CAPITULO - 03 PYTHON

EXECUÇÃO CONDICIONAL

EXPRESSÃO BOOLEANA
→ Verdadeira ou falsa
→ são valores especiais que pertencem à CLASSE BOOL, não são STRINGS

x != y	x diferente de y
x > y	x maior que y
x < y	x menor que y
x >= y	x maior ou igual a y
x <= y	x menor ou igual a y
x is y	x é o mesmo que y
x is not y	x não é y

OBSERVAÇÃO:

OPERADOR DE ATRIBUIÇÃO : =

OPERADOR DE COMPARAÇÃO : ==

OPERADORES LÓGICOS

AND
OR
NOT

→ QUALQUER NÚMERO DIFERENTE DE ZERO SIGNIFICA "TRUE".

```
>>> 17 AND TRUE
>>> TRUE
```

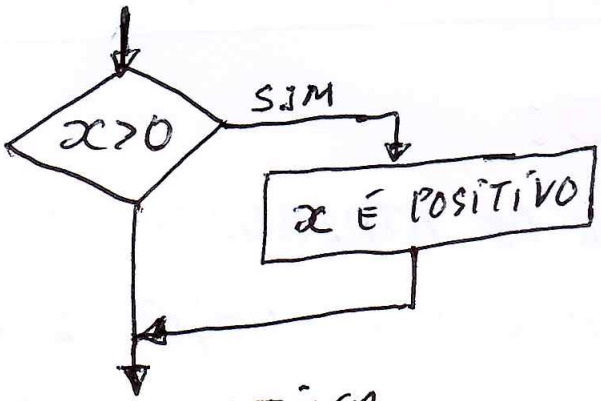
→ OPERANDOS DE EXPRESSÕES LÓGICAS DEVEM SER EXPRESSÕES BOOLEANAS, MAS O PYTHON NÃO É RÍGIDO QUANTO A ISTO.

```
>>> 0 AND TRUE
>>> 0
```

EXECUÇÃO CONDICIONAL

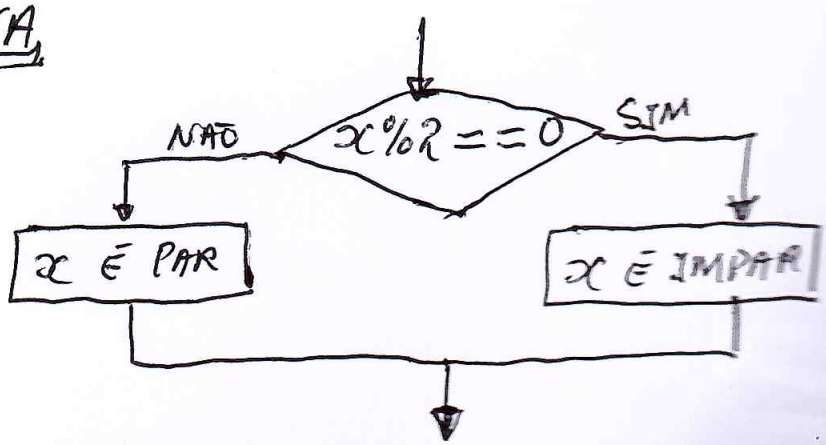
→ VERIFICAR CONDIÇÕES E EM FUNÇÃO DAS CONDIÇÕES MODIFICAR O COMPORTAMENTO DO PROGRAMA:

```
if x>0:
    print('x É POSITIVO')
```



EXECUÇÃO ALTERNATIVA

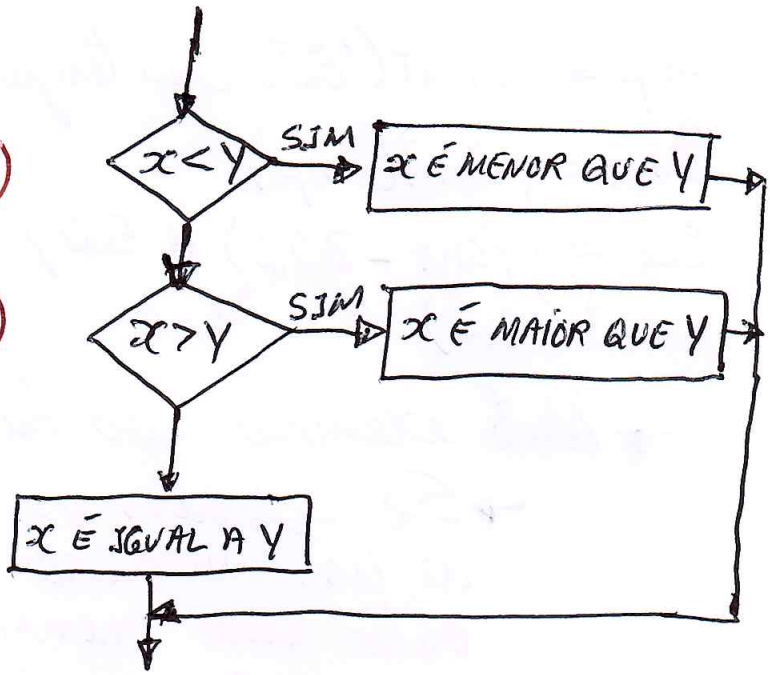
```
if x%2==0:
    print('x É PAR')
else
    print('x É IMPAR')
```



CAPÍTULO – 03

CONDIÇÕES ENCADEADAS

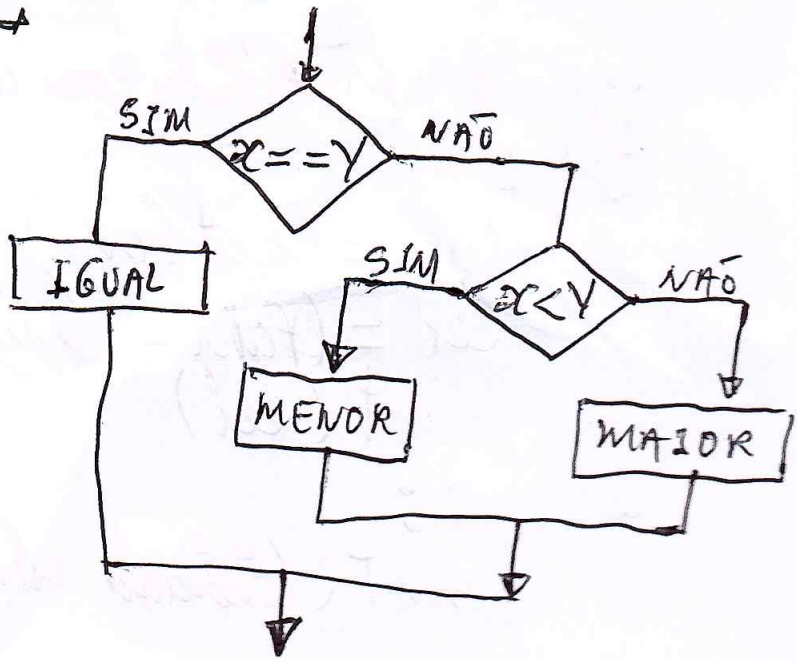
```
if x< y:
    print ('x É MENOR QUE y')
elif x>y:
    print('x É MAIOR QUE y')
else:
    print('x É IGUAL A y')
```



→ NÃO HÁ LIMITES PARA OS COMANDOS "elif"
→ CLÁUSULA "else", se existir ficará sempre no final.
→ As condições são verificadas em ordem até que um resultado TRUE seja corrigido.
- Se mais de uma condição for TRUE somente a primeira é verificada.

CONDIÇÕES ANINHADAS

```
if x==y:
    print('É IGUAL')
else:
    if x<y:
        print('MENOR')
    else:
        print('MAIOR')
```



OBSERVAÇÃO: EVITAR SE POSSÍVEL

TRATANDO CONDIÇÕES DE EXCEÇÃO


```
inp = input('Entra com temperatura em FAHRENHEIT')
Fahr = Float(inp)
Cel = (Fahr - 32.0) * 5.0/9.0
print(Cel)
```

→ Neste exemplo não há tratamento da entrada.

→ Se colocarmos uma "STRING" ao invés de um número como entrada, iremos obter uma mensagem de erro, mais ou menos como esta a seguir:

```
ValueError: could not convert string
to float: 'Brasil'
```

→ Neste caso particular da mensagem ao invés de um número foi informado como temperatura Fahrenheit, a palavra 'Brasil'

→ ESTRUTURA DE TRATAMENTO DE EXCEÇÃO

TRY / EXCEPT

```
inp = input('Entra com a temperatura em Fahrenheit')
try:
    Fahr = Float(inp)
    Cel = (Fahr - 32.0) * 5.0/9.0
    print(Cel)
except:
    print('Entrada deve ser numérica')
```

CIRCUITO CURTO DE AVALIAÇÃO DAS EXPRESSÕES LÓGICAS

SHORT-CIRCUIT

→ SEJA A EXPRESSÃO: `x>=2 AND (x/y)>2`

→ A LÓGICA AND determina que para que a EXPRESSÃO completa seja VERDADEIRA, ambas as expressões componentes devem ser verdadeiras.

→ Ou seja:

`(x>=2 AND (x/y)>2)`

- `x>=2`: TEM QUE SER VERDADEIRA
- `(x/y)>2`: TEM QUE SER VERDADEIRA

PARA QUE A EXPRESSÃO COMPLETA SEJA VERDADEIRA

→ O PYTHON faz o processamento da expressão lógica da ESQUERDA para a DIREITA, quando ele detectar que a primeira parte da expressão é FALSA, não precisará verificar a segunda parte da expressão

OBSERVAÇÃO:

```
>>> x=1
>>> y=0
>>> x>=2 AND (x/y)>2
```

→ Não haverá falha na execução da EXPRESSÃO `x>=2` é FALSO

→ Restante da expressão não será verificado

(PERFIL GUARDIÃO / GUARDIAN PATTERN)

```
>>> x=1
>>> y=0
>>> x>=2 AND (x/y)>2
```

→ Será gerado o 'RUNTIME ERROR', pois o PYTHON necessitará avaliar a segunda expressão que contém uma divisão por ZERO.