
6 Funciones

En *Mathematica* se pueden definir nuevas funciones, una característica que resulta muy útil y conveniente en una muy amplia variedad de contextos.

6.1. Definiendo funciones

- Esto define, en *Mathematica*, la función $f(x) = (x + 1)^2$. (El signo de guión bajo, `_`, que aparece tras la variable en la parte izquierda, es muy importante).

```
f[x_] := (x+1)^2
```

Observa que la definición no produce ninguna respuesta como resultado.

- El argumento pasado a la función f puede ser un número.

```
f[4]
```

- Y también, una variable o cualquier otra expresión.

```
f[a]
```

```
f[x + Sin[x]]
```

- Una vez definida, la función puede utilizarse en cualquier cálculo.

```
Expand[f[x+1+y]]
```

```
Sin[f[x+a]]
```

```
f[f[a]]
```

- Esto muestra la definición de f .

```
?f
```

- Esto libera la definición de f

```
Clear[f]
```

```
?f
```

6.2. Definiciones múltiples

- Una función con dos variables.

```
f[x_, y_] := (x - y)^2 / y
```

```
2 + f[a, b]
```

- Una nueva definición para `f` que, en este caso, sobrescribe a la anterior.

```
f[x_, y_] := (x - y)^4
```

```
2 + f[a, b]
```

- ¿Cuál es la definición de `f`?

```
?f
```

- Esta nueva definición de `f` no sobrescribe la anterior porque tiene otro tipo de argumentos.

```
f[x_] := Sin[x^2 + 1]
```

```
2 + f[a]
```

```
2 + f[a, b]
```

- ¿Cuál es la definición de `f` ahora?

```
?f
```

- El mecanismo de definición múltiple es, a veces, muy conveniente pero frecuentemente conduce a situaciones confusas. Una definición *limpia* de `f`.

```
Clear[f]
```

```
f[x_] := Sin[x^2 + 1]
```

```
2 + f[a]
```

```
?f
```

- Siempre es conveniente anular las definiciones de funciones que no se vayan a usar más. Ello evita una buena cantidad de problemas y errores.

```
Clear[f]
```

6.3. Definiciones condicionales

Mathematica permite definiciones múltiples, mezclando definiciones para expresiones específicas como `f[1]` o `f[a]` con definiciones para variables como `f[x_]`. Esto permite definir muchas funciones matemáticas.

- La definición de una función que presenta una singularidad.

```
f[x_] := Sin[x+1]/x
f[Pi/2-1]
```

- Aquí se obtiene un error porque f no está definida en 0.

```
f[0]
```

- Así que definimos f en cero por el límite.

```
f[0] = 1
f[0]
f[Pi/2-1]
```

La definición anterior es matemáticamente equivalente a

$$f(x) = \begin{cases} \frac{\text{sen}(x+1)}{x} & \text{si } x \neq 0 \\ 1 & \text{si } x = 0 \end{cases}$$

Cuando se hace una secuencia de definiciones, *Mathematica* sigue el principio de aplicar primero las definiciones más específicas y después las definiciones más generales. Los casos especiales son aplicados antes que los casos generales, independientemente del orden en que se definan; un comportamiento que es particularmente importante.

- Una función definida únicamente para valores racionales de la variable.

```
g[n_Rational] := 1
g[-23]
g[2/3]
g[Sqrt[2]]
```

- La definición de g cuando la variables es real.

```
g[r_Real] := 0
g[2/3]
g[Sqrt[2]]
g[3+4I]
```

- La definición de g en cualquier otro caso.

```
g[x_]:= 1/2
g[z]
```

El mecanismo de definiciones condicionales permite definir en *Mathematica* lo que habitualmente se conoce como funciones a trozos.

- Un primer procedimiento para una definición condicional.

```
f[x_]:= If[x < 0, Sin[x], 3x^2 - 1]
f[-2], f[2]
```

- Un segundo procedimiento.

```
g[x_]:= 3 x^2 - 1 /; x >= 0
g[x_]:= Sin[x] /; x < 0
g[-2], g[2]
```

6.4. Definición inmediata y definición diferida

En realidad hay dos formas posibles de definición. La primera, más frecuente, utiliza el operador `:=` y se denomina *definición diferida*; la segunda utiliza el operador `=` y se llama *definición inmediata*. La diferencia entre ambas es muy sutil y a veces causa confusión en el usuario poco experimentado.

- Cuando se utiliza el operador `:=`, la definición de la función no se evalúa (y como consecuencia no se obtiene resultado alguno).

```
ExpandeA[x_]:= Expand[(1+x)^2]
```

- Cuando se utiliza el operador `=`, la definición de la función se evalúa.

```
ExpandeB[x_]= Expand[(1+x)^2]
```

- Cuando se utiliza la función `ExpandeA`, primero se sustituye el argumento en el cuerpo de definición y después se ejecuta el procedimiento de desarrollo de la expresión.

```
ExpandA[a+2]
```

- Sin embargo, `ExpandB` sustituye su argumento en la expresión ya desarrollada, dando una respuesta diferente.

```
ExpandB[a+2]
```

Ambos operadores, `=` y `:=` pueden utilizarse para definir funciones, pero sus significados son diferentes. Hay que ser cuidadoso acerca de cuál utilizar en cada caso. La regla es: ante la duda, usar `:=`.

- Algunas veces, el uso de uno u otro es indiferente.

```

f[x_] := Sin[Log[x]]

f[1+a]

g[x_] = Sin[Log[x]]

g[1+a]

```

- En otras ocasiones, el uso del operador := es imprescindible.

```

Clear[f,g]

f[x_] := Simplify[x]

g[x_] = Simplify[x]

f[Sin[x]^2 + Cos[x]^2]

g[Sin[x]^2 + Cos[x]^2]

```

- El uso del operador := es posiblemente más frecuente, pero también hay ocasiones en las que debe usarse necesariamente el operador =.

```

f[x_] := D[Log[Sin[x]], x]

g[x_] = D[Log[Sin[x]], x]

f[1+a]

g[1+a]

```

Deberías...

- ...saber definir funciones y utilizarlas; al respecto es muy importante que recuerdes el uso de _ para definir las variables;
- ...saber que la orden ? se utiliza para obtener ayuda acerca de una función;
- ...saber que la orden Clear se utiliza para eliminar la definición de una función;
- ...saber cómo hacer definiciones múltiples;
- ...saber cómo hacer definiciones condicionales;
- ...entender muy bien la diferencia en la asignación inmediata (=) y la asignación diferida (:=).