3 + 1/4 + \dots + 1/n$$

Faça um programa que leia um valor  $n$  inteiro e positivo e apresente o valor de  $H(n)$ .

# Mais exercícios (usando for ou while)

---

9. A série de Fibonacci é formada pela seqüência 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Faça um programa capaz de gerar a série até o n-ésimo termo.
10. A série de Fibonacci é formada pela seqüência 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,... Faça um programa que gere a série até que o valor seja maior que 500.
11. Faça um programa que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.:  $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ . A saída deve ser conforme o exemplo abaixo:

```
Fatorial de: 5
```

```
5! = 5 . 4 . 3 . 2 . 1 = 120
```

# Dúvidas???

---



Fonte: <https://institutoseculoxxi.com.br/duvidas-entramos-em-contato-com-voce/>