

Matemáticas para ingeniería con Python

Matemáticas

Este concepto de 'entes abstractos' incluye a los números, los símbolos y las figuras geométricas, entre otros.

Aplicaciones de las matemáticas

Esta ciencia formal traduce todo lo que nos rodea a números. Y estos números son los que nos permiten dar cohesión a y entender el mismo universo. A lo largo de la historia, las Matemáticas han evolucionado mucho y, hoy en día, existen distintas disciplinas dentro de ellas que comentaremos a continuación.

Aritmética - La Aritmética es la rama de las Matemáticas que se centra en el estudio exclusivo de los números y de **las operaciones que pueden realizarse con ellos**. En este sentido, la Aritmética es la disciplina matemática que se encarga de las operaciones elementales de suma, resta, división y multiplicación. Se trata, pues, de la base sobre la que descansan las otras ramas.

Álgebra - El álgebra es la rama de las Matemáticas que **estudia la combinación de elementos de estructuras abstractas** de acuerdo a relaciones lógicas y reglas preestablecidas. Esto permite tanto realizar operaciones aritméticas con los números naturales y enteros como resolver ecuaciones que introducen símbolos más allá de los números

Geometría - La Geometría es la rama de las Matemáticas que **se encarga del estudio de figuras en el espacio**. En este sentido, la Geometría se centra en analizar las propiedades de extensión y forma de figuras en el plano o en el espacio, así como las relaciones entre puntos, líneas, rectas y otras figuras. Son las Matemáticas aplicadas a un espacio de distintas dimensiones, generalmente de dos o tres.

Trigonometría - La Trigonometría es la rama de las Matemáticas que, a grandes rasgos, **se centra en el estudio de los triángulos**. Más concretamente, en las relaciones entre las conocidas como razones trigonométricas: seno, coseno, tangente, secante, cosecante y cotangente. Se trata de una disciplina dentro de la Geometría con importantes aplicaciones especialmente en Astronomía y en sistemas de navegación por satélites.

Probabilidad y estadística - La Probabilidad y estadística es la rama de las Matemáticas que **estudia los fenómenos aleatorios**. En otras palabras, es la disciplina matemática que hace posible el análisis de las tendencias sobre la base de un muestreo a partir de estudios numéricos acerca de los patrones en los que nos centremos. Hoy en día hay un gran impacto con lo que es data science y el big data o data analysis

Análisis - El Análisis es la rama de las Matemáticas que, a grandes rasgos, estudia el cambio. El Análisis es la disciplina matemática que **inspecciona los conjuntos numéricos y los cambios que experimentan** desde el punto de vista tanto algebraico como topológico. Es similar al Álgebra, pero a diferencia de este, utiliza sucesiones numéricas infinitas.

Combinatoria - La Combinatoria es la rama de las Matemáticas que estudia la construcción y enumeración de configuraciones que hacen posible la existencia de otras condiciones establecidas. En otras palabras, es la disciplina matemática que **inspecciona de cuántas formas posibles se pueden agrupar unos elementos** para obtener el resultado esperado.

Matemática computacional - La Matemática computacional. Es una rama de las matemáticas aplicadas esencial para el funcionamiento de los programas de software/informáticos que existen, pues todos ellos funcionan a partir de **algoritmos que solo entienden el lenguaje matemático**.

Física Matemática - La Física Matemática es la disciplina que marca la conexión entre la Física, una ciencia natural que estudia la naturaleza de la materia y de la energía, y las Matemáticas, una ciencia formal. Ambas ciencias están estrechamente unidas ya que las predicciones matemáticas permiten **estudiar, de forma cuantitativa, los fenómenos físicos que tienen lugar en el Universo**. Sin matemáticas, no habría física.

Teoría de números - La Teoría de números es la rama de las Matemáticas que **estudia las propiedades de los números enteros**

Ejemplos de aplicación en campo de software, astronomía y programación

Tesla

Esto es lo que ve un auto que se conduce solo de Tesla.

El Software analiza las imágenes de sus sensores y cámaras para detectar objetos, calcular distancias y profundidad. La inteligencia artificial evalúa el diseño de las carreteras, la infraestructura estática y los objetos en movimiento para aprender de distintos escenarios y tomar decisiones en tiempo real.

El piloto automático está diseñado para usarse solo con un conductor totalmente atento que tiene las manos en el volante y está preparado para asumir el control en cualquier momento. En su forma actual no es un sistema de conducción autónoma, no convierte un Tesla en un vehículo autónomo y no permite que el conductor renuncie a la responsabilidad. **Cuando se usa correctamente, el piloto automático reduce la carga de trabajo general del conductor, y la presencia de ocho cámaras externas, radar y 12 sensores ultrasónicos para realizar el cálculo, evaluación y seguir con el aprendizaje de su IA**

Nasa

SpaceX y la NASA realizan con éxito el histórico lanzamiento de la cápsula Crew Dragon hacia la Estación Espacial Internacional, esto fue en 2020

Los astronautas Bob Behnken y Doug Hurley viajan hacia la EEI en la cápsula Crew Dragon, construida por SpaceX, que actúa como una suerte de taxi contratado por la NASA para que lleve a sus astronautas hasta la estación en órbita.

La nave tardó nueve minutos en alcanzar su órbita inicial

Crew Dragon en órbita

Posteriormente, la cápsula Crew Dragon se separó del cohete para dirigirse hacia la EEI.

Velocidad en órbita: 7,66 km/s
Velocidad máxima: 28.000 km/h
Altura de órbita: 408 km

Ejemplo en whiteboard dibujado de como alcanza el cohete a la estación simple xd

¿Por qué Python?

Python es el mejor lenguaje de programación tanto para aprender cuando se es **principiante**, como para especializarse cuando se tiene mucha más **experiencia**, dado que une la **simpleza**, **elegancia** y **eficacia**.

Características de Python

Es un lenguaje de programación multiparadigma, fuertemente tipado, de tipado dinámico y de sintaxis clara.

Es el único lenguaje con el potencial de poder crear scripts en segundos y a la vez responsable de aplicaciones gigantescas como **Instagram** o **YouTube**.

Todos los bloques de código se separan por indexación lo que consigue **claridad** y **elegancia**.

Los programas escritos en Python **se asemejan más a escribir una receta en inglés que a la programación de algoritmos**, cosa que en otros lenguajes pues es muy difícil

Operadores aritméticos

Los operadores aritméticos o *arithmetic operators* son los más comunes que nos podemos encontrar, y nos **permiten realizar operaciones aritméticas** sencillas, como pueden ser la suma, resta o exponente. A continuación, condensamos en la siguiente tabla todos ellos con un ejemplo, donde `x=10` y `y=3`.

Operador	Nombre	Ejemplo
+	Suma	$x + y = 13$
-	Resta	$x - y = 7$
*	Multiplicación	$x * y = 30$
/	División	$x/y = 3.333$
%	Módulo	$x\%y = 1$
**	Exponente	$x ** y = 1000$
//	Cociente	3

```
x = 10; y = 3
print("Operadores aritméticos")
print("x+y =", x+y)      #13
print("x-y =", x-y)      #7
print("x*y =", x*y)      #30
print("x/y =", x/y)      #3.3333333333333335
print("x%y =", x%y)      #1
print("x**y =", x**y)    #1000
print("x//y =", x//y)    #3
```

Operador +

El operador `+` suma los números presentes a la izquierda y derecha del operador. Recalcamos lo de números porque no tendría sentido sumar dos cadenas de texto, o dos listas, pero en Python es posible hacer este tipo de cosas.

```
print(10 + 3) # 13
```

Es posible sumar también dos cadenas de texto, pero la suma no será aritmética, sino que se unirán ambas cadenas en una. También se pueden sumar dos listas, cuyo resultado es la unión de ambas.

```
print("2" + "2")      # 22
print([1, 3] + [6, 7]) # [1, 3, 6, 7]
```

Operador -

El operador `-` resta los números presentes a la izquierda y derecha del operador. A diferencia el operador `+` en este caso no podemos restar cadenas o listas.

```
print(10 - 3) #7
```

Operador *

El operador `*` multiplica los números presentes a la izquierda y derecha del operador.

```
print(10 * 3) #30
```

Como también pasaba con el operador `+` podemos hacer cosas “raras” con `*`. Explicar porque pasan estas cosas es un poquito más complejo, por lo que lo dejamos para otro capítulo, donde explicaremos como definir el comportamiento de determinados operadores para nuestras clases.

```
print("Hola" * 3) #HolaHolaHola
```

Operador /

El operador `/` divide los números presentes a la izquierda y derecha del operador. Un aspecto importante a tener en cuenta es que si realizamos una división cuyo resultado no es entero (es decimal) podríamos tener problemas. En Python 3 esto no supone un problema porque el mismo se encarga de convertir los números y el resultado que se muestra si es decimal.

```
print(10/3) #3.3333333333333335
print(1/2) #0.5
```

Sin embargo, en Python 2, esto hubiera tenido un resultado diferente. El primer ejemplo `10/3=3` y el segundo `1/2=0`. El comportamiento realmente sería el de calcular el cociente y no la división.

Para saber más: Si quieres saber más acerca de este cambio del operador de división, puedes leer la [la PEP238](#)

Operador %

El operador `%` realiza la operación módulo entre los números presentes a la izquierda y la derecha. Se trata de calcular el resto de la división entera entre ambos números. Es decir, si dividimos 10 entre 3, el cociente sería 3 y el resto 1. Ese resto es lo que calcula el módulo.

```
print(10%3) # 1
print(10%2) # 0
```

Operador `**`

El operador `**` realiza el exponente del número a la izquierda elevado al número de la derecha.

```
print(10**3) #1000
print(2**2)  #4
```

Si ya has usado alguna vez Python, tal vez hayas oído hablar de la librería `math`. En esta librería también tenemos una función llamada `pow()` que es equivalente al operador `**`.

```
import math
print(math.pow(10, 3)) #1000.0
```

Para saber más: Puedes consultar más información de la librería `math` [en la documentación oficial de Python](#)

Operador `//`

Por último, el operador `//` calcula el cociente de la división entre los números que están a su izquierda y derecha.

```
print(10//3) #3
print(10//10) #1
```

Tal vez te hayas dado cuenta que el operador cociente `//` está muy relacionado con el operador módulo `%`. Volviendo a las lecciones del colegio sobre la división, recordaremos que el Dividendo `D` es igual al divisor `d` multiplicado por el cociente `c` y sumado al resto `r`, es decir `D=d*c+r`. Se puede ver como en el siguiente ejemplo, `10//3` es el cociente y `10%3` es el resto. Al aplicar la fórmula, verificamos que efectivamente `10` era el dividendo.


```
D = 10 # Número que queremos dividir
d = 3  # Número entre el que queremos dividir
print(3 * (10//3) + 10%3) # 10
```

Orden de aplicación

En los ejemplos anteriores simplemente hemos aplicado un operador a dos números sin mezclarlos entre ellos. También es posible tener varios operadores en la misma línea de código, y en este caso es muy importante tener en cuenta las prioridades de cada operador y cual se aplica primero. Ante la duda siempre podemos usar paréntesis, ya que todo lo que está dentro de un paréntesis se evaluará conjuntamente, pero es importante saber las prioridades.

El orden de prioridad sería el siguiente para los operadores aritméticos, siendo el primero el de mayor prioridad:

- `()` Paréntesis
- `**` Exponente
- `-x` Negación
- `*` `/` `//` Multiplicación, División, Cociente, Módulo
- `+` `-` Suma, Resta

```
print(10*(5+3)) # Con paréntesis se realiza primero la suma
# 80
print(10*5+3)   # Sin paréntesis se realiza primero la multiplicación
# 53
print(3*3+2/5+5%4) # Primero se multiplica y divide, después se suma
#10.4
print(-2**4)     # Primero se hace la potencia, después se aplica el signo
#-16
```

Algebra Python

Una de las herramientas matemáticas más utilizadas en machine learning y data mining es el Álgebra lineal; por tanto, si queremos incursionar en el fascinante mundo del aprendizaje automático y el análisis de datos es importante reforzar estos conceptos.

En la universidad vemos lo que es Álgebra lineal es una rama de las matemáticas que es sumamente utilizada en el estudio de una gran variedad de ciencias, como ser, ingeniería, finanzas, entre otras . Es una extensión del álgebra que aprendemos en la escuela secundaria, hacia un mayor número de dimensiones; en lugar de trabajar con incógnitas a nivel de escalares comenzamos a trabajar con matrices y vectores.

El estudio del Álgebra lineal implica trabajar con varios objetos matemáticos, como ser:

- **Los Escalares:** Un *escalar* es un solo número, en contraste con la mayoría de los otros objetos estudiados en Álgebra lineal, que son generalmente una colección de múltiples números.

Los Vectores: Un vector es una serie de números. Los números tienen un orden preestablecido, y podemos identificar cada número individual por su índice en ese orden. Podemos pensar en los vectores como la identificación de puntos en el espacio, con cada elemento que da la coordenada a lo largo de un eje diferente. Existen dos tipos de vectores, los vectores de fila y los vectores de columna. Podemos representarlos de la siguiente manera, donde f es un vector de fila y c es un vector de columna:

$$f = [0 \quad 1 \quad -1]; c = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

Las Matrices: Una matriz es un arreglo bidimensional de números (llamados entradas de la matriz) ordenados en filas (o renglones) y columnas, donde una fila es cada una de las líneas horizontales de la matriz y una columna es cada una de las líneas verticales. En una matriz cada elemento puede ser identificado utilizando dos índices, uno para la fila y otro para la columna en que se encuentra. Las podemos representar de la siguiente manera, A es una matriz de 3x2.

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 2 \\ -2 & 3 \end{bmatrix}$$

Sobre estos temas podemos realizar las operaciones matemáticas básicas, como ser suma, multiplicación, resta y división, es decir que vamos a poder sumar vectores con matrices, multiplicar escalares a vectores y demás.

Vectores

Un vector de largo n es una secuencia (o array, o tupla) de n números. La solemos escribir como $x=(x_1,...,x_n)$ o $x=[x_1,...,x_n]$

En Python, un vector puede ser representado con una simple lista, o con un array de Numpy; siendo preferible utilizar esta última opción.

EJEMPLOS EN PYTHON

Operaciones con vectores

Las operaciones más comunes que utilizamos cuando trabajamos con vectores son la suma, la resta y la multiplicación por escalares.

Cuando sumamos dos vectores, vamos sumando elemento por elemento de cada vector.

$$x + y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} := \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

Resta de Vectores

$$x - y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} := \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \\ \vdots \\ x_n - y_n \end{bmatrix}$$

La multiplicación por escalares es una operación que toma a un número γ , y a un vector x y produce un nuevo vector donde cada elemento del vector x es multiplicado por el número γ .

$$\gamma x := \begin{bmatrix} \gamma x_1 \\ \gamma x_2 \\ \vdots \\ \gamma x_n \end{bmatrix}$$

MATRICES

Las matrices son una forma clara y sencilla de organizar los datos para su uso en operaciones lineales.

Una matriz $n \times k$ es una agrupación rectangular de números con n filas y k columnas; se representa de la siguiente forma:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{bmatrix}$$

En la matriz A , el símbolo a_{nk} representa el elemento n -ésimo de la fila en la k -ésima columna. La matriz A también puede ser llamada un vector si cualquiera de n o k son iguales a 1. En el caso de $n=1$, A se llama un vector fila, mientras que en el caso de $k=1$ se denomina un vector columna.

Las matrices se utilizan para múltiples aplicaciones y sirven, en particular, para representar los coeficientes de los sistemas de ecuaciones lineales o para representar transformaciones lineales dada una base. Pueden sumarse, multiplicarse y descomponerse de varias formas.

OPERACIONES CON MATRICES

Multiplificación por escalares:

$$\gamma A = \begin{bmatrix} \gamma a_{11} & \cdots & \gamma a_{1k} \\ \vdots & \vdots & \vdots \\ \gamma a_{n1} & \cdots & \gamma a_{nk} \end{bmatrix} := \begin{bmatrix} \gamma a_{11} & \cdots & \gamma a_{1k} \\ \vdots & \vdots & \vdots \\ \gamma a_{n1} & \cdots & \gamma a_{nk} \end{bmatrix}$$

Suma de matrices:

$$A + B = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \vdots & \vdots \\ b_{n1} & \cdots & b_{nk} \end{bmatrix} := \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1k} + b_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} + b_{n1} & \cdots & a_{nk} + b_{nk} \end{bmatrix}$$

Resta de matrices:

$$A - B = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix} - \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \vdots & \vdots \\ b_{n1} & \cdots & b_{nk} \end{bmatrix} := \begin{bmatrix} a_{11} - b_{11} & \cdots & a_{1k} - b_{1k} \\ \vdots & \vdots & \vdots \\ a_{n1} - b_{n1} & \cdots & a_{nk} - b_{nk} \end{bmatrix}$$

Para los casos de suma y resta, hay que tener en cuenta que solo se pueden sumar o restar [matrices](#) que tengan las mismas dimensiones, es decir que si tengo una [matriz](#) A de dimensión 3x2 (3 filas y 2 columnas) solo voy a poder sumar o restar la [matriz](#) B si esta también tiene 3 filas y 2 columnas.

Multiplicación o Producto de matrices

La regla para la multiplicación de matrices generaliza la idea del producto interior que vimos con los vectores; y está diseñada para facilitar las operaciones lineales básicas. Cuando multiplicamos matrices, el número de columnas de la primera matriz debe ser igual al número de filas de la segunda matriz; y el resultado de esta multiplicación va a tener el mismo número de filas que la primera matriz y el número de columnas de la segunda matriz. Es decir, que si yo tengo una matriz A de dimensión 3×4 y la multiplico por una matriz B de dimensión 4×2 , el resultado va a ser una matriz C de dimensión 3×2 .

Algo a tener en cuenta a la hora de multiplicar matrices es que la propiedad conmutativa no se cumple. $A \times B$ no es lo mismo que $B \times A$.

Veamos los ejemplos en Python.

En caso de no tener mismas columnas y filas

Este último ejemplo vemos que la propiedad conmutativa no se cumple, es más, [Python](#) nos arroja un error, ya que el número de columnas de B no coincide con el número de filas de A, por lo que ni siquiera se puede realizar la multiplicación de $B \times A$.

Para una explicación más detallada del proceso de [multiplicación de matrices](#), pueden consultar el siguiente [tutorial](#).

La matriz identidad, la matriz inversa, la matriz transpuesta y el determinante

La matriz identidad es el elemento neutro en la multiplicación de matrices, es el equivalente al número 1. Cualquier matriz multiplicada por la matriz identidad nos da como resultado la misma matriz. La matriz identidad es una matriz cuadrada (tiene siempre el mismo número de filas que de columnas); y su diagonal principal se compone de todos elementos 1 y el resto de los elementos se completan con 0. Suele representarse con la letra I

Por ejemplo la matriz identidad de 3x3 sería la siguiente:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ahora que conocemos el concepto de la matriz identidad, podemos llegar al concepto de la matriz inversa. Si tenemos una matriz A, la matriz inversa de A, que se representa como A^{-1} es aquella matriz cuadrada que hace que la multiplicación $A \times A^{-1}$ sea igual a la matriz identidad I. Es decir que es la matriz recíproca de A.

$$A \times A^{-1} = A^{-1} \times A = I$$

Tener en cuenta que esta matriz inversa en muchos casos puede no existir. En este caso se dice que la matriz es singular o degenerada. Una matriz es singular si y solo si su determinante es nulo.

El determinante es un número especial que puede calcularse sobre las matrices cuadradas. Se calcula como la suma de los productos de las diagonales de la matriz en una dirección menos la suma de los productos de las diagonales en la otra dirección. Se representa con el símbolo $|A|$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$|A| = (a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}) - (a_{31}a_{22}a_{13} + a_{32}a_{23}a_{11} + a_{33}a_{21}a_{12})$$

Por último, la matriz transpuesta es aquella en que las filas se transforman en columnas y las columnas en filas. Se representa con el símbolo A^T

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}^T := \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

SISTEMAS DE ECUACIONES

Una de las principales aplicaciones del Álgebra lineal consiste en resolver problemas de sistemas de ecuaciones lineales.

Una ecuación lineal es una ecuación que solo involucra sumas y restas de una variable o mas variables a la primera potencia.

Para resolver en forma numérica los sistema de ecuaciones, existen varios métodos:

El método de sustitución

El método de igualacion

El método de reduccion

El método gráfico

El método de Gauss

El método de Eliminación de Gauss-Jordan

El método de Cramer

La idea no es explicar cada uno de estos métodos, sino saber que existen y que Python nos hace la vida mucho más fácil, ya que para resolver un sistema de ecuaciones simplemente debemos llamar a la función `solve()`.

Computación Simbólica:

En matemáticas y ciencias de la computación, el álgebra computacional, también conocida como cálculo simbólico, es un área científica que se refiere al estudio y desarrollo de algoritmos/ software para la manipular expresiones matemáticas y otros objetos matemáticos.

Sabiendo esto sabemos que podemos: simplificar expresiones, calcular derivadas, integrales y límites, resolver ecuaciones, trabajar con matrices y mucho, mucho más.

Derivadas

¿ Para que sirven las derivadas?

La derivada te permite conocer lo sensible que es al cambio una variable con respecto a otra. Eso resulta muy útil en ciencias (velocidades, aceleraciones, distribuciones que dependen del tiempo o de la cantidad de materia, son ejemplos sencillos), en ingeniería y en economía.

Para poder calcular derivadas con SymPy, solo hace falta usar el método `diff()`,

Integrales

Las integrales son la herramienta para calcular «El área bajo la curva» como lo describen en ingeniería, se trata pues del espacio comprendido entre el tramo de recta real delimitado por dos puntos y los dos puntos perfectamente paralelos de la curva que esta siendo estudiada.

estos procesos son tan usados en el análisis matemático en los estudios y aplicaciones de la ingeniería,

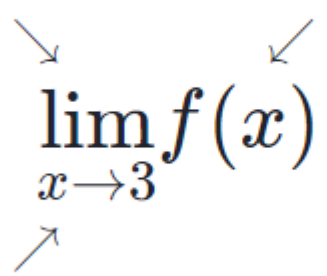
Como pueden observar, el método `integrate()` toma como argumento: La expresión matemática para calcular una integral indefinida.

Al momento de calcular una integral definida debemos pasarle a parte de la expresión matemática, (`variable_integración`, `limite_inferior`, `limite_superior`), como se muestra en el primer ejemplo. En el primer ejemplo como pueden ver, se hace uso de `exp()` el cual es la función exponencial, también vemos `"sympy.oo"`, esto es equivalente al símbolo de infinito.

Límites

Tenemos también una notación especial para hablar de límites. Así es como escribimos el límite de f cuando x se acerca (o tiende a) 3:

"El límite de..." "...la función f ..."


$$\lim_{x \rightarrow 3} f(x)$$

...cuando x tiende a 3."

El símbolo \lim significa que tomamos el límite de algo.

La expresión a la derecha de \lim es la expresión de la cual tomamos el límite. En nuestro caso, se trata de la función f

La expresión $x \rightarrow 3$ que aparece debajo de \lim significa que tomamos el límite de f a medida que los valores de x se acercan a 3

Para calcular límites, SymPy nos proporciona el método `lim`

La sintaxis para usar el método es la siguiente:

$$\lim_{x \rightarrow x_0} f(x)$$

`sympy.limit(f(x), x, x0)`

El primer argumento es la función, el segundo es el símbolo x , y el tercero es hacia donde tiende acercarse x .

Si se dieron cuenta, en el segundo ejemplo, que x tiende al infinito, esto lo hice con el atributo `oo` de SymPy que equivale a ∞ .

Ecuaciones diferenciales ordinarias y parciales

Ecuaciones diferenciales lineales

Series