Expressões condicionais

1.1. Se

Os programas do capítulo *Operações Lógicas* não são "amigáveis" para o usuário. Ao invés de mostrar *True* ou *False*, por exemplo, seria mais útil exibir se o aluno foi "Aprovado" ou "Reprovado".

Para que possamos escrever na tela as mensagens "Aprovado" ou "Reprovado", é necessário que haja em algum ponto do código o trecho *print('Aprovado')* e o trecho *print('Reprovado')*. Porém, não gostaríamos que ambos fossem exibidos ao mesmo tempo.

Precisamos *ramificar* o fluxo de execução de nosso programa: em certas circunstâncias, o fluxo deve executar algumas linhas de código e ignorar outras.

Uma **condicional** é uma instrução em Python que decide se outras linhas serão ou não executadas dependendo do resultado de uma condição. A condição nada mais é do que uma expressão lógica. Se a condição for verdadeira, as linhas são executadas. Senão, são ignoradas.

A condicional mais básica em Python é o **if** (se):

```
nota1 = float(input('Digite a nota 1: '))
nota2 = float(input('Digite a nota 2: '))

media = (nota1 + nota2)/2

if media >= 6.0:
    print('Aprovado')

print('Média: ', media)
```

Execute o programa acima. Note que **se** (if) a média é maior ou igual a 6.0, ele exibe a mensagem "Aprovado" e depois a média. Caso contrário, ele apenas exibe a média.

> Para dizermos que uma ou mais linhas "pertencem" ao nosso **if**, usamos um símbolo de parágrafo (tecla "Tab" no teclado). O programa sabe que o *if* "acabou" quando as linhas param de ter "tabs". Esses tabs são chamados de **indentação**. Tanto no *if* quanto no restante das estruturas de controle que estudaremos é obrigatório ter pelo menos 1 linha indentada abaixo da linha de controle.

1.2. Senão

Note que conseguimos fazer nosso programa decidir se ele exibe a mensagem "Aprovado" ou não. O próximo passo seria fazer ele decidir entre 2 mensagens diferentes: "Aprovado" ou "Reprovado". Um primeiro jeito de fazer isso seria um segundo **if** invertendo a condição:

```
nota1 = float(input('Digite a nota 1: '))
nota2 = float(input('Digite a nota 2: '))

media = (nota1 + nota2)/2

if media >= 6.0:
    print('Aprovado')
if media < 6.0:
    print('Reprovado')

print('Média: ', media)
```

O programa acima funciona. Porém, conforme nossos programas começam a ficar mais complexos e nossos **if** começam a ter linhas demais, podemos nos perder e esquecer que esses 2 **if** são 2 casos mutuamente exclusivos. Pior ainda, podemos vir a acrescentar condições novas em um e esquecer de atualizar no outro.

Nos casos em que temos condições mutuamente exclusivas, podemos utilizar um par **if/else** (se/senão). **Se** a condição for verdadeira, faça tal coisa. **Senão**, faça outra coisa.

```
nota1 = float(input('Digite a nota 1: '))
nota2 = float(input('Digite a nota 2: '))

media = (nota1 + nota2)/2

if media >= 6.0:
    print('Aprovado')
else:
    print('Reprovado')

print('Média: ', media)
```

Note que o **else** não possui condição. A condição dele é implícita: é a negação da condição do **if**. Se o **if** executar, o **else** não executa e vice-versa. Consequentemente, **o else não pode existir sem um if**.

1.3. Aninhando condições

É possível *aninhar* condições: ou seja, colocar um novo **if** dentro de outro **if** ou **else**. Imagine que nossa escola não reprova direto o aluno com nota inferior a 6, e sim permite que ele faça uma recuperação. Porém, o aluno precisa ter tirado no mínimo média 3 para que permitam que faça a recuperação. Assim temos:

Se nota maior ou igual a 6: aprovado.
Senão:
    Se nota entre 6 e 3: recuperação.
    Senão: reprovado.

Em Python:

```
nota1 = float(input('Digite a nota 1: '))
nota2 = float(input('Digite a nota 2: '))

media = (nota1 + nota2)/2

if media >= 6.0:
    print('Aprovado')
else:
    if media >= 3.0:
        print('Recuperação')
    else:
        print('Reprovado')

print('Média: ', media)
```

1.4. Senão-se

Note que se começarmos a aninhar muitas condições (if dentro de else dentro de else dentro de else...), nosso código pode começar a ficar confuso, com a aparência de uma "escadinha":

```
Se
Senão:
    Se
    Senão:
        Se
        Senão:
            Se:
            Senão:
                Se:
                 ...
```

Isso pode tornar o código bastante complexo e difícil de atualizar ou corrigir erros posteriormente. Para quebrar a "escadinha", existe a possibilidade de juntarmos o "se" do próximo nível com o "senão" do nível anterior: o **elif**: **else** + **if** (senão + se).

O elif só é executado se um **if** der errado (ou seja, ele é um **else**), mas ele também tem uma condição que deve ser respeitada (ou seja, ele também é um **if**). Podemos reescrever nosso código anterior utilizando um **elif**:

```
nota1 = float(input('Digite a nota 1: '))
nota2 = float(input('Digite a nota 2: '))

media = (nota1 + nota2)/2

if media >= 6.0:
   print('Aprovado')
elif media >= 3.0:
   print('Recuperação')
else:
   print('Reprovado')

print('Média: ', media)
```

Podemos usar quantos **elif** nós quisermos. Sempre que um deles der errado, o próximo será testado. Quando algum deles der certo, todo o restante será ignorado.

Opcionalmente, podemos ter um **else** ao final do bloco, que só será executado se o **if** e todos os **elif** derem errado.

O bloco, **obrigatoriamente**, deve ser iniciado com um *if*.

Atenção

Você lembra dos valores *truthy* e *falsy*? Nós conversamos sobre eles no capítulo *Operações Lógicas*. Uma variável, qualquer que seja seu tipo, pode ser interpretada pelo **if** como se fosse uma expressão lógica.

Se `x` for um inteiro, o bloco `if x:` será executado caso `x` seja diferente de zero, por exemplo.

Uma fonte comum de erros em iniciantes envolve o uso de **and** ou **or** em condicionais e a forma como Python lida com valores *truthy* e *falsy*. Execute o trecho de código abaixo:

```
seguro = input('Deseja adquirir um seguro opcional (sim/não): ')

if seguro != 'sim' and 'não':
    print('Você não digitou uma opção válida')
```

Você verá que ele nem sempre se comporta como você imaginaria. O Python não irá interpretar a condição do **if** como "seguro diferente de 'sim' e seguro diferente de 'não'", e sim como "(seguro diferente de 'sim') e ('não').

Isso ocorre porque no **if** temos uma expressão lógica do tipo `expressão1 and expressão2`. Nossa expressão1 é `seguro != 'sim'`, e nossa expressão2 é apenas a string `'não'`.

A expressão2 é, portanto, uma string não-vazia, portanto ela é *truthy*. O Python irá implicitamente convertê-la para o valor lógico **True**. Portanto, temos a expressão `(seguro !='sim') and (True)`. Logo, a condição será verdadeira se `seguro !='sim'` e falsa caso contrário. Logo, se você digitar "não", a expressão é falsa e o programa dirá que você digitou algo inválido.

Para evitar esse problema, você precisa ser explícito em suas condições:

```
seguro = input('Deseja adquirir um seguro opcional (sim/não): ')

if seguro != 'sim' and seguro !='não':
    print('Você não digitou uma opção válida')
```