My Mathematica cheat sheet

Nasser M. Abbasi

August 19, 2025 Compiled on August 19, 2025 at 5:58am

Contents

Chapter 1

introduction

This is my Mathematica cheat sheet. I keep in it useful things I learn about Mathematica and keep forgetting, and some things I see on the net. If something here is from the net, I try to make sure I put a reference or credit to where I saw it, else it will be something I wrote myself, in that case, all blames to me.

This was written using Latex and converted to HTML using tex4ht and to pdf using pdflatex.

1.1 Mathematica links?

Few are here

1. Archive of Mathematica weekly newsletters
2. Mathematica stackexchange forum
3. The large links page for Mathematica at stackexchnage
4. Language reference
5. What is new in V 10
6. Summary of New Features in 10
7. wolfram cloud
8. How to solve basic engineering and mathematics problems using Mathematica and Matlab

Chapter 2

Signal processing

2.1 Mathematica old signal processing package

These are the PDF file of the old Mathematica signal processing package. Currently as of Mathematica version 9, it contains number of build-in DSP functions.

1. introduction_chapter.pdf
2. analyzing_signals_chapter.pdf
3. chapter_5_transforms_chapter.pdf
4. filter_design_chapter.pdf
5. representing_signals_and_systems_chapter.pdf

Chapter 3

Differential equations, DSolve related

3.1 How to parse an ODE?

This code takes one ode, using same input as given to Mathematica's DSolve and verifies it is valid. I use this in my ode solver. This is the very first function called to verify the input is valid.

```
symbolQ[_Symbol] = True;
symbolQ[_]       = False;
has[e_,h_]       := Not[FreeQ[e,h]]

(*This function thanks to Carl Woll,see https://mathematica.stackexchange.com/questions/151850
getPatterns[expr_,pat_]:=Last@Reap[expr/. a:pat:>Sow[a],_,Sequence@@#2&];

(*Version Dec 2, 2024 by Nasser M. Abbasi*)
(*Takes ode and IC, using same syntax as Mathematica's DSolve, and verifies it is valid*)
(*Returns list of ode,IC,ode order if valid, else it generates Abort message*)

parseOneOde[odeAndIc_,y_Symbol[x_Symbol],x_Symbol]:=Module[{ode,ic,constX,alg,LHS,z,n,indeps,d

If[Not[symbolQ[y]],
    Print["Invalid dependent variable given ",y];
    Abort[]
];

If[Not[symbolQ[x]],
     Print["Invalid independent variable given ",x];
     Abort[]
];

If[Head[odeAndIc]===List,
    {constX,ic,alg,ode}=Internal`ProcessEquations`SeparateEquations[Flatten[odeAndIc],{x},{y}]
    ,
    {constX,ic,alg,ode}=Internal`ProcessEquations`SeparateEquations[{odeAndIc},{x},{y}]
];

If[Length[Flatten[constX]]!=0,
     Print["Does not support constraints on independent variable ",constX];
     Abort[]
];

If[Length[Flatten[alg]]!=0,
    Print["Does not support algebratic equations ",alg];
    Abort[]
];

If[Length[ode]>1,
    Print["Does not support more than one ode at this time",ode];
    Abort[]
];

If[Length[ode]==0,
    Print["No ode in ",y[x]," found"];
    Abort[]
];

(*extract the ode from the list now we know there is only one*)
ode=First[Flatten@ode];

(*this makes it easier to parse*)
LHS=First@ode-Last@ode;

If[FreeQ[LHS,y],
     Print["Error: ode ",ode," has no ",y];
     Abort[]
];

If[FreeQ[LHS,x],
    Print["Error. ode ",ode," has no ",x];
    Abort[]
];
```

```
deps=getPatterns[LHS,y[]];

If[deps=!={},
     Print["makeODE:: Can not have ",y[]," in ",ode];
     Abort[]
];

deps=getPatterns[LHS,y[z_.]];

(*this checks there is no y on its own*)
If[Length[Cases[Variables[LHS],Verbatim[y]]]!=0,
     Print["makeODE:: Can not have ",y," with no argument ",x," in ode ",ode];
     Abort[]
];

(*this code checks all the independent variable in y[x] are x*)
If[deps=!={},
     deps=Union[Cases[deps,y[z_]:>z]];(*this extracts all the independent variables*)
     If[Length[deps]>1,
         Print["makeODE:: independent variable must be ",x," in ",ode];
         Abort[]
     ];

     If[First[deps]=!=x,
         Print["makeODE:: independent variable must be ",x," in ",ode];
         Abort[]
     ]
];

der = getPatterns[LHS,Derivative[n_][y][x]];

If[der==={},
    Print["makeODE::No differential equation found in ",ode];
    Abort[]
];

der = getPatterns[LHS,Derivative[n_][y][_]];

indeps = Union@Cases[der,Derivative[n_][y][z_]:>z];

If[Or[Length@indeps>1,First@indeps=!=x],
     Print["Found wrong independent variable in derivative in ",ode];
     Abort[]
];
```

```

ic = Union@ic;

odeOrder = First@Flatten@Internal`ProcessEquations`DifferentialOrder[ode,{x},{y}];

If[Length@ic>0,
    If[Not[AllTrue[ic,Head[#]===Equal&]],
         Print["Invalid initial conditions found in ",ic];
         Abort[]
    ];

    If[Length@ic>odeOrder,
         Print["Too many initial conditions given for the order of the ode ",ic];
         Abort[]
    ]
];

Print["Successful Parsing"];
Print["ode=",ode];
Print["y=",y];
Print["x=",x];
Print["ode order=",odeOrder];
Print["IC=",ic];

{ode,ic,odeOrder}
]
```

These are examples using the above function

```
parseOneOde[{y''[x] == y[x]*Sin[x], y[0] == 0, y'[0] == 1}, y[x], x]

    {y''[x] == Sin[x] y[x], {y[0] == 0, y''[0] == 1}, 2}
```

```
parseOneOde[{y''[x] == y[x]*Sin[x], y[x] == 0, y'[0] == 1}, y[x], x]

    "Does not support algebratic equations ", {y[x] == 0}
    $Aborted
```

```
parseOneOde[{y''[x] == y[]*Sin[x], y[Pi] == 0, y'[0] == 1}, y[x], x]

    makeODE:: Can not have y[] in y''[x]==Sin[x] y[]
    $Aborted
```

```
parseOneOde[{Sin[x] y''[x] + Derivative[y[x], {x, 7}] == y[x]*Sin[x] - y[x], y[Pi] == 0, y'[0]

   {Derivative[y[x],{x,7}]+Sin[x] y''[x]==-y[x]+Sin[x] y[x],{y[Pi]==0,y'[0]==1},2}
```

```
parseOneOde[{Sin[x] y''[x] == y[z]*Sin[x] - y[x], y'[0] == 1}, y[x],  x]

   makeODE:: independent variable must be , x, in , Sin[x] Derivative[2][y][x] == -y[x] + Sin[
   $Aborted
```

3.2 Find the dependent variable of an ode

Given a single ode, how to find the dependent variable?

```
Internal`ProcessEquations`FindDependentVariables[y''[x] + y[x]*y'[x] == x, x]

   {y}
```

```
Internal`ProcessEquations`FindDependentVariables[x'[t] == 0, t]

   {x}
```

Notice the second argument must be the independent variable.

3.3 How to find if ode is linear or not?

There is no builtin function I could find in Mathematica to do this. But one way is the following. We will have a function that returns the coefficients of the ode and the RHS. i.e. given an ode $Ay'' + By' + Cy = f(x)$, it will return a list $\{A, B, C, f\}$. Next to check if the ode is linear or not, all what we have to do is check that A, B, C do not have y in them. This include y, y', y'' and so on.

So for a first order ode, the result of calling this function, which we will get `getODEcoeff` is always a list. The Length of the list is 2 more than the order of the ode. If a coefficient is missing, zero is placed in that slot. For example, for `3 y’’[x] + x y[x]==Sin[x]` then `getODEcoeff[ode,y[x],x]` will return `{3,0,x,Sin[x]` and for `x y’’’’[x] + x y[x]==Sin[x]` it will return `{x,0,0,0,x,Sin[x]}` and for first order ode `y’[x]+f[x]*y[x]==9` it will return

`{1,f[x],9}`.

Now we simply check that all element of this list have no y in them. Here is the function

(to do)

3.4 DSolveIntegrals package

Where did I get this from?

"Mathematica can handle partial differential equations via the DSolveIntegrals package. These arise in chemical contexts in the 1D wave equation, 3D wave equation, 3D diffusion equation, Time-dependent and Time independent Schrödinger equation. Hermite showed that the quintic equation could be solved by elliptic functions"

3.5 Extracting DSolve solutions

one way

```
Clear[y, x]
DSolve[{y'[x]^2 == 1 - y[x]^2, y[0] == 0}, y[x], x]

{{y[x] -> -Sin[x]}, {y[x] -> Sin[x]}}
```

And now

```
\verb|y[x] /. %|

   {-Sin[x],Sin[x]}
```

3.6 Solving ODE using power series method

For an ode, that has just ordinary point at $x = x_0$, we can find power series solution near x_0 as follows. Assume the ode is

$$u''(t) + \frac{1}{10}u(t)^3 u'(t)^2 + 4u(t) = 0$$

with intitial conditions $u(0) = 1, u'(0) = 1$, then

```
findSeriesSolution[t_, nTerms_] := Module[{pt = 0, u, ode, s0, s1, ic, eq, sol},
  ic = {u[0] -> 1, u'[0] -> 1};
  ode = u''[t] + 4 u[t] + 1/10 u[t]^3 u'[t]^2;
  s0 = Series[ode, {t, pt, nTerms}];
  s0 = s0 /. ic;
  roots = Solve@LogicalExpand[s0 == 0];
  s1 = Series[u[t], {t, pt, nTerms + 2}];
  sol = Normal[s1] /. ic /. roots[[1]]
  ]
```

and now call it with

```
seriesSol = findSeriesSolution[t, 6]
```

It returns

$$\frac{445409479t^8}{840000000}+\frac{8878343t^7}{10500000}-\frac{277427t^6}{600000}-\frac{12569t^5}{50000}+\frac{1607t^4}{2000}-\frac{29t^3}{50}-\frac{41t^2}{20}+t+1$$

Update: As of recent versions, we can just do this

```
ode = u''[t] + 4 u[t] + 1/10 u[t]^3*u'[t]^2 == 0;
ic = {u[0] == 1, u'[0] == 1};
AsymptoticDSolveValue[{ode, ic}, u[t], {t, 0, 6}]
```

$$-\frac{277427t^6}{600000}-\frac{12569t^5}{50000}+\frac{1607t^4}{2000}-\frac{29t^3}{50}-\frac{41t^2}{20}+t+1$$

3.7 Replacing y, y', y'' in an equation

Suppose we have $u''(t)+u'(t)+u(t)=3\cos(2t)$ and we wanted to find a particula solution by replacing u in the differential equation by some guess for a particular solution. Then do

```
ode = Derivative[2][u][t] + Derivative[1][u][t] + u[t] == 3*Cos[2*t];
ode /. u -> (c1*Cos[#1] + c2*Sin[#1] & )

Out[2]= c2*Cos[t] - c1*Sin[t] == 3*Cos[2*t]
```

3.8 How to classify singular points for ODE?

Given an ode, such as $x(1-x)y''(x)+(c-(a+b+1)x)y'(x)-aby(x)=0$ and we want to classify the singular points (finite and infinite).

We write it as $y''(x)+p(x)y'(x)+q(x)=0$ and then follow standard procedures. In this example, we want to classify points $0,1,\infty$.

This small function I wrote for a HW will do this. Pass it $p(x),q(x)$ and list of the points to classify.

```
checkForSingularity[p_, q_, at_List, x_] := Module[{p0, q0, t, r, r0},
  r = First@Last@Reap@Do[
       If[at[[n]] === Infinity,
        p0 = p /. x -> (1/t);
        p0 = (2 t - p0)/t^2;
        q0 = (q /. x -> (1/t))/t^4;
        r0 = checkForSingularity[p0, q0, {0}, t];
        Sow[Flatten[{Infinity, Rest[Flatten@r0]}]];
        ,
        r0 = {at[[n]], Limit[(x - at[[n]])  p, x -> at[[n]]],
          Limit[(x - at[[n]])^2 q, x -> at[[n]]]};
        Sow@r0
        ]
       , {n, 1, Length@at}
       ];
  r
  ]
```

To use it on the above example

```
ClearAll[c, a, b, x];
m = checkForSingularity[(c - (a + b + 1) x)/(x (1 -
        x)), (-a b)/(x (1 - x)), {0, 1, Infinity}, x];
Grid[Join[{{"point", "limit x p(x)", "limit x^2 q(x)"}}, m],
 Frame -> All]
```

Gives

```
{{"point", "limit x p(x)", "limit x^2 q(x)"},
 {0,            c,                 0},
 {1,       1 + a + b - c,          0},
 {\[Infinity], 1 - a - b,        a b}
}
```

Since all points have finite limits, then all $0, 1, \infty$ are removable singularities.

I did not test this function too much, but it works for the HW I did :)

3.9 How to replace $y(x)$ by $e^{s(x)}$ in an ode?

Given ode $y^{'''} = xy(x)$ we want to replace $y(x)$ by $e^{s(x)}$

```
ClearAll[y,x,s]
eq = y'''[x] == x y[x]
eq = eq /. y -> Function[{x}, Exp[s[x]]]
Simplify@Thread[eq/Exp[s[x]], Equal]
```

Gives $x = s^{'''}(x) + s'(x)^3 + 3s'(x)s''(x)$

3.10 rewrite ODE as $y'' =$ RHS

Given `y’[x]+3 Sin[x]-3-Cos[x]-2 y[x]==0` how to rewrite it to be `y’[x]==3+Cos[x]-3 Sin[x]+2 y[x]` i.e. put `y’[x]` on one side, and the rest on the other side?

```
ClearAll[f,x,y]
expr=y'[x]+3 Sin[x]-3-Cos[x]-2 y[x]==0;
lhs=expr/.lhs_==rhs_:>lhs;
rhs=expr/.lhs_==rhs_:>rhs;
rhs=-(lhs/.(y'[x]+any_.):>any)+rhs;
expr=y'[x]==rhs
```

This makes it easier to make it in form $y'(x) = f(x, y)$.

3.11 How to trace DSolve running?

Use this

```
Block[{DSolve`print=Print},
    DSolve[{ode,ic},y[t],t]
]
```

3.12 How to put time out on integrate in DSolve?

see https://mathematica.stackexchange.com/questions/120364/why-cant-dsolve-find-a-solution-for-this-ode/120650#120650 by Michael E2.

```
ClearAll[withTimedIntegrate];
SetAttributes[withTimedIntegrate, HoldFirst];
withTimedIntegrate[code_, tc_] := Module[{$in},
   Internal`InheritedBlock[{Integrate},
    Unprotect[Integrate];
    i : Integrate[___] /; ! TrueQ[$in] :=
     Block[{$in = True},
      TimeConstrained[i, tc, Inactivate[i, Integrate]]
      ];
    Protect[Integrate];
    code
    ]
   ];

withTimedIntegrate[{dsol} = DSolve[ode == 0, y, x], 1]; // AbsoluteTiming
dsol
```

3.13 How to find the order of an ODE?

Use

```
Clear["Global`*"];
ode=y''[x]+3*y'[x]+3==0;
Internal`ProcessEquations`DifferentialOrder[ode,{x},{y}]

(*   {{2}} *)
```

So it is order 2. Notice the input must be lists `[ode,{x},{y}]`, it will hang if you use `[ode,x,y]`

3.14 How to find the degree of an ODE?

Degree of an ode is the degree of the highest derivative in the oder. This needs a helper function

```
(*This function thanks to  Carl Woll, see https://mathematica.stackexchange.com/questions/1518
Clear["Global`*"];
getPatterns[expr_, pat_] :=   Last@Reap[expr /. a : pat :> Sow[a], _, Sequence @@ #2 &];
```

And now do

```
getDegreeOfOde[ode_,y_,x_]:=Module[{maxDer,p,d,der},
    der    = getPatterns[ode,Power[Derivative[_.][y][x],_.]] ;
    p      = Flatten[Internal`ProcessEquations`DifferentialOrder[#,{x},{y}]&/@der];
    maxDer = Extract[der,Position[p,Max[p]]];
    Abs[First[maxDer/.(Derivative[_.][y][x])^(d_.):>d]]
]
```

Examples of usage

```
ode=y'''[x]+y''[x]^4+3*y'[x]^8+3*y[x]^8==0;
getDegreeOfOde[ode,y,x]
   (*1*)

ode=y''[x]^4+3*y'[x]^8+3*y[x]^8==0;
getDegreeOfOde[ode,y,x]
   (*4*)

ode=y''[x]+3*y'[x]^8+3*y[x]^8==0;
getDegreeOfOde[ode,y,x]
   (*1*)
```

3.15 How to move all terms with y to one side of ode?

Given an ode (or any equation), and we want to move all terms with $y(x)$ to left side and everything else to right side. This makes it easier to see the forcing function. Select is used for this

```
moveAllYToOneSide[ode_Equal,y_Symbol,x_Symbol]:=Module[{expr=ode,termsWithNoY,termsWithY},
expr=expr[[1]]-expr[[2]];
termsWithNoY=Select[expr,FreeQ[#,y]&];
termsWithY=Select[expr,Not[FreeQ[#,y]]&];
expr=termsWithY==-termsWithNoY
]
```

Call it as

```
ode=y[x]*y'[x]+Sin[x]+3-1/y[x]==Sin[y[x]]+Pi*x*y[x];
moveAllYToOneSide[ode ,y,x]
```

Gives

```
-Sin[y[x]]-1/y[x]-Pi x y[x]+y[x] y'[x]==-3-Sin[x]
```

Chapter 4

Dynamics, Manipulate, animations and related

4.1 Making animated GIF of a manipulate

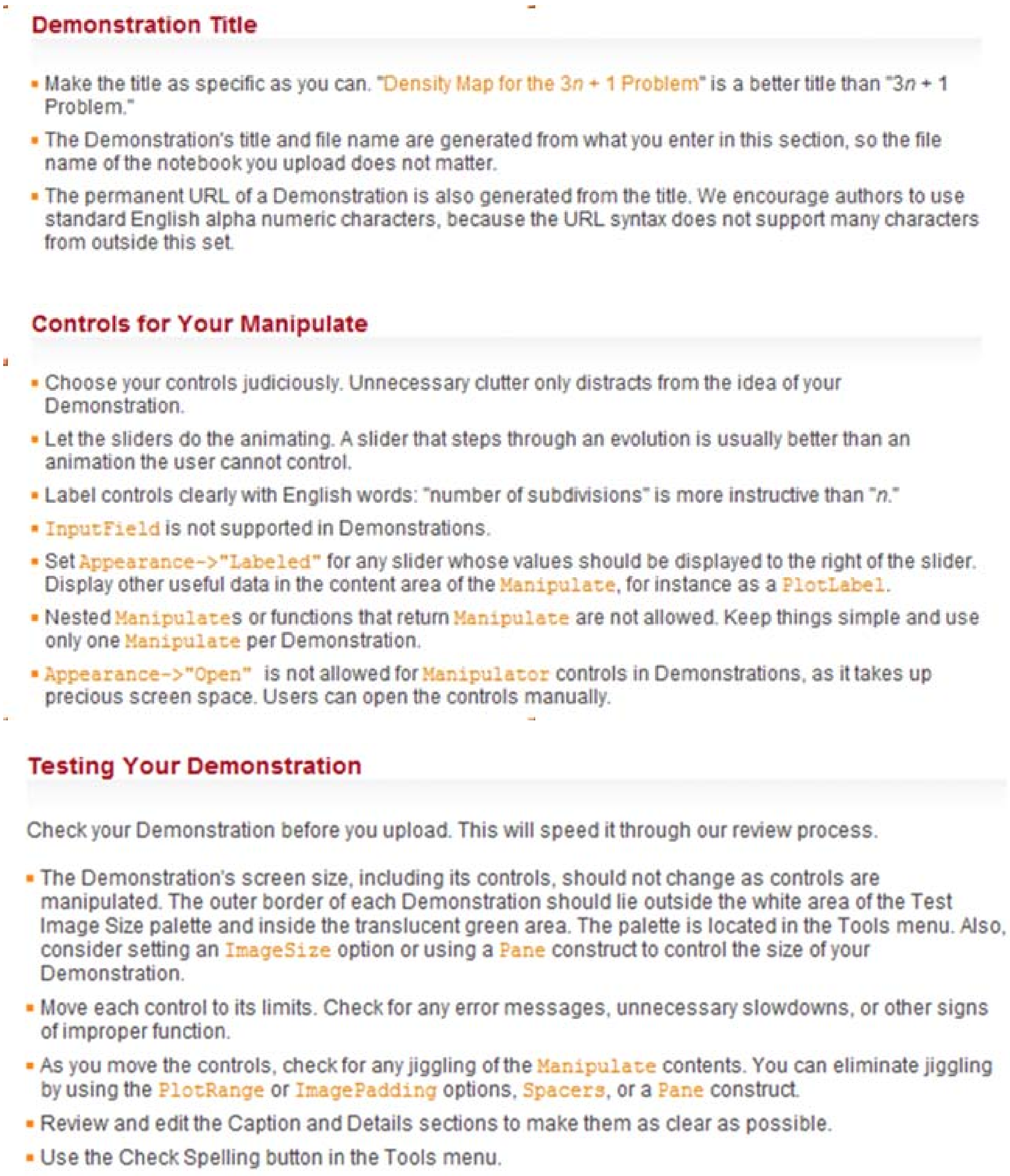
One way is to use Vitaliy Kaurov ManToGif. Another way is to run the manipulate and do screen capture using program such as `LICEcap`

4.2 Guidelines for writing demonstration

```
In math italicize single Roman letters that are variables or
functions  (example, x,y,f(x),t
2. Exception to above is capital letters for points like P and Q in
geometry.
3. Do not italicize Greek letters (example, alpha, gamma, beta, etc..),
   and units like sec or rad, or  punctuation like ( ).
4. Styling the control labels is optional.
5. Do not use strings with <> for such formatting. Use Row[{ăăă }]
6. put () around units in plot labels. As (sec) or (hz)
7. do not italicize function names longer than one letter. So Style["exp",Italic]
   should just be "exp"
8. The t in delta(t) should be italic--but not the delta, Greek letter are not
   Italian letters is how I remember that.
9. Log should be log.
10. Is j^2= -1? Better say so in the caption for non EE.
```

see also `http://demonstrations.wolfram.com/guidelines.html`

I made small copy here so I do not have to keep looking for this all the time.



Demonstration Title

- Make the title as specific as you can. "Density Map for the 3*n* + 1 Problem" is a better title than "3*n* + 1 Problem."
- The Demonstration's title and file name are generated from what you enter in this section, so the file name of the notebook you upload does not matter.
- The permanent URL of a Demonstration is also generated from the title. We encourage authors to use standard English alpha numeric characters, because the URL syntax does not support many characters from outside this set.

Controls for Your Manipulate

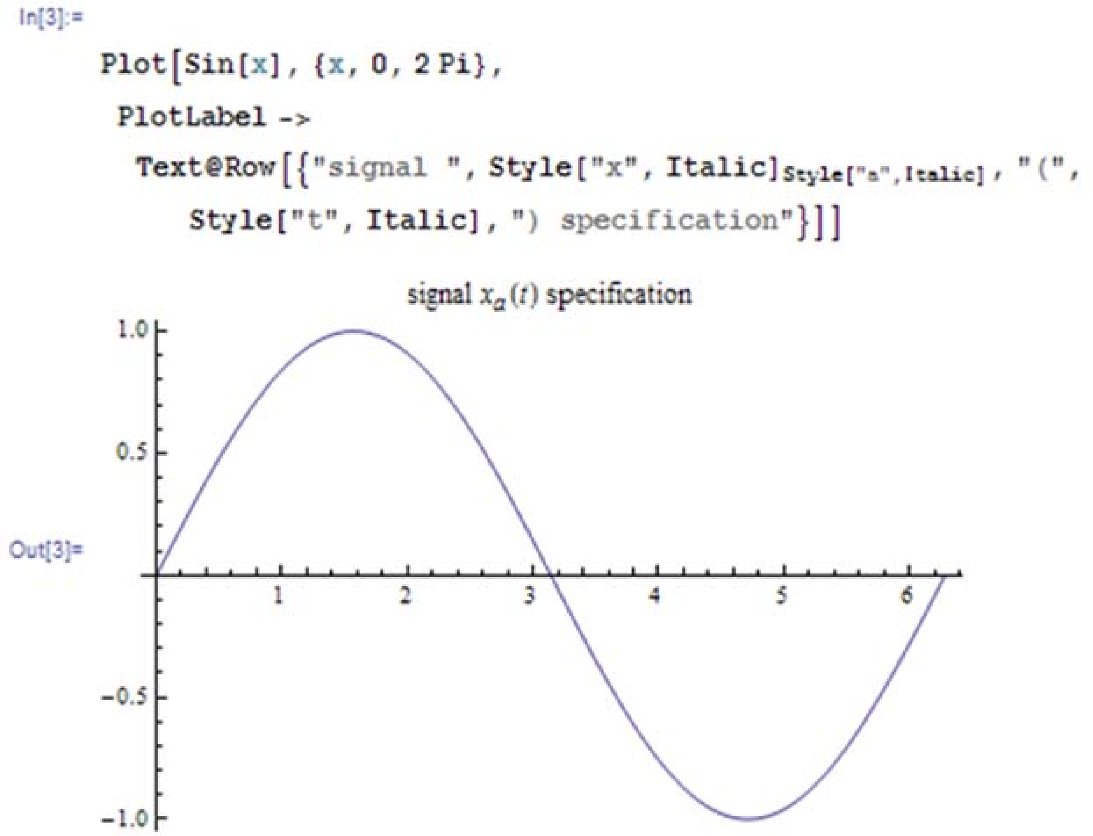
- Choose your controls judiciously. Unnecessary clutter only distracts from the idea of your Demonstration.
- Let the sliders do the animating. A slider that steps through an evolution is usually better than an animation the user cannot control.
- Label controls clearly with English words: "number of subdivisions" is more instructive than "*n*."
- `InputField` is not supported in Demonstrations.
- Set `Appearance->"Labeled"` for any slider whose values should be displayed to the right of the slider. Display other useful data in the content area of the `Manipulate`, for instance as a `PlotLabel`.
- Nested `Manipulate`s or functions that return `Manipulate` are not allowed. Keep things simple and use only one `Manipulate` per Demonstration.
- `Appearance->"Open"` is not allowed for `Manipulator` controls in Demonstrations, as it takes up precious screen space. Users can open the controls manually.

Testing Your Demonstration

Check your Demonstration before you upload. This will speed it through our review process.

- The Demonstration's screen size, including its controls, should not change as controls are manipulated. The outer border of each Demonstration should lie outside the white area of the Test Image Size palette and inside the translucent green area. The palette is located in the Tools menu. Also, consider setting an `ImageSize` option or using a `Pane` construct to control the size of your Demonstration.
- Move each control to its limits. Check for any error messages, unnecessary slowdowns, or other signs of improper function.
- As you move the controls, check for any jiggling of the `Manipulate` contents. You can eliminate jiggling by using the `PlotRange` or `ImagePadding` options, `Spacers`, or a `Pane` construct.
- Review and edit the Caption and Details sections to make them as clear as possible.
- Use the Check Spelling button in the Tools menu.

To put label on plot, example



```
In[3]:=
    Plot[Sin[x], {x, 0, 2 Pi},
     PlotLabel ->
      Text@Row[{"signal ", Style["x", Italic]Style["a",Italic], "(",
        Style["t", Italic], ") specification"}]]
```

Out[3]=

To typeset math for display on the demo use this type of coding

```
Text@Style[TraditionalForm[HoldForm[Sin[x] + Cos[y]], 12]]
```

To add invisible space use `ESC is ESC`

4.3 Notes on dynamics

Useful notes taken from different places from Mathematica documentation. And some added by me.

1. (added 7/5/14) There is a race condition between when updating a variable in the second argument of dynamics, which is also updated in the Manipulate expression. The problem is easy to explain

   ```
   Manipulate[.... n=n+1;...., Manipulator[Dynamic[f,{f=#;n=0}&....]
   ```

   Now, what happens, sometimes, is that when moving the slider, and setting $n = 0$, that this update to n can be lost, since it depends on the timing of when this happens. The Manipulate expression could be in the middle on updating n itself. This is classical lost update problem in multi-threading. The way to avoid this, is to follow this rule of thumb: When using second argument of dynamics in a Manipulate control, do not update a shared variable which can also be updated inside the Manipulate expression, as the update can be lost. The write to the shared variable/updating should only happen either inside the Manipulate expression or in the second argument of dynamics. But not in both places

2. Dynamic is wrapped around the whole expression, so evaluation of the Table command is delayed until the output is displayed in the notebook. Any time the value of x is changed, the Table command will be reevaluated.

3. Remember that Dynamic has the effect of delaying evaluation until the expression reaches the front end

4. Because it has the attribute HoldFirst, Dynamic does not evaluate its first argument. This is fundamental to the workings of Dynamic, but it can lead to a somewhat unexpected behavior

5. Ordinary variables in Mathematica are owned by the kernel. Their values reside in the kernel, and when you ask Mathematica to display the value in the front end, a transaction is initiated with the kernel to retrieve the value.

6. Variables declared with DynamicModule, on the other hand, are owned by the front end. Their values reside in the front end, and when the front end needs a value, it can be retrieved locally with very little overhead.

7. The most important is the fact that values of all `DynamicModule` variables are saved in the file when the notebook is saved.

8. By default, dynamic outputs triggered by changes in variable values are updated no faster than twenty times per second (this rate can be changed with the `SystemOption "DynamicUpda`

9. Dynamic outputs are only updated when they are visible on screen.

10. Remember to add `synchorization->False` to all dynamics, else will time out. When using `Refresh` also.

11. Never make a `refresh[]` tracks on 2 of my own symbols (not control variables). Use tick, only. Causes major synchronization problems with I update 2 variables inside a refresh, and have the refresh tracks both. Only make one track local variable, such as ticks

12. Ok, Found out that `finishDynamic[]` can causes annoying refresh on the UI whenever I move sliders. So removed it.

13. Remember to use `:>` and not `->` for `TrackedSymbols`

4.4 John Fultz collected on dynamics

```
Module variables should *never* appear inside Dynamics or
Manipulates internal to that Module.

To be clear with some examples (all using Dynamic, but they could equally well
use Manipulate, which is implemented using Dynamic)...

(* OK *) Dynamic[Module[{a}, a]]
(* OK *) Module[{a},
                (* expression involving a*);
                Dynamic[(* expression *not* involving a *)]
(* BAD *) Module[{a}, Dynamic[a]]
```

By John Fultz on math group, jan 24/2012

"Generally, you should construct controls so that they're not inside Dynamics that will trigger while you're interacting with those controls, since this can create instability"

By John Fultz on math group, feb 3/2012

"CDF files which you expect to deploy cannot rely on Shift+Enter evaluations to prime the pump. You need to make sure that all of the code dependencies are in the dynamic evaluations somewhere. Some possible ways of doing this, all of which have been discussed at various points on MathGroup, include:

* Using the Initialization option of Dynamic, DynamicModule, or Manipulate

* Using the SaveDefinitions option of Manipulate

* Adding code to the NotebookDynamicExpression option of the notebook (if it's initialization code, then wrapped in `Refresh[#,None]&` to prevent it from evaluating more than once per session).

* Not such a good idea for function definitions, but if you simply have code that needs to run before Dynamic code runs, nesting a `DynamicWrapper` around it might be appropriate, too."

4.5 WRI tech support notes on Dynamics/Manipulate

```
This is the support explanation of why this error came showed up:

The issue is specifically with the section:
Evaluate@env[{{age, 100, "age"}, 10, 200, 1}]

Manipulate doesn't really evaluate until it gets to the Initialization
option, but it will check its input for correct form. Mathematica
reads the main body of the Manipulate before running the
Initialization option. This is can be verified by using a Print statement:

Initialization -> (Print["Test"];
  makeCustomEnvironmentAlt =
   Function[Null, Function[code, With @@ Hold[{##}, code], HoldAll],
    HoldAll];
  env = makeCustomEnvironmentAlt[$age = 1, $salary = 2];
  Print["Test"])

Test does not print.

Getting around this will be probably not be clean.
....
Having  the code for the controller for age depend on evaluation of
some function which must be initialized does not appear to be possible
with simply Manipulate.
```

see `how-to-define-constants-for-use-with-with-in-one-place-and-then-apply-them-lat`

4.6 Some useful posts and links on dynamics

`why-wont-this-work-dynamic-in-a-select`

When `DynamicModule` is first evaluated, initial assignments for local variables are made during the evaluation. Any setting for the Initialization option is evaluated only when the output of `DynamicModule` is displayed.

see `how-to-make-dynamicmodule-work-without-an-extra-enter`

Here is a trick to allow one to control one slider based on another

```
Manipulate[{a, b},
 Grid[{

   {"a", Manipulator[Dynamic[a, {(a = #) &, (a = #; If[b > a, b = a]) &}],
          {1, 10, 1}], Dynamic[a]},

   {"b", Manipulator[Dynamic[b, {(b = #) &, (b = #; If[b > a, b = a]) &}],
          {1, 10, 1}], Dynamic[b]}}
  ],

 {{a, 5}, None},
 {{b, 3}, None}

 ]
```

4.7 Dynamically change the layout of Manipulate

```
Manipulate[x,

 {x, {True, False}},
 Grid[{
   {Dynamic@If[x,
      Control[{y, {True, False}}],
      Control[{z, 0, 1, .1}]
      ]
    }
   }
  ]
 ]
```

4.8 Compare Manipulate to DynamicModule

See this and the notebook is convert_manipulate_to_dynamicModule.nb

4.9 Usages of Manipulate

I wrote these for an answer here `http://community.wolfram.com/groups/-/m/t/153862?p_p_auth=GLyok3xN`

The manipulate expression is anything between the start of Manipulate and the first ","

```
Manipulate[Plot[Sin[x (1 + a x)], {x, 0, 6}], {a, 0, 2}]
```

It has the form

```
Manipulate[expression, controlVariables, Initialization :> ()]
```

4.9.1 case 1

```
foo[] := Plot[Sin[x (1 + a x)], {x, 0, 6}]
Manipulate[Evaluate@foo[], {a, 0, 2}]
```

4.9.2 case 2

move `foo[]` in the above example to inside Manipulate, in the initialization section, add literal symbol `a` so Manipulate will track it

```
Manipulate[a;
 foo[], {a, 0, 2},
Initialization :>
 (foo[] := Plot[Sin[x (1 + a x)], {x, 0, 6}])
]
```

Notice the trick above. `a` was added so that it appears in the Manipulate expression. Otherwise, it will not be tracked. You'll get an initial plot, but nothing will happen when moving the slider

4.9.3 case 3

move `foo[]` to global context, but have to tell Manipulate that `LocalizeVariable` is false

```
foo[] := Plot[Sin[x (1 + a x)], {x, 0, 6}];
Manipulate[a;
 foo[], {a, 0, 2}, LocalizeVariables -> False]
```

But notice, we still need to put `a` somewhere in the expression for it to tracked. Not enough just to say `LocalizeVariables -> False`

4.9.4 case 4

It is not enough to use `TrackedSymbols :>a`, if the symbol itself do not show up in the expression. Hence this does not work

```
foo[] := Plot[Sin[x (1 + a x)], {x, 0, 6}];
Manipulate[foo[], {a, 0, 2}, LocalizeVariables -> False, TrackedSymbols :> a]
```

Notice there is no `a` in the expression. Manipulate will not track `a` even though we told it to !

4.9.5 case 5

Same as case 4, even if we put the function inside the Initialization section, it will still not track `a` . One will get an initial plot, but that is all.

```
Manipulate[foo[], {a, 0, 2},
  TrackedSymbols :> a,
  Initialization :>
   (foo[] := Plot[Sin[x (1 + a x)], {x, 0, 6}];)]
```

4.9.6 case 6

Putting the function definition of `foo[]` itself inside Manipulate expression, now Manipulate sees `a` there and will automatically track it. No need to do anything more:

```
Manipulate[Module[{}, foo[] := Plot[Sin[x (1 + a x)], {x, 0, 6}];
  foo[]], {a, 0, 2}]
```

Or simply

```
Manipulate[Plot[Sin[x (1 + a x)], {x, 0, 6}], {a, 0, 2}]
```

4.9.7 case 7

This is the method I use myself. Put all the functions inside the initialization section, but pass the dependent variables by argument call.

```
Manipulate[foo[a], {a, 0, 2},
  Initialization :> (
    foo[a_] := Module[{x}, Plot[Sin[x (1 + a x)], {x, 0, 6}]]
)]
```

I like this, becuase it achieves both the goal of having all the slider symbols inside the Manipulate expression, hence Manipulate will track them, and at the same time, it avoid the function be in global context, and it is the modular way, since one can see which parameter each function depends on by looking at the signature of the function.

4.9.8 case 8

Similar to case 7, but the function itself is now in the global context. This is a fine solution as well, if this function needs to be called from somewhere else as well other than from the Manipulate. Otherwise, it is best not to have in the global context and use case 7.

```
foo[a_] := Module[{x}, Plot[Sin[x (1 + a x)], {x, 0, 6}]];
Manipulate[foo[a], {a, 0, 2}]
```

4.10 How to run CDF in Chrome browser?

Note: This is now (2024) no longer works. Can not run CDF any more in browsers.

Type this in the chrome window

`chrome://flags/#enable-npapi`

and click enable to enable NPAPI. Chrome now disables NPAPI and so CDF no longer runs inside browser. After doing the above, restart the browser again. Now it should run

Chapter 5

package related, reading/writing Mathematica source code to/from files, contexts and related

5.1 How to write a package?

Small note here

5.2 How to use context and packages?

```
?$Packages
```

"gives a list of the contexts corresponding to all packages which have been loaded in your current Mathematica session."

```
{"JLink`", "GetFEKernelInit`", "ResourceLocator`",
"PacletManager`", "QuantityUnits`", "WebServices`",
 "System`", "Global`"}
```

5.3 How to find what packages are loaded?

```
$ContextPath
```

```
{"PacletManager`", "QuantityUnits`", "WebServices`",
"System`", "Global`"}
```

5.4 Finding packages in specific context

```
Names["System‘ComplexExpand‘*"]
```

```
{"System`ComplexExpand`AbsExpr",
"System`ComplexExpand`ArgExpr",
"System`ComplexExpand`ConjugateExpr",
"System`ComplexExpand`ReImExpr",
"System`ComplexExpand`ReImFail",
"System`ComplexExpand`SignExpr"}
```

5.5 How to find what contexts are loaded?

```
?Contexts
```

```
Contexts[] gives a list of all contexts.
Contexts["string"] gives a list of the contexts
                   which match the string.
```

5.6 How to load an m file or a package?

Append to the Path the folder name where the package is located in. In this example, assuming there is a package `control.m` located in folder `C:\data` then type the following to load the package

```
AppendTo[$Path, "C:\\data"]
<< control.m
```

5.7 Finding names of functions in a package

$Packages	{"JLink`", "GetFEKernelInit`", "ResourceLocator`", "PacletManager`", "QuantityUnits`", "WebServices`", "System`", "Global`"}
Names["JLink‘*"]	{"JLink`AddPeriodical", "JLink`AddToClassPath",...

5.8 Finding a package that has specific function

use Context.

```
Context[Integrate]

    "System`"
```

5.9 Finding all contexts that belong to a package?

```
Contexts["packageName*"]
```

5.10 How to remove packages?

to do

5.11 How to list files in a directory?

```
SetDirectory[$BaseDirectory];
FileNames["*"]
```

5.12 Where is init.m and how to use it?

Possible locations for init.m files include the following:

1. `$BaseDirectory/Kernel` kernel initialization code for all users
2. `$UserBaseDirectory/Kernel` kernel initialization code for the currently logged-in user
3. `$BaseDirectory/FrontEnd` front end initialization code for all users
4. `$UserBaseDirectory/FrontEnd` front end initialization code for the currently logged-in user
5. I have my `init.m` in the following folder

   `C:\Documents and Settings\All Users\Application Data\Mathematica\Kernel\init.m`

5.13 Reading Mathematica example data and location

on windows, V 8, example data is located in

```
C:\Program Files\Wolfram Research\Mathematica\8.0\Documentation\English\System\ExampleData
```

and it can be read like this

```
str = OpenRead["ExampleData/USConstitution.txt"]

Out[147]= InputStream[ExampleData/USConstitution.txt, 127]
```

5.14 Finding which folders are on trusted path

Thanks to Mike for these commands, see `http://stackoverflow.com/questions/8583521/why-do-i-get-security-warning-message-this-file-contains-potentially-unsafe-dyn`

```
CurrentValue[$FrontEnd, {"NotebookSecurityOptions", "TrustedPath"}]

Out[212]= {FrontEnd`FileName[{$InstallationDirectory}],
          FrontEnd`FileName[{$BaseDirectory}],
          FrontEnd`FileName[{$UserBaseDirectory}]}

CurrentValue[$FrontEnd, {"NotebookSecurityOptions", "UntrustedPath"}]

Out[213]= {FrontEnd`FileName[{FrontEnd`$DesktopDirectory}],
           FrontEnd`FileName[{FrontEnd`$DownloadsDirectory}],
  FrontEnd`FileName[{FrontEnd`$LocalApplicationDataDirectory}],
  FrontEnd`FileName[{FrontEnd`$RemoteApplicationDataDirectory}],
  FrontEnd`FileName[{FrontEnd`$ProgramFilesDirectory}],
  FrontEnd`FileName[{FrontEnd`$ProgramFilesX86Directory}],
  FrontEnd`FileName[{$TemporaryPrefix}]}
```

Now to find if your current notebook is on the trusted path type `NotebookDirectory[]` and see if the output shows up in the trusted path of not. To add a folder to trusted path go to "Preferences > Advanced > Open Options Inspector". Then under Global Preferences search for trusted

5.15 save matrix to file, and read it again

```
SetDirectory[NotebookDirectory[]];
list = {{3, 4, 5}, {4, 5, 6}};
Export["data.txt", list]
```

To read it later, say after closing and restarting Mathematica again to continue working on the data

```
SetDirectory[NotebookDirectory[]];
list = ToExpression@Import["data.txt", "List"]

    {{3, 4, 5}, {4, 5, 6}}
```

5.16 run Mathematica m file as script

Thanks to Rolf Mertig reference for help on this.

Make a file foo.m such as

```
AppendTo[$Echo, "stdout"]
SetOptions[ $Output, FormatType -> OutputForm ];
Integrate[Sin[x],x]
```

Now type, from DOS window

```
"C:\Program Files\Wolfram Research\Mathematica\10.1\math.exe" < foo.m
```

This will send input and output to screen. To send output to file do

```
"C:\Program Files\Wolfram Research\Mathematica\10.1\math.exe" < foo.m > log.txt
```

You can modify the PATH on windows to add the above to environment variable so not to have to type the long command each time

Chapter 6

General items

6.1 How to do convolution?

see `?ListConvolve`

6.2 printing definitions of internal functions

```
LaplaceTransform[x,x,t];
ClearAttributes[LaplaceTransform,ReadProtected]
??LaplaceTransform
```

But better to use this:

```
Needs["GeneralUtilities`"];
PrintDefinitions@Charting`FindTicks
```

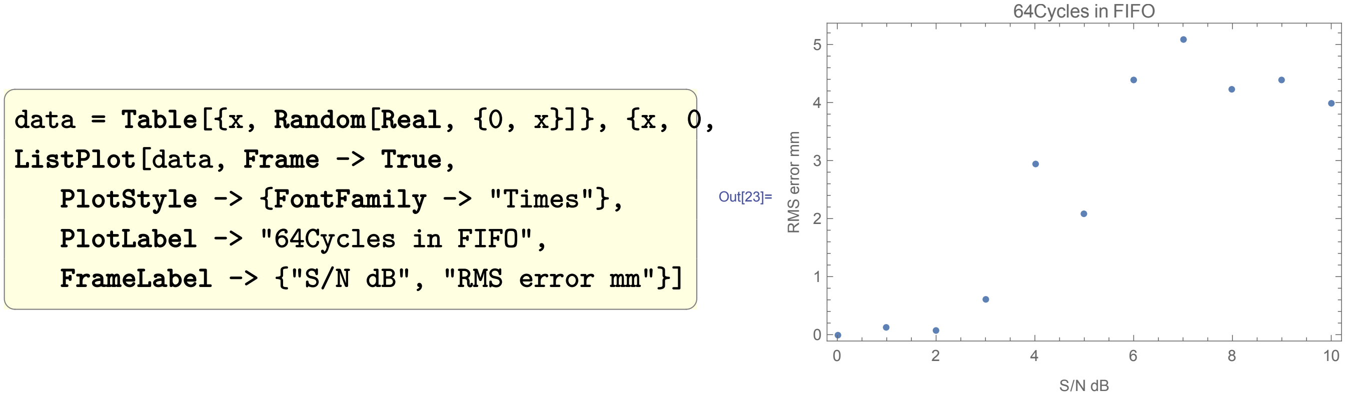
Thanks to `https://mathematica.stackexchange.com/questions/132568/extract-ticks-from-plot`

The above works if the function is not read protected.

See also this link

6.3 Making labeles for frame plot

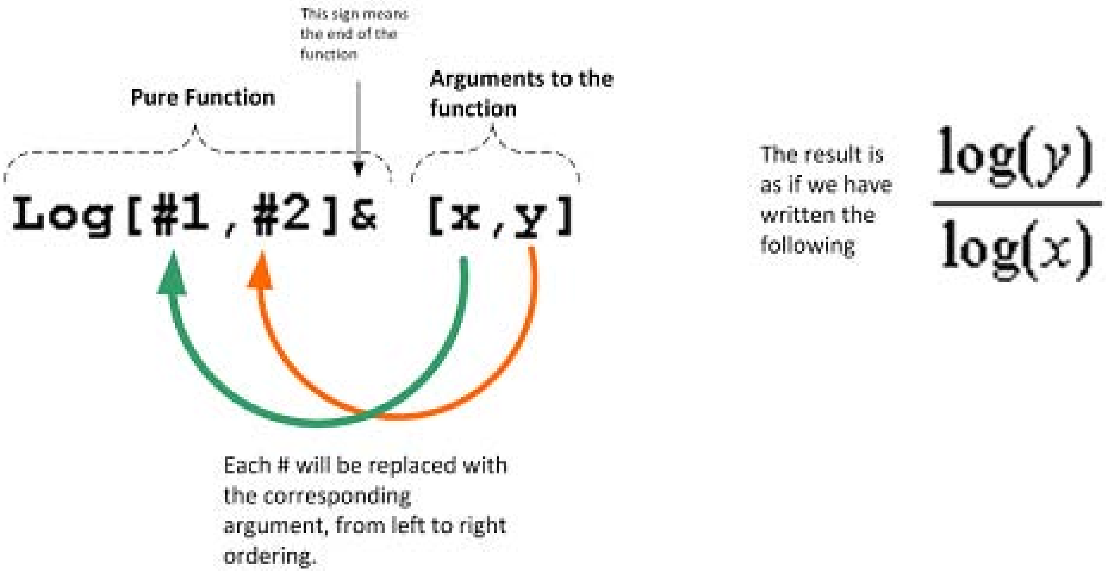
from the net. Using Times font family is the idea.



```
data = Table[{x, Random[Real, {0, x}]}, {x, 0,
ListPlot[data, Frame -> True,
   PlotStyle -> {FontFamily -> "Times"},
   PlotLabel -> "64Cycles in FIFO",
   FrameLabel -> {"S/N dB", "RMS error mm"}]
```

6.4 How do pure functions work?

I made this simple diagram to help me understand pure functions.



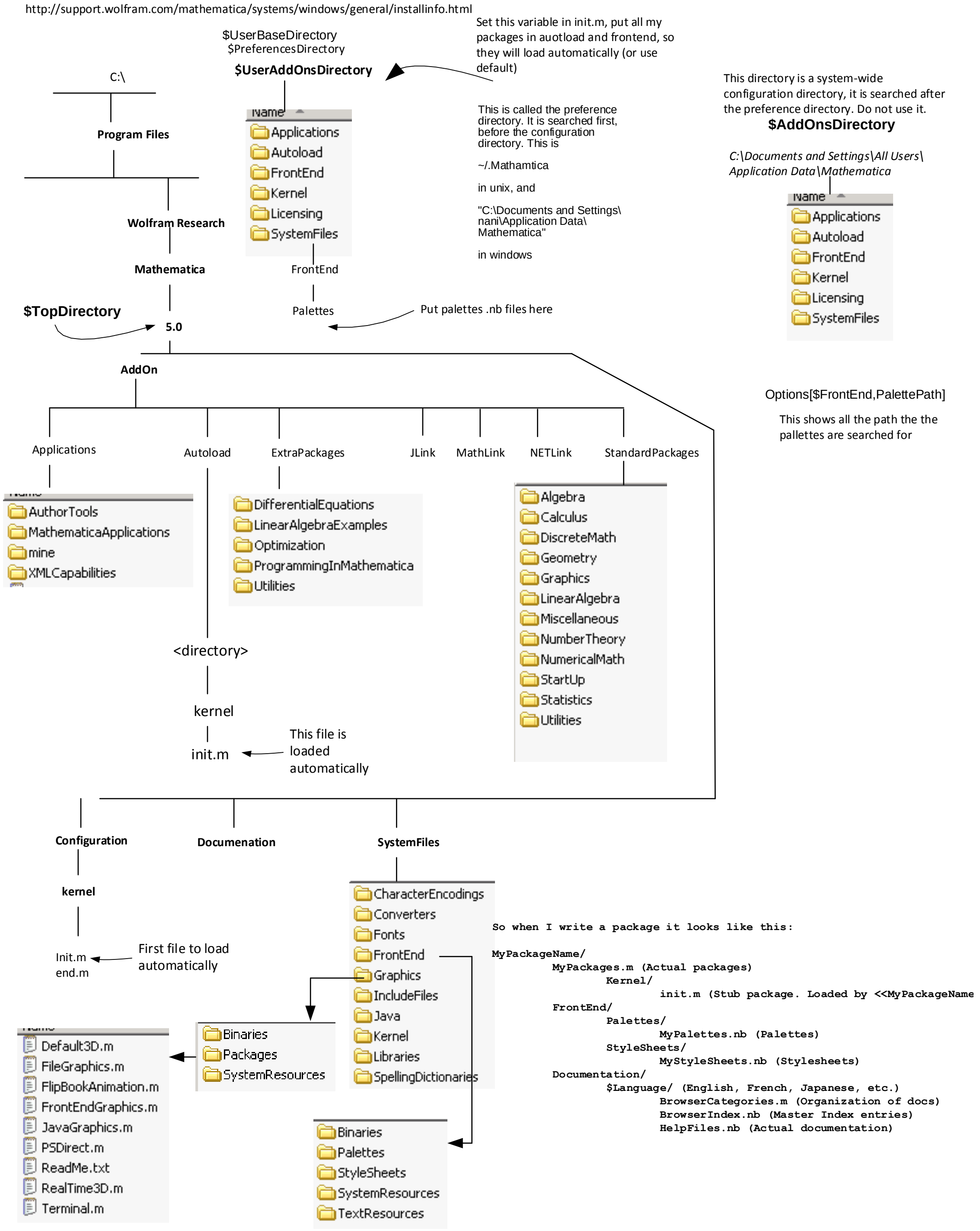
This sign means the end of the function
Pure Function
Arguments to the function
Log[#1,#2]& [x,y]
The result is as if we have written the following

$$\frac{\log(y)}{\log(x)}$$

Each # will be replaced with the corresponding argument, from left to right ordering.

6.5 Mathematica directories after installation

I made this diagram to show the installation tree structure.



Set this variable in init.m, put all my packages in auotload and frontend, so they will load automatically (or use default)

This is called the preference directory. It is searched first, before the configuration directory. This is

~/.Mathamtica

in unix, and

"C:\Documents and Settings\nani\Application Data\Mathematica"

in windows

This directory is a system-wide configuration directory, it is searched after the preference directory. Do not use it.

Put palettes .nb files here

Options[$FrontEnd,PalettePath]

This shows all the path the the pallettes are searched for

This file is loaded automatically

First file to load automatically

```
So when I write a package it looks like this:

MyPackageName/
        MyPackages.m (Actual packages)
                Kernel/
                        init.m (Stub package. Loaded by <<MyPackageName
        FrontEnd/
                Palettes/
                        MyPalettes.nb (Palettes)
                StyleSheets/
                        MyStyleSheets.nb (Stylesheets)
        Documentation/
                $Language/ (English, French, Japanese, etc.)
                        BrowserCategories.m (Organization of docs)
                        BrowserIndex.nb (Master Index entries)
                        HelpFiles.nb (Actual documentation)
```

6.6 Finding partial fraction expansion

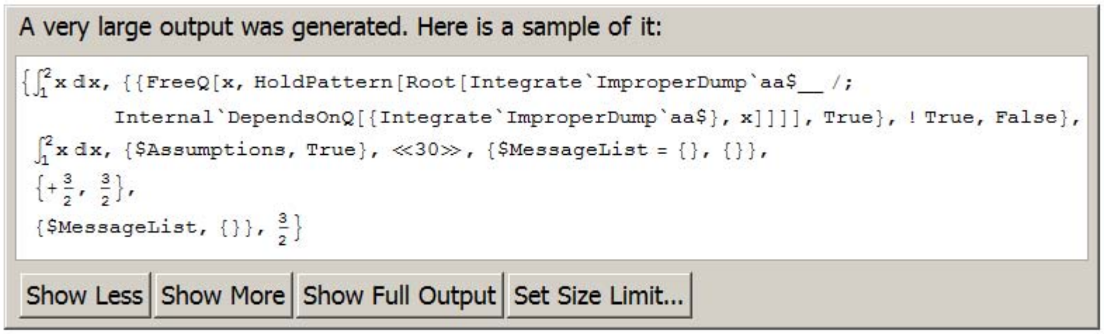
```
?Apart
```

6.7 Some keyboard shortcuts

To insert I	`esc ii esc`
To enter π	`esc p esc`
To enter E	`esc ee esc`
`ctrl-6` wil make exponent	
`ctrl^` will make superscript	

6.8 Tracing a function

```
Trace[Integrate[x, {x, 1, 2}], TraceInternal -> True]
```



A very large output was generated. Here is a sample of it:

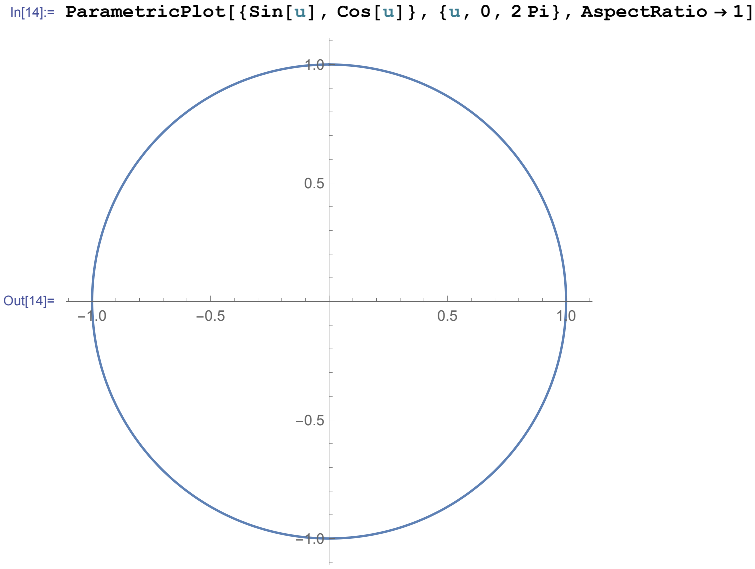
```
{∫_1^2 x dx, {{FreeQ[x, HoldPattern[Root[Integrate`ImproperDump`aa$__ /;
        Internal`DependsOnQ[{Integrate`ImproperDump`aa$}, x]]]], True}, ! True, False},
 ∫_1^2 x dx, {$Assumptions, True}, <<30>>, {$MessageList = {}, {}},
 {+3/2, 3/2},
 {$MessageList, {}}, 3/2}
```

6.9 Removing Big O notation from Series

`Normal[Series[f[x], {x, 0, 3}]]`	`f[0]+x (f^')[0]+1/2 x^2 (f^'')[0]+1/6 x^3 (f^(3))[0]`

6.10 How to plot circle?

```
ParametricPlot[{Sin[u], Cos[u]}, {u, 0, 2 Pi}, AspectRatio -> 1]
```



```
In[14]:= ParametricPlot[{Sin[u], Cos[u]}, {u, 0, 2 Pi}, AspectRatio → 1]
Out[14]=
```

6.11 Solve implicit differentiation of equations

Suppose we are given $z = xe^{-y}, x = \cosh(t), y = \cos(s)$ and need to find $\frac{dz}{ds}$

```
x[t_] := Cosh[t]
y[s_] := Cos[s]
z[x_, y_] := x[t] Exp[-y[s]]
D[z[x, y], s]
```

```
(Cosh[t]*Sin[s])/E^Cos[s]
```

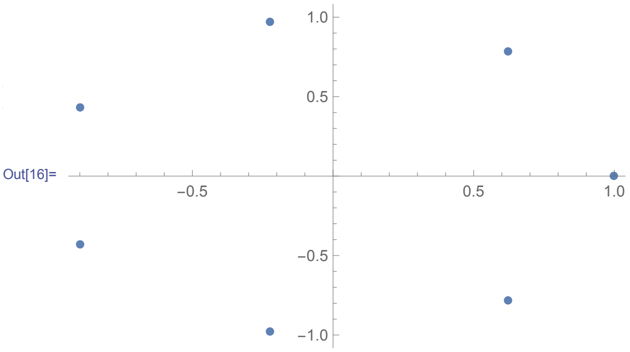
Another example: $u = x^2y^3z, x = \sin(s+t), y = \cos(s+t), z = e^{st}$ and we need to find $\frac{du}{ds}$ and $\frac{du}{dt}$

```
x[s_, t_] := Sin[s + t];
y[s_, t_] := Cos[s + t];
z[s_, t_] := Exp[s t];
u[s_, t_] := x[s, t]^2 y[s, t]^3 z[s, t];
Clear[s, t];
D[u[s, t], s]
```

```
2*E^(s*t)*Cos[s + t]^4*Sin[s + t] +
  E^(s*t)*t*Cos[s + t]^3*Sin[s + t]^2 +
  3*E^(s*t)*Cos[s + t]^2*Sin[s + t]^3
```

6.12 Drawing roots of complex equation?

```
rootsPlot[poly_, z_] := ListPlot[{Re[z], Im[z]}
  /. NSolve[poly == 0, z],
  PlotStyle -> Directive[PointSize[0.015]]];

rootsPlot[z^7 - 1, z]
```



6.13 How to simplify with conditions?

For example, to integrate this below, for n positive integer we do

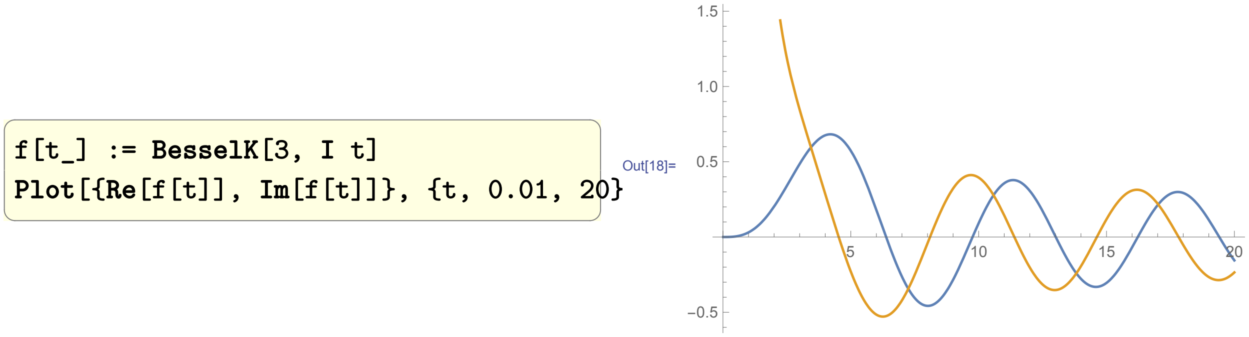
`Integrate[Sin[n x]^2, {x, 0, Pi}]`	`Pi/2 - Sin[2*n*Pi]/(4*n)`

`Assuming[Element[n, Integers] && n > 0,`
`   Integrate[Sin[n x]^2, {x, 0, Pi}]]`
	`Pi/2`

6.14 Smart replacement everywhere?

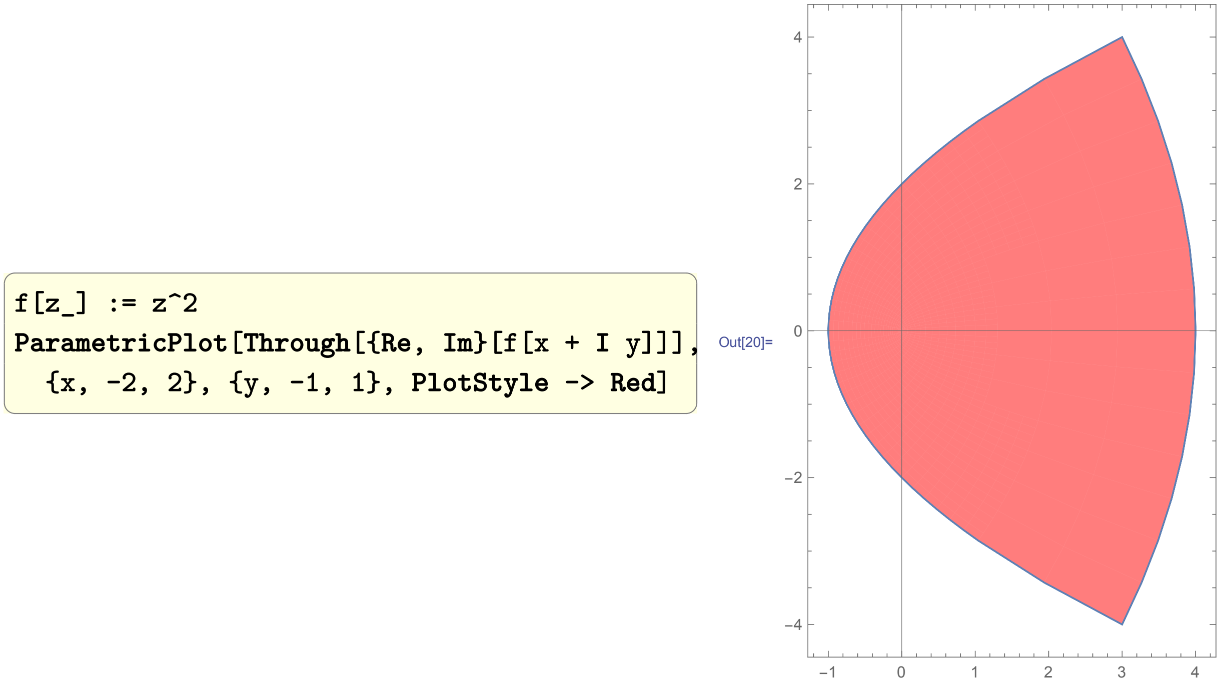
`eq = f == x^3 + 6/x^3;`
`Reduce[{eq, x^3 == z}]`
	`(x == z^(1/3) || x == (-(-1)^(1/3))*z^(1/3) ||`
	`  x == (-1)^(2/3)*z^(1/3)) &&`
	`   z != 0 && f == (6 + z^2)/z`

6.15 Plotting real and imaginary parts



```
f[t_] := BesselK[3, I t]
Plot[{Re[f[t]], Im[f[t]]}, {t, 0.01, 20}]
```

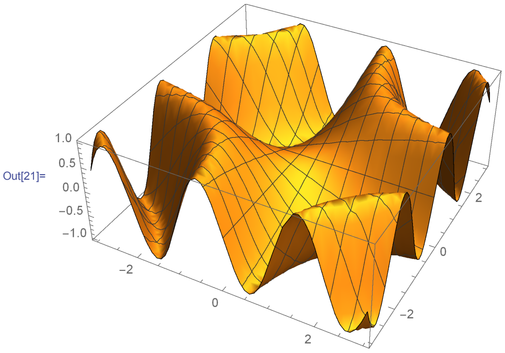
6.16 Plotting mapping of complex numbers?



```
f[z_] := z^2
ParametricPlot[Through[{Re, Im}[f[x + I y]]],
  {x, -2, 2}, {y, -1, 1}, PlotStyle -> Red]
```

6.17 Saving a plot as eps

```
Plot3D[Sin[x y], {x, -3, 3}, {y, -3, 3}]
Export["plot.eps", %]
```



6.18 Mathematica not evaluate its arguments?

```
In[60]:= s = HoldForm[1 + 2]
Out[60]= HoldForm[1 + 2]
In[61]:= ReleaseHold[s]
Out[61]= 3
```

6.19 Combining more than plot

One way is to use `Show`

6.20 Speed of functional and procedural

By Bill Rowe `http://forums.wolfram.com/mathgroup/archive/2004/Apr/msg00357.html`

```
In[1]:=
Timing[For[sum = 0; n = 1,  n < 100001, n++, sum += n]]

Out[1]=
{1.2999999999999998*Second,  Null}

In[2]:=
Timing[Plus @@ Range[10000]]

Out[2]=
{0.009999999999999787*Second, 50005000}
```

6.21 Using subscript variables in function def

Use notation package

6.22 Using zero as index

See `using_zero_index_in_Mathematica`

6.23 Extracting the LHS and RHS of equation

```
In[62]:= eq = x^2 + Sin[4*a] == 3 - Derivative[1][y][t]
Out[62]= x^2 + Sin[4*a] == 3 - Derivative[1][y][t]

In[63]:= lhs = eq /. (lhs_) == (rhs_) -> lhs
Out[63]= x^2 + Sin[4*a]

In[64]:= rhs = eq /. (lhs_) == (rhs_) -> rhs
Out[64]= 3 - Derivative[1][y][t]
```

6.24 Doing some matrix operations

Rememebr: `Position` and `Cases` return result that can be used by `Extract` directly. But can't be used by `Part` directly.

6.24.1 How to extract first column in matrix

`a = Table[RandomInteger[100], {4}, {4}]`	`Out[69]= {{55, 63, 78, 45},`
	`{13, 45, 67, 1},`
	`{94, 32, 48, 90},`
	`{31, 75, 43, 60}}`

`a[[All,1]]`	`Out[70]= {55,`
	`13,`
	`94,`
	`31}`

6.24.2 How to extract first 3 rows in the first column?

`a[[1 ;; 3,1]]`	`Out[71]= {55, 13, 94}`

6.24.3 How to find some matrix rows based on some condition on value in say the first column??

Find rows which has elements in first column less than 3 in the following

```
a = {{1, 2, 3}, {4, 5, 8}, {7, 8, 9}}
```

Reference: `how-to-extract-rows-from-matrix-based-on-value-in-first-entry`

The solution using pattern below (by WRech) is interesting since the same pattern can be used by `Cases` and `Position`.

solution by me

```
pos = Position[a[[All,1]], _?(#1 <= 4 & )]
Out[73]= {{1}, {2}}

Extract[a, pos]
Out[74]= {{1, 2, 3}, {4, 5, 8}}
```

by Simon

```
pos = Position[a, _List?(First[#1] <= 4 & ), {1}]
Out[75]= {{1}, {2}}

Extract[a, pos]
Out[76]= {{1, 2, 3}, {4, 5, 8}}
```

By Asim

```
Pick[a, a[[All,1]], _?(#1 <= 4 & )]
Out[77]= {{1, 2, 3}, {4, 5, 8}}
```

By WReach

```
Cases[a,{n_,__}/;n<=4,{}]
```

```
Out[78]= {{1,2,3},{4,5,8}}
```

By WReach

```
pos=Position[a,{n_,__}/;n<=4,{}]
Extract[a,pos]

Out[79]= {{1},{2}}
Out[80]= {{1,2,3},{4,5,8}}
```

6.24.4 How to generate a diagonal matrix?

Random values on the diagonal

```
DiagonalMatrix[Table[Random[], {3}]]
```

Ones on the diagonal

```
DiagonalMatrix[Table[1, {3}]]
```

6.24.5 How to generate upper diagonal matrix?

one way, using SparseArray

```
Normal[SparseArray[{{i_, i_} :> 2*i, {i_, j_} :> i + j /; i < j}, {5, 5}]]
Out[81]= {{2, 3, 4, 5, 6},
          {0, 4, 5, 6, 7},
          {0, 0, 6, 7, 8},
          {0, 0, 0, 8, 9},
          {0, 0, 0, 0, 10}}
```

Or using Table. But notice that in SparseArray, the 'zeros' are already the default case, so using SparseArray is simpler.

```
Table[If[i == j, 2*i, If[i < j, i + j, 0]], {i, 5}, {j, 5}]
Out[82]= ... same as above
```

6.24.6 How to find the trace of a matrix?

see ?Tr[a]

6.24.7 How to find product of elements on the Trace?

```
a = {{1, 2, 3},
     {4, 5, 8},
     {7, 8, 9}}
Tr[a, Times]

Out[84]= 45
```

6.24.8 How to check if a Matrix is diagonal matrix?

by Jon MacLoone

```
DiagonalQ[m_List] /; ArrayDepth[m] === 2 && Equal @@ Dimensions[m] :=
   And @@ Flatten[MapIndexed[#1 === 0 || Equal @@ #2 & , m, {2}]];

DiagonalQ[m_] := Return[False];
a = {{1, 2}, {2, 4}}
b = {{1, 0}, {0, 2}}

DiagonalQ[a]
Out[89]= False

DiagonalQ[b]
Out[90]= True
```

By Paul Abbott

```
Clear[DiagonalQ];
DiagonalQ[(m_)?MatrixQ] /; SameQ @@ Dimensions[m] := m === DiagonalMatrix[Tr[m, List]]
DiagonalQ[a]

Out[93]= False

DiagonalQ[b]
Out[94]= True
```

6.24.9 How to find locations of all zeros (or any other value) in a matrix?

Find location of zeros in this matrix

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}
```

one way

```
Position[a, 0]
Out[96]= {{2, 2}, {3, 3}}
```

Another way

```
Position[a, _?(#1 == 0 & )]
Out[97]= {{2, 2}, {3, 3}}
```

6.24.10 How to find locations of elements subject to some test?

find all elements between 4 and 8

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}

Position[a, _?(#1 >= 4 && #1 <= 8 & )]
Out[99]= {{2, 1}, {2, 3}, {3, 1}, {3, 2}}

Extract[a, %]
Out[100]= {4, 8, 7, 8}
```

6.24.11 How to insert an element in specific position?

Using `Part` to inser 99 in position (1,1)

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}
a[[1,1]] = 99;
a

Out[103]= {{99, 2, 3},
           {4, 0, 8},
           {7, 8, 0}}
```

6.24.12 How to insert a row into a matrix?

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}
```

To insert this row in the second row in matrix above

```
row = {97, 98, 99};
newa = Insert[a, row, {2}]

Out[106]= {{1, 2, 3},
           {97, 98, 99},
           {4, 0, 8},
           {7, 8, 0}}
```

or just use '2', it will also work

```
newa = Insert[a, row, 2]

Out[107]= {{1, 2, 3},
           {97, 98, 99},
           {4, 0, 8},
           {7, 8, 0}}
```

6.24.13 How to insert a column into a matrix?

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}
```

To insert this column in the second column position in above matrix

```
column = {97, 98, 99};
```

one way

```
newa = Transpose[Insert[Transpose[a], column, 2]]
Out[110]= {{1, 97, 2, 3},
           {4, 98, 0, 8},
           {7, 99, 8, 0}}
```

another way

```
Normal[SparseArray[{{i_, j_} :> column[[i]] /; j == 2,
       {i_, j_} :> a[[i,j]] /; j == 1, {i_, j_} :> a[[i,j - 1]] /; j > 1},
   {3, 4}]]

Out[111]= {{1, 97, 2, 3},
           {4, 98, 0, 8},
           {7, 99, 8, 0}}
```

Another way by Leonid Shifrin `how-to-insert-a-column-into-a-matrix-the-correct-mathematica-way`

```
MapThread[Insert, {a, column, Table[2, {Length[column]}]}]

Out[112]= {{1, 97, 2, 3},
           {4, 98, 0, 8},
           {7, 99, 8, 0}}
```

Another by Leonid Shifrin

```
ArrayFlatten[{{a[[All,1 ;; 1]], Transpose[{column}], a[[All,2 ;; All]]}}]

Out[113]= {{1, 97, 2, 3},
           {4, 98, 0, 8},
           {7, 99, 8, 0}}
```

6.24.14 How to build a large matrix from blocks of smaller matrices?

Given

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}
```

and we want to make matrix `{ {a,a},{a,a} }`

```
b = ArrayFlatten[ {{a, a}, {a, a}}]

Out[118] {{1, 2, 3, 1, 2, 3},
          {4, 0, 8, 4, 0, 8},
          {7, 8, 0, 7, 8, 0},
          {1, 2, 3, 1, 2, 3},
          {4, 0, 8, 4, 0, 8},
          {7, 8, 0, 7, 8, 0}}]
```

6.24.15 How to apply a function to each element in a 2D matrix?

Given

```
a = {{1, 2, 3},
     {4, 0, 8},
     {7, 8, 0}}
```

and we want to apply the this function to it

```
f[x_] := x + 2*Sin[x]
```

Then using Map

```
r = Map[f[#1] & , a, {2}]

Out[123]= {{1 + 2*Sin[1], 2 + 2*Sin[2], 3 + 2*Sin[3]},
           {4 + 2*Sin[4], 0, 8 + 2*Sin[8]},
           {7 + 2*Sin[7], 8 + 2*Sin[8], 0}}
```

6.25 Find if an expression implies another

```
Remove["Global`*"]
Refine[Sin[x]^2 + Cos[x]^2 == q, q == 1]

Out[125]= True
```

6.26 Displaying matrices in MatrixForm

One way

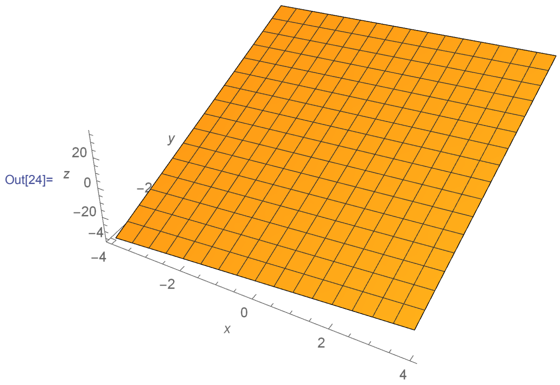
```
$PrePrint = If[MatrixQ[#], MatrixForm[#], #] &;
m = {{1, 2}, {3, 4}}
```

Another otpion is to use `TraditionalForm`. Can change default form from the menu, so this way no need to change `$PrePrint`

6.27 Making 3D axes normal instead of boxed

Use `Boxed -> False`

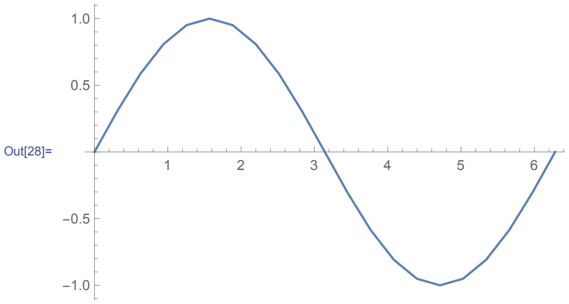
```
Plot3D[2 x + 7 y, {x, -4, 4}, {y, -4, 4},
 Boxed -> False,
 AxesEdge -> {{-1, -1}, {-1, -1}, {-1, -1}},
 AxesLabel -> {x, y, z}]
```



6.28 How to use ListPlot on set of x, y data?

One way to use `Transpose`

```
x = Table[i, {i, 0, 2 Pi, Pi/10}];
y = Sin[x];
data = Transpose[{x, y}];
ListPlot[data, Joined -> True]
```



6.29 How to do autocorrelation

```
v = {1, 2, 3};
ListCorrelate[v, v, {-1, 1}, 0]

Out[139]= {3, 8, 14, 8, 3}
```

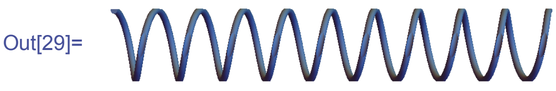
In Matlab it is

```
v=[1,2,3]
xcorr(v,v)

ans =
     3     8    14     8     3
```

6.30 How to make a spring?

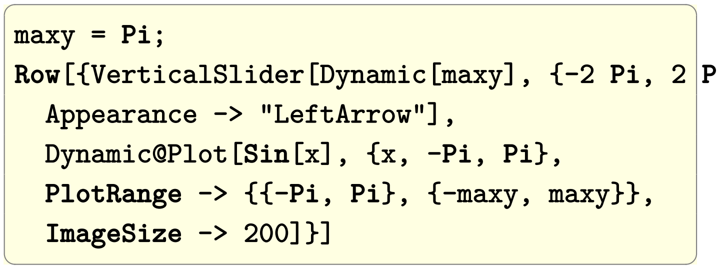
From the net, lost reference

```
ParametricPlot3D[{Sin[u], Cos[u], 0.2 u}, {u, 0, 10*2 Pi},
  PlotStyle -> {Tube[0.1]},
  ViewPoint -> {3.38378, 0, 0},
 ViewVertical -> {0, 1, 0},
 ViewAngle -> Automatic,
 Axes -> False, Boxed -> False,
 ImageSize -> 200]
```

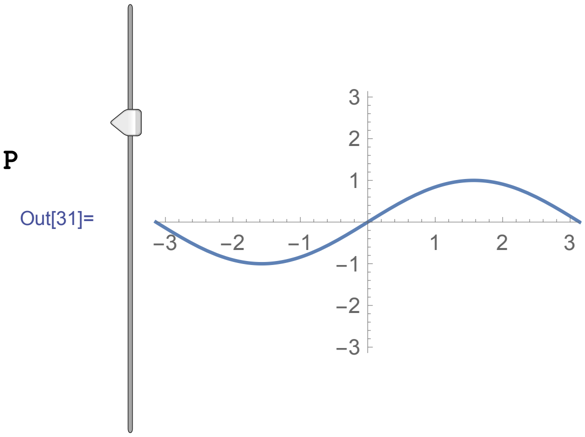


See also `how-to-draw-a-spring`

6.31 Making plot range a slider



```
maxy = Pi;
Row[{VerticalSlider[Dynamic[maxy], {-2 Pi, 2 P
  Appearance -> "LeftArrow"],
  Dynamic@Plot[Sin[x], {x, -Pi, Pi},
  PlotRange -> {{-Pi, Pi}, {-maxy, maxy}},
  ImageSize -> 200]}]
```



6.32 On Mathematica accuracy and precision

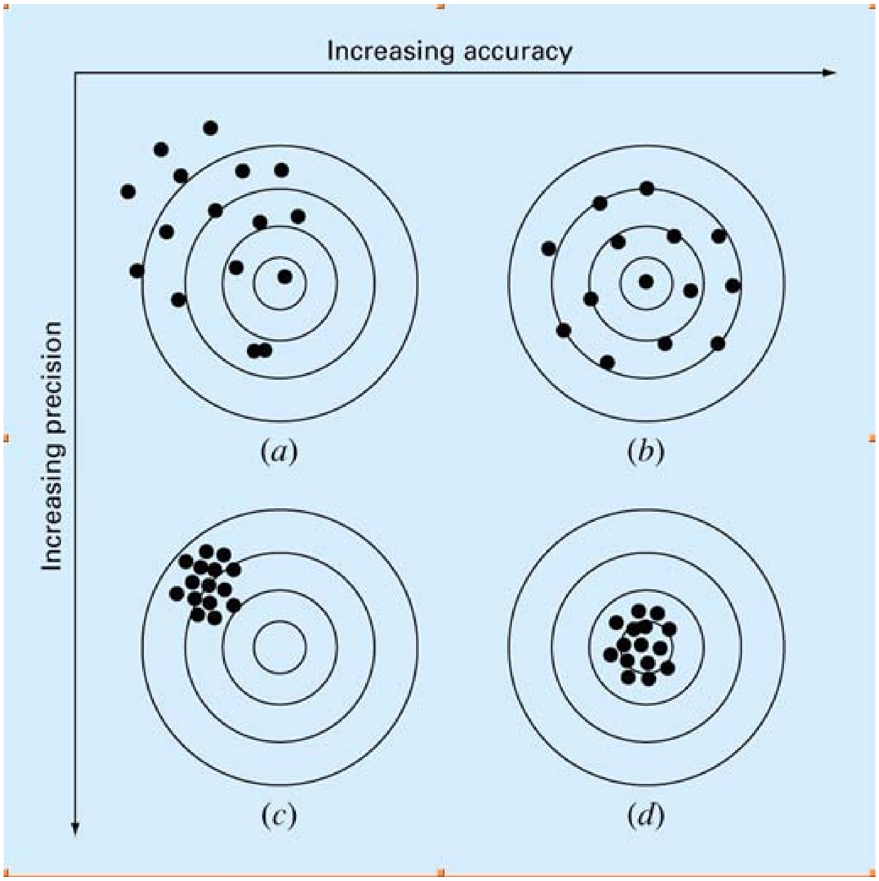
6.32.1 From class notes: UC Davis, Dr Rocke

some notes below

Precision means the variability between estimates Accuracy means the amount of deviation between the estimate and the "true value"

The condition number is the ratio of the output error to the input error. if the condition number is about 10k, then one loses about k digits of accuracy.

The main sources of inaccuracy (= error) is truncation error and round-off error.



From the above dart diagram, then we can say this: a value is accurate if it is near the bull-eye. But if is away from the bull-eye, but it is always away from the bull-eye and in the same place, then it is precise. So something can be precise but not accurate. So precise has to do with repeated values. i.e. one can't say a value is precise, but must talk about an experiment being precise, it is produced same result each time (or very close results each time).

So, it is best of course to be both accurate and precise. So what does it mean to be accurate but not precise? using the above dart diagram, it means values generated from the experiment are always close to the pull eye, but not in the same locations.

6.32.2 by Andrzej Kozlowski

From `http://forums.wolfram.com/mathgroup/archive/2010/Jan/msg00917.html`

```
The definition of precision in Mathematica is this. Suppose x is a
number known up to an error of epsilon, that is it can be viewed as
lying in the interval (x-epsilon/2,x+epsilon/2). Then its precision is

-Log[10,epsilon/x]. Its accuracy is -Log[10,epsilon]. The two are
related by the equation:
```

```
Precision[x] - Accuracy[x] == RealExponent[x]

The interpretation in terms of digits is only approximate. Both accuracy
and precision can be negative - this depends on the scale of the number
i.e. RealExponent. A number will have negative accuracy if its absolute
error is large. It is easy to produce such numbers by cancellation

With[{x = N[10^100, 50] - N[10^100, 50]},
     Accuracy[x]]

-50.301

On the other hand, since

$MinPrecision

0

You won't normally in Mathematica see numbers with negative Precision.
Precision is the main concept, Accuracy is only used because Precision
is singular at 0 (remember - its relative error).

It's all perfectly documented so this tired scape goat is not available
this time.
```

6.33 Making condition as pattern

By Bob Hanlon from math group:

```
Clear[x, $PrePrint]
expr = {E^x, x, x^2, Log[x]};
Position[expr, _?( !PolynomialQ[#1, x] & ), 1]

Out[146]= {{1}, {4}}
```

6.34 Stopping 3D plot from changing size

Use `SphericalRegion->True`

```
ListPlot3D[Table[RandomReal[], {5}, {5}], AxesLabel -> {"x", "y", "z"},
  ImageSize -> {200, 200}, ImagePadding -> 20,
  SphericalRegion -> True];
```

6.35 Select elements from a list on condition

This question was posted on the net. Given

```
b:=Table[{x,y},{x,1,6},{y,1,6}]
```

select from it elements x, y which satsify $x + y > 9$

some answers

```
(me)
Select[Flatten[b, 1], #1[[1]] + #1[[2]] > 9 & ]

Out[152]= {{4, 6},
           {5, 5},
           {5, 6},
           {6, 4},
           {6, 5},
           {6, 6}}
```

Adriano Pascoletti answer

```
Cases[b, {x_Integer, y_Integer} /; x + y > 9, {2}]
```

Bill Row answer

```
Cases[Flatten[b, 1], _?(Total[#1] > 9 & )]
```

Murray Eisenberg answer

```
Select[Flatten[b, 1], First[#1] + Last[#1] > 9 & ]
```

6.36 Selecting and replace elements from matrix

Given a matrix, say which has Indeterminate and we want to change all these entries in the matrix by zero.

```
mat = {{-1., -1., Indeterminate, -1., -1.},
       {-1., -1., Indeterminate, -1., -1.},
       {Indeterminate, Indeterminate, Indeterminate, Indeterminate, Indeterminate},
       {-1., -1., Indeterminate, -1., -1.},
       {-1., -1., Indeterminate, -1., -1.}}

p = Position[mat, Indeterminate]
mat = ReplacePart[mat, p -> 0]

Out[169]= {{-1., -1., 0, -1., -1.},
           {-1., -1., 0, -1., -1.},
           {0, 0, 0, 0, 0},
           {-1., -1., 0, -1., -1.},
           {-1., -1., 0, -1., -1.}}
```

another example, given a matrix of values, replace those values which are less than 0.5 by NULL

```
n = 5;
a = Table[RandomReal[], {n}, {n}];
p = Position[a, x_ /; x < 0.5];
a = ReplacePart[a, p -> Null]

Out[173]= {{Null, Null, Null, 0.6781657418995635, 0.7290662037036753},
           {Null, 0.7084980071179792, Null, Null, 0.5811489862295911},
           {Null, Null, 0.8467863882617719, Null, 0.8891915946646993},
           {0.8173279058333203, 0.7272894246356278, Null, Null, 0.8665880423275274},
           {Null, Null, 0.662026816962838, 0.5982839657423036, 0.6603967280952212}}
```

6.37 What are the AppearanceElements names?

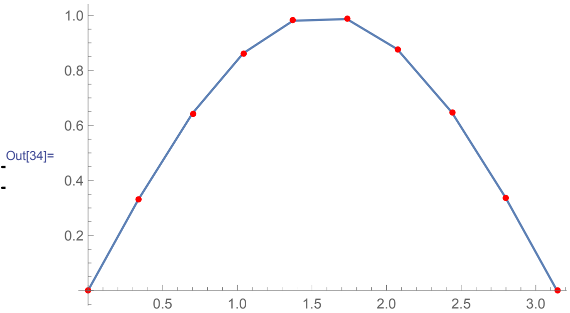
see `full-documentation-for-appearanceelements`

```
list = {
        "AutorunPlayButton",
        "BookmarksButton",
        "BookmarksPlayButton"
        "ContentResizeArea",
        "DirectionButton",
        "FasterSlowerButtons",
        "HideControlsButton",
        "InteractiveTradingChartMenu",
        "InteractiveTradingChartSnapshotButton",
        "InteractiveTradingChartResetButton",
        "InputField",
        "InlineInputField",
        "ManipulatePlayButton",
        "ManipulateMenu",
        "PlayPauseButton",
        "ProgressSlider",
        "ResetButton",
        "SnapshotButton",
        "StepLeftButton",
        "StepRightButton",
        "UpdateButton",
        None};
```

6.38 How to make listplot

one way

```
p = Plot[Sin[x], {x, 0, Pi}, MaxRecursion -> 0,
         PlotPoints -> 10];
data = Cases[Normal[p], x_Line :> First[x], Infinity
Show[p, ListPlot[data, PlotStyle -> Red]]
```



6.39 Getting points from plot once it is plotted

```
p = ContourPlot3D[x^2 + y^3 - z^3 == 0, {x, -2, 2}, {y, -2, 2}, {z, -2, 2},
    PlotPoints -> Automatic]
data = (InputForm@p)[[1, 1, 1]]
```

and

```
data = Reap[DensityPlot[Sin[x*y], {x, 0, 2 Pi}, {y, 0, 2 Pi},
     EvaluationMonitor :> Sow[{x, y, Sin[x*y]}]]][[2, 1]];

ListPlot3D[data]
```

6.40 Making struct/record and array of structs

There is no build-in struct or record in Mathematica. But this is what I do. Since in M a matrix can include non-numeric data, I use a list for a record, and then use a matrix to make an array of records (or array of structs). I just need to make a constant to indicate the field name in the record, to make it easier to reference. Here is an example

```
id = 1;  (*first field name*)
pop = 2; (*second field name*)
name = 3; (*third field name*)

(*now build the array of record, each row is a record*)
m = {{1, 3000, "London"},
     {1, 6000, "New York"},
     {3, 7300, "Moscow"}};

(*now can iterate over the records*)

Do[
 Print@m[[i, id]];
 m[[i, pop]] += 1,
 {i, Length[m]}
]
```

Ok, not very fancy, but easy to setup and data works with all M other functions, since it is just a list of lists.

Some more links on the subject

1. struct-data-type-in-mathematica
2. setting-up-a-struct-in-mathematica-safely
3. using-a-struct-inside-manipulate-to-help-manage-control-variables-how-to-initia
4. struct-equivalent-in-mathematica

6.41 Applying a function using 2 arguments from a list

```
Remove[a, b, c, d, e, f]
Apply[#1 + 3*#2 & , {{a, b}, {c, d}, {e, f}}, 1]

Out[183]= {a + 3*b, c + 3*d, e + 3*f}
```

Or

```
Apply[#1 + 3*#2 & , {{a, b}, {c, d}, {e, f}}, {1}]

Out[184]= {a + 3*b, c + 3*d, e + 3*f}
```

6.42 Using Sow and Reap

```
ListAnimate[Flatten[Reap[Do[Sow@Plot[Sin[x - c], {x, 0, 4 Pi},
      Ticks -> None], {c, 0, 2 Pi - Pi/10, Pi/10}]]]]
```

6.43 Making comments use monospaced fonts

Thanks to Alexey Popkov for this

```
SetOptions[EvaluationNotebook[],
 AutoStyleOptions -> {"CommentStyle" -> {FontWeight -> Plain,
     FontColor -> GrayLevel[0.6], ShowAutoStyles -> False,
     ShowSyntaxStyles -> False, AutoNumberFormatting -> False,
     FontFamily -> "Consolas"}}]
```

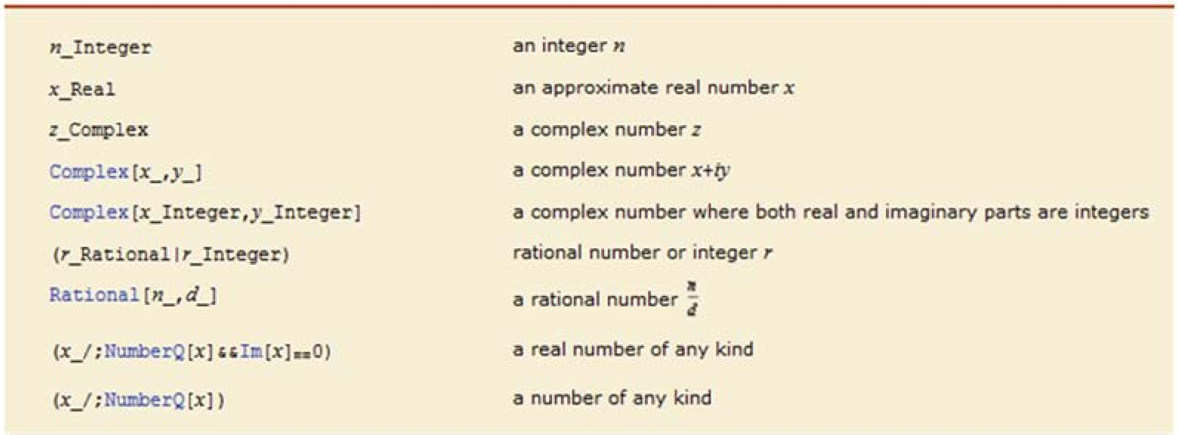
6.44 How to do long division of 2 polynomials?

This came about when I was trying to convert `1/(1-x^2/2)` to normal form, i.e. tell Mathematica to change the above to `1+x^2/2` But doing `Simplify[1/(1-x^2/2)]` or `Expand` does not work. So the only solution I found is to use Series command, as follows

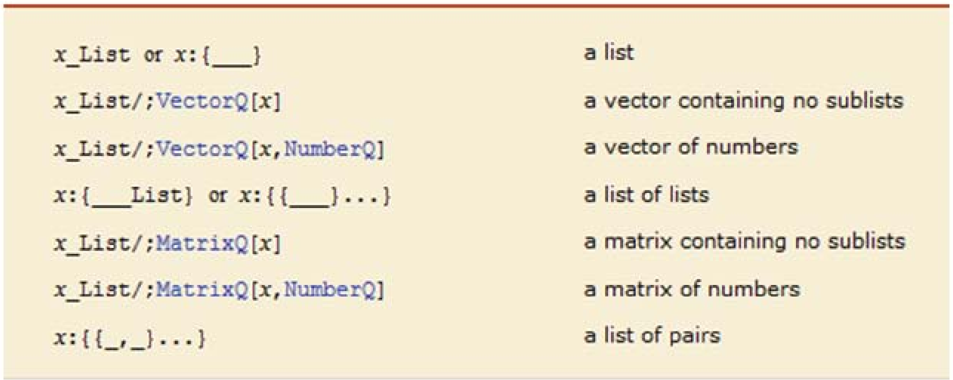
```
Normal[Series[1/r, {x, 0, 2}]]
Out[189]= 1 + x^2/2
```

6.45 Common patterns for function parameters

Use these in parameter "declaration" of functions to make more robust. From the help



n_Integer	an integer n
x_Real	an approximate real number x
z_Complex	a complex number z
Complex[x_,y_]	a complex number $x+iy$
Complex[x_Integer,y_Integer]	a complex number where both real and imaginary parts are integers
(r_Rational|r_Integer)	rational number or integer r
Rational[n_,d_]	a rational number $\frac{n}{d}$
(x_/;NumberQ[x]&&Im[x]==0)	a real number of any kind
(x_/;NumberQ[x])	a number of any kind



x_List or x:{___}	a list
x_List/;VectorQ[x]	a vector containing no sublists
x_List/;VectorQ[x,NumberQ]	a vector of numbers
x:{___List} or x:{{___}...}	a list of lists
x_List/;MatrixQ[x]	a matrix containing no sublists
x_List/;MatrixQ[x,NumberQ]	a matrix of numbers
x:{{_,_}...}	a list of pairs

6.46 How to add rational polynomials?

```
Clear[s];
f1 = 2/(s + 3);
f2 = 7/(s^2 + 2.5*s + 7);
Simplify[Together[f1 + f2]]

Out[193]= (35. + 12.*s + 2.*s^2)/(21. + 14.5*s + 5.5*s^2 + s^3)
```

6.47 How to use options in functions?

```
Options[myFun] = {form -> "linear"};
myFun[x_, OptionsPattern[]] := Module[{}, Print["x=", x, "form=", OptionValue[form]]; ]
myFun[3, form -> "quadratic"]
```

This below is also a useful post by David Park on the net on options usage in packages

`msg00335.html`

6.48 How to replace patterns in expressions

6.48.1 example 1

by Andrzej Kozlowski on math group, July 2010:

Suppose in the expression `2/3 I + x/y I` you wish to replace all fractions (that is `2/3` and `x/y`) by `r` and `I` by `d`. Without worrying about evaluation you can do this as follows:

```
Unevaluated[Unevaluated[(2/3)*I + (x/y)*I] /. HoldPattern[(x_)/(y_)] -> r]
      /. HoldPattern[I] -> d

Out[200]= -d + d*(1 - x^2/2)
```

If you allow the expression to evaluate the patterns will no longer match. For example, with only one Unevaluated you will get

```
Unevaluated[(2/3)*I + (x/y)*I] /. HoldPattern[(x_)/(y_)] ->
      r /. HoldPattern[I] -> d

Out[201]= -I + I*(1 - x^2/2)
```

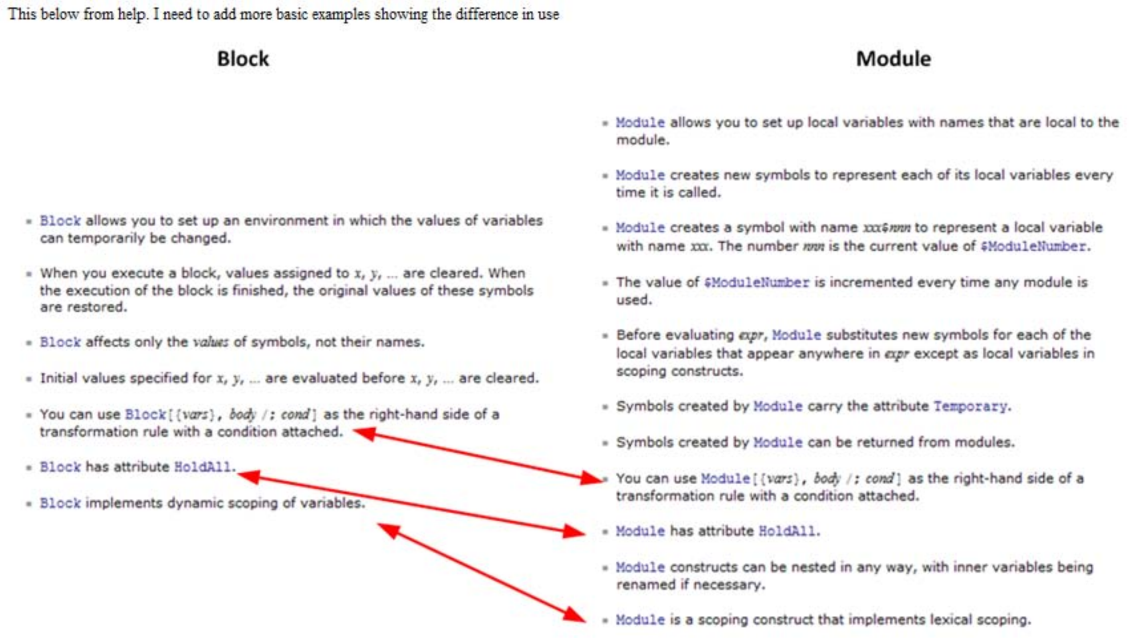
6.48.2 example 2

question: I want to replace y for x everywhere except in Exp[x].

Answer by Bob Hanlon on the net. `messageID=7120881&tstart=0`

```
Remove["Global`*"]
expr = a*x + b*x^2 - c*Exp[x];
expr /. {Exp[x] -> z, x -> y} /. z -> Exp[x]

Out[211]= (-c)*E^x + a*y + b*y^2
```

6.49 Difference between Block, With and Module



This below from help. I need to add more basic examples showing the difference in use

Block

- Block allows you to set up an environment in which the values of variables can temporarily be changed.
- When you execute a block, values assigned to *x*, *y*, ... are cleared. When the execution of the block is finished, the original values of these symbols are restored.
- Block affects only the *values* of symbols, not their names.
- Initial values specified for *x*, *y*, ... are evaluated before *x*, *y*, ... are cleared.
- You can use Block[{*vars*}, *body* /; *cond*] as the right-hand side of a transformation rule with a condition attached.
- Block has attribute HoldAll.
- Block implements dynamic scoping of variables.

Module

- Module allows you to set up local variables with names that are local to the module.
- Module creates new symbols to represent each of its local variables every time it is called.
- Module creates a symbol with name *xxx*$*nnn* to represent a local variable with name *xxx*. The number *nnn* is the current value of $ModuleNumber.
- The value of $ModuleNumber is incremented every time any module is used.
- Before evaluating *expr*, Module substitutes new symbols for each of the local variables that appear anywhere in *expr* except as local variables in scoping constructs.
- Symbols created by Module carry the attribute Temporary.
- Symbols created by Module can be returned from modules.
- You can use Module[{*vars*}, *body* /; *cond*] as the right-hand side of a transformation rule with a condition attached.
- Module has attribute HoldAll.
- Module constructs can be nested in any way, with inner variables being renamed if necessary.
- Module is a scoping construct that implements lexical scoping.

Block and Module have values, the last expression evaluated is their value, we can see this by making a Grid (or just printing). But module leaked symbols have $ signs

```
Remove["Global`*"]
Grid[{{Module[{x}, x]}}, Frame -> All]
```

Modules and Blocks both execute if they are in the path of code, without calling them. Block:

```
Remove["Global`*"]
x = 4;
Block[{}, If[x == 4, x = 3]];
```

```
x

Out[217]= 3
```

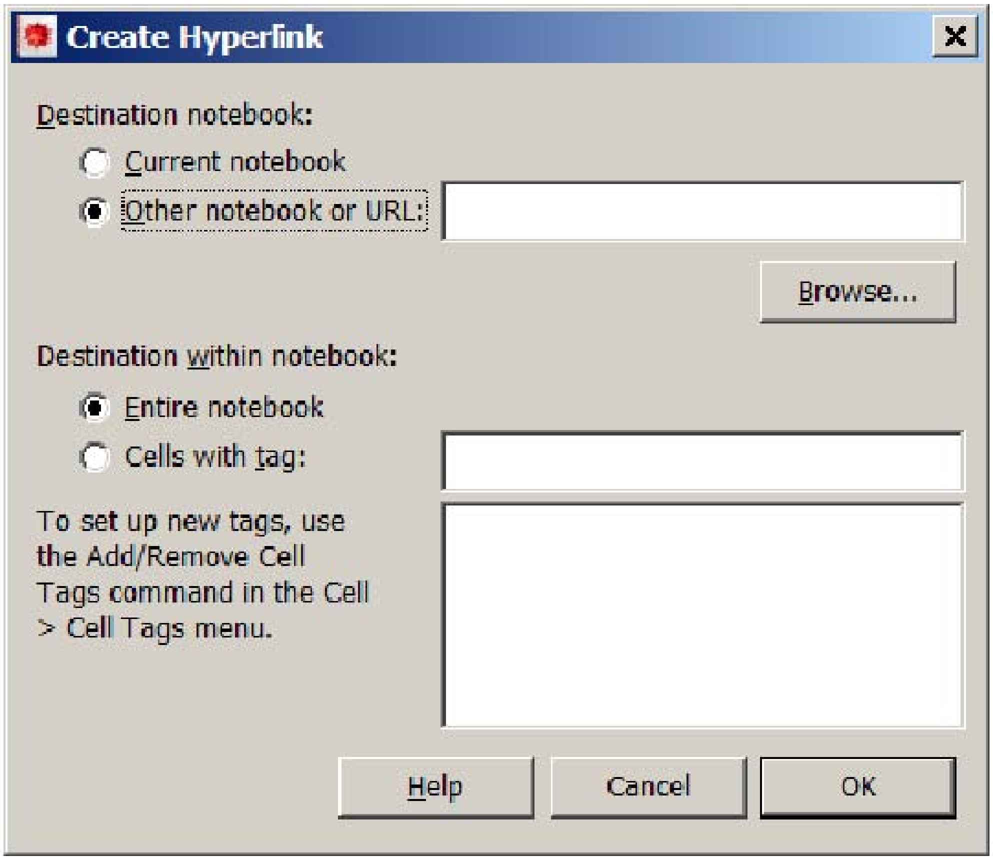
Module:

```
Remove["Global`*"]
x = 4;
Module[{}, If[x == 4, x = 3]];
x

Out[221]= 3
```

6.50 Generating T.O.C. to sections in a notebook

These are the steps I use to make TOC which is just HTML links to internal tags in the notebook, where these cell tags are sections. This way, when I exprt to HTML, I end up with TOC which is hyperlinks to internal locations within the web page.

1. create the section as normal. As in right-click the mouse, and select INSERT new cell, and select Section. Now in the new cell, write the section title.
2. Copy, using the mouse the title of the Section you just wrote so that the title is in the buffer. Now go to `Cell->Cell tags->Add/remove` and in the little window, paste the title of the section there and click `Add`.
3. Go to the top of the document itself, where the TOC is located, and also PASTE the name of the section there. It will be plain text now.
4. Now, using the mouse again, select the text you just pasted, and do right-click and select MAKE hyperlink. This will bring up a menu like this



Destination notebook:

○ Current notebook

● Other notebook or URL:

Destination within notebook:

● Entire notebook

○ Cells with tag:

To set up new tags, use the Add/Remove Cell Tags command in the Cell > Cell Tags menu.

5. Select the option Current notebook from above, and this will bring up a list of all cell tags below. Scroll down looking for the same title there and click on it. This will make the TOC entry HTML link.
6. Now do SAVE AS HTML, and the notebook will be saved as HTML and the TOC will be links to the sections

6.51 Extracting values in a list of the form x->value

```
lst = {{x -> 4, y -> 7}, {x -> 2, y -> 5}, {x -> -1, y -> 10}}
```

one way

```
({#1[[1,2]], #1[[2,2]]} & ) /@ lst

Out[223]= {{4, 7}, {2, 5}, {-1, 10}}
```

another way

```
Cases[lst, {_ -> n_, _ -> m_} :> {n, m}]
```

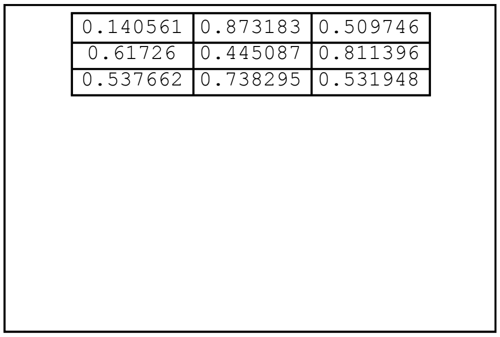
```
Out[224]= {{4, 7}, {2, 5}, {-1, 10}}
```

6.52 Aligning inside Framed environment

One way us to use `Item`

```
mat = Table[Random[], {3}, {3}];
Framed[
  Item[Grid[mat, Frame -> All, Alignment -> Center],
  Alignment -> {Center, Top}], ImageSize -> {300, 20
```

Out[36]=

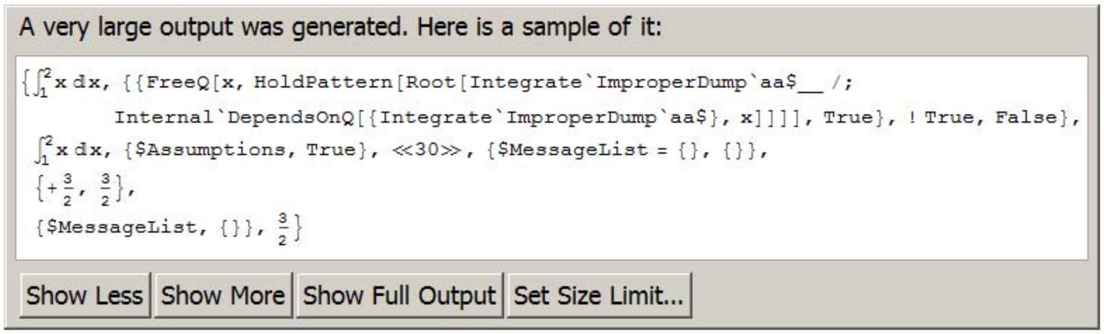


0.140561	0.873183	0.509746
0.61726	0.445087	0.811396
0.537662	0.738295	0.531948

6.53 Aligning individual row of a grids

One way us to use `Item`

```
Grid[{{"row1,row1"}, {"row2"}, {"row3"}},
    Frame -> All]

Grid[{{"row1,row1"},
  {Item["row2", Alignment -> Left]},
  {"row3"}}, Frame -> All]
```



A very large output was generated. Here is a sample of it:

```
{$\int_1^2 x\,dx$, {{FreeQ[x, HoldPattern[Root[Integrate`ImproperDump`aa$__ /;
        Internal`DependsOnQ[{Integrate`ImproperDump`aa$}, x]]]], True}, ! True, False},
 $\int_1^2 x\,dx$, {$Assumptions, True}, <<30>>, {$MessageList = {}, {}},
 {+$\frac{3}{2}$, $\frac{3}{2}$},
 {$MessageList, {}}, $\frac{3}{2}$}
```

6.54 Printing numerical value not in scientific notation?

Use `NumberForm`

```
NumberForm[1./10^6, ExponentFunction -> (Null & )]

Out[227]  0.000001
```

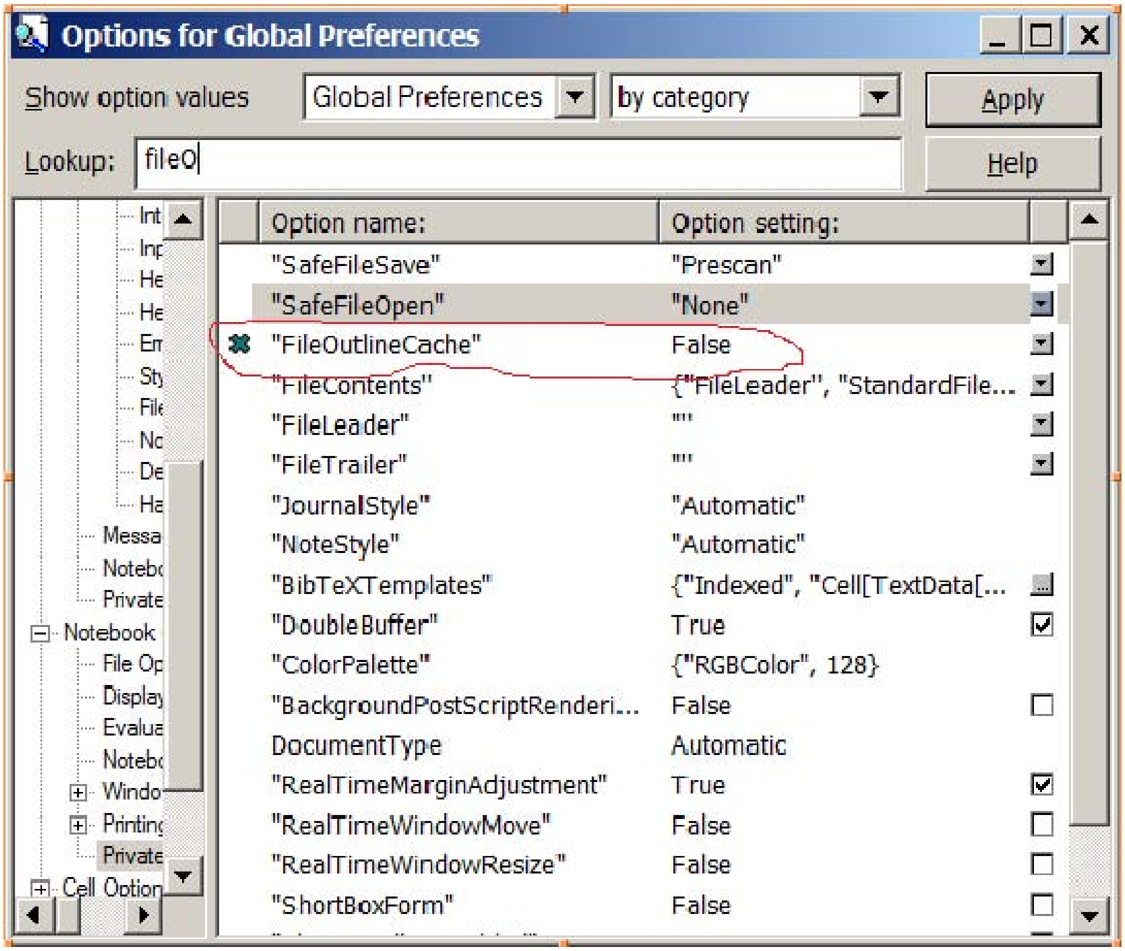
6.55 Clearing notebook cache from any old symbols

Sometimes I get the case that the notebook retain old definitions and symbols even after I deleted them from the notebook. This happened when I was using a Demonstration stylesheet and had an separate initilization cell, and had added `SaveDefinitions->True` in the Manipulate cell.

To make sure the notebook clears any old symbols, enter this command in the notebook once

```
SetOptions[EvaluationNotebook[],
  PrivateNotebookOptions -> {"FileContents" -> {"NotebookData"},
   "FileOutlineCache" -> False}]
```

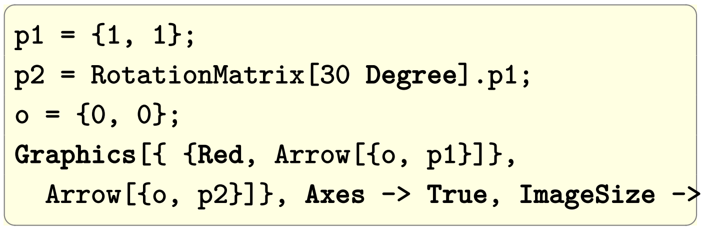
In addition, I change the preferences like this:



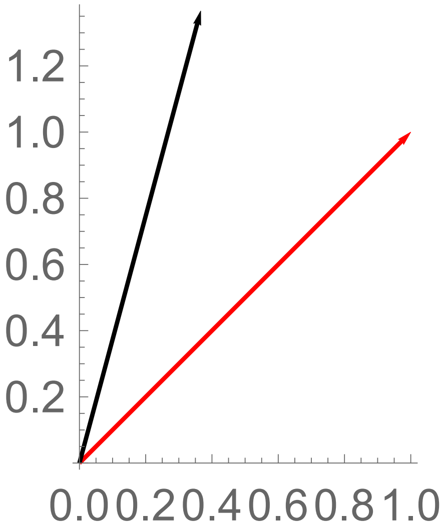
Option name:	Option setting:
"SafeFileSave"	"Prescan"
"SafeFileOpen"	"None"
"FileOutlineCache"	False
"FileContents"	{"FileLeader", "StandardFile...
"FileLeader"	""
"FileTrailer"	""
"JournalStyle"	"Automatic"
"NoteStyle"	"Automatic"
"BibTeXTemplates"	{"Indexed", "Cell[TextData[...
"DoubleBuffer"	True ☑
"ColorPalette"	{"RGBColor", 128}
"BackgroundPostScriptRenderi...	False □
DocumentType	Automatic
"RealTimeMarginAdjustment"	True ☑
"RealTimeWindowMove"	False □
"RealTimeWindowResize"	False □
"ShortBoxForm"	False □

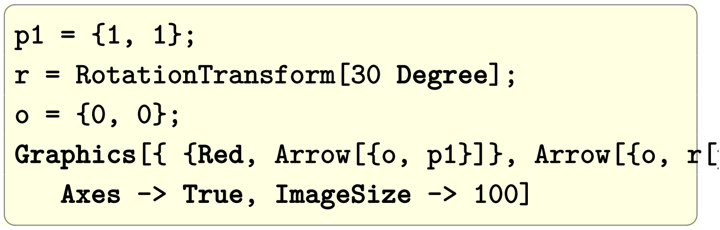
6.56 RotationMatrix and RotationTransform

The call for each is as follows

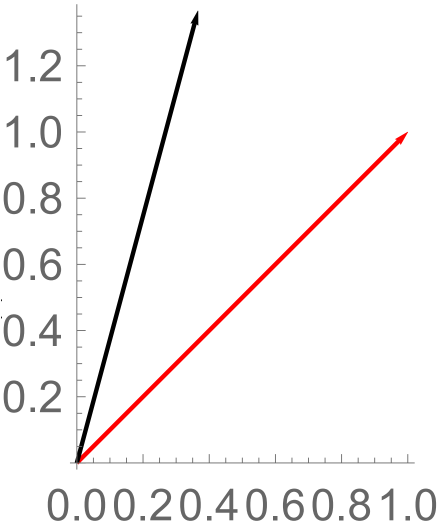


```
p1 = {1, 1};
p2 = RotationMatrix[30 Degree].p1;
o = {0, 0};
Graphics[{ {Red, Arrow[{o, p1}]},
  Arrow[{o, p2}]}, Axes -> True, ImageSize ->
```



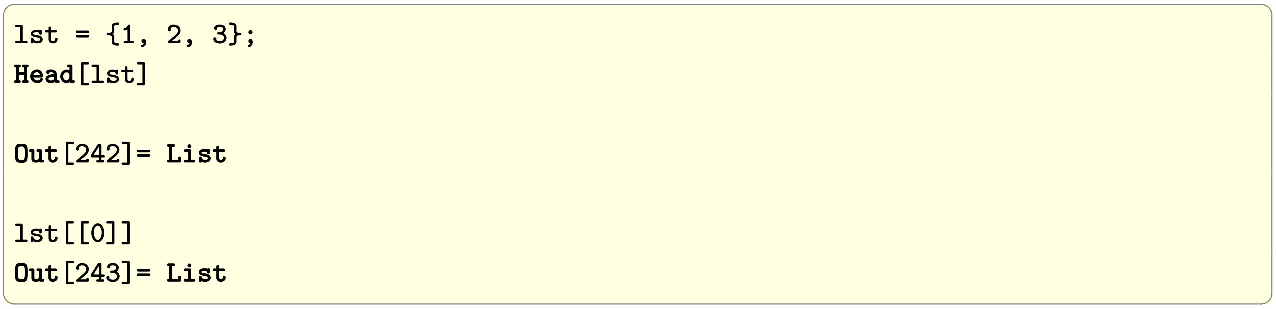


```
p1 = {1, 1};
r = RotationTransform[30 Degree];
o = {0, 0};
Graphics[{ {Red, Arrow[{o, p1}]}, Arrow[{o, r[
   Axes -> True, ImageSize -> 100]
```



6.57 How to change the head of a list?

A list has a Head at its zero index position



```
lst = {1, 2, 3};
Head[lst]

Out[242]= List


lst[[0]]
Out[243]= List
```

By changing the head we use Apply. For example, to add the numbers of the above lst, we need to change the Head from List to Plus. There is a command in Mathematica to change the Head, called `Apply`

```
Plus @@ lst

Out[244]= 6
```

We could have used the zero index trick, but it is better to use Apply:

```
lst[[0]] = Plus
Out[245]= Plus

lst
Out[246]= 6
```

If we have a list of lists, like this

```
lst = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
Out[247]= {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
```

And we wanted to change the head of each list in it, for example, we want the product of each list shown, then we need to change the head of each list to Times. To do that, the follwing short but might be strange looking command

```
Apply[Times, lst, {1}]
Out[248]= {6, 120, 504}
```

Another way to do the above is to Map the Apply function

```
(Times @@ #1 & ) /@ lst
Out[249]= {6, 120, 504}
```

or, little shorter version of the above:

```
(Times @@ #1 & ) /@ lst
Out[250]= {6, 120, 504}
```

6.58 Displaying polynomial from higher to lower order

Mathematica default display of polynomial is reverse the traditional form:

```
poly = x^3 + a*x + b*x^2 + c + d
Out[251]= 27 + 3*a + 9*b + c + d
```

use `Traditional` Form with `ParameterVariables` to make it appear as in text books

```
TraditionalForm[poly, ParameterVariables :> {a, b, c, d}]
```

6.59 How to understand symbol shadwing?

See this article by David Wagner `http://www.mathematica-journal.com/issue/v6i2/columns/wagner/wagner62.pdf`

And `why-are-some-function-names-red`

6.60 How to do OO in Mathematica?

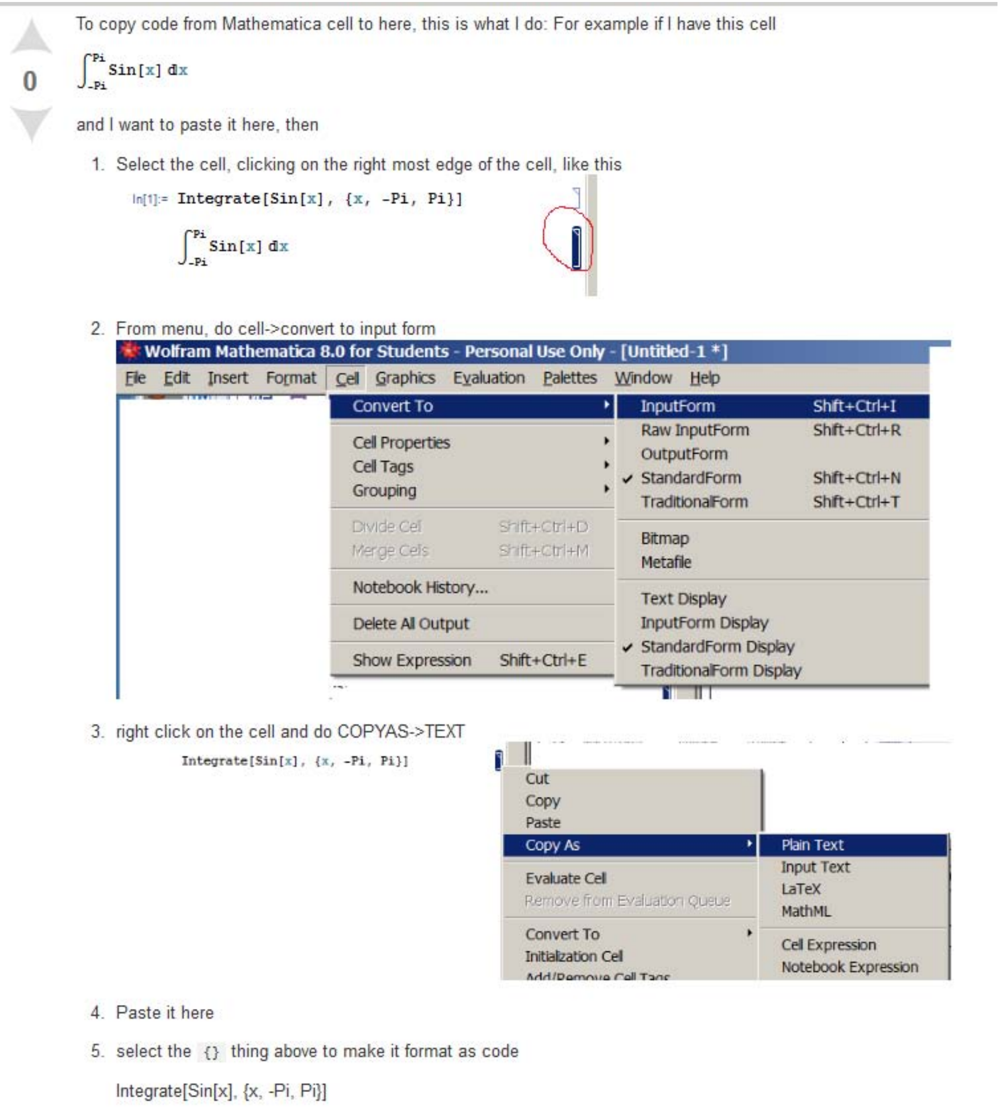
Some note here

6.61 How to sort a list of numbers?

`Sort[]` sorts numbers from small to large by default. By supplying a function, one can change this as needed

```
lst = {1, 2, 5, 3, 7};
Sort[lst]
Out[255]= {1, 2, 3, 5, 7}

Sort[lst, #1 < #2 & ]
Out[256]= {1, 2, 3, 5, 7}

Sort[lst, #1 > #2 & ]
Out[257]= {7, 5, 3, 2, 1}
```

6.62 Copy/paste code to Stackexchange?



0

To copy code from Mathematica cell to here, this is what I do: For example if I have this cell

$\int_{-Pi}^{Pi} \texttt{Sin[x]}\, dx$

and I want to paste it here, then

1. Select the cell, clicking on the right most edge of the cell, like this

```
In[1]:= Integrate[Sin[x], {x, -Pi, Pi}]
```

$\int_{-Pi}^{Pi} \texttt{Sin[x]}\, dx$

2. From menu, do cell->convert to input form

Wolfram Mathematica 8.0 for Students - Personal Use Only - [Untitled-1 *]
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
Convert To
Cell Properties
Cell Tags
Grouping
Divide Cell Shift+Ctrl+D
Merge Cells Shift+Ctrl+M
Notebook History...
Delete All Output
Show Expression Shift+Ctrl+E
InputForm Shift+Ctrl+I
Raw InputForm Shift+Ctrl+R
OutputForm
✓ StandardForm Shift+Ctrl+N
TraditionalForm Shift+Ctrl+T
Bitmap
Metafile
Text Display
InputForm Display
✓ StandardForm Display
TraditionalForm Display

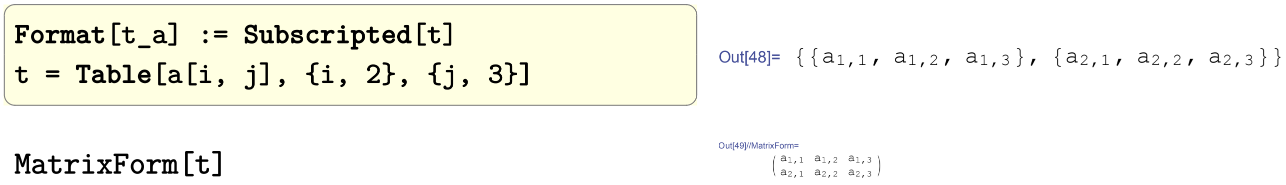
3. right click on the cell and do COPYAS->TEXT

```
Integrate[Sin[x], {x, -Pi, Pi}]
```

4. Paste it here
5. select the {} thing above to make it format as code

```
Integrate[Sin[x], {x, -Pi, Pi}]
```

6.63 Making automatic subscripted matrix notation



```
Format[t_a] := Subscripted[t]
t = Table[a[i, j], {i, 2}, {j, 3}]
```

Out[48]= $\{\{a_{1,1}, a_{1,2}, a_{1,3}\}, \{a_{2,1}, a_{2,2}, a_{2,3}\}\}$

```
MatrixForm[t]
```

Out[49]//MatrixForm=

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{pmatrix}$$

6.64 Finding Names of buildin contexts and options

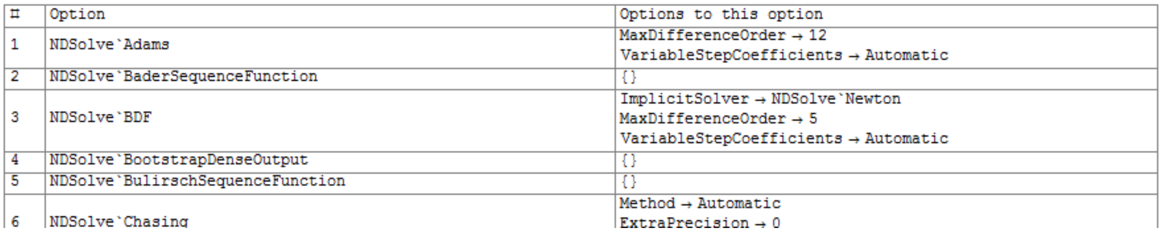
To find say all names in NDSolve and options used by that name if any do (this example is for NDSolve)

```
getList[name_String] := Module[{options, idx},
   options = Names[name <> "`*"];
   options = ToExpression /@ options;
   options = {#, Options[#]} & /@ options;
   idx = Range[Length[options]];
   options = {#[[1]], TableForm[#[[2]]]} & /@ options;
   options = Insert[options[[#]], #, 1] & /@ idx;
   options = Insert[options, {"#", "Option", "Options to this option"}, 1];
   Grid[options, Frame -> All, Alignment -> Left, FrameStyle -> Directive[Thickness[.005], Gra
   ];
```

then call it with

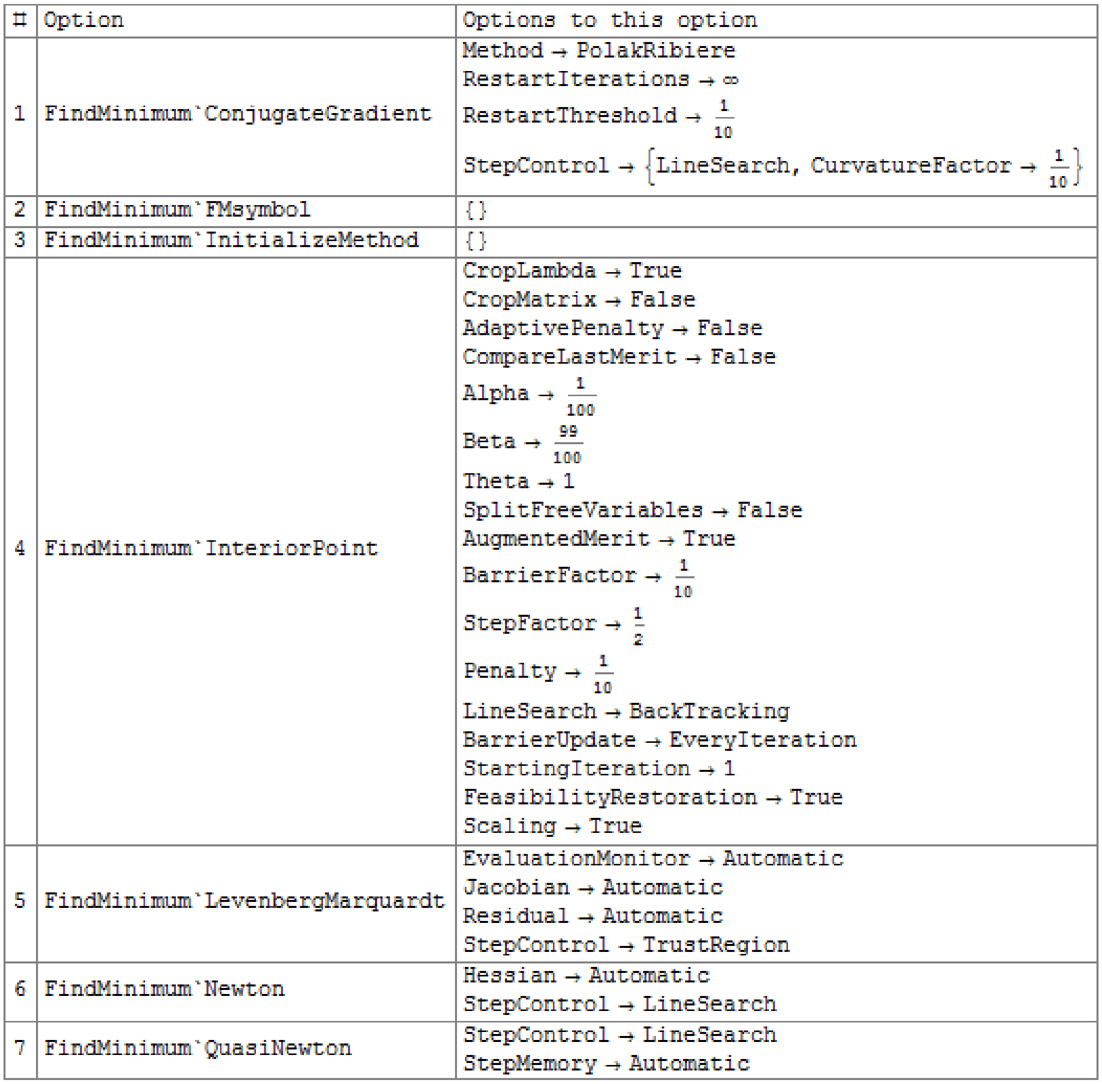
```
getList["NDSolve"]
```

It will produce large table. Here is part of it



#	Option	Options to this option
1	NDSolve`Adams	MaxDifferenceOrder → 12 VariableStepCoefficients → Automatic
2	NDSolve`BaderSequenceFunction	{}
3	NDSolve`BDF	ImplicitSolver → NDSolve`Newton MaxDifferenceOrder → 5 VariableStepCoefficients → Automatic
4	NDSolve`BootstrapDenseOutput	{}
5	NDSolve`BulirschSequenceFunction	{}
6	NDSolve`Chasing	Method → Automatic ExtraPrecision → 0

```
getList["FindMinimum"]
```



♯	Option	Options to this option
1	FindMinimum`ConjugateGradient	Method → PolakRibiere RestartIterations → ∞ RestartThreshold → $\frac{1}{10}$ StepControl → $\left\{\right.$LineSearch, CurvatureFactor → $\left.\frac{1}{10}\right\}$
2	FindMinimum`FMsymbol	{}
3	FindMinimum`InitializeMethod	{}
4	FindMinimum`InteriorPoint	CropLambda → True CropMatrix → False AdaptivePenalty → False CompareLastMerit → False Alpha → $\frac{1}{100}$ Beta → $\frac{99}{100}$ Theta → 1 SplitFreeVariables → False AugmentedMerit → True BarrierFactor → $\frac{1}{10}$ StepFactor → $\frac{1}{2}$ Penalty → $\frac{1}{10}$ LineSearch → BackTracking BarrierUpdate → EveryIteration StartingIteration → 1 FeasibilityRestoration → True Scaling → True
5	FindMinimum`LevenbergMarquardt	EvaluationMonitor → Automatic Jacobian → Automatic Residual → Automatic StepControl → TrustRegion
6	FindMinimum`Newton	Hessian → Automatic StepControl → LineSearch
7	FindMinimum`QuasiNewton	StepControl → LineSearch StepMemory → Automatic

6.65 Making escape key add around elements

if one types in 1,2,3,4 is there is a way to select these and have {} automatically put around them to make a list {1,2,3,4} using escape key shortcut?

Answer by Chris Degnen who wrote this

```
FrontEndExecute[FrontEnd`AddMenuCommands["DuplicatePreviousOutput",
   {Delimiter, MenuItem["Make List", FrontEnd`KernelExecute[nb = SelectedNotebook[];
sel = NotebookRead[nb];
       NotebookWrite[nb, Cell[BoxData[RowBox[{"{", sel, "}"}]]]]],
     MenuKey["u", Modifiers -> {"Control"}],
     MenuEvaluator -> Automatic]}]]
```

put it in the init file to load at start-up. See `wrap-text-selection-in-brackets-in-mathematica`

6.66 Searching for substring inside a larger string

I needed to do this when I was parsing some output. The problem is like this: given a string say `"foo[] boo[] more goo[] more"` and wanted to look for pattern like this `"__["`

In other words, a letter or more that end with `"["`, but needed to find the first one. Hence in the above, I wanted to find `"foo"`.

2 ways to do this:

```
s = "foo[] boo[] more goo[] more";
StringCases[s, RegularExpression["^\\w*\\["]]
Out[265]= {foo[}
```

and

```
StringCases[s, Shortest[StartOfString~~__~~"["], Overlaps -> False]
Out[266]= {foo[}
```

6.67 What are most common commands?

6.67.1 Mape or /@

Takes function and applies it to each element in a list

```
f /@ {a, b, c}
Out[267]= {f[a], f[b], f[c]}

(1 + g[#1] & ) /@ {a, b, c}
Out[268]= {1 + g[a], 1 + g[b], 1 + g[c]}

f /@ {a, b, c}
Out[269]= {f[a], f[b], f[c]}
```

6.67.2 Thread

Use when function needs to be called with arguments taken from more than one list, else use Map if argument come from one list

```
Thread[f[{a, b, c}]]
Out[270]= {f[a], f[b], f[c]}

f /@ {a, b, c}
Out[271]= {f[a], f[b], f[c]}

Thread[f[{a, b, c}, {1, 2, 3}]]
Out[272]= {f[a, 1], f[b, 2], f[c, 3]}
```

6.67.3 MapThread

```
MapThread[f, {{a, b, c}, {1, 2, 3}}]
Out[273]= {f[a, 1], f[b, 2], f[c, 3]}
```

In this case gives the same answer as using Thread

```
Thread[f[{a1, a2, a3}, {b1, b2, b3}]]
Out[274]= {f[a1, b1], f[a2, b2], f[a3, b3]}
```

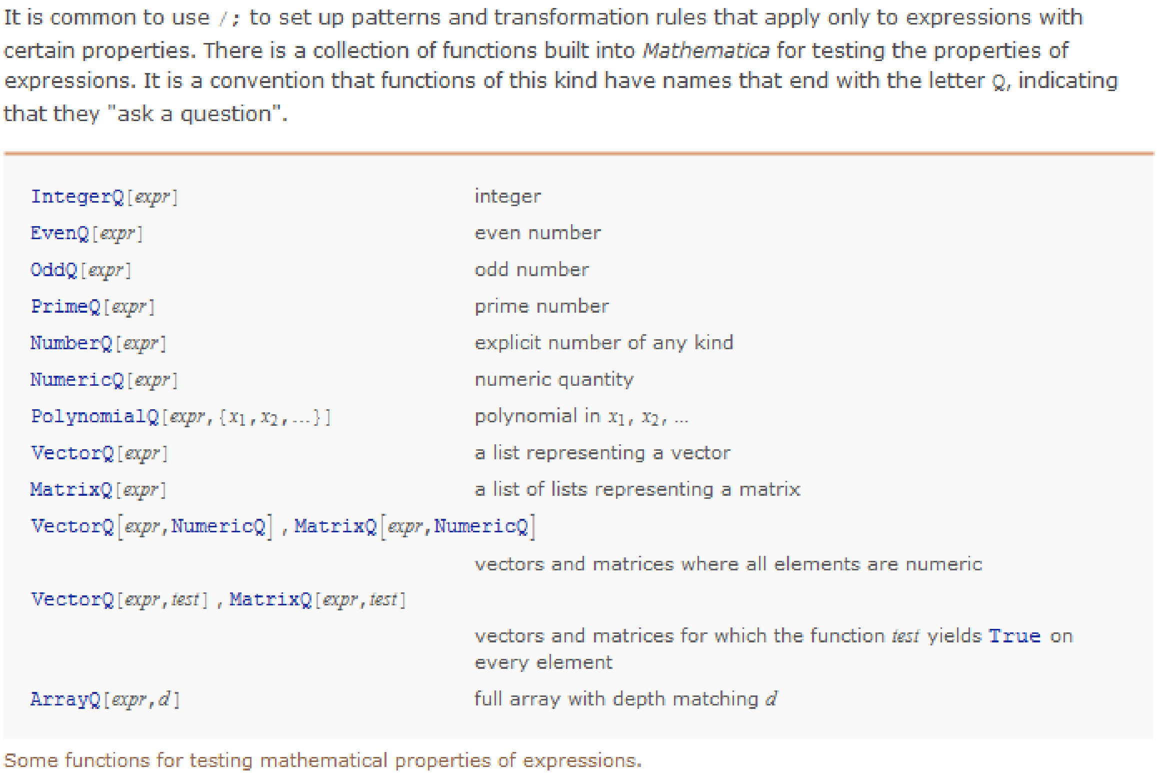
This is only when the lists are one level. For 2 levels we have to use MapThread. This shows the difference

```
MapThread[f, {{{a, b}, {c, d}}, {{1, 2}, {3, 4}}}]
Out[275]= {f[{a, b}, {1, 2}], f[{c, d}, {3, 4}]}

Thread[f[{{{a, b}, {c, d}}, {{1, 2}, {3, 4}}}]]
Out[276]= {f[{{a, b}, {c, d}}], f[{{1, 2}, {3, 4}}]}
```

6.68 common signature definitions

see `tutorial/PatternsOverview` this below from `tutorial/PuttingConstraintsOnPatterns`

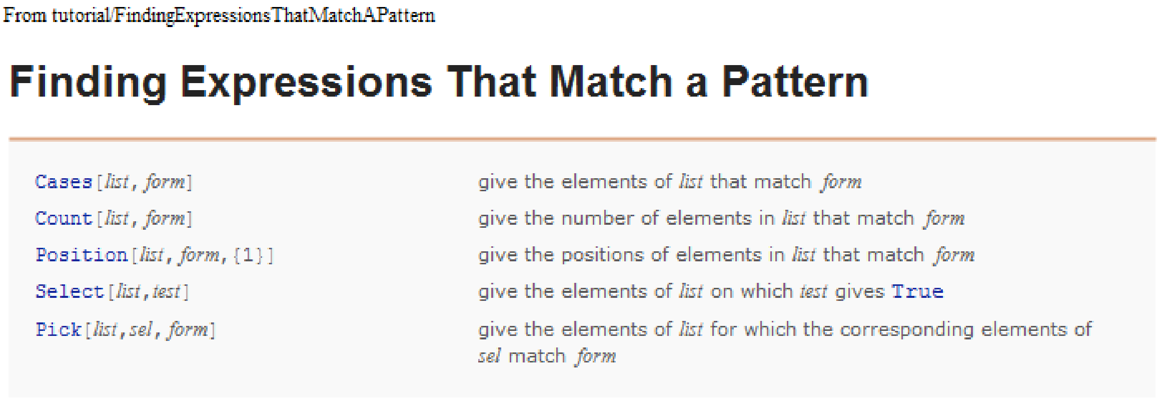
See also `what-is-the-recommended-way-to-check-that-a-list-is-a-list-of-numbers-in-argumen`



It is common to use `/;` to set up patterns and transformation rules that apply only to expressions with certain properties. There is a collection of functions built into *Mathematica* for testing the properties of expressions. It is a convention that functions of this kind have names that end with the letter `Q`, indicating that they "ask a question".

`IntegerQ[`*expr*`]`	integer
`EvenQ[`*expr*`]`	even number
`OddQ[`*expr*`]`	odd number
`PrimeQ[`*expr*`]`	prime number
`NumberQ[`*expr*`]`	explicit number of any kind
`NumericQ[`*expr*`]`	numeric quantity
`PolynomialQ[`*expr*`, {`x_1`,`x_2`,…}]`	polynomial in x_1, x_2, …
`VectorQ[`*expr*`]`	a list representing a vector
`MatrixQ[`*expr*`]`	a list of lists representing a matrix
`VectorQ[`*expr*`,NumericQ]` , `MatrixQ[`*expr*`,NumericQ]`	vectors and matrices where all elements are numeric
`VectorQ[`*expr*`,`*test*`]` , `MatrixQ[`*expr*`,`*test*`]`	vectors and matrices for which the function *test* yields `True` on every element
`ArrayQ[`*expr*`,`*d*`]`	full array with depth matching *d*

Some functions for testing mathematical properties of expressions.



From tutorial/FindingExpressionsThatMatchAPattern

Finding Expressions That Match a Pattern

`Cases[`*list*`,` *form*`]`	give the elements of *list* that match *form*
`Count[`*list*`,` *form*`]`	give the number of elements in *list* that match *form*
`Position[`*list*`,` *form*`, {1}]`	give the positions of elements in *list* that match *form*
`Select[`*list*`,`*test*`]`	give the elements of *list* on which *test* gives `True`
`Pick[`*list*`,`*sel*`,` *form*`]`	give the elements of *list* for which the corresponding elements of *sel* match *form*

6.68.1 some signatures collection

integer

```
foo[(x_)?(Element[#1, Integers] & )] := x
foo[x_Integer] := x
foo[x_Integer] := x
```

integer strictly positive

```
foo[(x_)?(IntegerQ[#1] && #1 > 0 & )] := x
foo[x_Integer /; x > 0] := x
foo[(x_Integer)?Positive] := x
foo[x_Integer /; x > 0] := x
```

integer strictly negative

```
foo[(x_)?(IntegerQ[#1] && #1 < 0 & )] := x
foo[x_Integer /; x < 0] := x
foo[(x_Integer)?Negative] := x
foo[x_Integer /; x < 0] := x
```

integer zero or positive

```
foo[(x_)?(IntegerQ[#1] && #1 >= 0 & )] := x
foo[x_Integer /; x >= 0] := x
foo[(x_Integer)?NonNegative] := x
foo[x_Integer /; x >= 0] := x
```

integer zero or negative

```
foo[(x_)?(IntegerQ[#1] && #1 <= 0 & )] := x
foo[x_Integer /; x <= 0] := x
foo[(x_Integer)?NonPositive] := x
foo[x_Integer /; x <= 0] := x
```

integer in some range

```
foo[x_Integer /; x > 3 && x < 7] := x
```

Real

```
foo[x_?(Element[#, Reals] &)] := x
foo[x_Real] := x
```

Real strictly positive

```
foo[x_Real /; x > $MachineEpsilon] := x
foo[x_Real /; x > $MachineEpsilon] := x
foo[x_Real /; Positive[x]] := x
foo[x_ (Element[#, Reals] && Positive[#] &)] := x
```

Real strictly negative

```
foo[x_Real /; x < $MachineEpsilon] := x
foo[x_Real /; x < $MachineEpsilon] := x
foo[x_Real /; Negative[x]] := x
foo[x_?(Element[#, Reals] && Negative[#] &)] := x
```

Real zero or positive

```
foo[x_Real /; x >= $MachineEpsilon] := x
foo[x_Real /; x >= $MachineEpsilon] := x
foo[x_Real /; Positive[x] || x == 0] := x
foo[x_ (Element[#, Reals] && (Positive[#] || # == 0) &)] := x
```

Real zero or negative

```
foo[x_Real /; x <= $MachineEpsilon] := x
foo[x_Real /; x <= $MachineEpsilon] := x
foo[x_Real /; Negative[x] || x == 0] := x
foo[x_ (Element[#, Reals] && (Negative[#] || # == 0) &)] := x
```

Real in some range

```
foo[x_ (Element[#, Reals] && ((# - 3) > $MachineEpsilon && (7 - #) > $MachineEpsilon) &)] := x
foo[x : _Real /; (x - 3) > $MachineEpsilon && (7 - x) > $MachineEpsilon] := x
```

Boolean

```
foo[x_?(Element[#, Booleans] &)] := x
```

any numerical parameter

```
foo[x_?(Element[#, Reals] &)] := x
foo[x_?(NumericQ[#] &)] := x
foo[x : _?NumericQ] := x
```

checks for Head Real, Integer, Ratioal and Complex

```
foo[x_?(NumberQ[#] &)] := x
```

general complex number

```
foo[x_Complex] := x
foo[x_?(Not@FreeQ[#, _Complex] &)] := x
```

list of any dimension, ragged lists, 1D vectors, 2D, any content

```
foo[x_List] := x
```

1D list (i.e. vector)

```
foo[x_?(VectorQ[#] &)] := x
```

Numeric 1D list

```
foo[x_?(VectorQ[#, NumericQ] &)] := x
```

Numeric 1D list

```
foo[x_?(VectorQ[#, NumericQ] &)] := x
foo[x : {_?NumericQ ..}] := x
foo[x : {__?NumericQ }] := x
foo[x_?(VectorQ[#, IntegerQ] &)] := x
```

2D matix of numbers

```
foo[x_?(MatrixQ[#, NumericQ] &)] := x
foo[x : {{_?NumericQ ..}}] := x
foo[x : {{__?NumericQ }}] := x
```

2D matrix numeric but contains no complex numbers

```
foo[x_?(MatrixQ[#, NumericQ] && FreeQ[#, _Complex] &)] := x
```

2D matrix of strings

```
foo[x_?(MatrixQ[#, StringQ] &)] := x
```

6.69 How to check for Head of expression?

use MatchQ

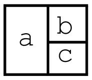
```
MatchQ[1/3, _Rational]
Out[277]= True

MatchQ[3, _Integer]
Out[278]= True

Or for the above can do

IntegerQ[3]
Out[279]= True
```

6.70 How to make different Grids

```
Grid[{
  {Item[a, Alignment -> Center], b},
  {SpanFromAbove, c}}, Frame -> All]

Grid[{
  {Item[a, Alignment -> Center], Item[Column[{b, c}]]}}, Frame -> All]
```

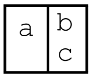
```
Grid[{
  {Item[a, Alignment -> Center], Item[Column[{b, c}, Frame -> All]]}}, Frame -> All]

Grid[{
  {a, Item[b, Alignment -> Center]},
  {c, SpanFromAbove}}, Frame -> All]

Grid[{
  {Item[a, Alignment -> Center], Item[b, Alignment -> Center], c},
  {SpanFromAbove, SpanFromAbove, d},
  {SpanFromAbove, e, f}}, Frame -> All]
```

Out[50]=



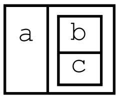
<table><tr><td rowspan="2">a</td><td>b</td></tr><tr><td>c</td></tr></table>

Out[51]=



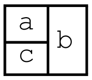
<table><tr><td>a</td><td>b
c</td></tr></table>

Out[52]=



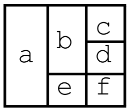
<table><tr><td>a</td><td><table><tr><td>b</td></tr><tr><td>c</td></tr></table></td></tr></table>

Out[53]=



<table><tr><td>a</td><td rowspan="2">b</td></tr><tr><td>c</td></tr></table>

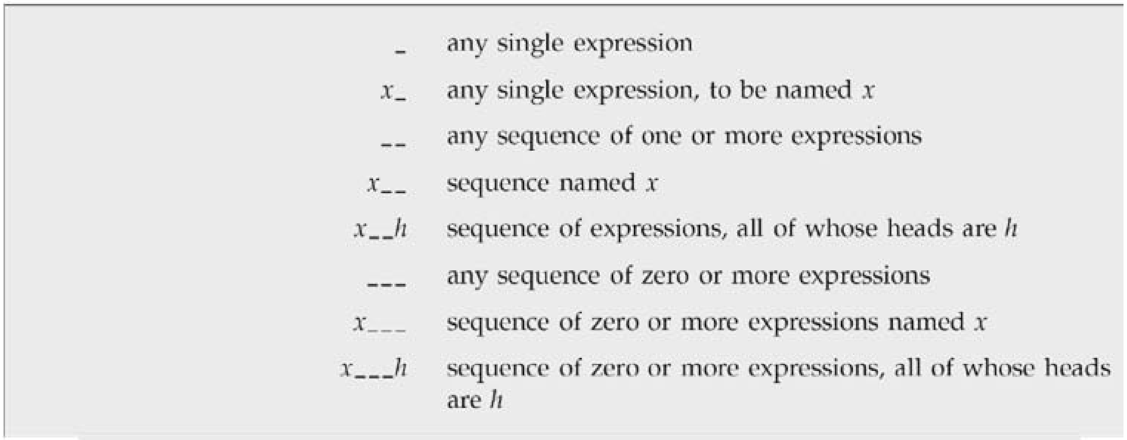
Out[54]=



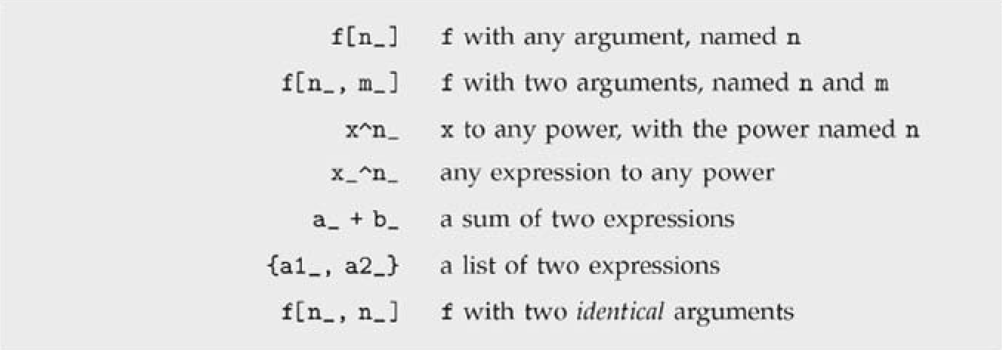
<table><tr><td rowspan="3">a</td><td rowspan="2">b</td><td>c</td></tr><tr><td>d</td></tr><tr><td>e</td><td>f</td></tr></table>

6.71 Common Patterns

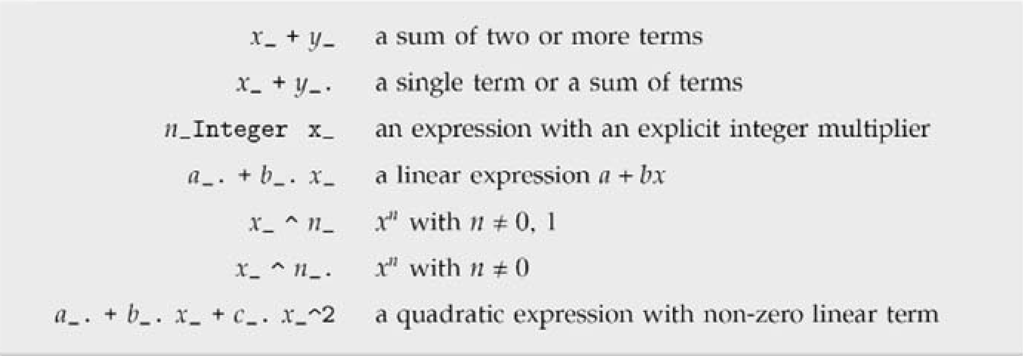
From help



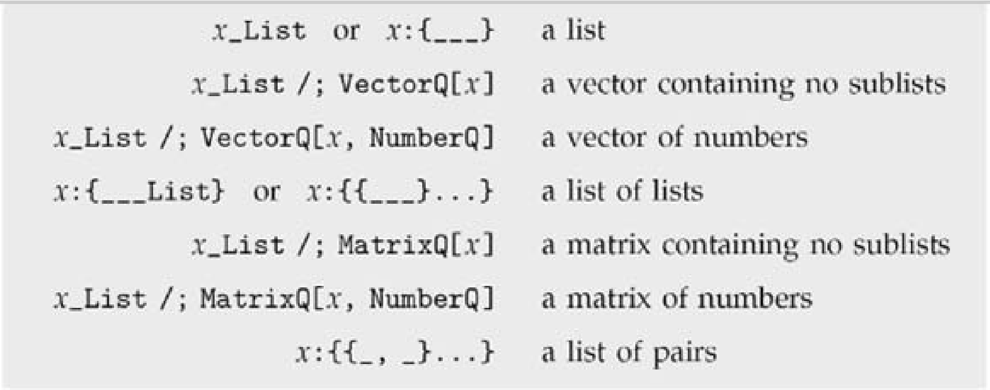
_	any single expression
x_	any single expression, to be named x
__	any sequence of one or more expressions
x__	sequence named x
x__h	sequence of expressions, all of whose heads are h
___	any sequence of zero or more expressions
x___	sequence of zero or more expressions named x
x___h	sequence of zero or more expressions, all of whose heads are h



f[n_]	f with any argument, named n
f[n_, m_]	f with two arguments, named n and m
x^n_	x to any power, with the power named n
x_^n_	any expression to any power
a_ + b_	a sum of two expressions
{a1_, a2_}	a list of two expressions
f[n_, n_]	f with two *identical* arguments



x_ + y_	a sum of two or more terms
x_ + y_.	a single term or a sum of terms
n_Integer x_	an expression with an explicit integer multiplier
a_. + b_. x_	a linear expression $a + bx$
x_ ^ n_	x^n with $n \neq 0, 1$
x_ ^ n_.	x^n with $n \neq 0$
a_. + b_. x_ + c_. x_^2	a quadratic expression with non-zero linear term



x_List or x:{___}	a list
x_List /; VectorQ[x]	a vector containing no sublists
x_List /; VectorQ[x, NumberQ]	a vector of numbers
x:{___List} or x:{{___}...}	a list of lists
x_List /; MatrixQ[x]	a matrix containing no sublists
x_List /; MatrixQ[x, NumberQ]	a matrix of numbers
x:{{_, _}...}	a list of pairs

6.71.1 string matching in list

See `select-and-blank`

```
test = {{"String1", "a"}, {"String2", "b"}, {"String3", "a"}, {"String4", "a"}};
Cases[test, {_String, "a"}]
Out[281]= {{String1, a}, {String3, a}, {String4, a}}

Select[test, MatchQ[#1, {_String, "a"}] & ]
Out[282]= {{String1, a}, {String3, a}, {String4, a}}
```

6.71.2 how to find if one symbolic term starts with minus sign or not?

See `given-a-symbolic-expression-how-to-find-if-starts-with-a-minus-or-not`

```
Clear[x]
p = (_.)*_?Negative;
MatchQ[-3*x^2, p]
Out[285]= True

MatchQ[3*x^2, p]
Out[286]= False

expr = -3*x^2;
(expr /. Thread[Variables[expr] -> 1]) < 0
Out[288]= True

expr = 3*x^2;
(expr /. Thread[Variables[expr] -> 1]) < 0
Out[290]= False
```

6.72 things to remember

Watch out for adding the extra third argument to trigger as show below (which is 1 now). This seems to cause a problem. Was using it in Manipulate and when I added, sometimes the trigger stops firing on its own. When I remove it, it never does stop.

```
Trigger[Dynamic[t0, {t0 = #} &], {0, 10000, 1}, ....]
ToString[#] & /@ a| is same as ToString /@ a
```

6.73 Copying outout cells to another notebook

see `how-to-select-and-delete-all-output-cells`

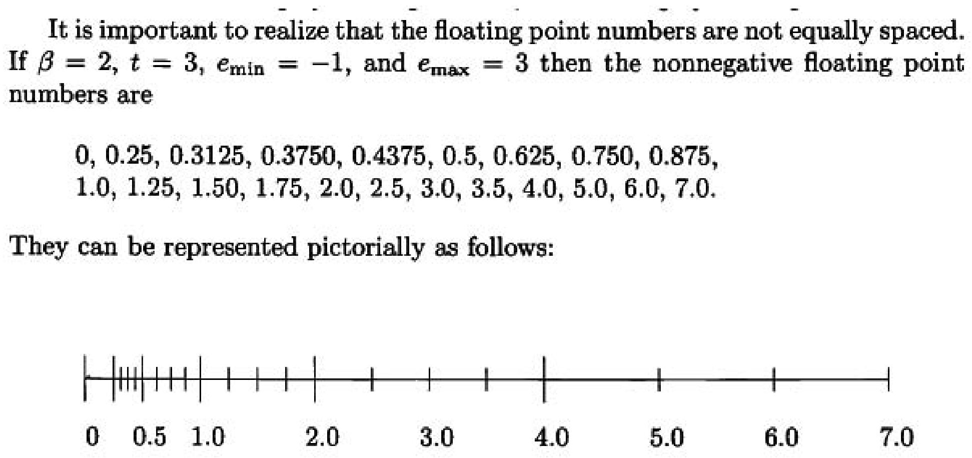
6.74 Mathematica command like Matlab blkdiag

To make a matrix, which contains on its diagonal matrices, Matlab uses the command `blkdiag`. In Mathematica use the following

```
a = {{1, 2, 3}, {4, 5, 6}}
b = {{7, 8}, {9, 10}}
SparseArray[Band[{1, 1}] -> {a, b}]
```

6.75 floating points stuff

Below is from "accuracy and stability of numerical algorithms", by Highma, page 36



It is important to realize that the floating point numbers are not equally spaced. If $\beta = 2$, $t = 3$, $e_{\min} = -1$, and $e_{\max} = 3$ then the nonnegative floating point numbers are

0, 0.25, 0.3125, 0.3750, 0.4375, 0.5, 0.625, 0.750, 0.875,
1.0, 1.25, 1.50, 1.75, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 7.0.

They can be represented pictorially as follows:

0
0.5
1.0
2.0
3.0
4.0
5.0
6.0
7.0

6.76 Mathematica functions that does the same thing

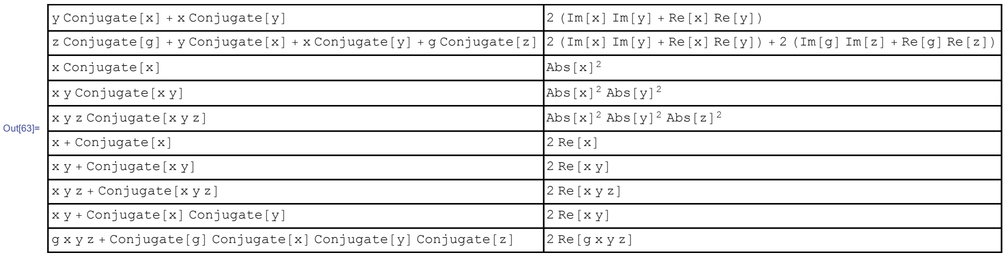
```
ClearAll["Global`*"]
lst = {{a, b}, {c, d}};
MapThread[f, lst]
Out[14]= {f[a, c], f[b, d]}

Thread[f[Sequence @@ lst]]
Out[15]= {f[a, c], f[b, d]}
```

6.77 Complex rules to help simplifications

```
ClearAll[x, y, z, g, foo];
p1 = Conjugate[x_]*Conjugate[y_] :> Conjugate[x*y];
p2 = (x_)*Conjugate[x_] :> Abs[x]^2;
p3 = Abs[(x_)*(y_)]^(n_.) :> Abs[x]^n*Abs[y]^n;
p4 = (x_)*Conjugate[y_] + (y_)*Conjugate[x_] :> 2*(Re[x]*Re[y] + Im[x]*Im[y]);
p5 = (x_) + Conjugate[x_] :> 2*Re[x];
allRules = {p1, p2, p3, p4, p5};
```

test it

```
expr = {{foo = x*Conjugate[y] + y*Conjugate[x]; foo, foo //. allRules},
 {foo = x*Conjugate[y] + y*Conjugate[x] + z*Conjugate[g] + g*Conjugate[z]; foo, foo //. allRule
 {foo = x*Conjugate[x]; foo, foo //. allRules}, {foo = x*y*Conjugate[x*y]; foo, foo //. allRule
 {foo = x*y*z*Conjugate[x*y*z]; foo, foo //. allRules}, {foo = x + Conjugate[x]; foo, foo //. a
 {foo = x*y + Conjugate[x*y], foo; foo //. allRules}, {foo = x*y*z + Conjugate[x*y*z]; foo, foo
 {foo = x*y + Conjugate[x]*Conjugate[y]; foo, foo //. allRules},
 {foo = x*y*z*g + Conjugate[x]*Conjugate[y]*Conjugate[z]*Conjugate[g]; foo, foo //. allRules}}
Grid[expr, Frame -> All, Spacings -> {0.5, 1}, Alignment -> Left]
```



Out[63]=

y Conjugate[x] + x Conjugate[y]	2 (Im[x] Im[y] + Re[x] Re[y])
z Conjugate[g] + y Conjugate[x] + x Conjugate[y] + g Conjugate[z]	2 (Im[x] Im[y] + Re[x] Re[y]) + 2 (Im[g] Im[z] + Re[g] Re[z])
x Conjugate[x]	Abs[x]^2
x y Conjugate[x y]	Abs[x]^2 Abs[y]^2
x y z Conjugate[x y z]	Abs[x]^2 Abs[y]^2 Abs[z]^2
x + Conjugate[x]	2 Re[x]
x y + Conjugate[x y]	2 Re[x y]
x y z + Conjugate[x y z]	2 Re[x y z]
x y + Conjugate[x] Conjugate[y]	2 Re[x y]
g x y z + Conjugate[g] Conjugate[x] Conjugate[y] Conjugate[z]	2 Re[g x y z]

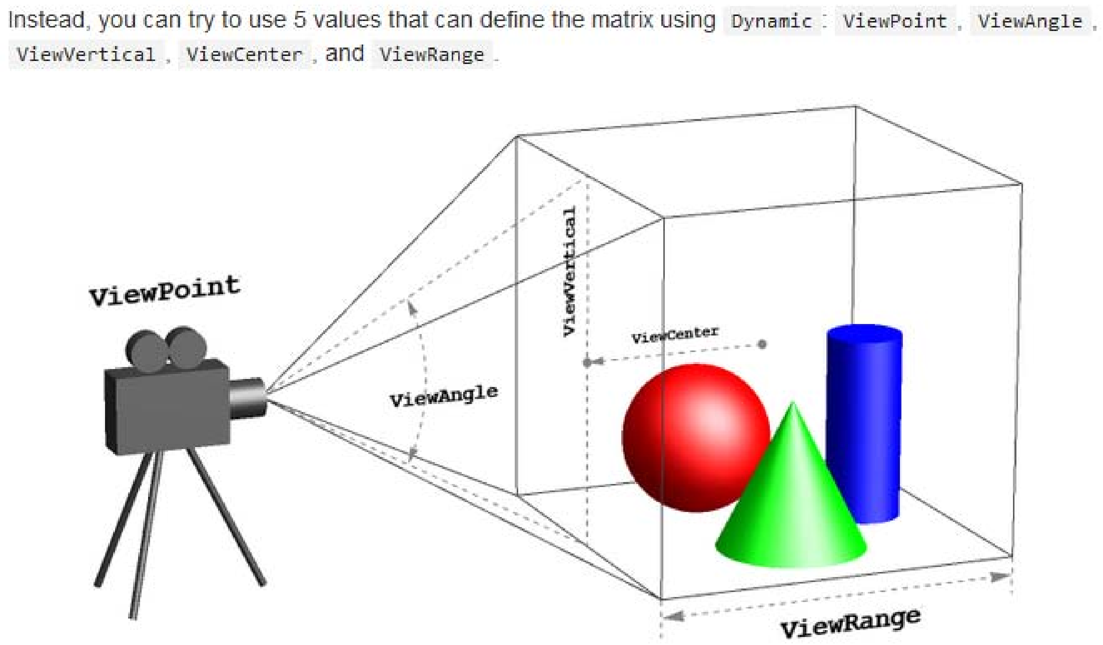
6.78 How to find names of named characters?

See `mathematica/guide/ListingOfNamedCharacters.html`

6.79 How to understand views for 3D Graphics?

From `extract-values-for-viewmatrix-from-a-graphics3d/3538`

by Yu-Sung Chang (Wolfram Research)



Instead, you can try to use 5 values that can define the matrix using `Dynamic` : `ViewPoint` , `ViewAngle` , `ViewVertical` , `ViewCenter` , and `ViewRange` .

6.80 On imagepadding, imageMargins etc.....

1. `ref/ImagePadding.html`
2. `tutorial/GridsRowsAndColumns.html`
3. `ref/PlotRangePadding.html`

6.81 How to thread functions over equations?

```
eq = E^(0.002/t) + E^(0.032/t) == 2*E^(0.03/t)
Thread[Log[eq], Equal]

Out[26]= Log[E^(0.002/t) + E^(0.032/t)] == Log[2*E^(0.03/t)]
```

6.82 Correct way to define function for Integrate use

Watch out when defining a function such as this:

```
f[x_] := Integrate[x - t, {t, 0, x}]
```

The problem is this:

```
In[45]:= f[t]
Out[45]= 0
```

This is becuase the replacement of "x" by "t" changed the integrand to zero.

The correct way is to always use Module symbols for everything inside the function, like this

```
f[x_] := Module[{t}, Integrate[x - t, {t, 0, x}]]
```

Now it gives the correct answer regardless of the symbol used as argument

```
In[46]:= f[t]
Out[46]= t^2/2

In[47]:= f[x]
Out[47]= x^2/2
```

6.83 How to find list of all distributions

```
StringCases[#, ___ ~~ "Distribution" ~~ ___ :> #] & /@ Names["System`*"];
DeleteCases[%, {}]
```

Gives

```
{{"ArcSinDistribution"}, {"BarabasiAlbertGraphDistribution"}, {"BatesDistribution
   "}, {"BeckmannDistribution"}, \
{"BenfordDistribution"}, {"BeniniDistribution"}, {"BenktanderGibratDistribution"},
     {"BenktanderWeibullDistribution"}, \
{"BernoulliDistribution"}, {"BernoulliGraphDistribution"}, {"
   BetaBinomialDistribution"}, {"BetaDistribution"}, \
{"BetaNegativeBinomialDistribution"}, {"BetaPrimeDistribution"}, {"
   BinomialDistribution"}, {"BinormalDistribution"}, \
{"BirnbaumSaundersDistribution"}, {"BorelTannerDistribution"}, {"
    CauchyDistribution"}, {"CensoredDistribution"}, \
{"ChiDistribution"}, {"ChiSquareDistribution"}, {"CompoundPoissonDistribution"},
    {"CopulaDistribution"}, {"CoxianDistribution"}, \
{"DagumDistribution"}, {"DataDistribution"}, {"DavisDistribution"}, {"
    DegreeGraphDistribution"}, {"DirichletDistribution"}, \
{"DiscreteUniformDistribution"}, {"DistributionChart"}, {"DistributionDomain"},
    {"DistributionFitTest"}, \
{"DistributionParameterAssumptions"}, {"DistributionParameterQ"}, {"
    EmpiricalDistribution"}, {"ErlangDistribution"}, \
{"EstimatedDistribution"}, {"ExpGammaDistribution"}, {"ExponentialDistribution"},
     {"ExponentialPowerDistribution"}, \
{"ExtremeValueDistribution"}, {"FailureDistribution"}, {"
    FindDistributionParameters"}, {"FirstPassageTimeDistribution"}, \
{"FisherHypergeometricDistribution"}, {"FisherZDistribution"}, {"
    FRatioDistribution"}, {"FrechetDistribution"}, \
{"GammaDistribution"}, {"GeometricDistribution"}, {"GompertzMakehamDistribution"},
     {"GraphPropertyDistribution"}, \
{"GumbelDistribution"}, {"HalfNormalDistribution"}, {"HistogramDistribution"}, {"
    HotellingTSquareDistribution"}, \
{"HoytDistribution"}, {"HyperbolicDistribution"}, {"HyperexponentialDistribution
    "}, {"HypergeometricDistribution"}, \
{"HypoexponentialDistribution"}, {"InverseChiSquareDistribution"}, {"
    InverseGammaDistribution"}, {"InverseGaussianDistribution"}, \
{"JohnsonDistribution"}, {"KDistribution"}, {"KernelMixtureDistribution"}, {"
    KumaraswamyDistribution"}, {"LandauDistribution"}, \
{"LaplaceDistribution"}, {"LevyDistribution"}, {"LindleyDistribution"}, {"
    LogGammaDistribution"}, {"LogisticDistribution"}, \
{"LogLogisticDistribution"}, {"LogMultinormalDistribution"}, {"
    LogNormalDistribution"}, {"LogSeriesDistribution"}, \
{"MarginalDistribution"}, {"MaxStableDistribution"}, {"MaxwellDistribution"}, {"
    MeixnerDistribution"}, {"MinStableDistribution"}, \
{"MixtureDistribution"}, {"MoyalDistribution"}, {"MultinomialDistribution"}, {"
    MultinormalDistribution"}, \
{"MultivariateHypergeometricDistribution"}, {"MultivariatePoissonDistribution"},
    {"MultivariateTDistribution"}, \
{"NakagamiDistribution"}, {"NegativeBinomialDistribution"}, {"
    NegativeMultinomialDistribution"}, {"NoncentralBetaDistribution"}, \
{"NoncentralChiSquareDistribution"}, {"NoncentralFRatioDistribution"}, {"
    NoncentralStudentTDistribution"}, {"NormalDistribution"}, \
{"OrderDistribution"}, {"ParameterMixtureDistribution"}, {"ParetoDistribution"},
    {"PascalDistribution"}, {"PearsonDistribution"}, \
{"PERTDistribution"}, {"PoissonConsulDistribution"}, {"PoissonDistribution"}, {"
    PolyaAeppliDistribution"}, {"PowerDistribution"}, \
{"PriceGraphDistribution"}, {"ProbabilityDistribution"}, {"ProductDistribution"},
     {"RayleighDistribution"}, \
{"ReliabilityDistribution"}, {"RiceDistribution"}, {"SechDistribution"}, {"
    SinghMaddalaDistribution"}, {"SkellamDistribution"}, \
{"SkewNormalDistribution"}, {"SliceDistribution"}, {"SmoothKernelDistribution"},
    {"SpatialGraphDistribution"}, \
{"SplicedDistribution"}, {"StableDistribution"}, {"StandbyDistribution"}, {"
    StationaryDistribution"}, {"StudentTDistribution"}, \
{"SurvivalDistribution"}, {"SuzukiDistribution"}, {"TransformedDistribution"}, {"
    TriangularDistribution"}, \
{"TruncatedDistribution"}, {"TsallisQExponentialDistribution"}, {"
    TsallisQGaussianDistribution"}, {"TukeyLambdaDistribution"}, \
{"UniformDistribution"}, {"UniformGraphDistribution"}, {"UniformSumDistribution"},
     {"VarianceGammaDistribution"}, \
{"VoigtDistribution"}, {"VonMisesDistribution"}, {"WakebyDistribution"}, {"
    WalleniusHypergeometricDistribution"}, \
{"WaringYuleDistribution"}, {"WattsStrogatzGraphDistribution"}, {"
    WeibullDistribution"}, {"WignerSemicircleDistribution"}, \
{"ZipfDistribution"}}
```

6.84 find number of points used in Plot command?

see `how-to-return-the-value-of-automatic-when-it-is-used-in-a-mathematica-function`

Thanks to Bob Hanlon for this method:

```
p1 = Plot[Sin[x], {x, 0, Pi}, PlotPoints -> Automatic]
Cases[p1, Line[pts_] :> Length[pts], Infinity]
  (*259*)
```

Another method due to Simon Woods

```
Trace[
 Plot[Sin[t], {t, 0, 2 Pi}],
 HoldPattern[PlotPoints | MaxRecursion -> _],
 TraceInternal -> True] // Flatten // Union

  {MaxRecursion -> 6, MaxRecursion -> Automatic, PlotPoints -> 50,
     PlotPoints -> Automatic}
```

6.85 How to delete all input cells?

link

make sure to copy the notebook first, just in case.

```
Module[{nb},
 nb = EvaluationNotebook[];
 NotebookFind[EvaluationNotebook[], "Input", All, CellStyle];
 NotebookDelete[nb]]
```

6.86 Use of ## &[]

reference

To understand what it does, these three do the same thing

```
Map[If[# == 1, Unevaluated@Sequence[], #] &, {1, 2, 3}];
If[# == 1, Unevaluated@Sequence[], #] & /@ {1, 2, 3};
If[# == 1, Sequence @@ {}, #] & /@ {1, 2, 3};
If[# == 1, ## &[], #] & /@ {1, 2, 3};
```

All above give `{2,3}`. So the effect of `## &[]` is to remove the entry completely (so we do not end up with a Null or empty slot in there).

6.87 How to use MapThread to map function on 2 lists?

Given lists `a={1,2,3}, b={4,5,6}` and we want to do operation from slot 1 from a with slot 1 from b, and so on. For this, we can use ‘MapThread‘. Suppose we want to add each corresponding slot, then

```
a = {1, 2, 3}
b = {4, 5, 6}
MapThread[(#1 + #2) &, {a, b}]

   (*  {5, 7, 9} *)
```

Of course, in this simple example, doing `a+b` will work, but this is just an example.

6.88 How to make shortcut to add [[and]] quickly?

see https://mathematica.stackexchange.com/questions/5212/automating-esc-esc-formatting

6.89 Installing Mathematica 10.1 on Linux

```
>sudo bash Mathematica_10.1.0_LINUX.sh
[sudo] password for me:
Mathematica 10.1.0 for LINUX Installer Archive

Verifying archive integrity.
Extracting installer. ............


Wolfram Mathematica 10.1 Installer

Copyright (c) 1988-2015 Wolfram Research, Inc. All rights reserved.

WARNING: Wolfram Mathematica is protected by copyright law
and international treaties. Unauthorized reproduction or
distribution may result in severe civil and criminalpenalties
and will be prosecuted to the maximum extent possible under law.

Enter the installation directory, or press ENTER to select
  /usr/local/Wolfram/Mathematica/10.1:

Now installing...

[******************

Type the directory path in which the Wolfram Mathematica script(s)
will be created, or press ENTER to select /usr/local/bin:



>



Installation complete.
```

```
>which Mathematica
/usr/local/bin/Mathematica
>export DISPLAY=:0
>Mathematica
```

6.90 get step-by-step solution from Alpha

One method is to just type `WolframAlpha["command here"]` and then click on the `show step by step` on top right corner of the result that displays on the notebook, assuming Wolfram Alpha gives an answer.

If the above does not work, try

```
WolframAlpha["Integrate[x Sin[x],{x,0,Pi}]", {{"Input", 2},
  "Content"}, PodStates -> {"Input__Step-by-step solution"}]
```

Another example

```
WolframAlpha["solve y''=-y",
              IncludePods -> "DifferentialEquationSolution",
              AppearanceElements -> {"Pods"},
              TimeConstraint -> {20, Automatic, Automatic, Automatic},
              PodStates -> {"DifferentialEquationSolution__Step-by-step solution"}]
```

or for text output

```
 WolframAlpha["solve y''=-y", {{"DifferentialEquationSolution", 2}, "Plaintext"},
        PodStates -> {"DifferentialEquationSolution__Step-by-step solution"}]
```

Another example

```
WolframAlpha["q*c'[x]==k*c[x], c[0]==c0",  PodStates -> {"Step-by-step solution"}]
```

6.91 How to nest Map inside Map?

see http://mathematica.stackexchange.com/questions/15480/how-do-i-designate-arguments-in-a-nested-map

From about url by Halirutan:

```
Map[Function[p2, Map[Function[p1, f[p1, p2]], list1]], list2]
```

6.92 TeXForm handling of derivative higher than two

How to make TeXForm handle higher derivatives better.

See http://mathematica.stackexchange.com/questions/134936/texform-handling-of-derivative-higher-than-two

From above URL by Carl Woll

```
Derivative /: MakeBoxes[Derivative[n_Integer?Positive][h_],TraditionalForm]
     := SuperscriptBox[MakeBoxes[h,TraditionalForm],
    StringJoin@ConstantArray["\[Prime]",n]
]

TeXForm[y'''''[t] + 2 x'''[t] - y'[t] == x[t]]
```

6.93 convert code from stackexchange to 2D math

sometime code is posed at Mathematica stacexchange which is hard to read. To convert it to clear 2D math code, copy into a mathematica cell (in an open notebook) on the computer, then do

CTRL-SHIFT-N

6.94 How to replace subexpression with condition?

Given

$$a\sqrt{z}+\sin\left(c\sqrt{r+4}+20\right)+\frac{e^{-c}\sqrt{9e^{2c}x^2+8}-x+99}{4x}+4\sqrt{h}e^{y}-99$$

Replace only the subexpression of the form `anything*Exp[anyting]*Sqrt[anything]` by the square of the pattern found if any.

```
expr /. patternName : (Exp[any3_]*any_.)*Power[Any_, Rational[1 | -1, 2]] -> patternName^2
```

This gives

$$a\sqrt{z}+\sin\left(c\sqrt{r+4}+20\right)+\frac{e^{-2c}(9e^{2c}x^2+8)-x+99}{4x}+16he^{2y}-99$$

The code `patternName :` above gives a name for the pattern found, this way we can use the name to easily do any transformation needed. In this example, it was just squaring it. But it can be anything else we want.

6.95 How to set up a function with optional arguments?

Optional arguments are very important and useful. It is easy to make these in Mathematica. lets say we want to make function `foo` that takes person name and age but the sex is provided as optional with default value as say `male`. To do this, two things are needed to be defined.

The function `foo` itself, but with additional argument called `opt : OptionsPattern[{foo}]` and we need to define an `Options[foo] =...` which sets up the default values. Here is an example

```
Clear["Global`*"];

(*this sets up the DEFAULT values for each optional argument *)
Options[processPerson] = {"sex" -> "male"};

(*this is the definition of the function itself*)
processPerson[name_String, age_?NumericQ, opt : OptionsPattern[{processPerson}]] /;  age > 0 :=

  (* the following line reads the optional value sex->"" if provided *)
  (* else Mathematica will automatically return the default value set up*)

  sexOfPerson = OptionValue["sex"];
  Print["Name = ", name];
  Print["age = ", age];
  Print["sex = ", sexOfPerson]
  ]
```

Now we are ready to call the function.

```
processPerson["me",10]
```

Gives

```
Name = me
age = 10
sex = male
```

We see in the above, that variable **sex** was automatically set to the default value thanks to the definitions `Options[foo]=...`

Now we call it with explicitly passing in the optional argument

```
processPerson["me",10,"sex"->"female"]
```

Gives

```
Name = me
age = me
sex = female
```

In the above, the option argument `sex->"..."` must always be the last one passed in. You can not do this for example

```
processPerson["joe doe","sex"->"female",10]
```

Here is another example

```
Clear["Global`*"];
Options[dsolve] = {"ic" -> {}, "hint" -> "None"};
dsolve[ode_, y_[x_], x_, OptionsPattern[]] :=  Module[{ic = OptionValue["ic"], hint = OptionVal
   Print["ic=", ic, " hint=", hint]
   ];

dsolve[y''[x] + y[x] == 1, y[x], x, "ic" -> {x[0] == 1}]
dsolve[y''[x] + y[x] == 1, y[x], x]
```

```
dsolve[y''[x] + y[x] == 1, y[x], x, "hint" -> "linear",  "ic" -> {x[10] == 0}]
```

Which prints

```
ic={x[0]==1} hint="None"
ic={} hint="None"
ic={x[10]==0} hint="linear"
```

To find all options supported by function do as an example `Options[LinearSolve]` and that will print `{Method -> Automatic, Modulus -> 0, ZeroTest -> Automatic}`

It is better to use string for the name of the hint as in the above, so not to clash with the same symbol being already defined.

6.96 checking optional values passed to function

The above showed how to use optional value. This below shows how to check and what to do if an optional value was not what is expected. This is done by explicitly checking, after reading the optional value, that its value is one that is expected. Like this

```
Clear["Global`*"];
Options[processPerson]={"sex"->"male"};

processPerson::msg="`1`";

processPerson[name_String,age_?NumericQ,opt:OptionsPattern[{processPerson}]]/;age>0:=Module[{s
    sex = OptionValue["sex"];

    If[Not[MemberQ[{"male","female"},sex]],
       Message[processPerson::msg,"Invalid value for sex found. "<>sex<>" but expected one of 
       Abort[]
    ];

    Print["Name = ",name];
    Print["age = ",age];
    Print["sex = ",sexOfPerson]
]
```

Now we are ready to call the function. Lets call it with bad value for the optional parameter

```
processPerson["me",10,"sex"->"car"]

    processPerson::msg: Invalid value for sex found. car but expected one of {"male", "female"
```

```
   $Aborted
```

6.97 post Mathematics code with Greek letters

Copy the Mathematica code from the notebook, paste it into `http://steampiano.net/msc/` and click convert. Then copy the output and paste that into the post at stackexchange.

6.98 How to find all variables in expressions?

```
Clear["Global`*"];
expr = y[z] + x + Pi + m/Sin[a] + b[0];
DeleteDuplicates@Cases[expr, any_ /; Head[any] === Symbol && Not[MemberQ[Attributes[any], Const
```

Gives `{x,n,a,x}`. Note that the above does not detected indexed variables, such as $b[0]$. The `Attributes` check is so not to include π since this is also a symbol. But we do not want to be there.

6.99 How to install Workbench on Linux Arch based?

I am now using CauchyOS linux, which is Arch based. These are the steps I did to insall and use Wolfram Workbench on linux.

First needed to install Eclispe. I did this using yay, like this

```
>yay -S eclipse
:: There are 7 providers available for eclipse:
:: Repository AUR
    1) eclipse-cpp-bin 2) eclipse-dsl-bin 3) eclipse-java-bin 4) eclipse-jee-bin
   5) eclipse-php-bin 6) eclipse-platform 7) eclipse-rcp-bin

Enter a number (default=1):
==> 3
AUR Explicit (1): eclipse-java-bin-2:4.36-1
Sync Dependency (1): webkit2gtk-2.48.5-1
:: (1/1) Downloaded PKGBUILD: eclipse-java-bin
  1 eclipse-java-bin                 (Build Files Exist)
==> Packages to cleanBuild?
==> [N]one [A]ll [Ab]ort [I]nstalled [No]tInstalled or (1 2 3, 1-3, ^4)
==>
  1 eclipse-java-bin                 (Build Files Exist)
==> Diffs to show?
==> [N]one [A]ll [Ab]ort [I]nstalled [No]tInstalled or (1 2 3, 1-3, ^4)
==>
==> Making package: eclipse-java-bin 2:4.36-1 (Fri 15 Aug 2025 09:36:05 PM CDT)
==> Retrieving sources...
  -> Found eclipse.desktop
  -> Downloading eclipse-java-2025-06-R-linux-gtk-x86_64.tar.gz...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  350M  100  350M    0     0  21.3M      0  0:00:16  0:00:16 --:--:-- 17.3M
==> WARNING: Skipping verification of source file PGP signatures.
==> Validating source files with sha512sums...
    eclipse.desktop ... Passed
==> Validating source_x86_64 files with sha512sums...
    eclipse-java-2025-06-R-linux-gtk-x86_64.tar.gz ... Passed
:: (1/1) Parsing SRCINFO: eclipse-java-bin
resolving dependencies...
looking for conflicting packages...

Package (1)       New Version  Net Change  Download Size

extra/webkit2gtk  2.48.5-1     126.19 MiB      33.50 MiB

Total Download Size:    33.50 MiB
Total Installed Size:  126.19 MiB

:: Proceed with installation? [Y/n]
:: Retrieving packages...
 webkit2gtk-2.48.5-1-x86_64                                              33.5
   MiB  34.3 MiB/s 00:01
   [------------------------------------------------------------] 100%
(1/1) checking keys in keyring

   [------------------------------------------------------------] 100%
(1/1) checking package integrity

   [------------------------------------------------------------] 100%
(1/1) loading package files

   [------------------------------------------------------------] 100%
(1/1) checking for file conflicts

   [------------------------------------------------------------] 100%
:: Processing package changes...
```

```
(1/1) installing webkit2gtk

    [--------------------------------------------------------------] 100%
Optional dependencies for webkit2gtk
    geoclue: Geolocation support [installed]
    gst-libav: nonfree media decoding [installed]
    gst-plugins-bad: media decoding [installed]
    gst-plugins-good: media decoding [installed]
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
==> Making package: eclipse-java-bin 2:4.36-1 (Fri 15 Aug 2025 09:37:07 PM CDT)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> Retrieving sources...
  -> Found eclipse.desktop
  -> Found eclipse-java-2025-06-R-linux-gtk-x86_64.tar.gz
==> Validating source files with sha512sums...
    eclipse.desktop ... Passed
==> Validating source_x86_64 files with sha512sums...
    eclipse-java-2025-06-R-linux-gtk-x86_64.tar.gz ... Passed
==> Removing existing $srcdir/ directory...
==> Extracting sources...
  -> Extracting eclipse-java-2025-06-R-linux-gtk-x86_64.tar.gz with bsdtar
==> Sources are ready.
==> Making package: eclipse-java-bin 2:4.36-1 (Fri 15 Aug 2025 09:37:08 PM CDT)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> WARNING: Using existing $srcdir/ tree
==> Entering fakeroot environment...
==> Starting package()...
==> Tidying install...
  -> Removing libtool files...
  -> Purging unwanted files...
  -> Removing static library files...
  -> Compressing man and info pages...
==> Checking for packaging issues...
==> Creating package "eclipse-java-bin"...
  -> Generating .PKGINFO file...
  -> Generating .BUILDINFO file...
  -> Generating .MTREE file...
  -> Compressing package...
==> Leaving fakeroot environment.
==> Finished making: eclipse-java-bin 2:4.36-1 (Fri 15 Aug 2025 09:37:10 PM CDT)
==> Cleaning up...
```

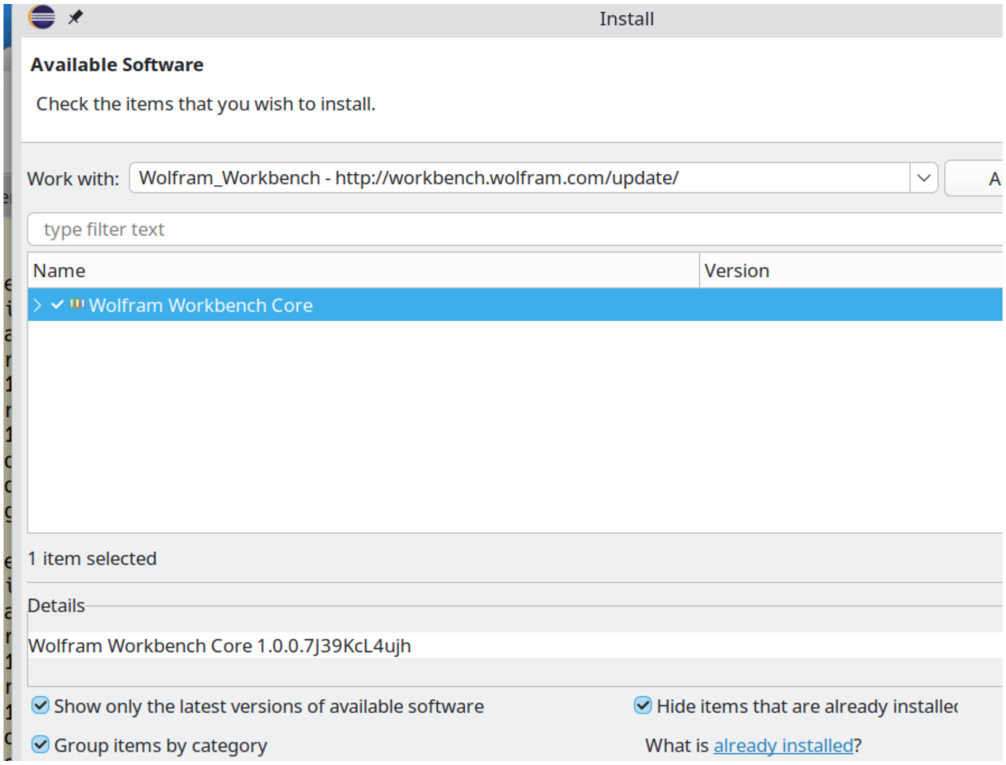
```
loading packages...
resolving dependencies...
looking for conflicting packages...

Package (1)       New Version  Net Change

eclipse-java-bin  2:4.36-1     428.73 MiB

Total Installed Size:  428.73 MiB

:: Proceed with installation? [Y/n]
(1/1) checking keys in keyring

    [------------------------------------------------------------] 100%
(1/1) checking package integrity

    [------------------------------------------------------------] 100%
(1/1) loading package files

    [------------------------------------------------------------] 100%
(1/1) checking for file conflicts

    [------------------------------------------------------------] 100%
:: Processing package changes...
(1/1) installing eclipse-java-bin

    [------------------------------------------------------------] 100%
:: Running post-transaction hooks...
(1/3) Arming ConditionNeedsUpdate...
(2/3) Updating icon theme caches...
(3/3) Updating the desktop file MIME type cache...
>which eclipse
/usr/bin/eclipse
>
```

Starting elipse it says the version is (clicking on About)

```
Eclipse IDE for Java Developers (includes Incubating components)

Version: 2025-06 (4.36.0)
Build id: 20250605-1316
```

Now we start the second phase. Which is to install the actual Wolfram workbench inside eclipse. For this click on `Help->Install new software` and follow instructions given on this web page https://support.wolfram.com/27221

These two images below I took during this process.



Available Software

Check the items that you wish to install.

Work with: Wolfram_Workbench - http://workbench.wolfram.com/update/

☑ Wolfram Workbench Core

1 item selected

Details

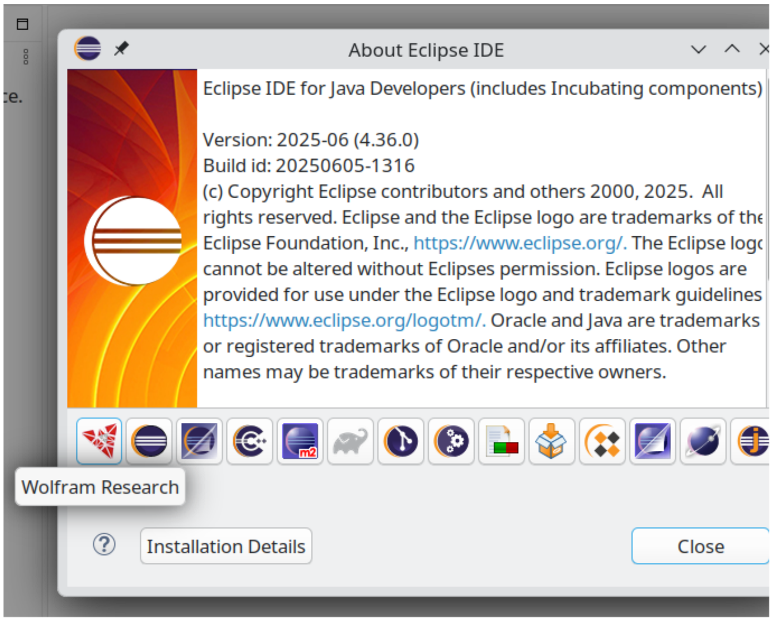
Wolfram Workbench Core 1.0.0.7J39KcL4ujh

☑ Show only the latest versions of available software

☑ Hide items that are already installec

☑ Group items by category

What is already installed?



Eclipse IDE for Java Developers (includes Incubating components)

Version: 2025-06 (4.36.0)
Build id: 20250605-1316
(c) Copyright Eclipse contributors and others 2000, 2025. All rights reserved. Eclipse and the Eclipse logo are trademarks of the Eclipse Foundation, Inc., https://www.eclipse.org/. The Eclipse logc cannot be altered without Eclipses permission. Eclipse logos are provided for use under the Eclipse logo and trademark guidelines https://www.eclipse.org/logotm/. Oracle and Java are trademarks or registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Make sure to select the Wolfram workbench by clicking on the little checkbox at the very end of the above process, when it asks to verify it and accept. If you do not, you will not see the OK button to finish.

Now Mathematica workbench is installed. Restart eclipse and start using it.

6.100 How to check call againt wrong number of arguments of wrong type?

In Mathematica, when calling function `foo[x]` and the function is defined to take two arguments (say), then Mathematica will not issue an error and will use `f[x]` unevaluated. This can cause hard to detect errors as the call is meant to be `f[x,y]`.

To check for this, add an error message, like this

```
foo[x_Integer,y_Integer]:=x*y;
foo::argerr="Wrong call in ``. Arguments are either wrong type or wrong number of arguments us
err:foo[___]:=(Message[foo::argerr,HoldForm@err]; $Failed);
```

Now if we do

```
foo[1,2]
```

Then it works and returns 2. But if we make mistake and type

```
foo[1]


    foo::argerr: Wrong call in foo[1]. Arguments are either wrong type or wrong number of argu
```

This makes the code more robust for wrong call made. This has to be done for each function defined if we want to detect such bad calls.