

## Aula 5

### Docupedia Export

Author:Goncalves Donathan (CtP/ETS)  
Date:29-Aug-2023 21:02

## Table of Contents

<b>1 Blocos de instrução</b>	<b>5</b>
<b>2 Loops e condicionais</b>	<b>6</b>
<b>3 Condicionais: if</b>	<b>7</b>
<b>4 Condicionais: elif</b>	<b>8</b>
<b>5 Condicionais: operadores</b>	<b>9</b>
<b>6 Exercício 1 – Número secreto</b>	<b>11</b>
<b>7 Correção</b>	<b>12</b>
<b>8 Exercício 2 - Eleição</b>	<b>13</b>
<b>9 Correção</b>	<b>14</b>
<b>10 Loops: while</b>	<b>15</b>
<b>11 Loops: while true</b>	<b>17</b>
<b>12 Exercício 4 – Calculadora</b>	<b>19</b>
<b>13 Correção</b>	<b>21</b>
<b>14 Loops: for</b>	<b>24</b>
<b>15 Exercício 5 – Números pares</b>	<b>26</b>
<b>16 Correção</b>	<b>27</b>
<b>17 Loops: loops aninhados</b>	<b>28</b>
<b>18 Exercício 6 – Tabuleiro</b>	<b>29</b>
<b>19 Correção:</b>	<b>30</b>
<b>20 Exercício 7 – Série de Fibonacci</b>	<b>31</b>
<b>21 Correção:</b>	<b>32</b>
<b>22 Exercício 8 - Números Primos</b>	<b>33</b>

**23 Correção****34**

```
#bloco 1  
#bloco 1  
#bloco 1  
#bloco 2  
#bloco 2  
#bloco 2  
#bloco 3  
#bloco 3  
#bloco 1  
#bloco 1
```

# 1 Blocos de instrução

Blocos de instrução são conjuntos de instruções que pertencem ao mesmo nível hierárquico.

Dependendo da linguagem a forma de delimitar um bloco pode mudar, em Python os blocos são delimitados a partir da **indentação**.

Indentação significa afastar o texto da margem, apertando a tecla **TAB** você estará indentando uma linha, fazendo com que ela passe de um nível hierárquico mais baixo para um mais alto.

## 2 Loops e condicionais

Agora veremos quais são os principais blocos de instruções do Python.

Existem os blocos **condicionais** e os de **loop**. Funcionam da seguinte maneira:

### Condicionais:

- Uma linha de código realiza a análise de uma condição, se ela for verdadeira o bloco referente à essa linha é executado, ao final do bloco o programa segue para a próxima linha, se for falsa o bloco é ignorado pelo código.

### Loop:

- Uma linha de código realiza a análise de uma condição, se ela for verdadeira o bloco referente à essa linha é executado, mas ao final do bloco o programa volta a analisar a condição, ou seja, enquanto a condição for verdadeira o bloco continuará a ser executado e a condição avaliada.



### 3 Condicionais: if

O condicional IF faz uma análise de uma condição, se essa análise resultar em TRUE, ou seja, se a condição for verdadeira, o bloco de instrução que segue a condição é executado.

```
if a == 0:  
    print("a é igual a zero!")
```

No exemplo acima o programa irá avaliar se a variável “a” tem um valor igual a zero. O bloco desta instrução só tem uma linha, mas que só será executada se essa condição for verdadeira.

Preste atenção na notação! Depois da condição deve ser colocado dois pontos, e na próxima linha o bloco já deve estar identado.

Mas e se a condição resultar em FALSE?

Podemos adicionar a expressão ELSE, fazendo com que se a condição for falsa, um bloco de instrução diferente seja executado.

Assim teremos uma condição que possui duas possibilidades, que são dois blocos de instrução.

```
if a == 0:  
    print("a é igual a zero!")  
else:  
    print("a é diferente de zero!")
```

No exemplo acima a condição da variável ser igual a zero será avaliada, a diferença agora é que também existe uma instrução caso seja falsa.

Preste atenção na notação! O comando **else** deve estar sempre alinhado com o **if**, do contrário você encontrará um erro.

## 4 Condicionais: elif

Também podemos utilizar a função ELIF para atribuir condições.

Ele funciona da seguinte forma:

- Primeiramente uma condição deve ser avaliada por um IF;
- Se for verdadeira o bloco do IF é executado e o programa segue o seu fluxo;
- Se for falsa, através do ELIF o programa avaliará outra condição;
- Assim podem ser avaliadas várias condições, gerando um efeito cascata;
- Por fim, se nenhuma das condições forem atendidas, pode ser usado um ELSE para as exceções.

```
if a == 0:  
    print("a é igual a zero!")  
elif a == 1:  
    print("a é igual a um!")  
else:  
    print("a é diferente de zero e de um!")
```

## 5 Condicionais: operadores

Em vários casos, precisamos avaliar mais de uma condição ao mesmo tempo, ou uma condição negada, para isso podemos utilizar os operadores AND, OR, NOT.

**Operador AND:** Dados dois valores booleanos A e B, o operador lógico **and** resulta em True apenas quando A e B forem ambos True, e retorna False caso contrário.

**Operador OR:** Dados dois valores booleanos A e B, o operador lógico **or** resulta em False apenas quando A e B forem ambos False, e retorna True caso contrário.

**Operador NOT:** O operador lógico **not** muda o valor de seu argumento, ou seja, not True é False, e not False é True.

### AND

```
if a == 0 and b == 2:  
    print("a é igual a zero!")  
    print("b é igual a dois!")
```

### OR

```
if a == 0 or b == 2:  
    print("a é igual a zero!")  
    print("ou b é igual a dois!")
```

### NOT

```
if not a == 0:  
    print("a não é igual a zero!")
```

**JOGO DA ADIVINHAÇÃO**

Digite seu palpite: 41  
Você Errou!

**JOGO DA ADIVINHAÇÃO**

Digite seu palpite: 42  
Você Acertou!



## 6 Exercício 1 – Número secreto

- Crie uma variável do número secreto = 42;
- Diga para o usuário dar um palpite;
- Diga se ele acertou ou errou o número secreto;
- Certifique-se que não ocorrerá um erro no programa se o usuário inserir uma string (Ex: 'a', 'b', etc.);
- DICA: Utilize a função `.isdigit()`;

## 7 Correção

```
7 print("JOGO DA ADIVINHAÇÃO")
8 print ("-*30)
9 tentativas = 3
10 numero_secreto = 42
11
12 palpite = int(input("Digite o seu palpite: "))
13 if palpate == numero_secreto:
14     print ("Você Acertou!")
15 else:
16     print ("Você Errou!")
```

## 8 Exercício 2 - Eleição

- Crie um script que o usuário pode entrar com a seu ano de nascimento;
- Printa a idade do usuário;
- Printa se o usuário é obrigado a votar, se é facultativo, ou se não é permitido votar:
  - de 0 a 16 anos não é permitido;
  - De 16 a 18 é facultativo;
  - De 18 à 70 é obrigatório;
  - De 70 para cima é facultativo.

Output:

Qual ano você nasceu? 1900  
Seu voto é facultativo!

Qual ano você nasceu? 2000  
Você TEM que votar!

Qual ano você nasceu? 2003  
Seu voto é facultativo!

## 9 Correção

```
8 print("-----ELEIÇÃO-----")
9 nascimento=int(input("Qual ano você nasceu? "))
10 idade = 2019 - nascimento
11
12 if idade < 16:
13     print("\nVocê não pode votar!")
14 elif idade > 18 and idade < 70:
15     print("\nVocê TEM que votar!")
16 elif idade > 16 and idade <18:
17     print("\nSeu voto é facultativo!")
18 else:
19     print("\nSeu voto é facultativo!")
```

## 10 Loops: while

O comando **while** faz com que um conjunto de instruções seja executado enquanto uma condição é atendida. Quando o resultado dessa condição passa a ser falso, a execução do loop é interrompida.

Podemos adicionar um valor a uma variável ao fim de cada ciclo do loop, podendo criar um loop com uma quantidade prevista de ciclos. O nome disso é **iteração**.

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

0  
1  
2  
3  
4  
5

### Exercício 3 – Soma de números

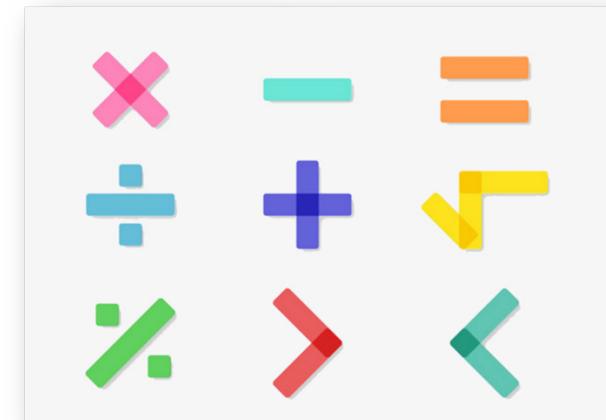
Crie um programa para somar todos os números naturais de 1 à escolha do usuário (limite)

**Exemplo:**

A soma dos números de 1 à 5 é:

$$1 + 2 + 3 + 4 + 5 = 15$$

```
Digite qual o limite: 5  
O valor total da soma é: 15
```



## Correção

```
7 limite = int(input("Digite qual o limite: "))
8
9 soma = 0
10 contador = 1
11
12 while contador <= limite:
13     soma = soma + contador
14     contador = contador + 1 # atualiza o contador
15
16 print("O valor total da soma é: ", soma)
```

## 11 Loops: while true

Quando queremos criar um loop sem uma quantidade de ciclos prevista, é usual utilizar **while: True**, esse comando inicia um loop infinito, que só pode ser quebrado pelo comando **break**.

Essa técnica é bem útil para criar programas que só acabam quando o usuário quiser.

```
while True:  
    print('python')  
    if input("Deseja parar? ") == 'sim' :  
        break
```

python

Deseja parar? não  
python

Deseja parar? não  
python

Deseja parar? sim

## 12 Exercício 4 – Calculadora

Crie um programa de uma calculadora;  
Ela deve fazer soma, subtração, divisão, e  
multiplicação entre dois números;  
O usuário deve escolher quando sair do  
programa;  
Dica: utilize if e while.

```
Digite o primeiro numero: 5
```

```
Digite o segundo numero: 2
```

```
Digite:
```

- 1 - Soma
- 2 - Subtração
- 3 - Multiplicação
- 4 - Divisão
- 0 - Sair

```
3
```

```
O resultado é: 10.0
```

```
Digite 0 se quiser sair, para continuar digite qualquer valor
```

```
2
```

```
Digite o primeiro numero: 6
```

```
Digite o segundo numero: 3
```

```
Digite:
```

- 1 - Soma
- 2 - Subtração
- 3 - Multiplicação
- 4 - Divisão
- 0 - Sair

```
4
```

```
O resultado é: 2.0
```

Digite 0 se quiser sair, para continuar digite qualquer valor  
0

## 13 Correção

```
8 print("Calculadora")
9 menu = 1
10
11 while(menu!=0):
12     oper1 = float(input("Digite o primeiro número: "))
13     oper2 = float(input("Digite o segundo número: "))
14     print("\nDigite o número que corresponde a operação ")
15     print("1 - Soma \n2 - Subtração \n3 - Multiplicação \n4 - Divisão \n0 - Sair")
16     calc = int(input(""))
17     resultado=0
18
19     if(calc==1):
20         resultado=oper1+oper2
21     elif(calc==2):
22         resultado=oper1-oper2
23     elif(calc==3):
24         resultado=oper1*oper2
25     elif(calc==4):
26         resultado=oper1/oper2
27     elif(calc==0):
28         menu=0
29     else:
30         print("Operação Inválida\n")
31
32     print("\n0 Resultado é: %.2f \n"%(resultado))
33     menu=int(input("Digite 0 caso desejar sair ou qualquer outro número para continuar\n"))
```



## 14 Loops: for

O comando **for** também é uma estrutura de loop, a sua diferença para o **while** é que a iteração da variável já é inclusa no próprio comando.

Ao fim de cada ciclo de loop a variável de controle é adicionada de um.

Utilizando a função **range** podemos criar um loop com a quantidade de ciclos que desejarmos.

```
for i in range(10):  
    print(i)
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

Também podemos usar o **for** para transitar sobre todos os elementos de uma estrutura de dados, usando o operador **in**, assim a variável de controle recebe o valor de um item diferente a cada ciclo executado.

```
lista = ['item_a','item_b','item_c','item_d']
for i in lista:
    print(i)
```

item\_a  
item\_b  
item\_c  
item\_d

## 15 Exercício 5 – Números pares

- Crie um programa que mostra todos os números pares de 0 a 30;
- DICAS: o resto da divisão de um número par por 2 é igual a 0;



OUTPUT:

```
0  
2  
4  
6  
8  
10  
12  
14  
16  
18  
20  
22  
24  
26  
28  
30
```

## 16 Correção

```
for i in range(31):
    if i%2 == 0:
        print(i)
```

## 17 Loops: loops aninhados

Podemos criar um loop dentro de outro loop, isso se chama **Loops Aninhados**, e tem múltiplas utilidades. Vamos ver um exemplo:

```
for i in range(5):
    for j in range(5):
        print('#',end = ' ')
print('@',end = ' ')
```

```
#####@#####@#####@#####@#####@
```

Em toda repetição do loop externo, o loop interno executa todas as suas repetições, ou seja, no final temos a execução externa realizada 5 vezes, e a interna é executada  $5 \times 5 = 25$  vezes!

Loops aninhados são muito úteis para fazer varreduras em listas, trabalhar com tabelas, matrizes, etc.

## 18 Exercício 6 – Tabuleiro

- Faça um programa em Python que solicite um número positivo inteiro ao usuário, e depois exiba um tabuleiro na tela.
- O tabuleiro deve ter número de linhas e colunas igual ao valor passado pelo usuário.

Exemplo:

- Se digitar 4:

```
x x x x  
x x x x  
x x x x  
x x x x
```



## 19 Correção:

```
num = int(input("Digite o tamanho do tabuleiro: "))
for i in range(num):
    for j in range(num):
        print('x',end=' ')
    print('\n',end='')
```

## 20 Exercício 7 – Série de Fibonacci

- Peça para o usuário escolher um número;
- O número que o usuário escolher será a quantidade de fatores da série de Fibonacci;
- Faça um print da série de Fibonacci;
- A série de Fibonacci é uma sequência que começa com 0, 1, e o próximo termo é sempre a soma dos dois últimos termos;
- Ex: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584...;



OUTPUT:

```
Digite o número de fatores : 10
0
1
1
2
3
5
8
13
21
34
```

## 21 Correção:

```
7 print ("Série Finonacci")
8 print ("-**30)
9
10 while True:
11
12     n=int(input("Qual a quantidade de itens deseja verificar? "))
13
14     anterior=0
15     proximo=1
16     soma=0
17     list=[]
18
19     for i in range(0,n):
20         list.append(anterior)
21         soma = proximo + anterior
22         anterior = proximo
23         proximo = soma
24
25     print (list)
```

## 22 Exercício 8 - Números Primos

- Peça para o usuário dizer um número;
  - O programa deve informar se o número escolhido é primo ou não;
- DICA 1: números primos somente são divisíveis por 1 e por eles mesmos;  
DICA 2: utilize a função %;



OUTPUT:

```
Digite um número: 2  
[1, 2]  
O número 2 é primo.
```

```
Digite um número: 10  
[1, 2, 5, 10]  
O número 10 não é primo.
```

## 23 Correção

```
8 print ("Números Primos")
9 print ("-"*30)
10
11 while True:
12     lista=[]
13     numero=int(input("Digite um número: "))
14     for x in range(1, numero+1):
15         if numero%x==0:
16             lista.append(x)
17
18     print(lista)
19     if len(lista)==2:
20         print("O número {} é primo.".format(numero))
21     else:
22         print("O número {} não é primo.".format(numero))
```