

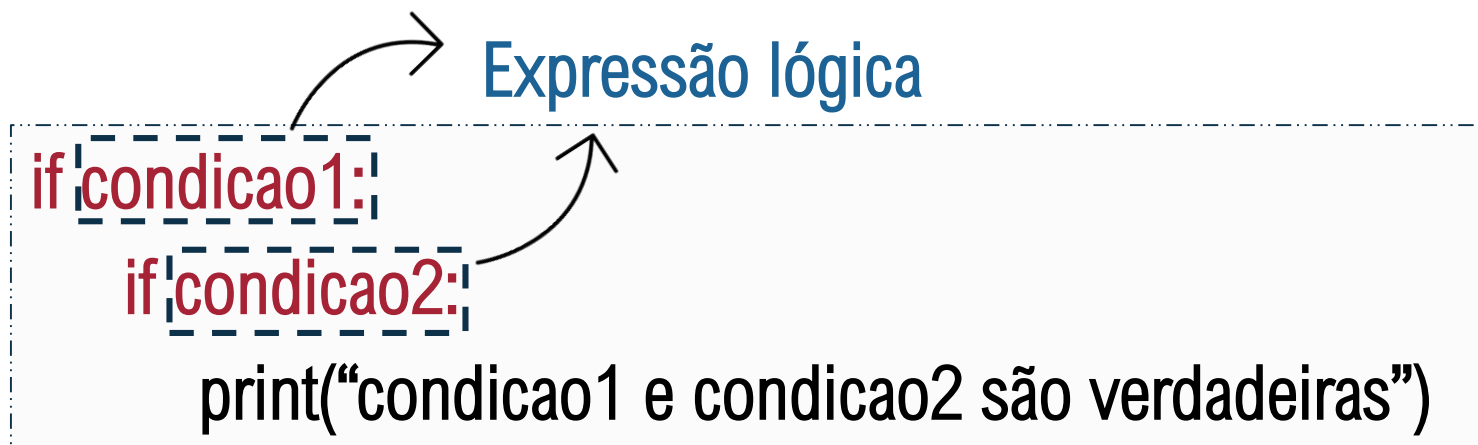
Programação de Computadores

🔗 ESTRUTURAS DE REPETIÇÃO

Na aula anterior...

Estruturas condicionais podem ser postas uma dentro da outra de forma a garantir uma melhor maneira de executar os comandos → **estruturas aninhadas**

Estrutura Condicional aninhada



```
if condicao1:  
    if condicao2:  
        print("condicao1 e condicao2 são verdadeiras")
```

The diagram shows a nested if statement structure. The first line is `if condicao1:` and the second line is `if condicao2:`. Both `condicao1` and `condicao2` are enclosed in dashed blue boxes. Two curved arrows point from these boxes to the text "Expressão lógica" (Logical Expression) written in blue above the code. The third line is `print("condicao1 e condicao2 são verdadeiras")`.



Na aula anterior...

1- Criar um programa em Python que leia a idade de uma pessoa e informe sua classe eleitoral:

- não-eleitor (abaixo de 16 anos)
- eleitor obrigatório (entre 18 e 65 anos)
- eleitor facultativo (entre 16 e 18 anos e maior de 65 anos)

```
main.py
1 idade = int(input("Digite a sua idade: "))
2 if idade < 16:
3     categoria = 'não-eleitor'
4 elif idade >= 18 and idade <= 65:
5     categoria = 'eleitor obrigatório'
6 else:
7     categoria = 'eleitor facultativo'
8 print("Sua classe eleitoral é:", categoria)
```



Conceitos abordados nesta aula

- A proposta desta aula é apresentar as estruturas de repetição (looping).



Exemplo: Tabuada

1- Escreva um algoritmo que solicite ao usuário um número, calcule e mostre a tabuada desse número.



algoritmo

inicio

inteiro numero

escreva("Entre com o número:")

leia(numero)

escreva (numero* 0)

escreva (numero* 1)

escreva (numero* 2)

escreva (numero* 3)

escreva (numero* 4)

escreva (numero* 5)

escreva (numero* 6)

escreva (numero* 7)

escreva (numero* 8)

escreva (numero* 9)

escreva (numero*10)

fim

Estruturas de repetição

- ✓ Também conhecidas como laços (loop, cycle, iteration, ...);
- ✓ Utilizadas para executar repetidamente uma instrução ou um bloco de instruções enquanto determinada condição for verdadeira.
- ✓ São três:



- ✓ para
- ✓ enquanto
- ✓ faça... enquanto



- ✓ for
- ✓ while

Estruturas de repetição

Para determinarmos qual é a estrutura mais adequada, devemos saber:

- ✓ o número de vezes que o trecho programa vai ser executado: laços contados
- ✓ ou a condição para que ela aconteça: laços condicionais

Laços contados: um **contador** irá auxiliar no laço. Neste laço, a repetição da estrutura repete-se até que o contador atinja o limite estipulado na condição.

Laços condicionais: o valor é desconhecido e devemos utilizar uma variável com valor predefinido em uma condição dentro do laço para finalizarmos a repetição.



Estruturas de repetição

Independente do tipo de laço, todos são constituídos de três partes:

- ✓ Inicialização(ões) da(s) variável(is) de controle
- ✓ Condição(ões)
- ✓ Atualização da(s) variável(is) de controle



Estrutura para (for)

É compacta, pois a inicialização, condição e atualização estão reunidos na declaração do laço.

Algoritmo

```
para(inicialização; condição; atualização)
{
    bloco de instruções
}
```



Python

```
for variável in lista:
    bloco de instruções
```

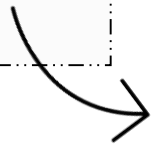
Instruções do bloco devem ser
endentadas corretamente



Comando para repetições na linguagem Python

Python

for variável **in** lista:
 bloco de instruções
 continue (opcional)
 break (opcional)



Repete os comandos para os valores de cada elemento na lista.

- ✓ O comando **continue** pode ser usada para ignorar os comandos e executar a próxima iteração ou passo do laço mais interno.
- ✓ O comando **break** permite sair do ciclo mais interno a qualquer momento.



O comando for na linguagem Python

Diferente de outras linguagens de programação, o comando for de Python itera sobre os itens de uma sequência (uma lista ou uma string), na ordem em que aparecem na sequência. Veja a saída deste programa:

main.py

```
1  # Exemplo simples comando for
2  animais = ['gato', 'cachorro', 'leão', 'camelo']
3  for a in animais:
4      print(a, "string de tamanho:", len(a))
5
6  sequencia = [0, 1, 2, 3, 4, 5]
7  for num in sequencia:
8      print(num)
```

Console

Shell

```
gato string de tamanho: 4
cachorro string de tamanho: 8
leão string de tamanho: 4
camelo string de tamanho: 6
0
1
2
3
4
5
➤ □
```



O comando for na linguagem Python

Diferente de outras linguagens de programação, o comando for de Python itera sobre os itens de uma sequência (uma lista ou uma string), na ordem em que aparecem na sequência. Veja a saída deste programa:

```
nome = 'aline'
for letra in nome:
    print(letra)
```

```
↳ a
   l
   i
   n
   e
```



O comando for na linguagem Python – função range

A função range é útil para iterar sobre sequências numéricas, porque gera progressões aritméticas.

```
for i in range(10):  
    print(i, end=" ")
```

➞ 0 1 2 3 4 5 6 7 8 9

```
for i in range(3, 8):  
    print(i, end=" ")
```

➞ 3 4 5 6 7

```
for i in range(0, 21, 2):  
    print(i, end=" ")
```

➞ 0 2 4 6 8 10 12 14 16 18 20

```
for i in range(5, 20, 3):  
    print(i, end=" ")
```

➞ 5 8 11 14 17



Exemplo: Tabuada

1- Escreva um algoritmo que solicite ao usuário um número, calcule e mostre a tabuada desse número.

```
algoritmo
inicio
    inteiro numero, i
    escreva("Entre com o número:")
    leia(numero)
    para (i=0; i<=10; i++){
        escreva ( numero * i )
    }
fim
```

Teste de mesa:

Por exemplo: numero = 3



i	Saída
0	0
1	3
2	6
3	9
4	12
5	15
6	18
7	21
8	24
9	27
10	30

Exemplo: Tabuada

1- Escreva um programa em Python que solicite ao usuário um número, calcule e mostre a tabuada desse número.

```
main.py
1 num = int(input("Digite um número inteiro: "))
2 for i in range(11):
3     print("%d * %d = %d" %(num, i, num*i))
```



Exemplo

2- Faça um algoritmo que solicite ao usuário 10 números reais, calcule e mostre a soma dos números digitados. Use a estrutura de repetição **for**

```
algoritmo exemplo2
    inicio
        real n, soma
        inteiro i
        soma = 0
        para(i=1; i <= 10; i++){
            escreva("Digite um número real:")
            leia(n)
            soma = soma+número
        }
        escreva("O somatório é:" + soma)
    fim
```

The diagram shows two dashed boxes with arrows pointing to variables in the code. The box labeled 'Acumulador' has an arrow pointing to the variable 'soma'. The box labeled 'Contador' has an arrow pointing to the variable 'i'.

i	n1	soma
		0.0
1	5.5	5.5
2	1.5	7.0
3	2.0	9.0
4	3.5	12.5
5	7.5	20.0
6	2.5	22.5
7	1.0	23.5
8	2.0	25.5
9	4.5	30.0
10	12.5	42.5

Exemplo

2- Faça um programa em Python que solicite ao usuário 10 números reais, calcule e mostre a soma dos números digitados. Use a estrutura de repetição **for**

main.py

```
1 soma = 0
2 for i in range(10):
3     num = float(input("Digite um número real "))
4     soma = soma + num
5
6 print ("O resultado da soma é: ", soma)
```



Estrutura enquanto (while)

Estrutura utilizada tanto para **laços contados** quanto para os **laços condicionais**, possui a seguinte sintaxe:

Algoritmo

```
iniciar a variável de controle  
enquanto (condição for verdadeira)  
{  
    bloco de instruções  
    atualizar a variável de controle  
}
```



Instrução que modifica o estado de algum elemento utilizado na condição

Python

```
iniciar a variável de controle  
while condição:  
    bloco de instruções  
    atualizar variável de controle
```

Instruções do bloco devem ser indentadas corretamente



Comandos para repetições na linguagem Python

Python

while condição:

bloco de instruções
continue (opcional)
break (opcional)

Repete os comandos enquanto a condição for verdadeira.

- ✓ O comando **continue** pode ser usada para ignorar os comandos e executar a próxima iteração ou passo do laço mais interno.
- ✓ O comando **break** permite sair do ciclo mais interno a qualquer momento.



Exemplo

4- Faça um algoritmo que calcula e mostra a média entre duas notas de 10 alunos. Use a estrutura de repetição **enquanto**

```
algoritmo exemplo3
    inicio
        real nota1, nota2, media
        inteiro contador
        contador = 1
        enquanto (contador <= 10) {
            escreva("Digite digite a primeira nota:")
            leia(nota1)
            escreva("Digite digite a segunda nota:")
            leia(nota2)
            media = (nota1 + nota2) / 2
            escreva("A média é:" + media)
            contador=contador +1
        }
    fim
```

Exemplo

4- Faça um algoritmo que calcula e mostra a média entre duas notas de 10 alunos. Use a estrutura de repetição **enquanto**

```
main.py
1 contador = 1
2 while contador <= 10:
3     nota1 = float(input("Digite a primeira nota: "))
4     nota2 = float(input("Digite a segunda nota: "))
5     media = (nota1 + nota2)/2
6     print("A média é %.2f" %media)
7     contador = contador +1
```



Exemplo

5- Faça um algoritmo que calcula e mostra a média de uma quantidade indeterminada de números inteiros digitados pelo usuário. Use a estrutura de repetição **enquanto**.

```
algoritmo exemplo4
inicio
    real num, media, soma
    inteiro contador
    caracter resp
    contador = 0
    soma = 0
    resp = 's'

    enquanto (resp == 's' ou resp == 'S') {
        escreva("Digite digite um número:")
        leia(num)
        soma = soma + num
        contador = contador + 1
        escreva("Deseja continuar (S/N)?")
        leia(resp)
    }
    media = soma/contador
    escreva("A média dos números digitados é:" + media)
fim
```

Contador

Acumulador

Exemplo

5- Faça um algoritmo que calcula e mostra a média de uma quantidade indeterminada de números inteiros digitados pelo usuário. Use a estrutura de repetição **enquanto**.

```
main.py
1 contador = 0
2 soma = 0
3 resp = "s"
4
5 while resp == "s" or resp == "S":
6     num = float(input("Digite um número: "))
7     soma = soma + num
8     contador = contador + 1
9     resp = input("Deseja continuar (S/N)? ")
10
11 media = soma / contador
12 print("A média dos números digitados é %.2f" %media)
```



Desafio

6- Adivinhe meu número: Crie um jogo onde o computador escolhe um número inteiro aleatório entre 0 e 100.

- ✓ Leia a entrada do usuário para tentar acertar o número;
- ✓ Se errar informar ao usuário se o número é maior ou menor;
- ✓ Repetir até o usuário acertar.

Dica: faça a importação da biblioteca random e gere um número aleatório inteiro utilizando a função **randint**(início, fim)

```
main.py
1 from random import *
2 num = randint(0,10)
```



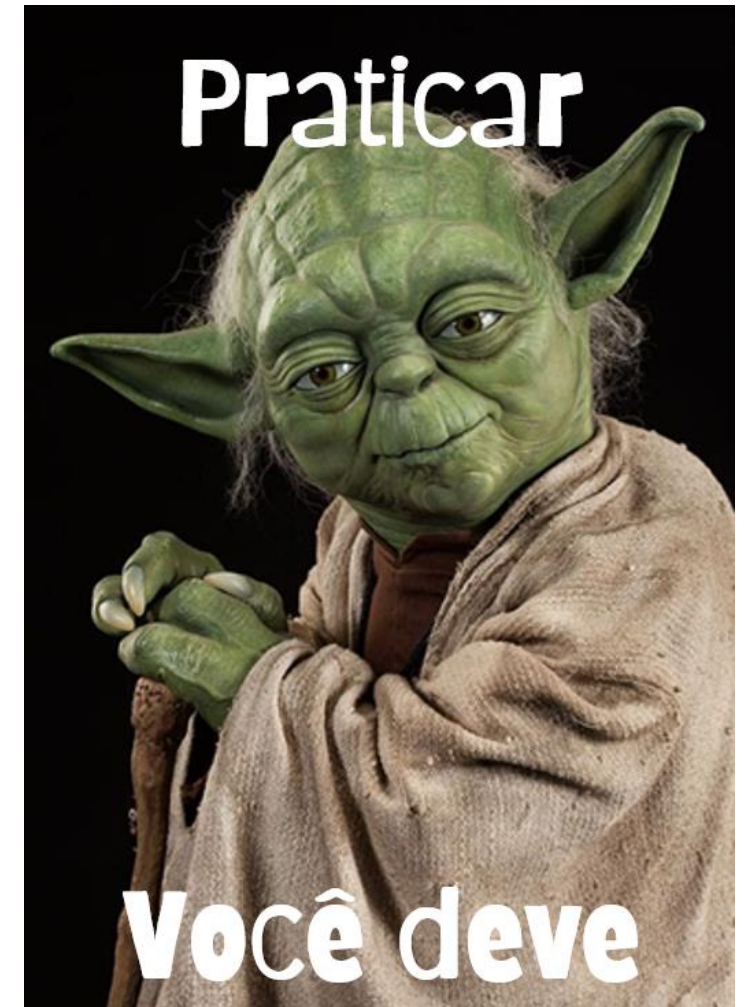
DESAFIO

Exercícios de aplicação



Observações sobre exercícios

- ✓ Todos os exercícios devem ser resolvidos em Python.
- ✓ O código Python pode ser feito no IDLE ou no Repl.it e deve ser salvo um arquivo por exercício com a extensão .py
- ✓ Após finalizar todos os exercícios da aula, compacte os arquivos .py e envie no Blackboard.



Exercícios

- 1- Faça um programa em Python que imprima os números pares entre 0 e 100
- 2- Faça um programa em Python que imprima os números de 1 a 50 de 1 em 1 e de 52 a 100 de 2 em 2.
- 3- Faça um programa em Python que leia um valor n , inteiro e positivo, calcule e mostre a seguinte soma:
$$S = 1 + 1/2 + 1/3 + 1/4 + \dots + 1/n$$
- 4- Escreva um algoritmo que leia um grupo de valores reais e determine quantos valores são positivos e quantos são negativos. Determine, também, qual é o menor desses valores. Utilize o comando de repetição que desejar.
- 5- Temos um grupo de pessoas. Escreva um programa em Python que leia o sexo e a altura de cada pessoa, calcule e mostre a altura média das mulheres e dos homens separadamente. Utilize o comando de repetição que desejar



That's all Folks!