Introducción al sistema Wolfram Mathematica

Expresiones

Usamos la interfaz gráfica ("Notebook") de Wolfram Mathematica.
Para calcular una expresión en Notebook, hay que oprimir `Shift-Enter` (`Mayús-Intro`).

- `3 * 5` (∗ *oprimir Shift-Enter, esto es, Mayús-Intro, después de cada comando* ∗)
- `% + 7` (∗ *el símbolo* `%` *denota la respuesta anterior* ∗)
- `10 ^ 100` (∗ *WM sabe manejar con números enteros muy grandes* ∗)
- `Factorial[4]` (∗ *argumentos de funciones se escriben entre corchetes* ∗)
- `factorial[4]` (∗ *WM distingue las letras mayúsculas de las minúsculas* ∗)
- `Sin[0]` (∗ *los nombres de las funciones estándares empiezan con mayúsculas* ∗)
- `?Sqrt` (∗ *descripción breve de la función* `Sqrt` ∗)
- `Sqrt[-2]` (∗ *WM sabe manejar con números complejos* ∗)
- `cos[0]` (∗ *¿por qué no quiere simplificar? corrija el error* ∗)
- `Sqrt(9)` (∗ *¿por qué no quiere simplificar? corrija el error* ∗)

Comparación de números

- `5 < 3`
- `9 >= 2`
- `Sqrt[9] == 3` (∗ *la relación de igualdad se denota por* `==`*, como en C* ∗)
- `5 != 7`

Operaciones lógicas

- `Not[5 < 3]`
- `True && False`
- `True || False`
- `5 != 7 && 7 <= 7`

Valores simbólicos y numéricos

Hay una diferencia entre valores simbólicos (exactos) y numéricos (aproximados, representados con el punto flotante):

- `20 / 6` (* *regresa el valor exacto* *)
- `N[20 / 6]` (* *convierte* 20/6 *en la representación numérica aproximada* *)
- `20 / 6.0` (* *regresa el resultado numérico* *)
- `Sqrt[2]` (* *representación exacta* *)
- `% ^ 2`
- `N[Sqrt[2], 50]` (* *representación aproximada con 50 dígitos decimales* *)
- `% ^ 2`

Algunas constantes predefinidas, sus valores exactos y aproximados

- `Pi` (* *representación exacta* *)
- `Sin[Pi]` (* *usa propiedades de la función* sen *y regresa el valor exacto* *)
- `Sin[Pi/4]` (* *usa propiedades de la función* sen *y regresa el valor exacto* *)
- `N[Pi]`
- `N[Pi, 100]`
- `Sin[N[Pi]]` (* *regresa valor muy pequeño, pero no es cero exacto* *)
- `Sin[N[Pi]] == Sin[Pi]`
- `N[E, 20]`
- `Exp[1]`
- `N[%, 20]`

Sustitución, expansión y simplificación

- `ReplaceAll[x ^ 2, x -> a + b]` (* *en la expresión* x^2 *sustituir* x *por* $a + b$ *)
- `Expand[%]`
- `Simplify[%]`
- `Factor[a ^ 3 + b ^ 3]`
- `Simplify[2 Cos[x] Sin[x]]`

Variables

- `a = 3` (∗ *asociar el símbolo* `a` *con el valor* `3` ∗)
- `Expand[(a + b) ^ 2]`
- `Clear[a]` (∗ *"limpiar" el símbolo* `a`*, esto es, borrar su asociación* ∗)
- `Expand[(a + b) ^ 2]`

Listas

Listas (tuplas, arreglos) se escriben usando llaves `{` y `}`:

- `a = {1, 9, 10, 5}` (∗ *definir una lista* ∗)
- `a[[3]]`
- `a[[2]] = 7`
- `a`
- `Length[a]`
- `Join[{7, 8}, {9, 10}]` (∗ *unir dos listas* ∗)
- `b = Append[a, 8]`

Muchas funciones internas de Wolfram Mathematica son *listadas*. Si una función es listada y en vez de un argumento le damos una lista, entonces la función se aplica a cada elemento de la lista y nos regresa la lista de los valores correspondientes:

- `Sqrt[{1, 9, 10, 4}]`

Listas se usan para representar muchos objetos complejos en Wolfram Mathematica, por ejemplo, vectores y matrices:

- `{1, 2, 3} + {4, 5, 6}`
- `{1, 2, 3} * {4, 5, 6}` (∗ *multiplicación por componentes* ∗)
- `{1, 2, 3} . {4, 5, 6}` (∗ *producto punto de dos vectores* ∗)
- `A = {{1, 2}, {2, 3}}` (∗ *definimos una matriz como lista de renglones* ∗)
- `MatrixForm[A]`
- `B = Inverse[A]; MatrixForm[B]`
- `MatrixForm[A . B]` (∗ *aquí el punto denota el producto de matrices* ∗)

Gráficas de funciones

Mathematica sabe dibujar gráficas de funciones, así como curvas definidas por ecuaciones paramétricas y gráficas tresdimensionales. Los dominios de funciones se describen de la siguiente manera:

{*variable*, *valor mínimo*, *valor máximo*}

Ejemplos:

- `Plot[1 / (1 + x ^ 2), {x, -5, 5}]`
- `ParametricPlot[{t*Cos[t], t*Sin[t]}, {t, 0, 6*Pi}]`
  (∗ *espiral de Arquímedes = espiral aritmética* ∗)
- `PolarPlot[1 + Cos[t], {t, 0, 2 Pi}]` (∗ *cardioide* ∗)
- `Plot3D[x ^ 2 - y ^ 2, {x, -1, 1}, {y, -1, 1}]` (∗ *silla* ∗)
- `PolarPlot[Cos[3*t], {t, 0, Pi}]`
- `PolarPlot[t, {t, 0, 6*Pi}]` (∗ *espiral de Arquímedes usando* `PolarPlot` ∗)
- `ParametricPlot3D[{t / 5, Cos[t], Sin[t]}, {t, -10, 10}]` (∗ *hélice* ∗)

Definición de funciones

Para definir una función nueva, se usa el símbolo `:=`.
Preste atención a la notación de los argumentos, especialmente a los guiones bajos.

- `Clear[All]`
- `f[x_] := x ^ 2`
- `f[3]`
- `Expand[f[a + b]]`
- `Plot[f[x], {x, -2, 2}]`
- `Clear[f]` (∗ *después de usar la función f, podemos liberar el símbolo f* ∗)
- `f[3]`
- `silla[u_, v_] := u ^ 2 - v ^ 2` (∗ *función de dos argumentos* ∗)
- `Plot3D[silla[u, v], {u, -1, 1}, {v, -1, 1}]`

Derivación e integración de funciones

- `D[x^p, x]`
- `f[x_] := x * Exp[x]`
- `D[f[x], x]`
- `Integrate[Sin[x], x]`
- `Integrate[Exp[- x ^ 2], x]` (∗ *conoce muchas funciones especiales* ∗)
- `Integrate[Sin[x], {x, -1, 2}]` (∗ *sacar el valor exacto* ∗)
- `NIntegrate[Sin[x], {x, -1, 2}]` (∗ *calcular aproximadamente* ∗)

Estructuras de control

Usamos una expresión condicional para definir una función "por trozos":

- `g[t_] := If[t > 0, t ^ 2, - 3 * t]`
- `{g[10], g[-10]}`
- `Plot[g[u], {u, -2, 2}]`

Los ciclos principales son `Do`, `For` y `While`. Cada uno de los siguientes comandos imprime los primeros 10 números primos:

- `Do[Print[Prime[k]], {k, 10}]`
- `For[k = 1, k <= 10, k++, Print[Prime[k]]]`
- `k = 1; While[k <= 10, Print[Prime[k]]; k++]`

Los comandos en el cuerpo del ciclo se separan con un punto y coma:

- `Do[Print[2]; Print[3], {10}]`

En muchos casos es posible evitar ciclos usando técnicas de *programación funcional*:

- `Map[Prime, Range[10]]`
- `Table[Prime[k], {k, 10}]`
- `Prime[Range[10]]`