

# **Introdução**

## Comandos de Repetição

Mauricio Ferste

## Comandos de Repetição

Comando `while`

Listas

Comando

`for`

Comandos

`break` e  
`continue`

Laços

## Comandos de Repetição

- Até agora, vimos como escrever programas capazes de executar comandos de forma sequencial e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.
- Entretanto, muitas vezes é necessário executar um bloco de comandos várias vezes para obter o resultado desejado.

- Imprimindo todos os números inteiros de 1 até 5.

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

print (4)

print (5)
```

- Imprimindo todos os números inteiros de 1 até 100.

```
1 print (1)
2
3 print (2)
4
5 print (3)
...
print (100)
```

# Exemplos

- Imprimindo todos os números inteiros de 1 até  $n$ .
- Note que só resolvemos o problema para  $n \leq 100$ .

```
1 n = int(input("Digite um número: "))
2 if n >= 1:
3
4     print(1)
5 if n >= 2:
6
7     print(2)
8 if n >= 3:
9
10    print(3)
11 if n >= 4:
12     print(4)
13 ...
14 if n >=
15 100:
16     print(10
17 0)
```

Comando

while



# Comando `while`

- O primeiro comando de repetição que aprenderemos é o `while`.

```
1 while <condição>:  
2     # este bloco irá repetir até a condição ser falsa  
3     <comando1>  
4     <comando2>  
5     ...  
6     <comandoY>
```

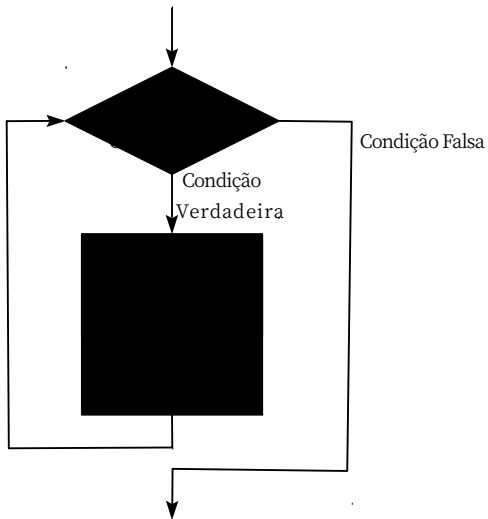
- Funcionamento:

Passo 1: Se a condição for verdadeira, vá para o Passo 2.

Caso contrário, encerre o bloco de repetição (`while`).

Passo 2: Execute o bloco de comandos. Passo 3: Volte para o Passo 1.

# Comando `while`



## Exemplo do Comando `while`

- Imprimindo todos os números inteiros de 1 até 100.

```
1 i = 1
2 while i <= 100:
3     print(i)
4     i = i +
5     1
6     print("Fim
7     do
8     programa!")
```

```
1 n = int(input("Digite um número inteiro positivo: "))
2 i = 1
3 while i <= n:
4     print(i)
5     i = i +
6     1
7     print("Fim
8     do
9     programa!")
```

# Exemplo do Comando `while`

- Contagem regressiva:

```
1 n = int(input("Digite um número inteiro positivo: "))
2
3 while n >= 0:
4     print(n)
5     n = n - 1
6
7 print("BOOM!")
```

## Exemplo do Comando `while`

- Dados dois números inteiros positivos, calcule o quociente e o resto da divisão inteira entre os dois, usando apenas somas e subtrações.

```
1 dividendo = int(input("Entre com o dividendo: "))
2 divisor = int(input("Entre com o divisor: "))
3
4 quociente = 0
5
6 while dividendo >= divisor:
7     dividendo = dividendo - divisor
8     quociente = quociente + 1
9
10 print("Quociente:", quociente)
11 print("Resto:", dividendo)
```

- Como calcular  $b^e$  (para  $b \in \mathbb{R}$ ,  $e \in \mathbb{N}$ )?
- Para  $e \geq 1$ , usando  $e - 1$  multiplicações:
  - $b^e = b \times b \times \dots \times b$
- Para  $e \geq 0$ , usando  $e$  multiplicações:
  - $b^e = 1 \times b \times b \times \dots \times b$

## Exemplo do Comando `while`

- Dados dois números inteiros positivos (base e expoente), calcule  $base^{expoente}$ , usando apenas somas e multiplicações.

```
1 base = int(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = base
5     i = 1
6
7 while i <
8 exp:
9     result = result * base
10     i = i + 1
11
12 print(result)
```

## Exemplo do Comando `while`

- Dados dois números inteiros não negativos (base e expoente), calcule  $base^{expoente}$ , usando apenas somas e multiplicações.

```
1 base = int(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = 1
5 i = 1
6
7 while i <= exp:
8     result = result * base
9     i = i + 1
10
11 print(result)
```

1  
1



## Comando `while`

O que acontece se a condição for falsa na primeira

```
1 a = 0
2 while a > 0:
3     a = a +
4     1
   print(a)
```

- Resposta: o programa nunca entra no bloco de repetição.

O que acontece se a condição for sempre verdadeira?

```
1 a = 0
2 while a >= 0:
3     a = a + 1
4     print(a)
```

- Resposta: o programa entra no bloco e nunca sai (*loop infinito*).

# Exponenciação

- Como calcular  $b^e$  (para  $b \in \mathbb{R}$ ,  $e \in \mathbb{Z}$ )?
- Para  $e > 0$ , usando  $e$  multiplicações:
  - $b^e = 1 \times b \times b \times \dots \times b$
- Para  $e < 0$ , usando  $|e|$  divisões:
  - $b^e = (((1/b)/b)/\dots)/b$

## Exemplo do Comando `while`

- Dados dois números inteiros (base e expoente), calcule  $base^{expoente}$ , usando apenas somas, subtrações, multiplicações e divisões.

```
1 base = int(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = 1
5
6 while exp > 0:
7     result = result * base
8     exp = exp - 1
9
10 while exp < 0:
11     result = result / base
12     exp = exp + 1
13
14 print(result)
```

## Exemplo do Comando `while`

- Dados dois números inteiros (base e expoente), calcule  $base^{expoente}$ , usando apenas somas, subtrações, multiplicações e divisões.

```
1 base = int(input("Base: "))
2 exp = int(input("Expoente: "))
3
4 result = 1
5
6 while exp < 0:
7     result = result / base
8     exp = exp + 1
9
10 while exp > 0:
11     result = result * base
12     exp = exp - 1
13
14 print(result)
```

## Exemplo do Comando `while`

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3 i = 1
4
5 while i <= n:
6     fat = fat * i
7     i = i + 1
8
9 print("Resultado: ", fat)
```

## Exemplo do Comando `while`

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 fat = 1
3 i = 2
4
5 while i <= n:
6     fat = fat * i
7     i = i + 1
8
9 print("Resultado: ", fat)
```

## Exemplo do Comando `while`

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: ")) fat = 1
2
3
4 while n > 0:
5     fat = fat * n    n = n - 1
6
7
8 print("Resultado:", fat)
9
```

## Exemplo do Comando `while`

- Dado um número inteiro positivo  $n$ , calcular o valor de  $n!$ .

```
1 n = int(input("Entre com um número inteiro positivo: ")) fat = 1
2
3
4 while n > 1:
5     fat = fat * n n = n -
6     1
7
8 print("Resultado:", fat)
9
```



Listas

- Uma lista é uma estrutura de Python que armazena múltiplos dados.

```
1 lista = [<dado_1>, <dado_2>, <dado_3>, ..., <dado_n>]
```

- Podemos ter todos os dados do mesmo tipo.

```
1 # Uma lista com dados do tipo int
2 lista_de_int = [40, 3, 61, 7, 3]
3
4 # Uma lista com dados do tipo bool
5 lista_de_bool = [True, False, True]
```

- Podemos ter dados de tipos diferentes misturados.

```
1 lista_mista = ["Gato", 42, True, 5.4, "Cachorro", 73]
```

- Podemos acessar um elemento em uma lista indicando a sua posição (o primeiro elemento fica na posição 0 da lista).

```
1 lista = ["Azul", 51, "Amarelo", 55, True, 7.2]
2 print(lista[0]) # imprime o primeiro elemento da lista
3 # Azul
4 print(lista[1]) # imprime o segundo elemento da lista
5 # 51
6 print(lista[2]) # imprime o terceiro elemento da lista
7 # Amarelo
8 print(lista[-1]) # imprime o último elemento da lista
9 # 7.2
```

- A função `len()` retorna o número de elementos de

```
1 print(len(lista))
2 # 6
```

- Podemos usar um `while` para percorrer todos os elementos de uma lista.

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 i = 0
3 while i < len(letras):
4     print(letras[i])
5     i = i + 1
```

## Exemplo do Comando `while` com Listas

- Encontrando o máximo de um conjunto de números positivos.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = 0
3
4 i = 0
5 while i < len(numeros):
6     if numeros[i] > maximo:
7         maximo = numeros[i]
8         i = i + 1
9
10 print(maximo) # 9
```

## Exemplo do Comando `while` com Listas

- Encontrando o máximo de um conjunto de números quaisquer.

```
1 numeros = [-3, -1, -7, -9, -4]
2 maximo = numeros[0]
3
4 i = 1
5 while i < len(numeros):
6     if numeros[i] > maximo:
7         maximo = numeros[i]
8         i = i + 1
9
10 print(maximo) # -1
```

Comando `for`

- Podemos percorrer uma lista de forma mais compacta com o comando `for`.

```
1 for <variavel> in <lista>:  
2 # este bloco irá repetir para todos os valores da lista  
3     <comando1>  
4     <comando2>  
5     ...  
6     <comandoY>
```



# Comando for

- Estes dois códigos são equivalentes.

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 i = 0
3 while i < len(letras):
4     print(letras[i])
5 i = i + 1
```

```
1 letras = ["A", "B", "C", "D", "E", "F", "G"]
2 for letra in letras:
3     print(letra)
```

## Exemplo do Comando `for`

- Encontrando o máximo de um conjunto de números positivos.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = 0
3
4 for numero in numeros:
5     if numero > maximo:
6         maximo = numero
7
8 print(maximo) # 9
```

## Exemplo do Comando `for`

- Encontrando o máximo de um conjunto de números.

```
1 numeros = [3, 1, 7, 9, 4]
2 maximo = numeros[0]
3
4 for numero in numeros:
5     if numero > maximo:
6         maximo = numero
7
8 print(maximo) # 9
```

# Função range

- Python possui a função `range`, que pode ser usada para criar uma lista de inteiros.

```
1 print(list(range(2, 6)))  
2 # [2, 3, 4, 5]
```

- Esta função recebe como argumentos os limites da sequência a ser gerada (o primeiro número é incluído na sequência, mas o último não).

```
1 print(list(range(1, 5)))  
2 # [1, 2, 3, 4]  
3 print(list(range(0, 4)))  
4 # [0, 1, 2, 3]  
5 print(list(range(0, -5)))  
6 # []
```

# Função range

- Se a lista começa em 0, o primeiro número pode ser omitido.

```
1 print(list(range(4)))  
2 # [0, 1, 2, 3]
```

- A função range pode receber um terceiro argumento (opcional), que determina o incremento usado na geração da lista.

```
1 print(list(range(2, 10, 2)))  
2 # [2, 4, 6, 8]  
3 print(list(range(1, 15, 3)))  
4 # [1, 4, 7, 10, 13]  
5 print(list(range(5, 0, -1)))  
6 # [5, 4, 3, 2, 1]
```

# Repetindo n vezes

- Note que podemos utilizar `range(n)` em conjunto com o `for` para repetir uma operação n vezes.

```
1 for i in range(100):  
2     print("Esta frase será impressa 100 vezes")
```

```
1 n = int(input("Digite um número inteiro positivo: "))  
2 for i in range(n):  
3     print("Esta frase será impressa", n, "vezes")
```

# Exemplo do Comando `for` com

## `range`

- Imprimindo todos os números inteiros de 1 até 100.

```
1 for i in range(1, 101):  
2     print(i)  
3 print("Fim do programa!")
```

- Imprimindo todos os números inteiros de 1 até n.

```
1 n = int(input("Digite um número inteiro positivo: "))  
2 for i in range(1, n+1):  
3     print(i)  
4 print("Fim do programa!")
```

## Exemplo do Comando `for` com `range`

- Imprimindo as  $n$  primeiras potências de 2, sem usar o operador de exponenciação ( $**$ ).

```
1 n = int(input("Digite um número inteiro positivo: "))
2 potencia = 1
3 for i in range(n):
4     potencia = potencia * 2
5     print(potencia)
```



# Variável Acumuladora

- Como no exemplo anterior, em vários problemas precisamos combinar dados em uma variável utilizando alguma operação.
- Esse tipo de variável é chamada de acumuladora.
- Exemplo: somando os números inteiros de 1 até n.

```
1 n = int(input("Entre com um número inteiro positivo: "))
2 soma = 0
3 for i in range(1, n+1):
4     soma = soma + i
5 print(soma)
```

- Calculando  $n!$

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(1, n+1):
4     fatorial = fatorial * i
5 print(fatorial)
```

- Calculando  $n!$

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(2, n+1):
4     fatorial = fatorial * i
5 print(fatorial)
```

- Calculando  $n!$

```
1 n = int(input("Digite um número inteiro positivo: "))
2 fatorial = 1
3 for i in range(n, 1, -1):
4     fatorial = fatorial *
5     i
print(fatorial)
```

**Comandos** break e  
continue

- Vimos que o comando `for` percorre a lista completa.
- Às vezes queremos interromper a execução do comando `for` antes dele percorrer a lista completa.
- O comando `break` faz com que a execução de um laço de repetição seja finalizada, passando a execução para o próximo comando após o laço.

- Exemplo: procurando um valor em uma lista sem elementos repetidos.

```
1 x = int(input("Digite um número: "))
2 for i in [2, 4, 7, 1, 0, 8, 9, 5]:
3     print("Verificando elemento", i)
4     if i == x:
5         print("Elemento", i, "encontrado!")
```

- Após encontramos o valor, podemos encerrar a busca.

```
1 x = int(input("Digite um número: "))
2 for i in [2, 4, 7, 1, 0, 8, 9, 5]:
3     print("Verificando elemento", i)
4     if i == x:
5         print("Elemento", i, "encontrado!")
6         break
```

- O que será impresso no seguinte código?

```
1 for i in range(1, 10):  
2     if i == 5:  
3         break  
4     print (i)  
5     print ("F  
im do  
programa")
```

- Resposta:

1  
2  
3  
4

Fim do programa



- Determinando a quantidade de números inteiros positivos fornecidos para um programa (até a leitura de um inteiro não positivo).

```
1 n = 0
2
3 while True:
4     x = int(input("Entre com um número inteiro positivo: "))
5     if x <= 0:
6         break
7     n = n + 1
8
9 print("Quantidade de números positivos fornecidos:", n)
```

## O Comando `break` e o Comando `else`

- Podemos usar o comando `else` para executar um bloco de comandos apenas caso o comando `for` ou comando `while` tenham sido executados sem interrupção (`break`).
- Encontrando um elemento `x` em uma lista.
- Quando o `break` é executado, o bloco `else` é ignorado.

```
1 x = int(input("Digite um número: "))
2 for i in [2, 4, 7, 1, 0, 8, 9, 5]:
3     if i == x:
4         print("Elemento", i, "encontrado!")
5         break else:
6             print("O elemento", x,
7                 "não está na lista.")
```

## Comando `continue`

- O comando `continue` faz com que a execução atual do bloco de comandos do laço de repetição seja finalizada, passando a execução para a próxima iteração do laço.

```
1 for i in range(1, 6):  
2     if i == 3:  
3         continue  
4         print(i)  
5     print("Fim  
do  
programa")
```

- Resposta:

1

2

4

5

Fim do programa

## Exemplo do Comando `continue`

- Estes dois códigos são equivalentes: imprimem apenas os elementos pares da lista.

```
1 for i in [2, 4, 7, 1, 0, 8, 9, 5]:  
2     if i % 2 == 0:  
3         print(i)
```

```
1 for i in [2, 4, 7, 1, 0, 8, 9, 5]:  
2     if i % 2 == 1:  
3         continue  
4     print(i)
```

Laços  
Aninhados

# Laços Aninhados

- Em muitas situações é necessário implementar um laço (bloco de repetição) dentro de outro laço.
- Estes blocos de comandos são conhecidos como laços aninhados.

# Exemplo de Laços Aninhados

- Imprimindo as tabuadas dos números de 1 a 10.

```
1 for i in range(1, 11):  
2     print("Tabuada do ", i, ":", sep = "")  
3     for j in range(1, 11):  
4         print(i, "x", j, "=", i * j)
```

# Exemplo de Laços Aninhados

- Imprimindo um retângulo com  $n$  linhas e  $m$  colunas ( $n \times m$  caracteres #).

```
1 n = int(input("Entre com o número de linhas: "))
2 m = int(input("Entre com o número de colunas:
3 "))
4
5 for i in range(n):
6     for j in range(m):
7         print("# ", end =
            "")
        print()
```



# Exemplo de Laços Aninhados

- Imprimindo um retângulo com n linhas e m colunas (n × m caracteres #).

```
1 n = int(input("Entre com o número de linhas: "))
2 m = int(input("Entre com o número de colunas:
3 "))
4
5 for i in range(n):
6     print("#" * m)
```

7

# Exemplo de Laços Aninhados

- Imprimindo um triângulo retângulo isóceles, com catetos de tamanho dado.

```
1 n = int(input("Entre com o tamanho dos catetos: "))
2   c = input("Entre com caractere a ser usado: ")
3
4 for i in range(1, n+1):
5     for j in range(i):
6         print(c, end = "")
7 print()
```

# Exemplo de Laços Aninhados

- Imprimindo um triângulo retângulo isóceles, com catetos de tamanho dado.

```
1 n = int(input("Entre com o tamanho dos catetos: "))
2   c = input("Entre com caractere a ser usado: ")
3
4 for i in range(1, n+1):
5     print(c * i)
```

6

7

# Exemplo de Laços Aninhados

- Imprimindo todos os horários de um dia, no formato HH:MM (horas e minutos).

```
1 for h in range(24):  
2     for m in range(60):  
3         print(h, m, sep =  
4             ":")
```

# Exemplo de Laços Aninhados

- Imprimindo todos os horários de um dia, no formato HH:MM (horas e minutos).

```
1 for h in range(24):  
2     for m in range(60): print('{:02d}:'  
3         '{:02d}'.format(h, m))
```

4

# Exemplo de Laços Aninhados

- Imprimindo todos os horários de um dia, no formato HH:MM:SS (horas, minutos e segundos).

```
1 for h in range(24):  
2     for m in range(60):  
3         for s in range(60): print('{:02d}:{:02d}:  
4             {:02d}'.format(h, m, s))
```

## Exercícios

## Exercícios

- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima os seus divisores.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima o número de seus divisores.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima sua fatoração em números primos.
- Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima seus fatores primos (cada fator deve ser impresso uma única vez).



## Exercícios

- Escreva um programa que leia um número inteiro ( $n \geq 1$ ) e, para todo inteiro  $j$  de 1 até  $n$ , imprima os valores:

# Exercício

## Imprimindo Divisores

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima os seus divisores.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2
3 for divisor in range(1, n+1):
4     if n % divisor == 0:
5         print(divisor)
```

# Exercício

## Contando Divisores

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima o número de seus divisores.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisores = 0
3 for divisor in range(1, n+1):
4     if n % divisor == 0:
5         divisores = divisores + 1
6 print(divisores)
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisores = 0
3 for divisor in range(1, n+1):
4     if n % divisor == 0: divisores =
5         divisores + 1
6
7 if divisores == 2:
8     print("Primo")
9 else:
10    print("Composto")
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))  primo
2 = True
3 for divisor in range(2, n):
4     if n % divisor == 0:  primo
5         = False
6
7 if primo:
8     print("Primo")  else:
9     print("Composto")
```

1  
0

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, n):
4     if n % divisor == 0:
5         primo = False break
6 if primo:
7     print("Primo")
8 else:
9     print("Composto
10 ")
```

0

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, n//2 + 1):
4     if n % divisor == 0:
5         primo = False
6         break
7 if primo:
8     print("Primo")
9 else:
10    print("Composto")
```

## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 primo = True
3 for divisor in range(2, int(n/2)+1):
4     if n % divisor == 0:
5         primo = False
6         break
7 if primo:
8     print("Primo")
9 else:
10    print("Compo
sto")
```



## Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e determine se ele é primo.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: ")) primo
2 = True
3 for divisor in range(2, int(n**0.5)+1): if n
4     % divisor == 0:
5     primo = False break
6 if primo:
7     print("Primo")
8 else:
9     print("Composto")
```

1  
0

## Fatoração em Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima sua fatoração em números primos.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisor = 2
3 while n != 1:
4     if n % divisor == 0:
5         print(divisor)
6         n = n / divisor
7     else:
8         divisor =
            divisor + 1
```

## Fatoração em Números Primos

Escreva um programa que leia um número inteiro positivo ( $n > 1$ ) e imprima sua fatoração em números primos.

- Possível Resposta:

```
1 n = int(input("Digite um número inteiro positivo: "))
2 divisor = 2
3 while n != 1:
4     while n % divisor == 0:
5         print(divisor)
6         n = n / divisor
7     divisor = divisor
8     + 1
```