Em geral, quando os dados são muito grandes e temos pouca memória disponível, é comum utilizar tipos de dados mais compactos para reduzir o consumo de memória. Mas, é sempre importante garantir que a **escolha do tipo de dados não prejudique a precisão ou a acurácia dos resultados**.

Quando trabalhamos com números inteiros com Python, podemos ter diversos tipos de dados, cada um com suas limitações e características. Durante as aulas, trabalhamos com o int64, um inteiro com precisão de 64 bits. Para entender o significado dessa precisão, é importante conhecer alguns termos técnicos, como byte e bit:

* **Byte**: é uma unidade de medida de informação, que representa um conjunto de 8 bits.
* **Bit**: é a menor unidade de informação utilizada em sistemas digitais, podendo assumir os valores de 0 ou 1.

## **Tipo Inteiro**

Com os conceitos de bit e byte claros, podemos, então, entender melhor o significado do int64, que é o tipo inteiro que utiliza 8 bytes de armazenamento - 8 bits em cada byte, resultando em 64 bits ao total. Esse tipo inteiro é capaz de representar números muito grandes, que podem variar de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807.

Além dele, temos outros inteiros que podem ter sua precisão definida, como o int32, um tipo de dado inteiro que utiliza 4 bytes - 8 bits em cada byte, resultando em 32 bits ao total. Ele é capaz de representar números inteiros menores do que os representados pelo int64, com um máximo de -2.147.483.648 a 2.147.483.647.

Pode ser mais comum encontrar os tipos int64 e int32, mas pode ser necessário, em algumas situações, utilizar outros tipos de dados inteiros, como o int8 ou o int16. Tipos de dados como esses são úteis quando é preciso economizar mais memória e não se está trabalhando com grandes valores. Você pode conferir os tipos de inteiros na tabela abaixo:

| **Tipo de dado** | **Quantidade de bits** | **Valor mínimo** | **Valor máximo** |
| --- | --- | --- | --- |
| int8 | 8 | -128 | 127 |
| int16 | 16 | -32768 | 32767 |
| int32 | 32 | -2.147.483.648 | 2.147.483.647 |
| int64 | 64 | -9.223.372.036.854.775.808 | 9.223.372.036.854.775.807 |

A escolha de qual valor de precisão vai depender da situação e da natureza dos dados sendo manipulados. Se os valores que estão sendo analisados são relativamente pequenos, o uso do int32, por exemplo, pode ser o suficiente, **podendo economizar espaço em memória.** Entretanto, se tivéssemos trabalhando com dados científicos, por exemplo, que precisam de valores bem grandes, precisaríamos talvez utilizar o int64.

## **Tipo float**

Além dos inteiros, outros tipos de dados, como o float, também utilizam essa opção de precisão como opção de controle de espaço na memória. De maneira semelhante aos números inteiros, o tipo float também apresenta opções de precisão: entre os tipos mais comuns estão o float32e o float64.

O tipo float64 é um número de ponto flutuante com 64 bits de precisão, que representa um número decimal com até 15 dígitos. Por outro lado, o 'float32' é menor tanto em sua capacidade de bits, com 32 ao total, quanto em sua capacidade de precisão de casas decimais, com a capacidade de precisão até 7 dígitos.