The Top 10 Maple Errors

10. The exponential function is exp.

To express e^x, you must use the **exp** function.

```
> exp(x);
```

$$e^x$$

```
> evalf(exp(1));
```

$$2.718281828$$

It's not **e^x**, because **e** is just another name to Maple.

```
> e^x;
```

$$e^x$$

```
> evalf(e^1);
```

$$e$$

And it's not **exp^x**.

9. Omitting * for multiplication

Maple complains if you try (in 1D Maple input)

```
> a b;
Error, missing operator or `;`
```

More of a problem, because Maple doesn't complain, is

```
> ab;
```

$$ab$$

Maple thinks this is a variable named **ab**.

```
> x(1+x);
```

$$x(1+x)$$

And here it thinks the first x is a function that you're evaluating at 1+x.

In 2D math input, a space is considered as implied multiplication, so this is OK:

```
> a b
```

$$a\,b \qquad (2.1)$$

However, leaving out the space is still bad: Maple thinks this is the variable **ab**:

```
> ab
```

$$ab \qquad (2.2)$$

... and that this is the function x evaluated at 1+x:

```
> x(1+x)
```

$$x(1+x) \qquad (2.3)$$

Moreover, if you do want a function, in 2D math input you'd better not put a space between the function name and the left parenthesis, because this is interpreted as sin * x:

```
> sin (x)
```

$$\sin x \tag{2.4}$$

8. Premature evaluation

Some functions only work with numerical arguments. If **f** is such a function, you must avoid using **f(x)** (e.g. in a plot command) where **x** is a symbolic variable. For example,

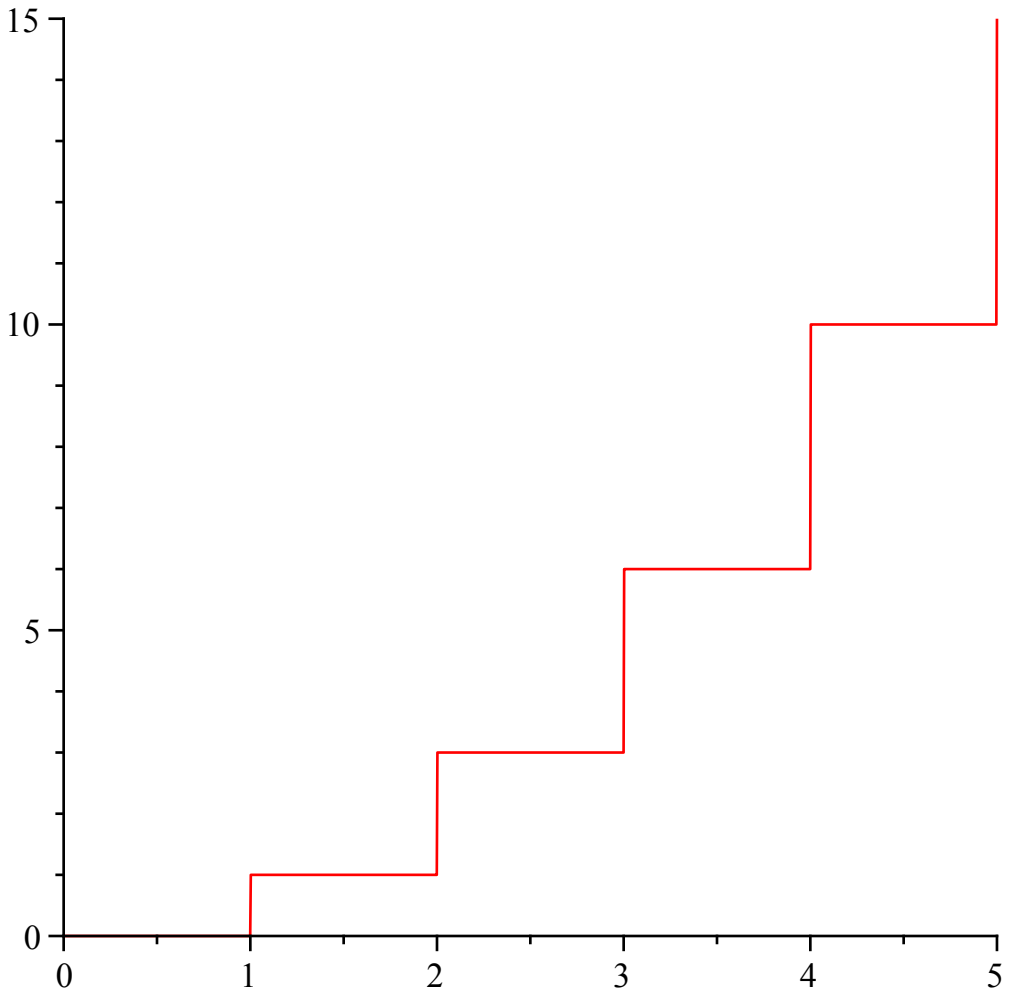
```
> f:= x -> add(j, j=0 .. floor(x));
  plot(f(x),x = 0 .. 5);
```

$$f := x \mapsto add(j, j = 0 \,..\, \lfloor x \rfloor)$$

```
Error, (in f) unable to execute add
```

In this case, you can get around it by using

```
> plot(f, 0..5);
```



7. Those annoying brackets

Every (, [or { needs a matching),] or }. If you miss one, Maple complains. For example,

```
> pointplot([seq([j,f(j)],j=1..10]);
```

Error, `]` unexpected

One way to help with this is to put in the),] or } at the same time as the (,[or {, rather than writing the whole command from left to right.

6. Spelling counts

Maple does not forgive spelling errors. Get one little letter in a name wrong and Maple thinks it's something completely different, probably an undefined variable or function. A common clue that something is wrong is that the name appears in the output.

```
> wiht(plots);
```

$$wiht(plots)$$

"Spelling" includes case: Maple is case-sensitive, so it treats x and X as completely different. Unfortunately, there is no consistent policy on when to use upper and lower case.
One particular case that often arises: the constant 3.14159... must be written as **Pi**, not **pi**.

5. To float or not to float

Maple can use either exact arithmetic, which writes rational numbers as fractions and irrational numbers as symbolic expressions such as $\sqrt{2\,\pi}$, or floating-point arithmetic, which writes numbers such as 2.506628274 using a certain number of decimal places. It is often important to decide which of these is more appropriate in a particular problem. Often there is a tradeoff to consider: exact arithmetic avoids any problems of roundoff error, but can be much slower than floating-point and may result in very complicated expressions.

In a midterm a couple of years ago one student wanted to find $\lim_{n\to\infty} \left(n^4\,(J - S(n))\right)$. If she had used the exact value for J, it would have worked. Unfortunately she used **evalf** to get J, and Maple's result was undefined. Here's a simpler version of that:

```
> limit(n*(1/3 - 1/(3+1/n)),n=infinity);
```

$$\frac{1}{9}$$

```
> limit(n*(evalf(1/3) - 1/(3+1/n)),n=infinity);
```

$$\text{Float}(-\infty)$$

4. Forgotten "with"

If a command is in a certain package, you must load the package using **with** or refer to the command using the package name (e.g. **orthopoly[T]**). Otherwise Maple doesn't know the command.

```
> implicitplot(x^2-y^2,x=-1..1,y=-1..1);
```

$$implicitplot\left(x^2 - y^2, x = -1\,..1, y = -1\,..1\right)$$

3. Variable has been assigned a value

If you want to use something as a symbolic variable, but that variable was already assigned a value, you'll have trouble. Maple sometimes tells you what went wrong. For example:

```
> x := 3;
```

$$x := 3$$

```
> sum(x,x=1..3);
```

Error, (in sum) summation variable previously assigned, second argument evaluates to 3 = 1 .. 3

But sometimes there's no obvious clue.

```
> solve(x^2=4);
```

To see what's going on here you'd have to look at the equation you gave to solve.

```
> x^2=4;
```

$$9 = 4$$

To use x as a variable you'd need to unassign it.

```
> x:= 'x';
```

$$x := x$$

```
> solve(x^2=4);
```

$$2, -2$$

2. Worksheet is out of order

Worksheets are meant to be read (and executed by Maple) from top to bottom. Unfortunately they are not always produced in the same order. When you go back and make changes, it's very easy to produce a worksheet where a certain command won't work properly without a later one. For example:

```
> J:=Int((x^(1/4))/(x^2+1), x=0..1);
```

$$J := \int_0^1 \frac{x^{1/4}}{x^2+1}\, dx$$

```
> Rule[change, x = s^4](%);
```

$$Rule_{change,\, x=s^4}\left(\int_0^1 \frac{x^{1/4}}{x^2+1}\, dx\right)$$

```
> with(Student[Calculus1]):
```

1. Functions versus expressions

This defines an expression:

```
> a := x^2;
```

$$a := x^2$$

And this defines a function:

```
> f := x -> x^2;
```

$$f := x \mapsto x^2$$

It's important to know which you have, and how to use them. For example, you might say

```
> f(t);
```

$$t^2$$

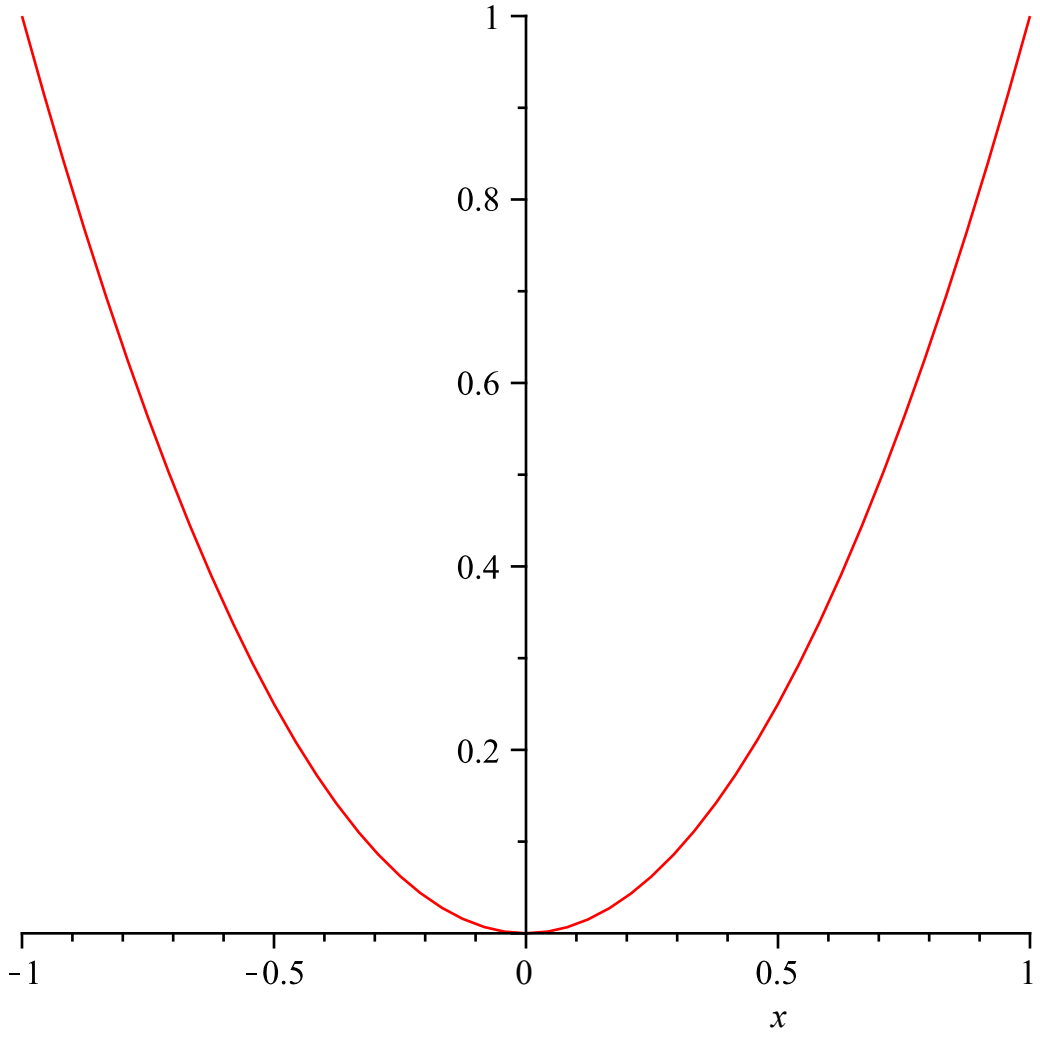
but you don't want

```
> a(t);
```

$$x(t)^2$$

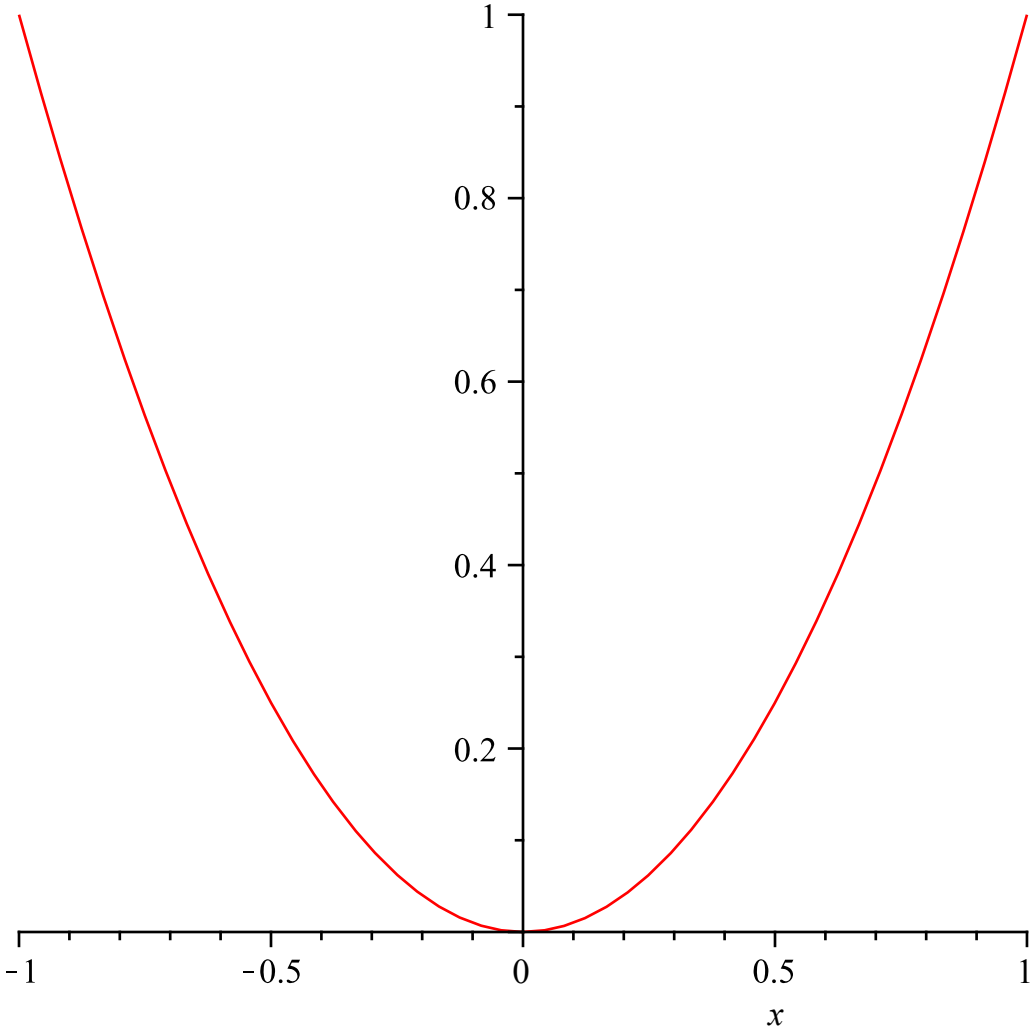
Or in plotting, this would be good:

```
> plot(a, x=-1..1);
```



Or this:

```
> plot(f(x), x=-1..1);
```



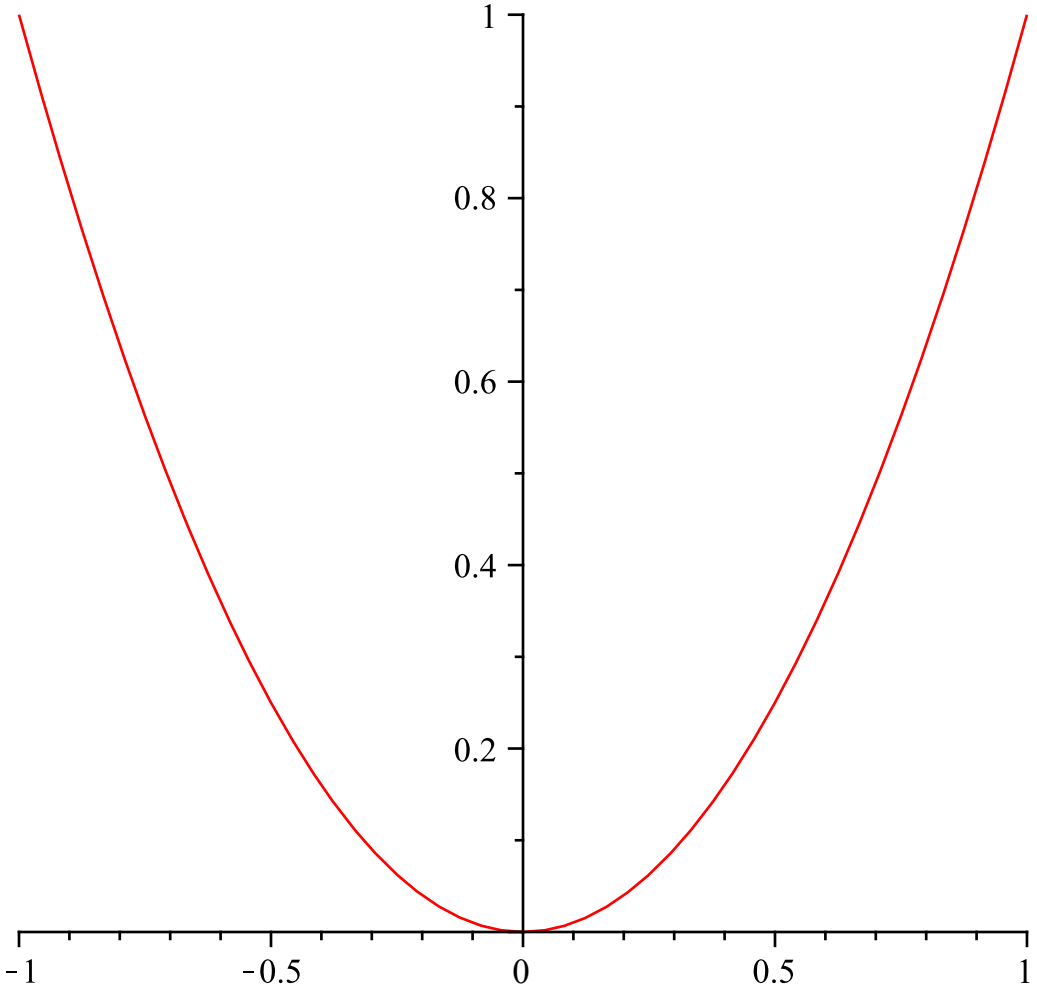
But not this:

```
> plot(f, x=-1..1);
```

```
Error, (in plot) expected a range but received x = -1 .. 1
```

You could use

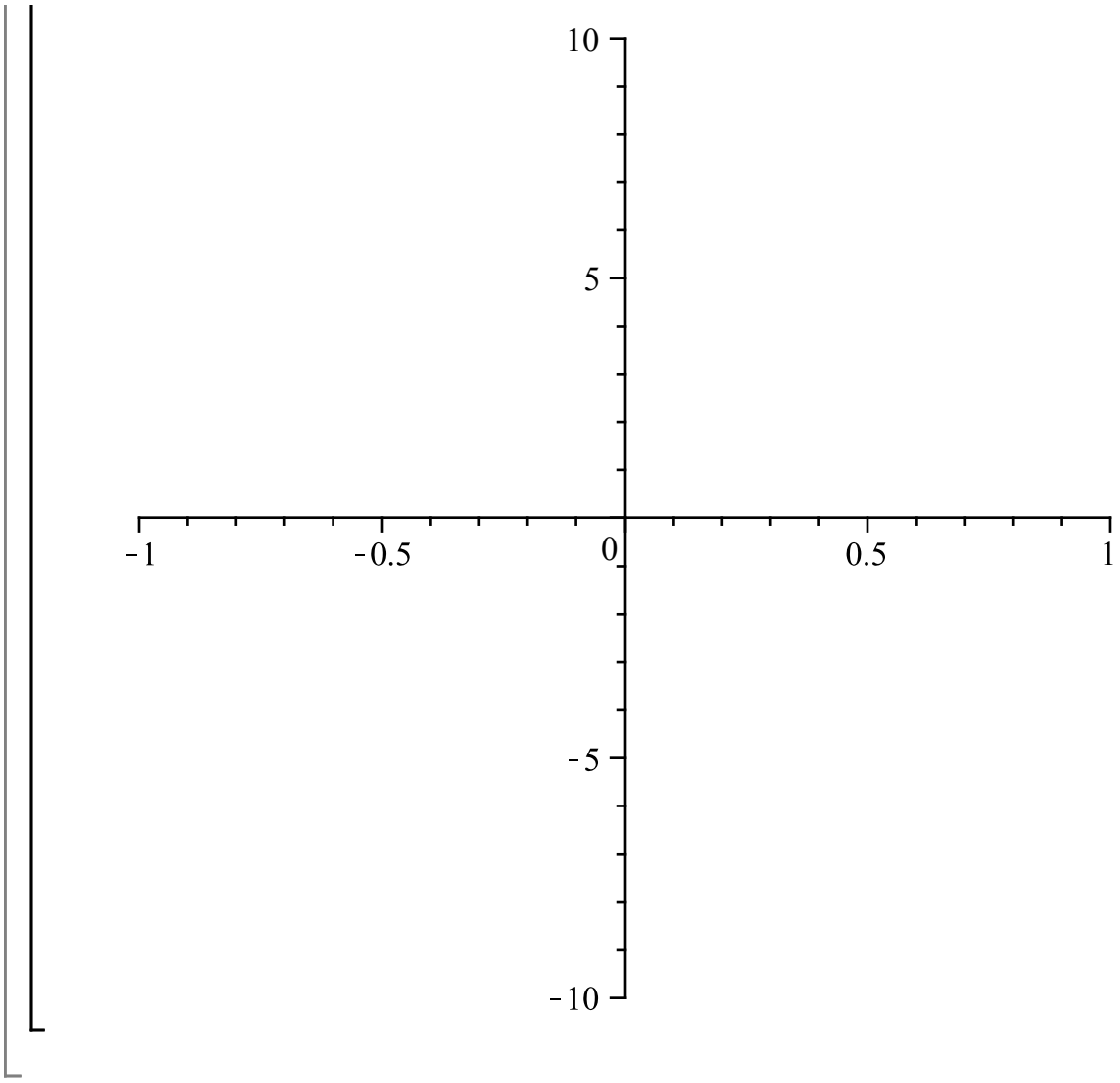
```
> plot(f, -1..1);
```



but not

```
> plot(a,-1..1);
```

Warning, unable to evaluate the function to numeric values in the region; see the plotting command's help page to ensure the calling sequence is correct


10
5
-1
-0.5
0
0.5
1
-5
-10