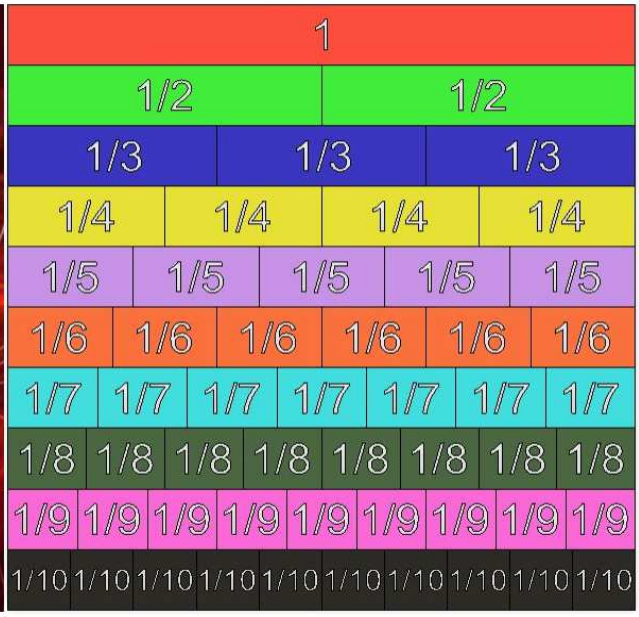
# Fracciones con Python

(Resumen: Castelli Horacio - 2022)


## Índice

## Para Comenzar, Deberías Saber Que:

Una fracción, representa una División no realizada, también podemos definirla como una representación matemática que, expresa la relación entre dos números, donde el numerador (el número de arriba) indica cuántas partes se toman y el denominador (el número de abajo) indica en cuántas partes iguales se divide un todo. Por ejemplo, en la fracción 3/4, el 3 representa tres partes de un total de cuatro partes iguales.

Las fracciones son útiles en cálculos Matemáticos: ya que nos permite el estudio de las relaciones numéricas, son exactas (evitan que trabajemos con aproximaciones) y se utilizan en operaciones como suma, resta, multiplicación y división. En resumen, las fracciones son herramientas esenciales para representar y trabajar con partes de un todo en diversas disciplinas.

$$\frac{a}{b} = \frac{numerador}{denominador}$$

## LAS FRACCIONES EN PYTHON

Python incluye un módulo (librería) llamado "fractions" que permite trabajar con fracciones de manera sencilla y eficiente. Este módulo proporciona la clase "**Fraction**", que representa números racionales como el cociente de dos enteros: un numerador y un denominador.

El uso de fracciones, es una herramienta poderosa para realizar cálculos precisos con números racionales. La clase "**Fraction**" te permite realizar operaciones matemáticas complejas de manera sencilla y eficiente.

### *Operaciones Básicas con Fracciones:*

**a) Suma y Resta**: Puedes sumar y restar fracciones de forma natural. Python se encarga de encontrar un denominador común automáticamente.
**b) Multiplicación y División**: Multiplicar fracciones es tan simple como multiplicar los numeradores entre sí y los denominadores entre sí. Para dividir, simplemente multiplicas por la fracción inversa.
**c) Conversión a Decimal**: Las fracciones pueden convertirse fácilmente a su representación decimal utilizando la función "**float()**". Esto es útil para cálculos que requieren precisión decimal.
**d) Simplificación Automática**: Una de las características más útiles de la clase "**Fraction**" es que cuando es posible, simplifica automáticamente las fracciones.
**e) Comparaciones**: Puedes comparar fracciones usando los operadores estándar de comparación, como mayor que, menor que, etc.
**f) Acceso a Numerador y Denominador**: La clase "**Fraction**" proporciona acceso directo al numerador y denominador a través de sus atributos "**numerator**" y "**denominator**".
**g) Detectar si una variable contiene una fracción**.
**h) Reconoce Notación Científica**.
**i)** etc.

A continuación te dejo el código de un programa donde encontraras ejemplos de cada una de las cosas que puedes hacer con las fracciones. Bájalo de la nube y analízalo. Es muy fácil usar las fracciones con Python.

| **Programa:** | Codigo/13_Fracciones_001_Operaciones_A_Ejemplos_Operaciones_Iniciales.txt |
|---|---|

Adicionalmente también esta el cálculo del **mínimo común múltiplo**, muy útil, pero que no usa funciones de la librería "**fractions**", es por este motivo que el ejemplo esta por separado.

```
import numpy as np

def mcm(*Valores):
    return np.lcm.reduce(Valores)  # Función para calcular el MCM usando numpy
# La función lcm.reduce, que pertenece a la Librería "numpy" aplica la función
# de mínimo común múltiplo a todos los números que le pases como argumento.

# Programa Principal
Numeros = [4, 6, 8, 5, 7]
resultado = mcm(*Numeros)
print(f"El Mínimo Común Múltiplo de {Numeros} es: {resultado}")
```

***Importante*:** el asterisco antes de "Numeros" cuando realizo el llamado a la función "mcm(*numeros)" se llama *operador de desempaquetado*. En este caso, permite pasar los elementos de la lista números como argumentos individuales a la función np.lcm.reduce. Sin el asterisco, la función "mcm( )" recibiría la lista completa como un solo argumento, y "np.lcm.reduce( )" espera varios números separados. El asterisco "*desempaqueta*" la lista, haciendo que cada elemento se pase como un argumento independiente. Es una forma concisa de pasar una cantidad variable de argumentos a una función.
No hay un límite explícito en la cantidad de elementos que puedes pasar a la función "np.lcm.reduce( )" a través del desempaquetado con el asterisco, excepto las limitaciones prácticas de la memoria de tu sistema. Podrías pasar una lista muy grande, pero si es extremadamente grande, podrías encontrarte con problemas de rendimiento o incluso un error de memoria. En la práctica, el límite dependerá de la capacidad de tu computadora.

Puedes descargar el Código completo desde la nube. Analízalo.

| **Programa:** | Codigo/13_Fracciones_001_Operaciones_B_Minimo_Comun_Multiplo_01.txt |
|---|---|