Condições

É muito comum em programas de computador que o fluxo de funcionamento mude de acordo com algumas condições, como por exemplo, o preenchimento de um campo, o pressionar de um botão, ou o navegar entre páginas.

Para dar mais dinâmica aos nossos scripts, podemos utilizar algumas estruturas de controle de fluxo, e nesta aula vamos aprender como fazer isso usando a instrução `if`, vamos direto a um exemplo:

```
pi = 3.14
if pi == 3.14:
        print("Sim, pi é igual a 3.14")
```

Muita coisa acontecendo de uma vez só neste exemplo, mas vamos dissecar linha por linha para entender o que está acontecendo:

1. Estamos declarando uma variável com nome (identificador) pi, cujo valor é 3.14;
2. Estamos usando a instrução if para verificar se pi é igual a 3.14 por meio do operador de igualdade (==), por fim, finalizamos a sentença usando dois pontos;
3. A terceira linha faz o display para o console com o texto, repare que esta linha começa um pouco mais a direita, essa distância é chamada de indentação, e serve para indicar que ali se inicia um novo bloco de código.

Ainda assim temos muita informação para digerir em uma pancada só, vamos focar primeiro em entender a sintaxe da instrução if, para só então dar sequência na explicação de indentação e bloco de código.

A instrução if avalia se uma determinada expressão é verdadeira ou falsa, e caso seja verdadeira executa a próxima instrução ou bloco de instruções, caso contrário, o próximo bloco é deixado de lado, e a execução do script continua ignorando aquelas instruções.

 A expressão que está sendo avaliada na instrução if precisa necessariamente resolver para um valor verdadeiro ou falso, estes dois valores compõem um novo tipo dentro da linguagem python, o tipo `bool`, que possui apenas dois valores possíveis `True` ou `False`.

No caso do exemplo, a variável pi possui o valor de 3.14, portanto a expressão é executada, retornando o valor True, que é do tipo bool, dessa forma o próximo bloco de código será executado.

Mas o que define um bloco de código? Bom, vamos tentar explicar da maneira mais didática possível.

Em python, a indentação é muito importante, enquanto outras linguagens utilizam chaves ({}) para definir blocos de instruções, em python a identação faz esse trabalho.

Considere que cada linha que começa um pouco mais a direita corresponde a um novo bloco de instruções, e cada vez que as instruções voltam um nível, começando um pouco mais a esquerda do que a linha anterior, aquele bloco está fechado. No exemplo abaixo, temos três blocos de código distintos, o bloco principal, o bloco secundário e o bloco terciário:

```
# Bloco principal, onde estão todos os comandos iniciais do script.
txt = "Bloco principal."
if True:
        # Bloco secundário, instruções começam quatro espaços para direita.
        greeted = True
        if greeted == True:
```

```
                    # Bloco terciário, instruções começam oito espaços para direita.
                    print("Estou no bloco Terciário.")
                    # Fim do bloco terciário, a próxima instrução já começa com apenas quatro espaços.
        print("Estou no bloco Secundário.")
        # Fim do bloco secundário, a próxima instrução começa sem espaços.
print("Estou no bloco Principal")
```

Vamos escrever o nosso primeiro programa utilizando o que aprendemos até agora, será um programinha bastante simples, vamos apresentar para o usuário no console uma saudação e ele pode responder de uma forma ou de outra, e dependendo da resposta dele vamos apresentar uma mensagem diferente.

```
print("Olá, seja bem vindo ao python!")
resp = input("Responda com Obrigado para uma mensagem especial")
if resp == "Obrigado":
        print("Aproveite a jornada!")
else:
        print("Não entendi o que você falou...")
```

Introduzimos um novo comando no exemplo acima, o comando `else`, ele é utilizado quando queremos executar instruções nos casos onde a condição verificada em `if` é falsa, como essa instrução é sempre executada quando o inverso de if é verdadeiro, não é necessário (nem possível) executar condições como parte de else.

No entanto, existem casos onde precisamos verificar mais de uma condição para isso podemos utilizar alguns operadores lógicos como `or` e `and`.

```
pi = 3.14
x = 5.12
if pi > 3 and x > 5:
        print("Ok!")
pi = 3.61
x = 4
if pi < 4 or x >= 4:
        print("Ok1")
```

No primeiro exemplo, and é utilizado para indicar que a condição é verdadeira, se e somente se pi for maior do que 3 e x for maior do que 5. De maneira adversa, no segundo exemplo o operador lógico or é utilizado para indicar que a expressão será verdadeira se e somente se pi for menor do que 4 ou x for maior ou igual a 4.

Nestes dois exemplos apresentamos alguns operadores novos, para ficar mais fácil, a tabela abaixo com algumas dicas de como utilizar cada um destes operadores.

Operadores de Comparação

Operador	Efeito	Exemplo True	Exemplo False
==	Igual a	"a" == "a"	"a" == "b"
!=	Diferente de	"b" != "a"	"a" != "a"
>	Maior que	3 > 2	2 > 3
<	Menor que	2 < 3	3 < 2
>=	Maior ou igual a	3 >= 3	3 >= 4
<=	Menor ou igual a	3 >= 3	4 <= 3

Operadores lógicos

Operador	Efeito	Exemplo
and	Retorna True se ambas expressões forem verdadeiras	2 < 5 and 3 < 5
or	Retorna True se uma das condições forem verdadeiras	2 < 5 or 6 < 5
not	Inverte o resultado.	not(2 < 5)