**O que são literais no contexto de Python?**

Os literais são notações que representam alguns valores fixos no código. O Python tem vários tipos de literais, podendo ser um número literal, por exemplo 123 ou uma string, por exemplo "Eu sou um literal".

**Octais e Hexadecimais em Python**

Em Python nós temos dois tipos de convensão além dos decimais, que são os números octais, que são múltiplos de 8 em vez de 10, como no conjunto decimal, e para converter um decimal para octal nós fazemos o número \* 8 elevado a x dependendo das casas, por exemplo:

123 decimal = 83 octal.

1 \* 8 elevado a 2 = 64

2 \* 8 elevado a 1 = 16

3 \* 8 elevado a 0 = 3

Resultando em 83.

Para converter para hexadecimal nós dividimos o número por 16 elevado ao número mais proximo do total do número em decimal e continuamos a dividir o resto(podemos calcular o resto em uma calculadora usando a seguinte fórmula( decimal - resultado da divisão \* 16)).

123 decimal = 7B hexadecimal

123/16 elevado a 1 = 7, sobrando 11

Como 11 é um número menor que 16 e maior que 9 nós usamos a sua forma em hexadecimal.

Nós podemos também em Python converter o valor de Octal ou Hexadecimal para decimal o representando por 0o123 para os números octais e 0x123 para os números hexadecimais.

**Float e a representação de números muito grandes em Python.**

Floats são basicamente números com valores após a vírgula, e que em Python, números que são abaixo de 0 podem ser representados sem apresentar o número 0 em si, como por exemplo 0.4 pode ser representado como .4 . Números inteiros convertidos para float também devem conter o "." mesmo que não sejam números fracionados, por exemplo para o número inteiro 6 ser considerado float ele precisa ser escrito como 4.0 .

Para a escrita de números muito grandes, como a velocidade da luz que é representada pelo número 300\_000\_000 no campo científico as pessoas representam esse número como 3 \* 10 elevado a 8, assim a compreensão fica mais fácil, e para representar isso em Python nós escrevemos 3E8, que tem o mesmo significado.

Obs.: O número expoente(depois do E) precisa ser inteiro, e o número base pode ser tanto um integer como um float.

Para escrever números muito pequenos nós fazemos o mesmo processo, porém nós colocamos um sinal de negativo antes do expoente. O Python ao representar números no console pode acabar escrevendo os números de forma diferente ao que você escreveu como literal. Por exemplo:

print(0.0000000000000000000001)

Teria como resultado no console:

1e-22

Isso por que o Python tenta usar a forma mais economica de representação de um número.

**Representando aspas e apóstrofos em Strings.**

Para escrever aspas e apóstrofos em Strings usando Python nós temos duas soluções, nós podemos chamar eles individualmente usando a barra invertida como 'I\' Monty Python' mas nós também podemos usar outra solução que é usar o apóstrofo ou aspas para apresentar o outro, por exemplo como vou escrever um apóstrofo eu cubro a String em aspas: "I'm Monty Python".