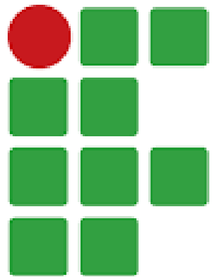


INSTITUTO FEDERAL
Santa Catarina

Lógica de Programação com Python

Estrutura de Repetição
WHILE / FOR

Jackson Meires
jackson.meires@ifsc.edu.br

Agenda

- Apresentar o conceito de estruturas de Repetição em Python
- While (Enquanto)
  - Como usar um Contador e Acumulador
- Operações Atribuição Especiais
- Repetições Aninhadas
- For (Para)
  - For Else

Por que usar estruturas de repetição?

- Por exemplo, um programa que imprima de 1 a 3 na tela.

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

- Imagine escrever de 1 a 100 na tela

Por que usar estruturas de repetição?

- Repetições representam a base de vários programa

- São utilizadas para executar a mesma parte de um programa várias vezes, normalmente dependendo de uma condição.

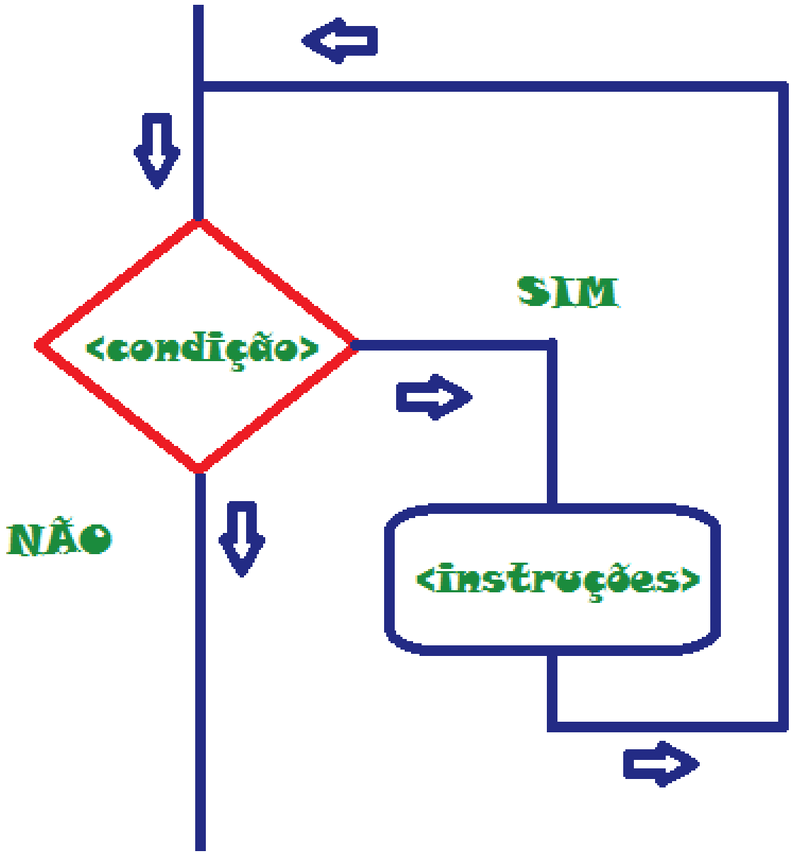
Estrutura de repetição Enquanto (while)

https://explorandoti.com.br/criando-repeticoes-com-while-no-python-3/

Estrutura de repetição enquanto

- Laço WHILE (Enquanto)

- A estrutura **while** repete um bloco de código enquanto a condição for verdadeira.
- A condição é uma expressão lógica, e o bloco representa linhas dos programas a repetir



Estrutura de repetição enquanto

- Uma estrutura de repetição **enquanto** pode ser utilizada quando o algoritmo precisa **testar determinada condição antes de executar um conjunto de comandos** repetidas vezes

- Se a condição avaliada for verdadeira, o conjunto de comandos dentro da estrutura de repetição **enquanto** é executado e após esta execução, a condição é novamente avaliada. Se o resultado da avaliação for falso, este conjunto de comandos não será executado e o fluxo do algoritmo segue normalmente.

Estrutura de repetição enquanto - pseudocódigo

Sintaxe da Estrutura de Repetição enquanto

<inicialização da variável de controle>;

enquanto *(<condição>)* ***faça***

<comando_1>;

<comando_2>;

...

<comando_n>;

<atualização da variável de controle>;

fimenquanto*;*

Estrutura de repetição enquanto - *Python*

Sintaxe da Estrutura de Repetição enquanto

<inicialização da variável de controle>;

while *<condição> :*

<comando_1>;

<comando_2>;

...

<comando_n>;

<atualização da variável de controle>;

Estrutura de repetição enquanto - *Python*

```
# Entrada de Dados
contador = 0
# condicional de repetição
while contador < 3:
    # saída de dados
    print("O valor de x é: ", contador)
    # Processamento de Dados
    contador = contador + 1
# saída de dados
print("Saiu do while")
```

- Tabela Verdade

Contador
0

Estrutura de repetição - *Contador*

- O poder das estruturas de repetições é muito interessante, principalmente quando utilizamos condições com mais de uma variável.
- Normalmente utilizamos variáveis para controlar a quantidade de repetições (contadores)
- Imagine um problema em que deveríamos imprimir os números inteiros entre 1 e um valor digitado pelo usuário.

Estrutura de repetição – *Contador Vs Acumulador*

- Nem sempre conseguimos resolver problemas que precisem de repetição usando contadores. Muitas vezes são necessários **acumuladores**.

- A diferença entre um **contador** e um **acumulador** é que nos **contadores o valor adicionado é constante** e, nos **acumuladores, variável.**

- Ex: Um programa que calcula a soma de 10 números.

Estrutura de repetição - *Acumuladores*

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

- Tabela Verdade

N	Soma
1	0

Contadores e Blocos de Decisão

```
pontos = 0
questao = 1
while questao <= 3:
    resposta = str(input(f"Resposta da questão {questao}"))
    if questao == 1 and resposta == "b":
        pontos = pontos + 1
    if questao == 2 and resposta == "a":
        pontos = pontos + 1
    if questao == 3 and resposta == "d":
        pontos = pontos + 1
    questao = questao + 1
print(f"O aluno fez {pontos} ponto(s)")
```

Operações Atribuição Especiais

Operador	Exemplo	Equivalência
+=	x += 1	x = x + 1
-=	x -= 1	y = y - 1
*=	c *= 2	c = c * 2
/=	d /= 2	d = d / 2
**=	e **= 2	e = e ** 2
//=	f //= 4	f = f // 4

Interrompendo a Repetição

- Embora muito útil, a estrutura **while** só verifica a condição de parada no início de cada repetição.

- Dependendo do problema, a habilidade de terminar o **while** dentro do bloco é interessante.

- A instrução **break** serve para esse propósito. Ele encerra o bloco **while** **atual** independente da condição de parada.

Interrompendo a Repetição

```
soma = 0
while True:
    valor = int(input("Digite um número a somar ou 0 para sair:"))
    if valor == 0:
        break # parando o laço
    soma += valor
print(soma)
```

Repetições Aninhadas

- Podemos combinar vários **while** de forma a obter resultados mais interessantes, como a repetição com incremento de duas variáveis.

- Imagine imprimir as tabuadas da multiplicação de 1 a 10.

Repetições Aninhadas

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

Exercício 1 – Estrutura de repetição

1. Desenvolva um programa que recebe números inteiros digitados pelo usuário e calcule a soma entre esses números e a média. Só parar de digitar os números quando o usuário digitar zero. **Utilizar while**.

2. Desenvolva um programa que receba 10 números reais digitados pelo usuário e some somente os números pares. **Utilizar while**.

Exercício 1 – Estrutura de repetição

```
# Exercício 01
# Desenvolva um programa que recebe números inteiros digitados pelo usuário e calcula
# a soma entre esses números e a média. Só parar de digitar os números quando
# o usuário digitar zero. Utilizar while.
# Inicialização das variáveis dados
numero = 5
cont = 0
soma = 0
# condicional do laço
while numero != 0:
    # entrada de dados
    numero = int(input("Digite um número: "))
    # condicional
    if numero != 0:
        # processamento de dados
        soma = soma + numero
        cont = cont + 1
# saiu do laço
media = soma / cont

# saída de dados
print("A soma é igual a %d e a média é igual a %.2f" % (soma, media))
```

- Tabela Verdade

numero	soma	cont
5	0	0

Estrutura de repetição *para* (for)

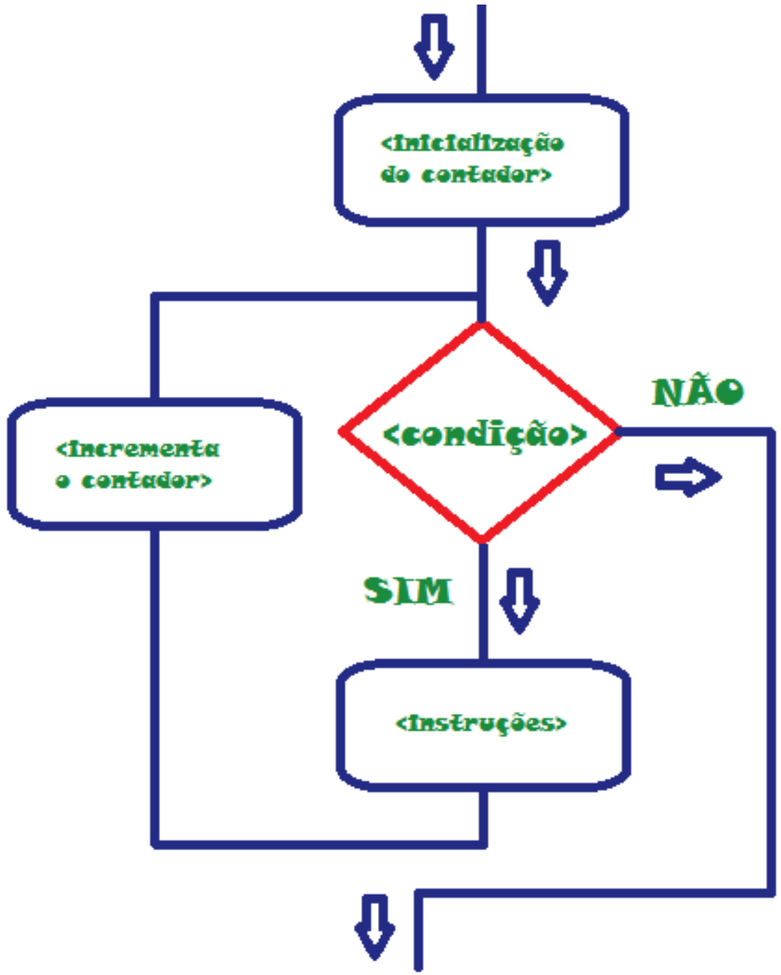
https://explorandoti.com.br/criando-repeticoes-com-for-em-python3/

Estrutura de repetição *para*

- Uma estrutura de repetição **para** pode ser utilizada quando o algoritmo precisa ter definido a quantidade de vezes que um conjunto de comandos deve ser executado
- Neste caso, a variável de controle, sua inicialização e finalização bem como sua atualização fazem parte do cabeçalho da estrutura de repetição **para** e o conjunto de comandos dentro da estrutura de repetição **para** é executado a quantidade de vezes determinado no cabeçalho desta estrutura
- Note que nesta estrutura de repetição, pode ocorrer do conjunto de comandos não ser executado nenhuma vez

Estrutura de repetição *para*

- O laço FOR (para) é utilizado quando a quantidade de repetições desejada é conhecida

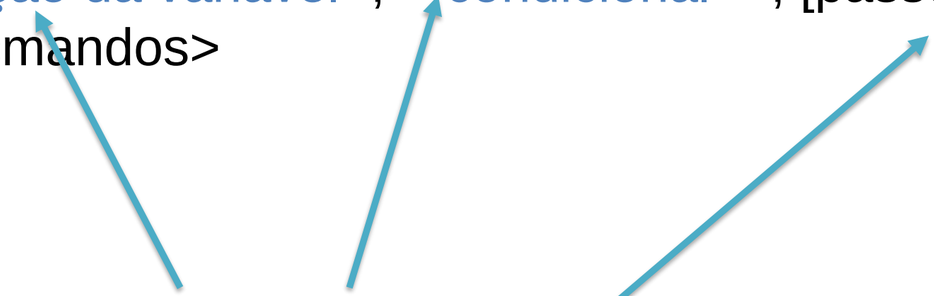


Estrutura de repetição ***para*** - *pseudocódigo*

Sintaxe da estrutura de repetição ***Para***

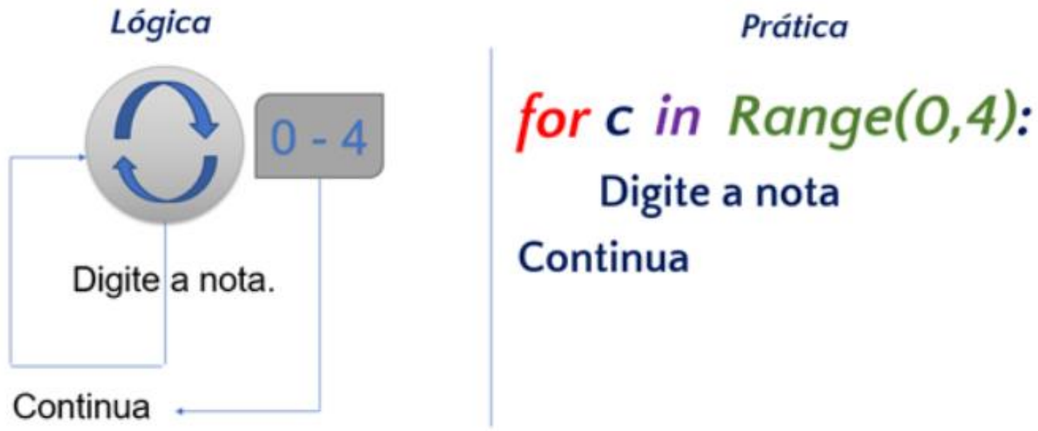
Para <inicialização da variável>; < condicional > ; [passo <incremento>] faça
    <bloco de comandos>

FimPara



```
Para i = 0 de i até 10 passo 1 faça
    Escreva(i)
FimPara
```

Estrutura de repetição ***para*** - *Python*




```
# Exemplo 01 - x vai de 0 a 2
x = 0
for x in range(3):
    print("O valor de x é: ", x)
print("Saiu do laço")
```

```
# Exemplo 02
for x in range(0, 3):
    print("O valor de x é: ", x)
print("Saiu do laço")
```

Estrutura de repetição ***para*** - *Python*

O *loop* for é um iterador de sequência genérico no Python: ele pode percorrer os itens de qualquer objeto sequência ordenada. O *loop* ***for*** funciona em *strings*, *tuplas*, listas e em novos objetos que criaremos posteriormente com classes. O *loop* for do Python começa como uma linha de cabeçalho que especifica um destino (ou destinos) de atribuição, junto com um objeto que você queira percorrer.

O loop `for` também aceita um bloco `else` que é executado, se o loop terminar, ou se ele encontrar uma instrução `break`:

```
for <destino> in <objeto> #Atribui itens do objeto ao destino
     <instruções>
     if <teste>:
          break#Sai do loop, pular clausula
     elseif <teste>:
          continue       #Vai para o início do loop agora
else:<instruções>
     #Se não atingimos uma instrução break
```

Estrutura de repetição ***para*** - *Python*

Else no Loop For

A palavra-chave ***else*** em um loop for especifica um bloco de código a ser executado quando o loop for concluído:

Exemplo:
Imprima todos os números de 0 a 5 e imprima uma mensagem quando o loop terminar:

```
# Exemplo 03
for x in range(0, 6):
     print(x)
else:
     print("Finalmente terminado!")
```

- Tabela Verdade

x
0

Exemplos – Estrutura de repetição

```
# Exemplo 04
for x in range(50, 100):  # x vai de 50 a 99
    if x == 88:
        break  # se x for igual a 88, sai do laço
    print(x)
print("Saiu do laço")
```

Exemplos – Estrutura de repetição

```
# Exemplo 05 – Soma dos números impares
total = 0
numero = int((input("Digite um número: ")))
if (numero % 2) == 0:
    numero = numero - 1

for i in range(numero, 0, 2): # 3 parametro é o incremento ou decremento
    total = total + i
    print("Valor de i ", i)
print("A soma dos número impares é ", total)
```

Exercício 2 – Estrutura de repetição

1. Desenvolva um programa que calcule o quadrado dos números inteiros compreendidos entre 10 e 150. **Utilizar for**.
2. Desenvolva um programa que receba um número inteiro, calcule e mostre o seu fatorial. (Exemplo de Fatorial: se o número 4 for digitado, o programa deverá fazer 1*2*3*4 e mostrar como resultado 24, se o número digitado for 5 o programa deverá fazer 1*2*3*4*5 e mostrar como resultado 120). **Utilizar for**.
3. Desenvolva um programa que recebe um número inteiro e mostra a tabuada desse número.
4. Desenvolva um programa que receba um número inteiro, verifique e mostre se esse número é primo ou não.

Exercício 2 – Estrutura de repetição

```
# Questão 01
# Desenvolva um programa que calcule e o quadrado dos número
s inteiros
# compreendidos entre 10 e 150. Utilizar for.

for i in range(10, 151):
    print("%d ao quadrado = %d" % (i, i*i))
```

Exercício 2 – Estrutura de repetição

```
# Questão 02
# Desenvolva um programa que receba um número inteiro, calcule e
# mostre o seu fatorial. (Exemplo de Fatorial: se o número 4 for digitado,
# o programa deverá fazer 1*2*3*4 e mostrar como resultado 24,
# se o número digitado for 5 o programa deverá fazer 1*2*3*4*5
# e mostrar como resultado 120). Utilizar for.

numero = int(input("Digite um número: "))
result = 1
for i in range(1, numero+1):
    result = result * i
print("Fatorial de ", numero, result)
```

Exercício 2 – Estrutura de repetição

```
# Questão 03
# Desenvolva um programa que recebe um número inteiro e mostra a tabuada
# desse número.
numero = int(input("Digite um número: "))
for i in range(1, 11):
    print("%d x %d = %d" % (numero, i, numero*i))
```

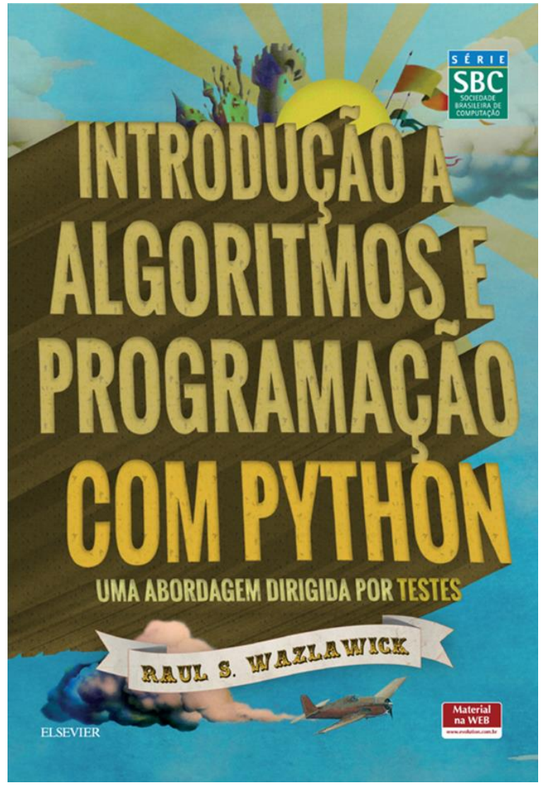
Exercício 2 – Estrutura de repetição

```
# Questão 04
# Desenvolva um programa que receba um número inteiro, verifique e mostre se
# esse número é primo ou não.
numero = int(input("Digite um número: "))
cont = 0
for i in range(2, numero):
    if numero % i == 0:
        cont = cont + 1
        break
if cont == 0:
    print("%d é primo " % numero)
else:
    print("%d não é primo " % numero)
```

Material

- **Livro**
  - WAZLAWICK , Raul. Introdução a Algoritmos e Programação com Python. *LTC, 2017.*

- **Documentação Oficial – Python**
  - https://docs.python.org/pt-br/3/
- **W3schools**
  - https://www.w3schools.com/python/
- IDE Online - Gratuita
  - https://replit.com/languages/python3



Leitura Complementar

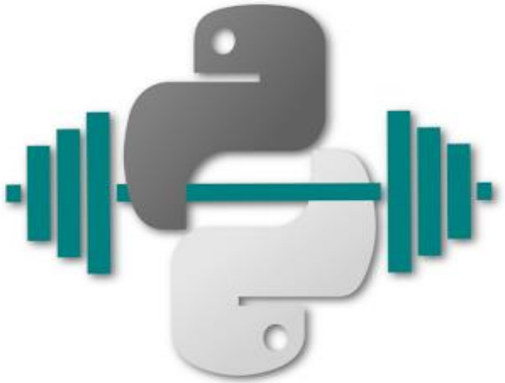
- Instrução **continue**

```
x = 0
while (x <= 10):
    x += 1
    if (x % 2 == 1):
        continue  # continua o laço
    print(x)
```

- http://excript.com/python/instrucao-continue-python.html
- https://docs.python.org/3/reference/simple_stmts.html#continue

Exercícios

Vamos lá!



Dúvidas?