Introdução à programação em Python: comandos condicionais

Você já teve contato com alguns elementos de programação em Python.

Você fez alguns programas que trocam mensagens com o usuário e fazem cálculos básicos.

Porém, até agora, seus programas não conseguem tomar rumos diferentes de acordo com a informação recebida: uma linha é executada depois da outra **sempre**.

Neste notebook, você vai aprender a tomar decisões sobre executar algum trecho de código ou não.

Comandos condicionais servem para isso: testam se uma condição é verdadeira ou falsa e então decidem quais ações tomar.

Começaremos vendo quais tipos de condições esses comandos podem testar.

Lembre-se que para executar uma célula de código, basta clicar nela e digitar `SHIFT+ENTER`.

Comparações numéricas

As condições mais simples são as que comparam números, por exemplo:

In [2]:
```
print(2 > 5)
```

```
False
```

A condição anterior vale `False` (falso), pois 2 **não é** maior do que 5.

A condição seguinte é `True` (verdadeiro), pois 3 é igual a 3:

In [3]:
```
print(3 >= 3)
```

```
True
```

Comparadores numéricos de Python

Na tabela abaixo estão todos os operadores de comparação:

operador	significado
==	igual a
!=	diferente de
>	maior que
<	menor que
>=	maior ou igual a
<=	menor ou igual a

E a seguir você pode ver alguns exemplos com todos eles.

Analise cada exemplo e entenda bem o resultado que foi impresso.

In [1]:
```
print("5 é igual a 5?", 5 == 5)
print("3 é igual a 5?", 3 == 5)
print("5 é igual a 5.0?", 5 == 5.0)
print("5 é igual a 5.0001?", 5 == 5.0001)
print("'carla' é igual a 'carla'?", "carla" == "carla")
print("'carla' é igual a 'Carla'?", "carla" == "Carla")
print("'5' é igual a 5?", "5" == 5)
print("int('5') é igual a 5?", int("5") == 5)
```

```
5 é igual a 5? True
3 é igual a 5? False
5 é igual a 5.0? True
5 é igual a 5.0001? False
'carla' é igual a 'carla'? True
'carla' é igual a 'Carla'? False
'5' é igual a 5? False
int('5') é igual a 5? True
```

In [2]:
```
print("5 é diferente de 5?", 5 == 5)
print("3 é diferente de 5?", 3 == 5)
print("5 é diferente de 5.0?", 5 == 5.0)
print("5 é diferente de 5.0001?", 5 == 5.0001)
print("'carla' é diferente de 'carla'?", "carla" == "carla")
print("'carla' é diferente de 'Carla'?", "carla" == "Carla")
print("'5' é diferente de 5?", "5" == 5)
print("int('5') é diferente de 5?", int("5") == 5)
```

```
5 é diferente de 5? True
3 é diferente de 5? False
5 é diferente de 5.0? True
5 é diferente de 5.0001? False
'carla' é diferente de 'carla'? True
'carla' é diferente de 'Carla'? False
'5' é diferente de 5? False
int('5') é diferente de 5? True
```

In [3]:
```
print("5 é maior do que 3?", 5 > 3)
print("5 é maior ou igual a 3?", 5 >= 3)
print("5 é maior do que 5?", 5 > 5)
print("5 é maior ou igual a 5?", 5 >= 5)
print("5 é menor do que 3?", 5 < 3)
print("5 é menor ou igual a 3?", 5 <= 3)
print("5 é menor do que 5?", 5 < 5)
print("5 é menor ou igual a 5?", 5 <= 5)
```

```
5 é maior do que 3? True
5 é maior ou igual a 3? True
5 é maior do que 5? False
5 é maior ou igual a 5? True
5 é menor do que 3? False
5 é menor ou igual a 3? False
5 é menor do que 5? False
5 é menor ou igual a 5? True
```

Faça você mesmo:

Os resultados a seguir são todos `False`. Mude os números ou operadores para que virem todos `True`.

In [8]:
```
# print(4-1 <= 2) False
print(4-1 >= 2)
```

```
True
```

In [9]:
```
# print(3.5/3 == 1) False
print(3.5/3 >= 1)
```

```
True
```

In [10]:
```
# x = 3
# print(2**x != 4*2) False

x = 4
print(2**x != 4*2)
```

```
True
```

In [11]:
```
# x = 4
# y = 2
# print(x/2 != y) False

x = 4
y = 3
print(x/2 != y)
```

```
True
```

Atenção!! Não confunda o operador de comparação de igualdade (==) com a atribuição (=)

Verdadeiro ou falso

Ao executar os exemplos da seção acima, você percebeu que o resultado de uma comparação é sempre **`True`** ou **`False`**. Estes são dois valores especiais em Python.

Você pode guardá-los em uma variável, se precisar. Veja os exemplos:

In [12]:
```
teste1 = 3 < 8 / 3
print(teste1)
```

```
False
```

In [13]:
```
teste2 = True
print(teste2)
```

```
True
```

Checando itens de uma lista

Podemos verificar se um item está armazenado dentro de uma lista usando a palavra `in`.

In [29]:
```
vogais = ["a", "e", "i", "o", "u"]
print("a" in vogais)
print("b" in vogais)
```

```
True
False
```

Condicionais

Finalmente, vamos ver como fazer para indicar que partes de um código somente serão executadas no caso em que uma condição é verdadeira.

Para isso, vamos usar o comando `if`, que significa "se" em inglês.

Veja o exemplo abaixo, execute-o e depois veja a explicação.

In [3]:
```
#A = 9
#B = 10
#if A > B:
#    print("A é maior que B")
#    print("===============")
#print("fim do programa")

inteiro_1 = int(input("Digite um número inteiro: "))
inteiro_2 = int(input("Digite outro número inteiro: "))

if inteiro_1 > inteiro_2:
    print("O número ",inteiro_1," é maior do que o número ",inteiro_2)
    print("==========================")
print("fim do programa")
```

```
Digite um número inteiro: 5
Digite outro número inteiro: 4
O número  5  é maior do que o número  4
==========================
fim do programa
```

Veja que os dois comandos `print(...)` que aparecem logo após o comando **`if`** não foram executados, pois `A > B` é falso para os valores atribuídos a `A` e `B`.

Faça você mesmo:

1. No código acima, mude o valor de `A` para qualquer valor maior do que 10, re-execute a célula e veja o que acontece.
2. Depois disso, mude o código acima para que `A` e `B` sejam inteiros digitados pelo usuário.

O comando `if`

O comando `if` tem a seguinte estrutura:

```
if condição:
    comando
```

```
    comando
    comando
    etc...
continuação do programa
```

As linhas de código que você quiser executar quando a condição for verdadeira devem vir *indentadas* (alinhadas um pouco mais a frente do que o `if`) logo após o comando `if`. Você pode usar a tecla `TAB` para alinhar as linhas.

Chamaremos o conjunto de comandos executados caso a condição do `if` seja verdadeira de **bloco de comandos do `if`**.

Uma linha sem *indentação* marca o fim do bloco de comandos do `if`.

Atenção: se você não *indentar* o código corretamente, o programa terá um comportamento diferente daquele que você espera!

Por exemplo, o código a seguir dá erro por falta de indentação (execute-o e veja a mensagem de erro).

In [9]:
```
#A = 2
#B = 10
#if A > B:
#print("A é maior que B")
#print("===============")
#print("fim do programa")

A = 2
B = 10
if A < B:
    print("A é maior que B")
    print("===============")
print("fim do programa")
```

```
A é maior que B
===============
fim do programa
```

O código a seguir sempre imprime "===============", pois o comando para imprimir isso já está *fora* do bloco do if.

In [13]:
```
#A = 2
#B = 10
#if A > B:
#    print("A é maior que B")
#print("===============")
#print("fim do programa")

A = int(input("Digite um número inteiro: "))
B = int(input("Digite outro número inteiro: "))

if A > B:
    print("A é maior que B")
print("===============")
print("fim do programa")
```

```
Digite um número inteiro: 3
Digite outro número inteiro: 1
A é maior que B
===============
fim do programa
```

Observação: É uma prática comum em programação, de forma geral, fazer indentação de código para que ele fique mais legível. Na maioria das linguagens de programação como Java e C, a indentação tem um propósito puramente estético. Em Python, no entanto, a indentação é **obrigatória** em vários casos. O bloco do `if` é um desses casos.

Importante: você pode escrever qualquer comando que nós já vimos dentro do bloco de comandos do `if`, inclusive outros comandos `if`:

In [15]:
```
A = int(input("Digite o primeiro número: "))
B = int(input("Digite o segundo número: "))
if B != 0:
    divisao = A // B
    resto = A % B
    if resto == 0:
        print(A, "é divisível por", B, "e o resultado da divisao é", divisao)
    if resto != 0:
        print("O resultado da divisão de", A, "por", B, "é", divisao, "e o resto é", resto)
print("FIM")
```

```
Digite o primeiro número: 31
Digite o segundo número: 15
O resultado da divisão de 31 por 15 é 2 e o resto é 1
FIM
```

Entenda bem o programa anterior: ele testa se um número é divisível pelo outro. Esse tipo de teste será muito utilizado posteriormente.

Faça você mesmo!

Complete o código a seguir de modo que ele tenha o seguinte comportamento:

- Pergunta ao usuário quantos anos ele tem.
- Se o usuário tem idade par, mostra uma mensagem a ele informando-o desse fato. No caso ímpar você não precisa fazer nada.

Dica: um número é par se ele é divisível por 2.

In [19]:
```
idade = int(input("Quantos anos você tem? "))
if idade != 0:
    resto = idade % 2
    if resto == 0:
        print("Sua idade é par.")
    if resto != 0:
        print("Sua idade é impar.")
print("")
print("Fim do programa.")
```

```
Quantos anos você tem? 0

Fim do programa.
```

Faça você mesmo!

Identifique o que o código a seguir está fazendo e complete a mensagem.

In [23]:
```
x = int(input("Digite um inteiro qualquer: "))
if x in range(1,100,2):
    print("O número digitado é positivo, menor do que 100 e é impar")
```

```
Digite um inteiro qualquer: 5
O número digitado é positivo, menor do que 100 e é impar
```

O comando `if..else`

Agora que você já entendeu o funcionamento do `if`, vamos aprender como executar uma ação (ou várias ações) caso a condição do `if` seja falsa.

Por exemplo, no exercício da idade par ou impar, se quiséssemos mostrar uma mensagem ao usuário também no caso ímpar, poderíamos fazer algo do tipo:

In [ ]:
```
idade = int(input("Quantos anos você tem? "))
if idade % 2 == 0:
    print("Sua idade é par.")
if idade % 2 == 1:
    print("Sua idade é ímpar.")
```

O código acima está funcionando da seguinte forma:

- primeiro, ele testa se `idade % 2 == 0`
- se for verdade, ele imprime a mensagem `Sua idade é par.` e se não for verdade ele não imprime essa mensagem
- em seguida, ele testa se `idade % 2 == 1`
- se for verdade, ele imprime `Sua idade é ímpar.` e se não for verdade ele não imprime nada

Não há nada de errado com esse comportamento, afinal nós já vimos que os comandos de um programa são executados dessa forma: um após o outro.

Inclusive, esse código está funcionando corretamente: ele vai indicar, corretamente, se o número digitado pelo usuário é par ou ímpar.

Acontece que nós sabemos que um número só pode ser par ou ímpar, e nunca ambos ao mesmo tempo.

Assim, se `idade` é um número par e a mensagem `Sua idade é par.` for impressa, não há necessidade em fazer o teste `idade % 2 == 1`.

Com o comando `else` (que significa "senão" em inglês) é possível melhorar esse programa. Veja e teste o exemplo abaixo.

In [1]:
```
idade = int(input("Quantos anos você tem? "))
if idade % 2 == 0:
    print("Sua idade é par.")
else:
    print("Sua idade é ímpar.")
```

```
Quantos anos você tem? 10
Sua idade é par.
```

Ou seja, o código no bloco do `else` só é executado se a condição do `if` for falsa.

Observação: você pode escrever qualquer comando que nós já vimos dentro do bloco de comandos do `if` e dentro do bloco de comandos do `else`, inclusive outros comandos `if` ou `if..else`.

Importante

O comando `if..else` só deve ser utilizado quando temos certeza de que apenas um dos caminhos deve ser seguido.

Teste o exemplo a seguir:

In [3]:
```
valor = int(input("Digite um inteiro qualquer: "))
if valor > 0:
    print("O número é positivo")
    if valor % 2 == 0:
            print("O número é par")
    else:
            print("O número é impar")
else:
    if valor < 0:
        print("O número é negativo")
    if valor % 2 == 0:
            print("O número é par")
    else:
            print("O número é impar")
```

```
Digite um inteiro qualquer: -13
O número é negativo
O número é impar
```

Note que um número pode ser positivo e par ao mesmo tempo.

Assim, supondo que a intenção do código acima é fornecer todas as informações possíveis sobre o número digitado, então ele está errado.

Volte e **corrija-o**.

Exemplo

O programa a seguir quer testar qual é o maior dentre dois números lidos na entrada.

Modifique-o para fazer isso de forma mais eficiente.

In [6]:
```
#a = int(input("Digite um número: "))
#b = int(input("Digite outro número: "))
#if a >= b:
#    print("O número", a, "é maior ou igual ao número", b)
#if a < b:
#    print("O número", b, "é maior do que o número", a)

a = int(input("Digite um número: "))
b = int(input("Digite outro número: "))
if a >= b:
    print("O número",a,"é maior ou igual ao número",b)
else:
    print("O número",b,"é maior do que o número",a)
```

```
Digite um número: 11
Digite outro número: 13
O número 13 é maior do que o número 11
```

O exemplo a seguir envolve listas.

Note que um elemento pode estar ou não em uma lista: não há outras opções e ele precisa satisfazer uma dessas duas.

In [9]:
```
minhas_sobremesas = ["bolo de chocolate", "sorvete", "pudim", "paçoca", "brigadeiro"]
x = input("Digite sua sobremesa favorita: ")
if x in minhas_sobremesas:
    print("Essa também é uma das minhas sobremesas favoritas!")
else:
    print("Hmm, eu não havia pensado nessa!!")
```

```
Digite sua sobremesa favorita: torta de maçã
Hmm, eu não havia pensado nessa!!
```

Faça você mesmo!

Profa. Alice ministra a disciplina de Fenômenos Quânticos na UFABC. Ela dará duas provas (P_1 e P_2), e o critério de aprovação é ter média maior ou igual a 5.0. Faça um programa em Python que ajude a Profa. Alice a decidir se um aluno foi aprovado ou não. Seu programa deve executar os seguintes passos:

- Perguntar a nota da P_1.
- Perguntar a nota da P_2.
- Calcular a média simples das provas.
- Se a média das provas for maior ou igual a 5.0, mostrar a mensagem "Aprovado".
- Senão, mostrar "Reprovado".

In [51]:
```
p1 = float(input("Informe a nota da P1: "))
p2 = float(input("Informe  anota da P2: "))

media = (p1 + p2) / 2

if media >= 5.0:
    print("Parabéns jovem, o senhorito está aprovado")
else:
    print("Tente outra vez")
```

```
Informe a nota da P1: 4.5
Informe  anota da P2: 5.5
Parabéns jovem, o senhorito está aprovado
```

..

Combinando condições

Podemos criar condições mais complexas combinando condições simples. Vejamos um exemplo.

Em algumas disciplinas da UFABC, não basta que a média das duas provas ($P1$ e $P2$) de um aluno seja maior ou igual a 5.0: ele também deve tirar uma nota mínima 3.0 na $P2$.

Ou seja, para que seja aprovado o aluno precisa que

$$\frac{P1 + P2}{2} \geq 5 \qquad \text{E} \qquad P2 \geq 3.$$

Em Python, podemos usar a palavra **and** para exigir que duas condições valham ao mesmo tempo:

In [13]:
```
#P1 = float(input("Qual a nota da P1? "))
#P2 = float(input("Qual a nota da P2? "))
#if (P1 + P2)/2 >= 5.0 and P2 >= 3.0:
#    print("Aprovado")
#else:
#    print("Reprovado")

P1 = float(input("Qual a nota da P1? "))
P2 = float(input("Qual a nota da P2? "))
if (P1 + P2)/2 >= 5.0 and P2 >= 3.0 and P1 >= 3.0:
    print("Aprovado")
else:
    print("Reprovado")
```

```
Qual a nota da P1? 7.01
Qual a nota da P2? 2.99
Reprovado
```

O operador **`and`** funciona assim: a condição composta `(condição1 and condição2)` é avaliada como **`True`** (verdadeira) só quando a `condição1` é **`True`** e, simultaneamente, a `condição2` também é **`True`** .

Faça você mesmo:

Modifique o código acima para que a nota da P_1 também tenha que ser pelo menos 3.

No código abaixo, altere o valor das variáveis `chuva` e `sol` para ver como isso afeta o programa.

In [16]:
```
#As duas condições precisam ser verdadeiras
chuva = True
sol = True
arcoiris = chuva and sol
print(arcoiris)
```

```
True
```

Faça você mesmo!

Vamos assumir que um professor universitário federal pode se aposentar desde que tenha pelo menos 35 anos de serviço e pelo menos 60 anos de idade. Faça um programa para ajudar o professor a descobrir se ele já pode se aposentar.

Seu programa deve fazer o seguinte:

- Pergunte ao professor pelo ano em que nasceu.
- Pergunte pelo ano em que começou a trabalhar.
- Pergunte ao professor pelo ano em que estamos.
- Baseando-se nas respostas que ele deu ao seu programa, diga se ele pode se aposentar este ano.

In [24]:
```
ano_nasc = int(input("Em que ano você nasceu? "))
comecou_trab = int(input("Em que ano você começou a trabalhar? "))
ano_atual = int(input("Informe em que ano estamos: "))

tempo_trab = ano_atual - comecou_trab
idade = ano_atual - ano_nasc
tempo_restante = 35 - tempo_trab
idade_restante = 60 - idade

if (tempo_trab >= 35) and (idade >= 60):
     print("Você já pode se aposentar.")
else:
    if (tempo_trab < 35):
        print("Você precisa trabalhar mais ",tempo_restante,"anos.")
    else:
        if (idade < 60):
            print("Faltam",idade_restante,"anos para você completar a idade obrigatória.")
```

```
Em que ano você nasceu? 1969
Em que ano você começou a trabalhar? 1983
Informe em que ano estamos: 2018
Falta 11 anos para vocÊ competar a iade obrigatória.
```

Outros operadores

Python tem ainda os operadores **`or`** e **`not`**.

O operador **`not`** pode ser usado para inverter a validade de uma condição. Veja:

In [9]:
```
not (2 > 5)
```

Out[9]:
```
True
```

Uma condição da forma `(condição1 or condição2)` será verdadeira se pelo menos uma das duas condições envolvidas for verdadeira.

Diferentemente do que estamos acostumados em português, o operador **`or`** não é um ou-exclusivo, ou seja, se as duas condições `condição1` e `condição2` forem ambas verdadeiras, então o resultado de `(condição1 or condição2)` também será verdadeiro.

Faça você mesmo:

No código abaixo, altere o valor das variáveis `chuva` e `sol` para ver como isso afeta o programa.

In [1]:
```
#Apenas uma das condições precisa ser verdadeira
frio = True
chuva = False
tempoRuim = frio or chuva
print(tempoRuim)
```

```
True
```

Mais exemplos

O resultado da condição composta abaixo é `True` porque, apesar da primeira condição ser falsa, a segunda é verdadeira.

In [15]:
```
print(2 > 5 or 1 < 2)
```

```
True
```

No próximo exemplo, as duas condições são verdadeiras.

In [16]:
```
print(4 == 2 + 2 or 2 > 0)
```

```
True
```

Observação: Os operadores `and`, `or` e `not` podem ser combinados em expressões maiores, de forma similar ao que fazemos com os operadores aritméticos (como `+` e `*`).

In [28]:
```
x = int(input("Digite um número inteiro: "))
# Nesse problema, um número é considerado válido se satisfaz uma das duas propriedades:
# * ele está no intervalo entre 100 e 1000
# * ele é múltiplo de 3
if (100 <= x and x <= 1000) or (x % 3 == 0):
    print("O número digitado é válido")
```

```
Digite um número inteiro: 8
```

Múltiplas possibilidades

É bastante comum lidarmos com situações em que não existem apenas duas possibilidades para o comportamento do programa.

Suponha que temos que criar um programa para classificar o nível do *low-density lipoprotein* (LDL), conhecido em português com o nome de colesterol "ruim". Segundo um estudo citado pelo *CDC* (http://www.cdc.gov/dhdsp/data_statistics/fact_sheets/fs_cholesterol.htm) Norte Americano, os níveis de LDL podem ser classificados segundo a tabela abaixo.

Nível de LDL	Classificação
menor que 100	Ótimo
entre 100 e 129	Baixo
entre 130 e 159	Limítrofe
entre 160 e 189	Alto
maior que 190	Muito alto

Se tivéssemos que fazer um programa que perguntasse pelo nível de LDL e que mostrasse a classificação correspondente, poderíamos usar os conhecimentos de `if..else` que adquirimos até agora e construir o seguinte programa.

In [24]:
```
LDL = float(input("Qual o nível de LDL? "))
if LDL < 100:
    print("Ótimo")
if 100 <= LDL and LDL < 130:
    print("Baixo")
if 130 <= LDL and LDL < 160:
    print("Limítrofe")
if 160 <= LDL and LDL < 190:
    print("Alto")
if 190 <= LDL:
    print("Muito alto")
```

```
Qual o nível de LDL? 159
Limítrofe
```

O programa acima está correto, mas essa não é a melhor forma de realizar esses testes.

No código acima, todos os testes serão feitos, mesmo sabendo que apenas um deles pode ser verdadeiro, qualquer que seja o valor de `LDL`.

Uma forma de melhorar isso é dada a seguir.

Estude bem essa forma para entender o que está acontecendo.

In [26]:
```
LDL = float(input("Qual o nível de LDL? "))
if LDL < 100:
    print("Ótimo")
else: # aqui sabemos que LDL >= 100
    if LDL < 130: # por isso basta testar se é menor do que 130
        print("Baixo")
    else: # aqui sabemos que LDL >= 130
        if LDL < 160: # por isso basta testar se é menor do que 159
            print("Limítrofe")
        else:
            if LDL < 190:
                print("Alto")
            else:
                print("Muito alto")
```

```
Qual o nível de LDL? 159
Limítrofe
```

Existe uma forma ainda melhor de escrevê-lo em Python, que é usando o comando **`elif`** (abreviação de "else if"; em português, "senão, se").

Você pode usar diversos comandos **`elif`** associados a um **`if`**, mas eles devem sempre vir atrelados a um comando **`if`**.

Veja como é mais simples classificar os níveis de LDL com o comando **`elif`**:

In [27]:
```
LDL = float(input("Qual o nível de LDL? "))
if LDL < 100:
    print("Ótimo")
elif LDL < 130:
    print("Baixo")
elif LDL < 160:
    print("Limítrofe")
elif LDL < 190:
    print("Alto")
else:
    print("Muito alto")
```

```
Qual o nível de LDL? 100
Baixo
```

Quando o teste `LDL < 100` resulta em verdadeiro, os comandos do seu bloco são executados e **nenhum** outro comando da construção `if .. elif .. else` é executado (em particular, nenhum teste é feito).

Por outro lado, se o teste `LDL < 100` for falso, então os outros testes serão feitos, mas para todos eles já sabemos que temos um valor de `LDL` maior ou igual a 100 e, por isso, não há necessidade de indicar esse teste nas condições seguintes.

No programa acima, o trecho `print("Alto")` só será executado se:

- a condição do **`if`** for falsa (isto é, se LDL >= 100) **e**
- a condição do primeiro **`elif`** for falsa (isto é, se LDL >= 130) **e**
- a condição do segundo **`elif`** for falsa (isto é, se LDL >= 160) **e**
- a condição do terceiro **`elif`** for verdadeira (isto é, se LDL < 190).

De forma mais geral, um trecho de código dentro de um bloco **`elif`** só será executado se a condição daquele **`elif`** for verdadeira, mas todas as condições dos **`elif`** que o precedem, incluindo a condição do **`if`**, forem todas falsas!

Por fim, fique sabendo que usar um comando **`elif`** dentro de um **`if`** não proíbe você de usar um **`else`** também, desde que o **`else`** venha por último.

O trecho de código dentro do **`else`** só será executado se **todas** as condições que o precederam forem falsas.

Faça você mesmo!

Se uma pessoa tem peso P e altura A, então o índice de massa corporal (*IMC*) dessa pessoa é

$$IMC = \frac{P}{A^2}.$$

Faça um programa que pede para o usuário digitar a sua altura e depois o seu peso. Calcule e mostre o IMC do usuário e diga em que categoria ele está. Use a seguinte tabela:

Faixa	Categoria
IMC < 18.5	Abaixo do peso
18.5 ≤ IMC < 25.0	Normal
25.0 ≤ IMC < 30.0	Sobrepeso
30 ≤ IMC	Obeso

In [38]:
```
A = float(input("Informe a sua altura (em metros):"))
P = float(input("Informe quantos kilos você pesa: "))

IMC = (P / A**2)

if IMC < 18.5:
    print("Abaixo do peso")
else: # aqui sabemos que IMC >= 18.6
    if IMC < 25: # por isso basta testar se é menor do que 25
        print("Normal")
    else: # aqui sabemos que IMC >= 25
        if IMC < 30: # por isso basta testar se é menor do que 30
            print("Sobrepeso")
        else:
            print("Obeso")
```

```
Informe a sua altura (em metros):1.8
Informe quantos kilos você pesa: 90
Sobrepeso
```

Outro exemplo

Suponha que queremos descobrir se três dados números são os lados de um triângulo.

Obviamente os três valores têm que ser positivos e não nulos mas, além disso, em um triângulo vale a propriedade de que cada um dos seus lados é menor do que a soma dos outros dois lados.

Os três códigos a seguir fazem essa verificação de formas diferentes.

Estude cada uma delas atentamente.

In [ ]:
```
a = int(input("Digite um número inteiro: "))
b = int(input("Digite outro número inteiro: "))
c = int(input("Digite um último número inteiro: "))

if a <= 0 or b <= 0 or c <= 0: # se algum dos números for negativo, não forma triângulo
    print("Esses valores não formam um triângulo.")
# a seguir sabemos, portanto, que todos os valores são positivos
elif a < b+c and b < a+c and c < a+b: # aqui basta testar se cada lado é menor que a soma dos outros dois
    print("Os valores formam um triângulo.")
# a seguir ainda sabemos que todos os valores são positivos
else: # porém algum dos lados deve ser maior do que a soma dos outros dois
    print("Esses valores não formam um triângulo.")
```

In [ ]:
```
a = int(input("Digite um número inteiro: "))
b = int(input("Digite outro número inteiro: "))
c = int(input("Digite um último número inteiro: "))

if (a > 0 and b > 0 and c > 0) and (a < b+c and b < a+c and c < a+b):
    print("Os valores formam um triângulo.")
else:
    print("Esses valores não formam um triângulo.")
```

In [41]:
```
a = int(input("Digite um número inteiro: "))
b = int(input("Digite outro número inteiro: "))
c = int(input("Digite um último número inteiro: "))

#     (a > 0 and b > 0 and c >  0) and    (a < b+c and b < a+c and c <  a+b):
if not(a <= 0 or b <= 0 or c <= 0) and not(a >= b+c or b >= a+c or c >= a+b):
    print("")
    print("Os valores formam um triângulo.")
else:
    print("")
    print("Esses valores não formam um triângulo.")
```

```
Digite um número inteiro: 10
Digite outro número inteiro: 12
Digite um último número inteiro: 13
Os valores formam um triângulo.
```

Outro exemplo

O problema agora é descobrir qual é o maior número dentre três números inteiros dados pelo usuário.

Note que não é dada nenhuma informação sobre quais são os números com relação a eles serem negativos ou positivos ou mesmo se eles estão todos dentro de algum intervalo.

In [40]:
```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))

# vamos usar uma variável de nome "maior" que vai manter qual dos três é o maior deles
if a >= b and a >= c:
    maior = a
elif b >= a and b >= c:
    maior = b
else:
    maior = c
print("")
print("O maior dos três números é", maior)
```

```
Digite o primeiro número: 2
Digite o segundo número: 3
Digite o terceiro número: 1

O maior dos três números é 3
```

Agora é a sua vez!

Resolva a lista de exercícios dessa aula para fixar melhor todo esse conteúdo!

Boa sorte e não se esqueça de ir ao horário de atendimento da professora caso esteja com dúvidas ou tenha algum problema.