Mathematica Notes

Mathematica Hints

1. All reserved words begin with capital letters. Eg. E, Sin, Solve, Plot, ...
2. Arguments of all functions are always in square brackets: Sin[x], Solve[x^2==1, x], ... See below for the four uses of brackets.
3.D is used for partial derivatives, but this is the same as (Calc I) derivatives if there is only one variable present. Example: D[x^2*y,x] results in output 2x y (since y was treated as a constant).
4. Spaces can be used for implied multiplication: x*y is the same as x y. But x y is NOT the same as xy. (xy is a singlevari−able name.) Spaces are not required after numbers. For example, 2x y, 2 x y, and 2*x*y are all the same.
5. The output from any *Mathematica* operation can be named for ease of future use. Example: sln=Solve[x^2==1,x]. This should almost always be done when the output from a *Mathematica*
operation will be used at some future point.
6. Substitutions (or replacements): Use of '/.' Example: y=2x+1/.x−>3 results in y being assigned the value of 7, but doesn't permanently change x to 3.

Initial Value Template for Calc II type problems

Solving x'(t) = t, x(0)=2 Comments to be added by the user.

```
Clear[f]

t =.
    Const =.

    t0 = 0

0

    x0 = 2

2

f[t_] := t

t

gensln = Integrate[f[t], t] + Const

Const + t^2/2

Csln = Solve [gensln == x0 /. t -> t0, Const]

{{Const → 2}}
```

```
Csln = Csln[[1]]
```

```
{Const → 2}
```

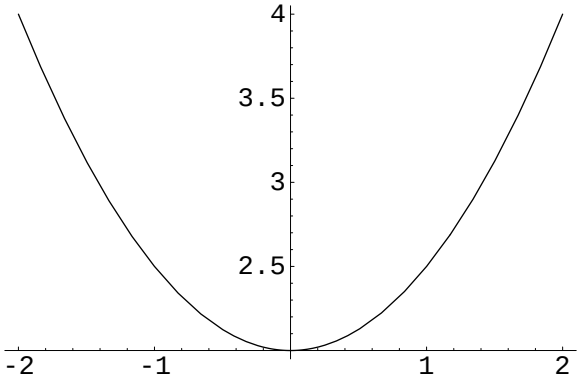
```
Csln = Csln[[1]]
```

```
Const → 2
```

```
sln = gensln /. Csln
```

$$2 + \frac{t^2}{2}$$

```
Plot[sln, {t, -2, 2}]
```




```
- Graphics -
```

From the Mathematica Book:

■ 1.2.5 The Four Kinds of Bracketing in *Mathematica*

Over the course of the last few sections, we have introduced each of the four kinds of bracketing used in *Mathematica*. Each kind of bracketing has a very different meaning. It is important that you remember all of them.

(*term*)	parentheses for grouping
f[*x*]	square brackets for functions
{*a*, *b*, *c*}	curly braces for lists
v[[*i*]]	double brackets for indexing (`Part[v, i]`)

The four kinds of bracketing in *Mathematica*.

When the expressions you type in are complicated, it is often a good idea to put extra space inside each set of brackets. This makes it somewhat easier for you to see matching pairs of brackets. *v*`[[ {a, b} ]]` is, for example, easier to recognize than *v*`[[{a, b}]]`.

■ 1.7.1 Defining Functions

In this part of the book, we have seen many examples of functions that are built into *Mathematica*. In this section, we discuss how you can add your own simple functions to *Mathematica*. Part 2 will describe in much greater detail the mechanisms for adding functions to *Mathematica*.
As a first example, consider adding a function called `f` which squares its argument. The *Mathematica* command to define this function is `f[x_] := x^2`. The _ (referred to as "blank") on the left−hand side is very important; what it means will be discussed below. For now, just remember to put a _ on the left−hand side, but not on the right−hand side, of your definition.

This defines the function `f`. Notice the _ on the left−hand side.

```
f[x_] := x^2
```

`f` squares its argument.

```
f[a+1]
```

```
(1 + a)^2
```

The argument can be a number.

```
f[4]
```

```
16
```

Or it can be a more complicated expression.

```
f[3x + x^2]
```

```
(3 x + x^2)^2
```

You can use `f` in a calculation.

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

```
1 + 2 x + x^2 + 2 y + 2 x y + y^2
```

This shows the definition you made for `f`.

```
?f
```

```
Global`f

f[x_] := x^2
```

`f[x_] := x^2`	define the function `f`
`?f`	show the definition of f
`Clear[f]`	clear all definitions for f

Defining a function in *Mathematica*.

The names like `f` that you use for functions in *Mathematica* are just symbols. Because of this, you should make sure to avoid using names that begin with capital letters, to prevent confusion with built−in *Mathematica* functions. You should also make sure that you have not used the names for anything else earlier in your session.

Mathematica functions can have any number of arguments.

```
hump[x_, xmax_] := (x − xmax)^2 / xmax
```

You can use the `hump` function just as you would any of the built−in functions.

```
2 + hump[x, 3.5]
```

```
2 + 0.285714 (−3.5 + x)^2
```

This gives a new definition for `hump`, which overwrites the previous one.

```
hump[x_, xmax_] := (x − xmax)^4
```

The new definition is displayed.

```
?hump
```

```
Global‘hump

hump[x_, xmax_] := (x − xmax)^4
```

This clears all definitions for `hump`.

```
Clear[hump]
```

When you have finished with a particular function, it is always a good idea to clear definitions you have made for it. If you do not do this, then you will run into trouble if you try to use the same function for a different purpose later in your *Mathematica* session. You can clear all definitions you have made for a function or symbol *f* by using `Clear[f]`.