Resumo – Controle de Fluxo

Condicionais: `if/else/elif`

> 💡 O Python utiliza as palavras `if/elif/else` para executar instruções caso uma expressão seja `verdadeira` (ou seja, `True`)

Exemplo:

```
idade = int(input("Qual a sua idade?"))  # Aqui estamos transformando o input em int
if idade > 18:  # Se idade é maior que 18, execute o código "dentro" do "if"
    print("Você é um adulto(a)")
elif idade >= 12:  # Se idade não é maior que 18 mas é maior ou igual a 12...
    print("Você é um(a) adolescente!")
elif idade >= 4:  # Podemos utilizar quantos elif's quisermos
    print("Você é uma criança!")
else:  # Nenhuma das condições acima foi considerada True
    print("Você é um bebê.")
```

Algumas linguagens utilizam `{ }` para identificar o **bloco de código** dentro de um `if/else` que tem que ser executado. Por exemplo, em *JavaScript*:

```
if (idade > 18) {
    console.log(idade);
} else {
    console.log("Só falo com maiores de idade.");
}
```

No caso, Python utiliza a identificação por **indentação,** logo, a quantidade de "espaços" antes de uma linha de código importa:

```
if (10 % 5 == 0):
print("É divísivel")
```

Vai dar um erro se você tentar executar, dizendo que "espera-se um bloco indentado", ou seja, com tabulação/espaçamento para identificar o bloco de código a ser executado caso a expressão `(10 % 5 == 0 )` seja verdadeira.

Repetição: `while`

💡 Chamamos de **iteração** cada "passada" no trecho de código dentro de um `while`. No exemplo anterior, quantas iterações serão realizadas?

💡 **loop infinito**: quando o programa fica infinitamente executando um conjunto de instruções. A causa mais comum é quando a `expressão booleana` sempre retorna `True`. Por exemplo, e se não incrementássemos o valor de `x` no exemplo anterior? `x` seria sempre igual a `10` e **sempre** `<= 20`.

A instrução `break` faz com que o programa "saia" de dentro do loop. Por exemplo:

```
while(True):
    break  # vai parar aqui
    print("Não vai imprimir isso na tela")
print("Mas isso vai.")
```

Execute o programa acima para ver o que acontece. Depois tente tirar o `break` e veja o que acontece.

💡 Lembre-se: você pode apertar `Ctrl + C` para parar a execução de um programa no *Python Shell.*