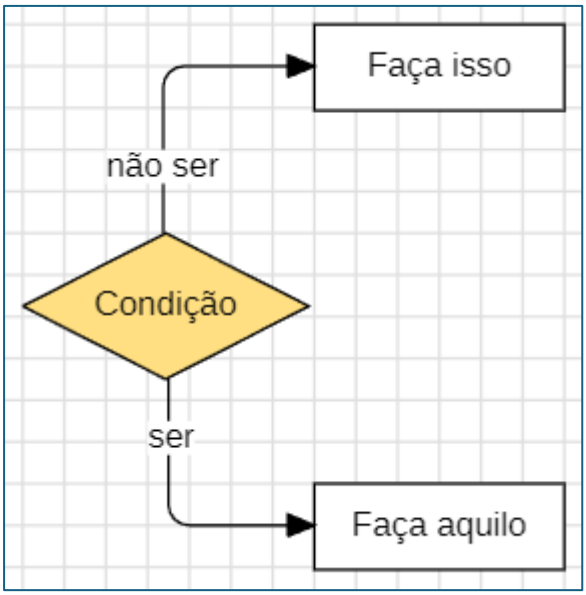
COMPUTAÇÃO 1 – AULA 4

Estrutura Condicional

Prof. Cesar Raitz

1. Introdução

- A frase de Hamlet: *"Ser ou não ser – eis a questão"*, denota uma escolha que o personagem precisa fazer.
- Obviamente, ele precisa decidir para que a história avance.
- Podemos expressar sua escolha através de um **fluxograma**:



- A decisão de Hamlet depende de uma ou mais **condições**.
- Agora veremos como fazer escolhas em seus programas.

2. Estrutura condicional

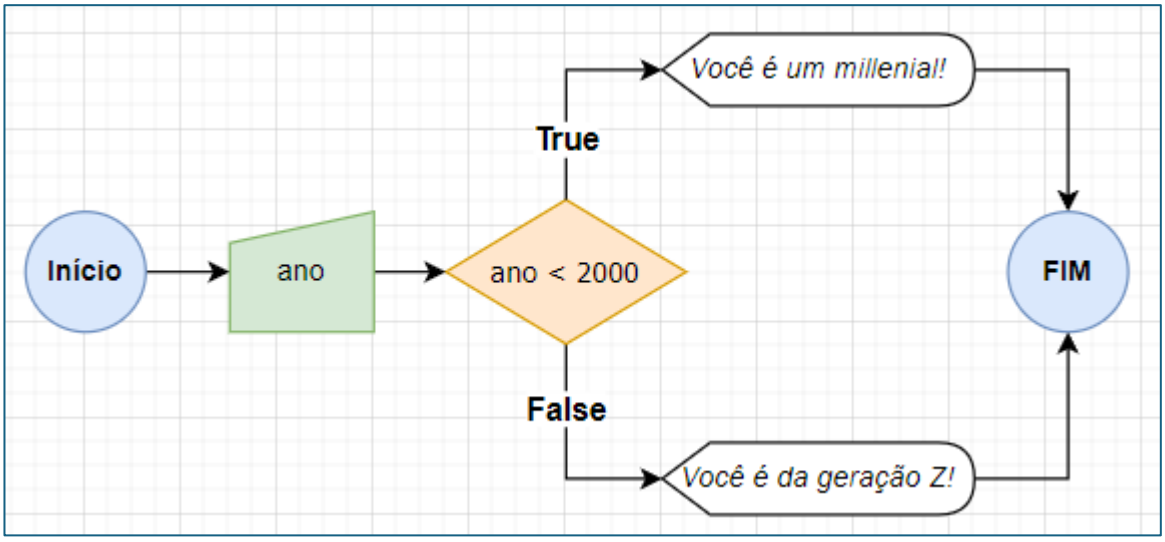
- Pra começar, vejamos um exemplo bem simples:

```
1. ano = int(input("Em que ano você nasceu? "))
2. if ano < 2000:
3.   print("Você é da geração Z!")
4. else:
5.   print("Você é um millenial!")
```

- A mensagem que será mostrada no Terminal depende da **condição** `ano < 2000`
    - Se `ano < 2000` é **verdade**, a instrução abaixo do **`if`** é lida
    - Se `ano < 2000` é **falso**, a instrução abaixo do **`else`** é lida
- Mas como pode `ano < 2000` ser verdade ou falso?
- Porque `ano` é uma **variável**. Cada vez que você roda o programa, a resposta ao **`input`** pode ser diferente, então *a condição pode ser verdadeira ou falsa*.
- Se você respondeu:
    - 2005: então 2005 < 2000 é **falso**, aparece *"Você é um millenial!"*
    - 1998: 1998 < 2000 é **verdade**, aparece *"Você é da geração Z!"*
    - 2000: 2000 < 2000 é **falso**, aparece *"Você é um millenial!"*

- Você compreendeu que o resultado da condição é *sempre* **verdade** ou **falso**?
- Em Python, o resultado de comparações é **`True`** ou **`False`**, ou seja, uma variável do tipo **`bool`**.

- Vou representar o programa acima por um fluxograma ok? Porque costuma deixar mais claro a ideia da **estrutura condicional**:



- E agora vamos abstrair, ou seja, falar da sintaxe da estrutura:

```
<instruções antes da estrutura>
if <condição>:
  <intruções para condição True>
else:
  < intruções para condição False>
<instruções depois da estrutura>
```

- Não esqueça de deixar uma **margem de dois espaços** em cada instrução do **`if/else`**, chamada de **indentação**. Assim, o Python sabe qual instrução está associada ao **`if`**, e qual está associada ao **`else`**.
- Normalmente, após pressionar `Enter` depois de `:`, o editor de código já coloca a indentação pra você. Se precisa, você pode apertar a tecla `Tab` para indentar.

3. Comparações

- A condição do `if` é resultado de uma **comparação**.
- A maioria dos operadores matemáticos funciona:

Símbolo	Python
$>$	`>`
$\geq$	`>=`
$<$	`<`
$\leq$	`<=`
$=$	`==`
$\neq$	`!=`

⚠ Note que **maior ou igual** é `>=` e não `=>`

⚠ Note que **comparação de igualdade** é `==`, para diferenciar de **atribuição** `=`.

Exercício 1. Verificando comparações.

Dadas as variáveis `a=4, b=10, c=5.0, d=1` e `f=5`. Qual é o resultado das seguintes expressões?

`a == c`	**False**
`a < b`	**True**
`d > b`	
`c != f`	
`a == b`	
`c < d`	
`b > a`	
`c >= f`	

`f >= c`	
`c <= c`	
`c <= f`	
`d == d`	
`c-d > a`	
`2*c == d`	
`abs(b) == d`	
`c == max(c,f)`	

- Aproveitando a conversa, *é possível verificar se duas strings são iguais*, mas para quê?

```
1. filme = input("Qual filme você assistiu? ")
2. if filme == "John Wick":
3.   print("Esse filme é muito bom!")
```

Mas cuidado! Pro computador, *maiúsculas e minúsculas não são a mesma coisa!* Portanto:

- `"john wick"` == `"John Wick"` equivale a **`False`**

Até um pequeno espaço extra pode atrapalhar sua comparação:

- `"Jonh  Wick"` == `"John Wick"` também é **`False`**

- É possível guardar o resultado de uma comparação numa variável, pode ser útil!

```
1. dinheiro = float(input("Quanto R$ você possui? "))
2. ryco = dinheiro > 1000
3. if ryco:
4.   print("Tá cheio de dinheiro hem!")
```

4. Operadores Booleanos

- Em algum momento, a condição exigirá mais de uma comparação.
- Por exemplo, o problema de decidir se uma variável x está dentro de um intervalo específico:

$$x \in [10, 100) \ ?$$

- Traduzindo par português:
  x é maior ou igual a 10 **e** x é menor que 100?

- Repare bem: temos duas comparações e um conectivo **e**.
- Agora traduzindo para *Pythonês*:
  `x >= 10 and x < 100`
- **and** é chamado de **operador Booleano** e
  resulta em `True` se os dois lados forem `True`.
  Por exemplo:
  - x = 5: `False and True = False`
  - x = 10: `True and True = True`
  - x = 100: `True and False = False`

- Também temos o operador **or**, que equivale ao conectivo **ou**
  e resulta em `True` quando um dos lados for `True`.
- Vamos verificar se o número y está fora do intervalo: $y \notin [10, 100)$
- A condição é: *y* é menor que 10 **ou** *y* é maior ou igual a 100
  então `y < 10 or y >= 100`.

```
1. y = float(input("y = "))
2. if y < 10 or y >= 100:
3.   print("y está fora do intervalo [10, 100)")
```

- Finalmente, temo o operador de negação **not** que é bem simples:
  - `not True  = False`
  - `not False = True`

```
1. dinheiro = float(input("Quanto R$ você possui? ")
2. ryco = dinheiro > 1000
3. if not ryco:
4.   print("Poxa, quer um dinheiro emprestado?")
5. else:
6.   print("Tá cheio de dinheiro hem!")
```

- Faça um passo-a-passo para com diferentes respostas do usuário.

Exercício 2. Verificando comparações.

Construa as *tabelas de verdade* para os operadores booleanos **and**, **or** e **not**.

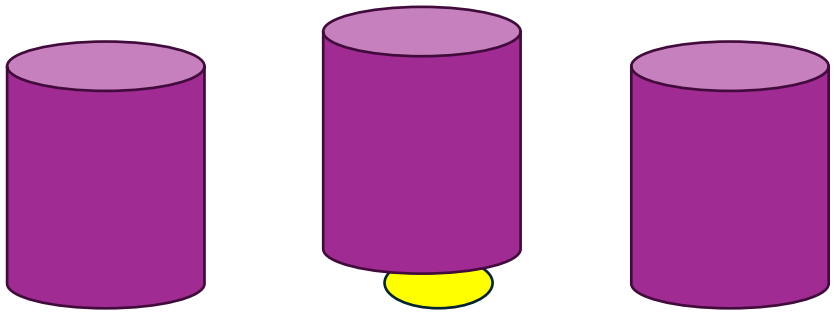
Exercício 3. Vou ao cinema?

O que aparece no Terminal, para o seguinte programa:

```
 1. dinheiro = 30.00
 2. filme = "Oppenheimer"
 3. tenho_carro = True
 4.
 5. print("Será que vou ao cinema?")
 6. if filme == "Barbie" and dinheiro > 20.00:
 7.   print("Acho que vou!")
 8.
 9. if dinheiro > 50.00 or tenho_carro:
10.   print("Estou na dúvida...")
11.
12. if filme == "John Wick" and dinheiro > 40.00 or
        tenho_carro:
13.   print("Vou com certeza!")
```

Exercício 4. Jogo dos copos.

Três copos virados para baixo, um deles esconde uma moeda de ouro. Qual seria a expressão correta para saber se a moeda está sob um dos copos?



`copo1=False` `copo2=True` `copo3=False`

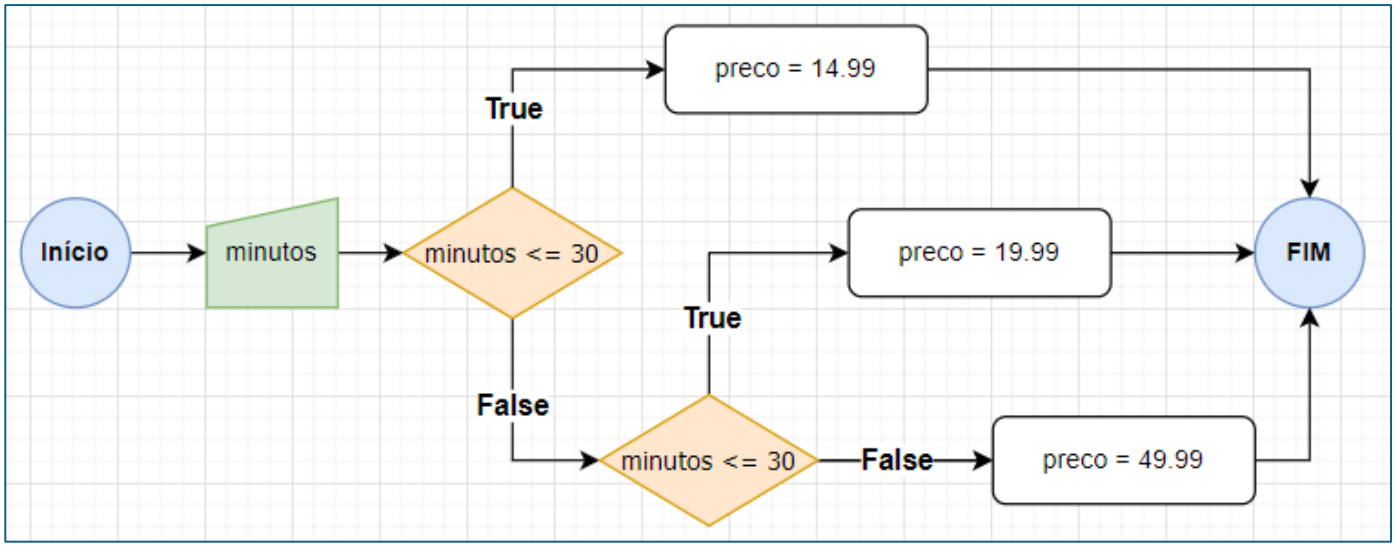
a) `copo1 or copo2 and copo3`
b) `copo1 and copo2 and copo3`
c) `copo1 and copo2 or copo3`
d) `copo1 or copo2 or copo3`

5. Múltiplas decisões

- Imagine uma operadora de telefonia que oferece vários planos:

Ligações	Preço
Até 30 min	R$ 14,99
Até 60 min	R$ 19,99
Ilimitado	R$ 49,99

- Vamos imaginar um fluxograma que permita decidir o preço do plano em função dos minutos de ligação desejados.



- Sim, teremos uma estrutura condicional dentro de outra:

```
1. minutos = int(input("Quantos minutos? ")
2. if minutos <= 30:      # primeira condição
3.   preco = 14.99
4. else:
5.   if minutos <= 60:    # segunda condição
6.     preco = 19.99
7.   else:                # última opção
8.     preco = 49.99
```

- Note que a instrução dentro do segundo **`if`** tem indentação adicional! Mesma coisa para o segundo **`else`**.
- É possível simplificar essa estrutura?
  Podemos juntar **`else`** com **`if`** formando um **`elif`**

```
1. minutos = int(input("Quantos minutos? ")
2. if minutos <= 30:      # primeira condição
3.   preco = 14.99
4. elif minutos <= 60:    # segunda condição
5.   preco = 19.99
6. else:                  # última opção
7.   preco = 49.99
```

- Retiramos a indentação adicional ⇒ ficou mais legível! 😎
- Podemos encadear tantos **`elif`** quanto forem necessários.

6. O operador in

- O problema é o seguinte: Como saber se uma string contém outra? Por exemplo: "tigre" está contida em "Um tigre, dois tigres, três tigres"?
- Uma simples solução: usamos a função **`str.find()`** – ela retorna -1 caso a string não tenha sido encontrada. Veja o programa:

```
1. oque = "gato"
2. onde = "Um tigre, dois tigres, três tigres"
3. if str.find(onde, oque) != -1:
4.   print(f"Encontrei!")
5. else:
6.   print(f"Não encontrei {oque} em: {onde}!")
```

- Também podemos usar **`str.count()`** – mas essa deixa seu programa um tiquinho mais lento:

```
1. oque = "tigres"
2. onde = "Um tigre, dois tigres, três tigres"
3. if str.count(onde, oque) > 0:
4.   print(f"Encontrei!")
5. else:
6.   print(f"Não encontrei {oque} em: {onde}!")
```

- Agora, uma alternativa muito melhor é o operador **`in`**.
- Basicamente, se você quer saber se a string a está contida em b, escreve: `a in b` – o resultado será sempre **`True`** ou **`False`**.
- Veja como o código fica mais limpo: 😎

```
1. oque = "tigres"
2. onde = "Um tigre, dois tigres, três tigres"
3. if oque in onde:
4.   print(f"Encontrei!")
5. else:
6.   print(f"Não encontrei {oque} em: {onde}!")
```