RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Mathematica - a First Glance

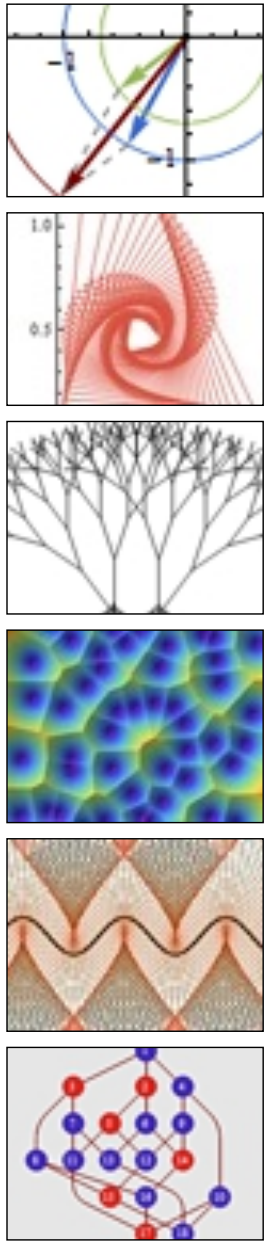
oder

'Nie wieder verrechnen!'

Peter Fischer, ZITI

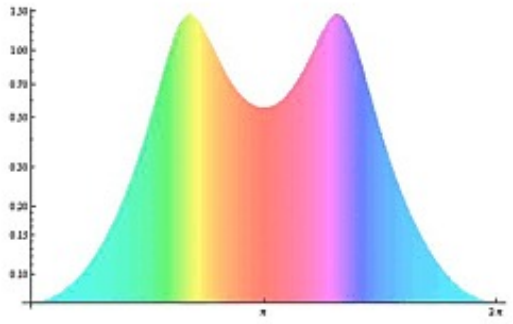
What is Mathematica ?

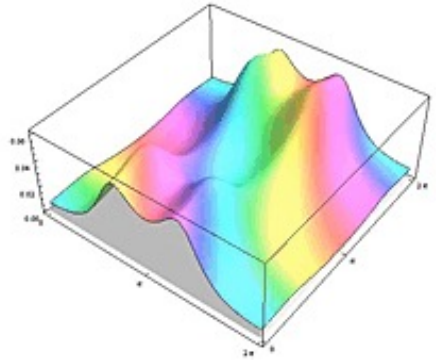
- Initially: **algebraic** manipulation of formulae
  - Integration
  - Differentiation
  - Many special functions
- Later:
  - Numerical methods
  - Fancy graphics
  - Sound, Image processing, lots of specialties…
- Recently:
  - ‘Demonstrations’ on the Web: http://demonstrations.wolfram.com/
  - Separate ‘CDF player’ to locally run demos (2.7GB!)
  - All notebooks can be downloaded, viewed, changed



What can WE use it for

- Function Plotting
  - For publications
  - For our understanding





- Equation solving, optimizations
- Verification of manual results
- Data visualization and analysis
- Educational display of relations / effects / …
- For fun!

Mathematica on Raspberry Pi

- There is a ***free*** license included in the 'Raspbian' OS for Pis

Mathematica Online

- https://mathematica.wolframcloud.com
- Free, but need registration

Using Mathematica in Heidelberg

- Mathematica is installed in CIP Pool of Physics Faculty. You can ssh there with your Uni ID:
- **`ssh -X <unid>@physik5.kip.uni-heidelberg.de`**
  (server can be physik1 - physik5)
- There, call
  `$ mathematica-13.3 &`

- You can also use a Web browser:
  https://physik1.kip.uni-heidelberg.de/guacamole/

- Best: Install Mathematica on your own computer. Heidelberg students get a 1 year licence via https://www.urz.uni-heidelberg.de/de/service-katalog/software-und-anwendungen/mathematica

Similar Tools?

- **Maple:** commercial, www.maplesoft.com
- **Magma**: commercial, magma.maths.usyd.edu.au/magma
  Mainly Computer Algebra, no graphics.

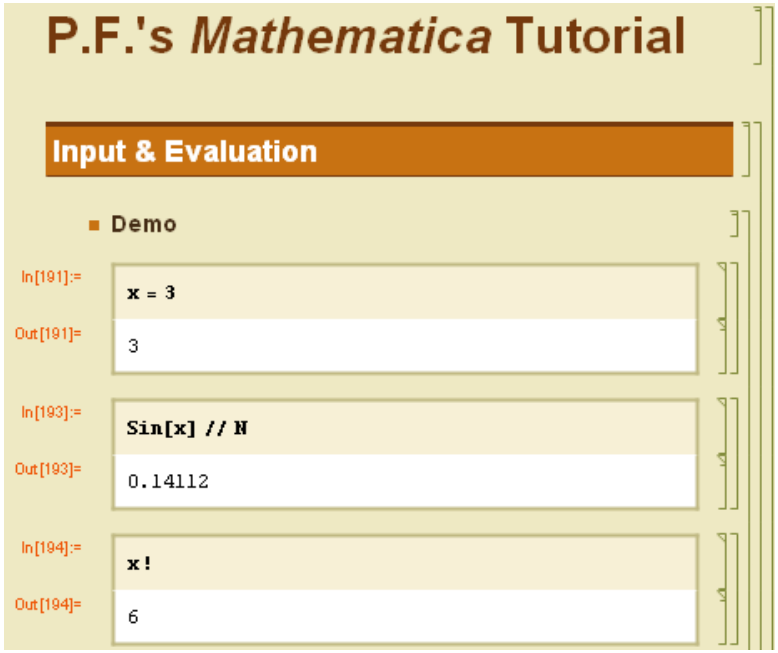
- **MATLAB:…** more for matrices, i.e. linear systems
- **Octave**: open source alternative to MATLAB

- **sage**: open source (pronounce ‘seetch’) www.sagemath.org
  Offers Python support.
  See: http://doc.sagemath.org/html/de/a_tour_of_sage/

- Jupyter Notebooks (?)

BASICS

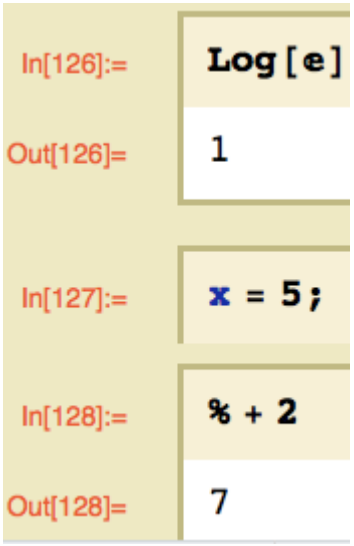
Notebooks



- User interface: *Notebooks*
- Each group has a specific *format*
  - Document title, subtitle,…
  - Sections, subsection,…
  - Input **- This must be used for all input !**
- Hierarchical Input: Blocks can be made invisible by clicking on rulers on right side
- Colors / Styles use templates
  - *Format → StyleSheet → …*
- Get format selection box in Window from
  - *Window → Show Toolbar*
- Before saving a notebook, the (memory consuming) outputs can be cleared with
  - *Cell → Delete All Output*
- File extension: *.nb

Entering Stuff

- An input is ‘executed’ by ending with **`SHIFT – Return`**
  - Just entering CR starts a new line but does *not* execute it
- The *input* line is evaluated and an *output* is produced
  - To disable output (for definitions…), end the input line with ‘`;`’
- Each input/output line gets a line number
  - The last input line can be referred to as ‘`%`’
  - The line before with ‘`%%`’ etc.
  - Any line can be referred to by its number with `%n` (no blank)
- **Parenthesis**
  - `(…)` for normal mathematical priorities
  - `[…]` for *function* arguments
  - `{…}` are *lists* (see later), used also for vectors, matrices,…
  - `[[ n ]]` are *indices* in lists (see later)
  - `(* … *)` are *comments*



Very useful later!

Getting Help

- Select a key word, then press F1

- In the Help Window
  - Use key words
  - Use ‘see also’ section at the bottom
  - Very nice feature:
    all examples can be executed and modified

- More help on wolfram pages, for instance
  - http://www.wolfram.com/support/learn/

Formatting Input

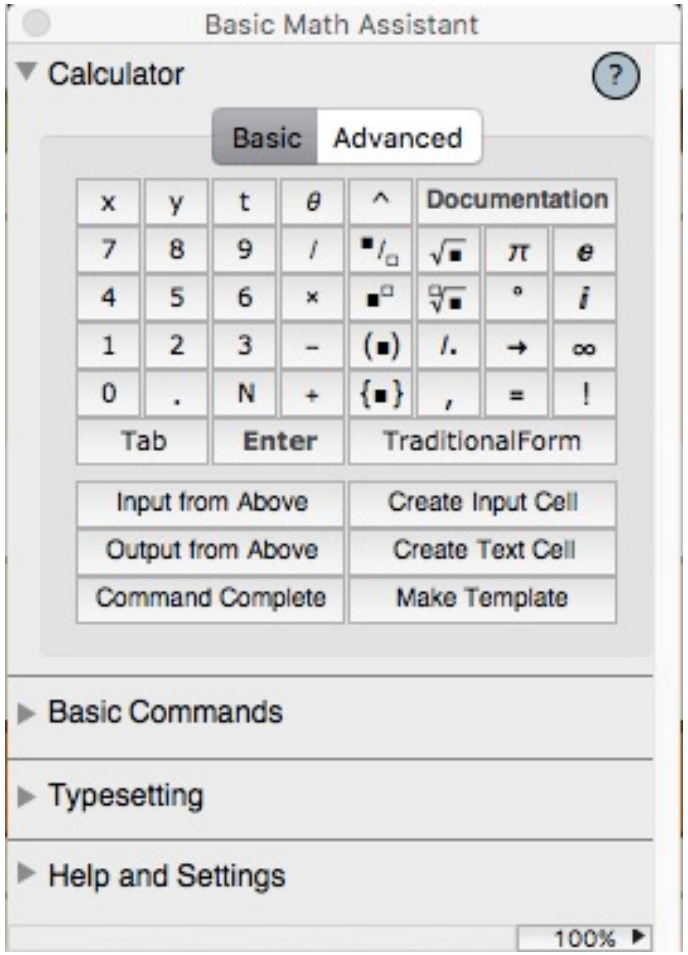
- The GUI *Palettes* → *Basic Math Assistant* helps a lot!
- Special characters are obtained with ESCAPE Commands:
  - **`ESC a ESC`** → α
  - **`ESC p ESC`** → π
  - **`ESC ii ESC`** → i (imaginary unit, also **`I`**)
  - **`ESC ee ESC`** → e (base of natural logarithm, also **`E`**)
  - **`ESC inf ESC`** → ∞ (infinity)
  - **`ESC elem ESC`** → ∈ (element of a set)
- Special formatting (exponent, fraction,..) is obtained with
  - **`CTRL ^ or CTRL 6`** → Exponent (PC, MAC)
  - **`CTRL _`** → lower index
  - **`CTRL SHIFT /`** → Fraction
  - **`CTRL 2`** → Square Root

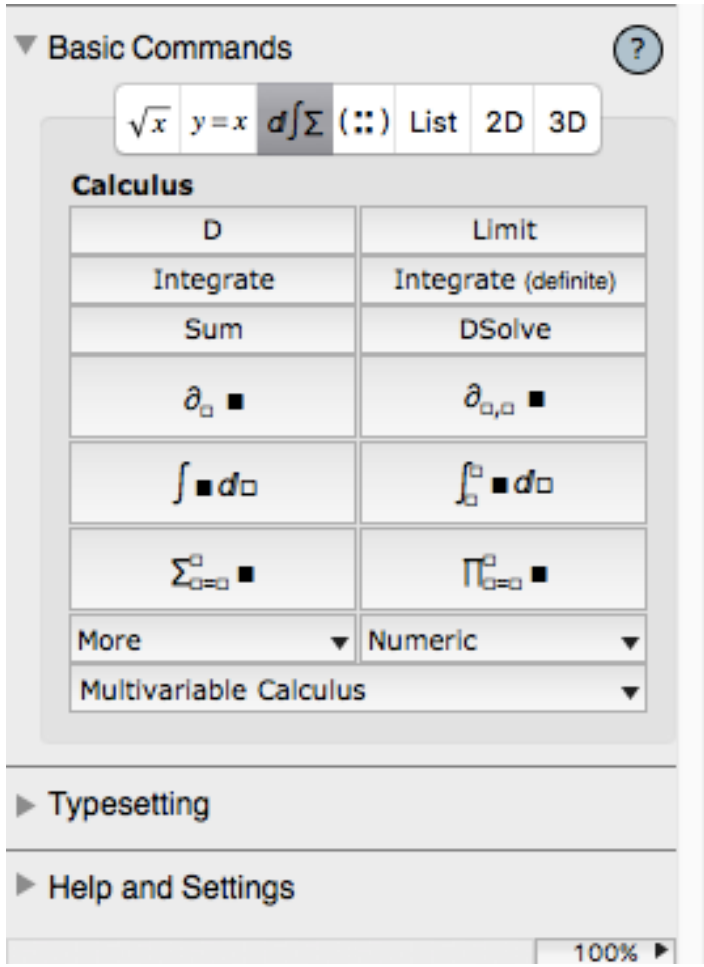
Note: The special character '2' on some keyboards does NOT work for squaring!

- Note: **All** Commands and Functions start with Uppercase (**`Sin[], Log[], Simplify[],…`**)

The Math Assistant

- All formatting, symbols, expressions etc. are collected in the Math Assistant
  - Palettes → Basic Math Assistant





Aborting Calculations

- When calculations take too long:
  - **`ALT .`** (or ⌘`.` on MAC)
  - or *Evaluation → Abort Evaluation*
- This sometimes does not work
  - Try several times
  - Kill the process in the operating system
- Remark:
  - Mathematica is divided in a graphical 'Front End' and a 'Kernel' which does the calculations.
  - The kernel can use several cores or run on a remote machine!

Numbers

```
In[199]:= 3 * 10 / 2
Out[199]= 15

In[207]:= 1/3 + 1/4
Out[207]= 7/12

In[208]:= % + 5/12
Out[208]= 1

In[212]:= 20!
Out[212]= 2 432 902 008 176 640 000

In[209]:= Sin[1]
Out[209]= Sin[1]

In[210]:= Sin[π/4]
Out[210]= 1/√2
```

- Normal numbers are EXACT, with *arbitrary* precision
- Fractions are kept, but simplified
- Functions are NOT evaluated numerically by default

- To get a *numerical* value, use
  `N[value]` or **`N[value,precision]`**

```
In[50]:= N[π]
Out[50]= 3.14159

In[51]:= N[Pi, 100]
Out[51]= 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706
```

Postfix / Prefix Notation

- Functions can be used in several ways:

- Normal Notation:
  - `Sin[π]`
    Some functions take multiple arguments:
  - `N[π,100]`

- **Postfix** syntax using `//` operator:
  - `π //Sin`
    works only with ONE parameter (… see later how to improve this ...)

- **Prefix** notation with `@`:
  - `Sin@π`

Note: You may not use these at the start. I mention this here because you find it in many examples and help on the web!

- The latter two are useful to nest functions:
  - `N@Cos@π` or `π//Cos//N`

Complex Numbers

```
a = 3 + 4 i; b = 1 + i;
```

```
a + b
```

$4 + 5\,i$

```
a b
```

$-1 + 7\,i$

```
a / b
```

$\frac{7}{2} + \frac{i}{2}$

```
Re[a]
```

3

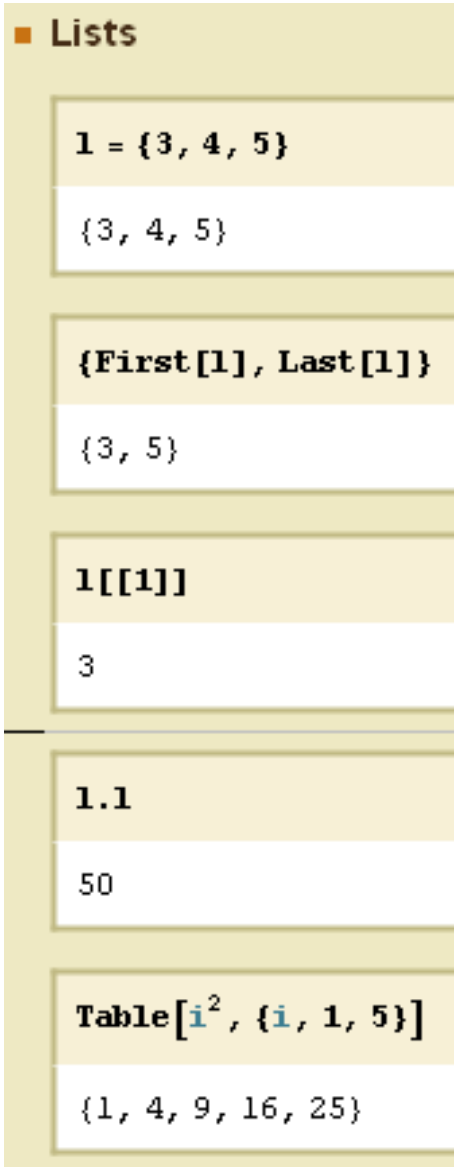
- Imaginary unit is `I` or i (entered as `ESC ii ESC`)

- Some functions with a complex number x are:
  - `Re[x]` or `x//Re` real part — `Re[3+I]` → `3`
  - `Im[x]` imaginary part — `Im[3+I]` → `1`
  - `Abs[x]` absolute value — `Abs[3+I]` → $\sqrt{10}$
  - `Arg[x]` angle — `Arg[3+I]` → `ArcTan[1/3]`
  - `Conjugate[x]` — `3+I//Conjugate` → `3-I`
  - `ComplexExpand[x]` simplifies assuming real variables
- Functions work with complex numbers:
  - `Sqrt[3+4I]` → `2+I`
  - `Exp[x+I y]//ComplexExpand`
    → E^x `Cos[y] + I` E^x `Sin[y]`

Exercise 1: Numbers and Expressions

- Calculate the square root of 2 with 100 digits precision
- What is Sqrt [-4] ?
- Add, subtract and multiply the fractions F1=4/5; F2=5/6;
- Can 20! be divided by 45 or by 46 ?
  - What is the general approach to this type of question?
  - Play with **`FactorInteger[…]`**. Can you guess what the output means? Try with some simple cases
- Stirling's formula $\mathrm{Sqrt}[2\,\pi\,k]\,(k/e)^k$ is an approximation of the **`Factorial[k]`** function k! (= 1×2×3×…×k).
  Compare for k=5,10,50
  How large is the relative error (in Percent) for k=10,50,100 ?

Lists



- `L = {a,b,c,…}` is linear list of (arbitrary!) elements
- Picking Elements:
  - The first element is `First[L]` or `L // First`
  - The last Element is `Last[L]`
  - The N-th Element is `L[[n]]`
    Note two parenthesis! Indices start with 1!
  - Append an element with `K = Append[L,elem]`
- Generating a list (**important**!):
  `Table[expression,{index,start,stop}]`
- Operations on lists:
  - Operators / Functions are applied to each element:
  - `1 + {1,2}` → `{2,3}`
  - `{1,2} + {1,2}` → `{2,4}`
  - `Sin[{1,2}]` → `{Sin[1],Sin[2]}` 'threads over lists'

Vectors = Lists, Matrices = Lists of Lists

- **`L={a,b,c,…}`** is a linear list of (arbitrary!) elements

- Lists can be interpreted as vectors
  - Display with **`MatrixForm[L]`** or **`L // MatrixForm`**
  - Scalar multiply with '`.`' operator

- Nested lists are matrices
  - '`.`' operator is Multiplication
  - Watch out:
    - if A,B are matrices
    - A.B is 'real' product
    - AB is element-wise product (see left!)
  - Watch out:
    - MatrixForm[…] is NO vector any more, but a 'graphics'. It cannot be used for further calculations.

```
A // MatrixForm
```

rixForm=

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
B // MatrixForm
```

rixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
A.B // MatrixForm
```

rixForm=

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
A B // MatrixForm
```

rixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

```
M = {{1, 2}, {3, 4}}; MatrixForm[M]
```

atrixForm=

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
M.{1, 2}
```

```
{5, 11}
```

Expressions

```
A = 3 x + 2; B = 5 y;
```

```
A B
```

```
5 (2 + 3 x) y
```

```
% / (3 x)
```

$$\frac{5\,(2+3\,x)\,y}{3\,x}$$

```
A
```

```
2 + 3 x
```

```
Clear[A]
```

```
A
```

```
A
```

- Mathematica works with ***symbolic*** expressions
- They can be assigned to ‘variables’
- Variables do **not** need to be declared

- Assignment can be cleared with `Clear[x]`
- *Common Error*: ‘old stuff’ stored in an expression
- -> Clear everything with `ClearAll["Global`*"]`
  - Note accent 'grave': ` (not ´or ')

- Get information about an expression with `?x`

- Note that the color of an expression changes when it is defined

```
In[40]:= Clear[x, y]

In[41]:= y = 3
Out[41]= 3

In[42]:= {x, y}
Out[42]= {x, 3}
```

Important and Tricky: Direct / Delayed Assignment

- `x=y` **evaluates** y and assigns the result
- `x:=y` keeps y as an **expression**. Y is evaluated newly whenever x is used (‚delayed assignment‘)

```
In[28]:= Clear[x, y]

In[29]:= y = 3
Out[29]= 3

In[30]:= x = y
Out[30]= 3

In[31]:= x
Out[31]= 3

In[32]:= y = 4
Out[32]= 4

In[33]:= x
Out[33]= 3
```

Now x is 3!

x is still 3!

```
In[34]:= Clear[x, y]

In[35]:= y = 3
Out[35]= 3

In[36]:= x := y

In[37]:= x
Out[37]= 3

In[38]:= y = 4
Out[38]= 4

In[39]:= x
Out[39]= 4
```

x is y.
y is 3 (here),
therefore x is 3

x is y.
y is now 4

Exercise 2A: Lists, Expressions

- Generate a list of the first 10 squares (n^2) (Hint: Use **`Table`**)
  - Pick the first, third and last element
- Define and multiply two 3-D vectors
- Define a 2×2 Matrix A. Use **`MatrixForm[…]`** to show it.
  Let **`B=Inverse[A]`** **`(or A//Inverse)`**. Multiply A and B.
- Define two expressions: A = 3 x + 5; B = 5 x^2 – 7
  - Define F = A B
  - Play with **`Expand[…], Factor[…]`**. Use Help to learn more.
  - Divide the **`expand`**ed F by A. Try to **`Simplify[…]`** the result

Exercise 2B:

- Create a 10 x 10 matrix A filled with random integer values between 0 and 10 (use RandomInteger[10])
- Calculate its inverse
- Check that the product is the Unit Matrix

Exercise 2C: COOL

- Calculate the numeric value of $e^{\pi \sqrt{50}}$
  We expect an ‘arbitrary’ real number…
- Get the digits after the comma with `FractionalPart[]//N`
- Generate a list of this for $e^{\pi \sqrt{k}}$ for k = 50 … 60
- Do you see the many ‚9‘ ? Can this be by chance?
- Try k = 160...170
- Some hints why this happens on this external link:
  https://math.stackexchange.com/questions/4544/why-is-e-pi-sqrt163-almost-an-integer

Using Functions - Recap

- Functions & Commands have **UpperCase** names
- Arguments are given in square brackets `[]`
  - `Sin[3 x + y]`
  - `FactorInteger[123456789]`

- Functions with **1 argument** can use **postfix** notation:
  - `arg // function` `(* = function[arg] *)`
  - `111 // FactorInteger` → `{{3, 1}, {37, 1}}`
  - `1 // Log` → `0`

  or with **prefix** notation:
  - `function @ arg`
  - `Log@7`
- There is also an **infix** possibility for functions with 2 arguments
  - `arg1 ~ function ~ arg2`
  - `{1,2}~Join~{3,4}` joins two lists -> {1,2,3,4}
  - Or: `π~N~10` -> `N[p,10]`

Defining Functions

- **Define** a function with '_' after the arguments:
  - `Average[x_,y_] := (x+y)/2;`
  - `Average[3,4]` → `7/2`
  - `DoSquare[x_] := x^2;`
  - `3 // DoSquare` → `9`
- **Important Detail** (as discussed before):
  - When defining with '=', all expressions are *immediately* evaluated
  - When defining with ':=', evaluation of expressions is *delayed* until function call. This is the normal case!

Important!
Common mistake!

```
x = 3;

f1[x_] = 3 + x;

f1[5]

6

f1[x_] := 3 + x;

f1[5]

8
```

(Argument Types and Defaults)

- Arguments can be constrained to a type:

```
In[64]:= g[x_Integer, y_] := x y

In[65]:= g[4, 5]
Out[65]= 20

In[66]:= g[4.5, 5]
Out[66]= g[4.5, 5]
```

- Multiple definitions are possible:

```
In[79]:= g[n_Integer] := n g[n - 1];
         g[1] = 1;

In[81]:= g[10]

Out[81]= 3 628 800
```

- Arguments can get a default value:

```
In[84]:= h[x_, k_Integer: 2] := x^k

In[85]:= h[π]

Out[85]= π^2

In[86]:= h[π, 1]

Out[86]= π
```

(Pure Functions) – for experts …

- Sometimes a function is only needed once, it has no 'name'
- Such a ***pure function*** is defined by (postfix) '**&**':

```
..expression with #.. &                    // # is the argument
```

for example

```
#+3 &              // take argument and add 3
```

- This function can be used once:

```
5 // #+3 &              → 8              (* postfix *)
#+3 & [7]               → 10             (* with [] *)
Pi // N[#,3] &          → 3.14           (* very useful! *)
```

- You will (maybe) see later that this can be ***very*** useful…

- Several arguments are labeled with **#1,#2,..**, e.g.

```
(#1^2 + #2^4) & [x, y]    → x^2 + y^4
```

Simplifying Expressions, Assumptions

```
Simplify[√(x^2), x > 0]
```

```
x
```

```
Sin[k π] // Simplify
```

```
Sin[k π]
```

```
Simplify[Sin[k π], k ∈ Integers]
```

```
0
```

```
Simplify[Cos[k π], k ∈ Integers]
```

```
(-1)^k
```

- **VERY** important function: '`Simplify[…]`'
  - `Simplify[Sin[x]^2+Cos[x]^2]` → `1`

  or

  `Sin[x]^2+Cos[x]^2 // Simplify` → `1`

- Often, we cannot simplify without further knowledge:
  - `Sqrt[x^2] // Simplify` → `Sqrt[x * x]`
- We can use **Assumptions** to help:
  - `Simplify[Sqrt[x^2],x>0]` → `x`

- Assumptions can be set globally:
  - `$Assumptions = {x>0, n∈Integers, …};`
  - `$Assumptions = True;  (* to clear *)`

More Powerful Simplification

- FullSimplify[…] makes a stronger effort

```
In[90]:= Sqrt[7 - 4 Sqrt[3]] // Simplify
Out[90]= Sqrt[7 - 4 Sqrt[3]]

In[91]:= Sqrt[7 - 4 Sqrt[3]] // FullSimplify
Out[91]= 2 - Sqrt[3]
```

Surprising ?
Do YOU understand why?

- Very similar: `Refine[expr, assumption]`

For fun: More strange formulae from Ramanujan

$$\sqrt[3]{\sqrt[3]{2}-1} == \sqrt[3]{\frac{1}{9}} - \sqrt[3]{\frac{2}{9}} + \sqrt[3]{\frac{4}{9}} \;//\; \text{FullSimplify}\; (*\text{Ramanujan}*)$$

True

$$\sqrt[3]{\sqrt[5]{\frac{32}{5}} - \sqrt[5]{\frac{27}{5}}} == \sqrt[5]{\frac{1}{25}} + \sqrt[5]{\frac{3}{25}} - \sqrt[5]{\frac{9}{25}} \;//\; \text{FullSimplify}$$

True

$$\sqrt[4]{\frac{3+2\sqrt[4]{5}}{3-2\sqrt[4]{5}}} == \frac{\sqrt[4]{5}+1}{\sqrt[4]{5}-1} \;//\; \text{FullSimplify}$$

True

$$\sqrt{\sqrt[3]{28} - \sqrt[3]{27}} == \frac{\sqrt[3]{98} - \sqrt[3]{28} - 1}{3} \;//\; \text{FullSimplify}$$

True

Manipulation Expressions

```
a = x + 5; b = 2 x + y;
```

```
a b
```

$$(5 + x)\ (2\,x + y)$$

```
a b // Expand
```

$$10\,x + 2\,x^2 + 5\,y + x\,y$$

```
1 + x + y + x y // Factor
```

$$(1 + x)\ (1 + y)$$

```
a / b
```

$$\frac{5 + x}{2\,x + y}$$

```
a / b // Expand
```

$$\frac{5}{2\,x + y} + \frac{x}{2\,x + y}$$

- Many Build-In functions to manipulate Expressions:
  - **Expand**
  - **Factor**
  - **TrigExpand**
  - **TrigFactor**
  - **TrigReduce**
  - **ComplexExpand**
  - …
- See help…

```
Sin[3 x] // TrigExpand
```

$$3\,\mathrm{Cos}[x]^2\,\mathrm{Sin}[x] - \mathrm{Sin}[x]^3$$

```
Sin[3 x] // TrigFactor
```

$$(1 + 2\,\mathrm{Cos}[2\,x])\,\mathrm{Sin}[x]$$

```
2 Sin[x] Cos[y] // TrigReduce
```

$$\mathrm{Sin}[x - y] + \mathrm{Sin}[x + y]$$

```
Exp[1 + I] // ComplexExpand
```

$$e\,\mathrm{Cos}[1] + i\,e\,\mathrm{Sin}[1]$$

Replacing Parts of an Expression with /.

Important!

```
Sin[x] /. x → 3 y
```

```
Sin[3 y]
```

```
EX = a x^2 + 13;
```

```
EX /. x → (1 + y)
```

```
13 + (1 + y)^2 (6 + y)
```

```
% // Expand
```

```
19 + 13 y + 8 y^2 + y^3
```

```
Log[t] /. t → {1, x, Sin[x]}
```

```
{0, Log[x], Log[Sin[x]]}
```

- A symbol in an expression can be replaced by
  `something /. expression -> newvalue`
  - The right hand side (**`expression -> newvalue`**) is a *transformation rule*
- **`xxx /. yyy`** means:
  - Apply the replacement rule yyy to expression xxx
  - yyy *must be* one (or a list of) transformation rules
- Examples
  - **`Sin[x] /. x -> 3 y`** → **`Sin[3y]`**
  - **`Log[x] /. x -> Sin[k]`** → **`Log[Sin[k]]`**
- Can sometimes be used in a more general way:
  - **`(1+x)^2 /.(1+x) -> y`** → **y^2**

(Map)

- A command related to `/.` is **`Map[f,expr]`** or **`f/@expr`**
- **`Map[f,expr]`** applies f to all elements of expr, i.e. **`Map[f,{a,b,c}]`** gives **`{f[a], f[b], f[c]}`**
- f can be 'anything'
- Try to understand: 5 # & / @ {1, 5, 7}
- For more experienced users, this is very handy. You find this approach very often in examples in the internet.

- Example:
- **`If[PrimeQ[#],Framed[#],#] &  /@  Range[10]`**

a pure function which puts a frame around a prime number	MAP it to	{1,2,..,10}

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Solving Equations

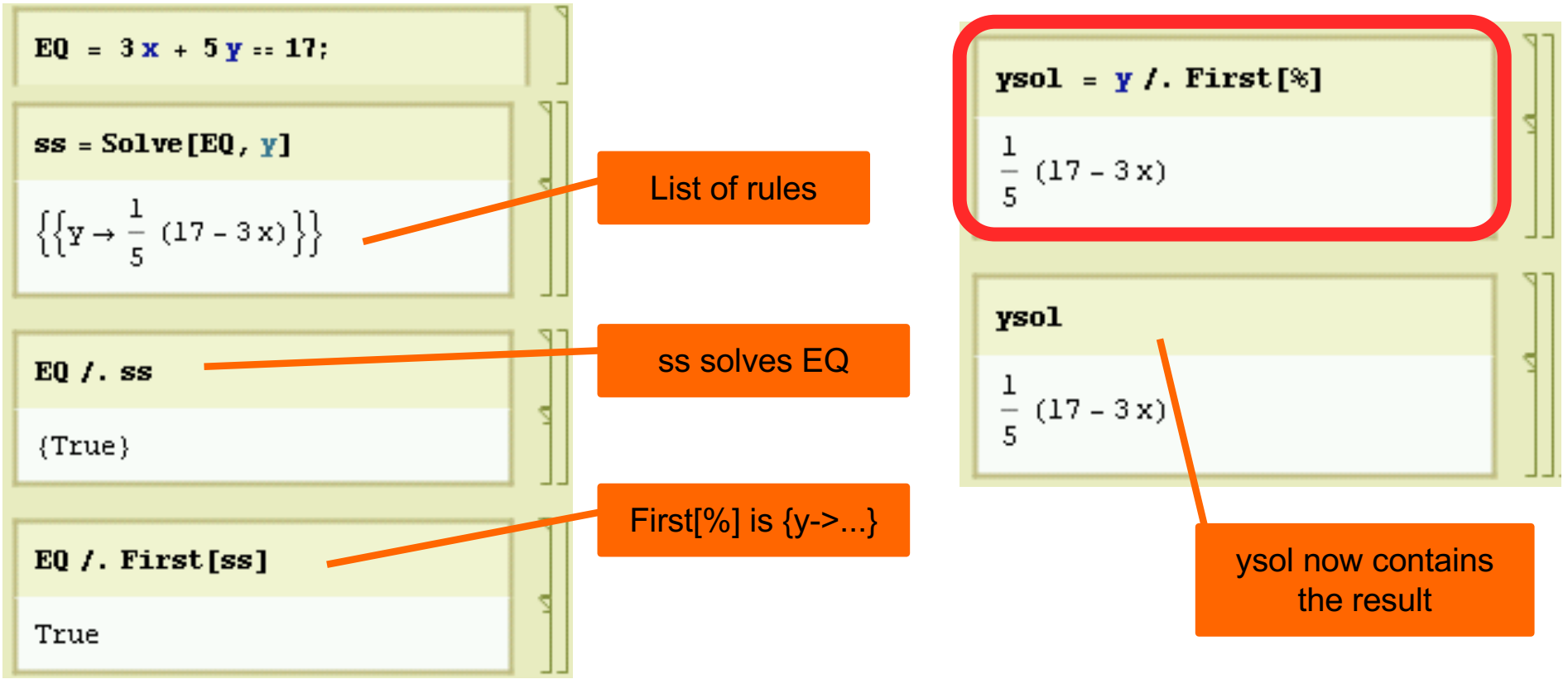
- **Equality** in Equations must use the '**==**' sign
- Solutions can be found with **Solve[equation, variable]**:
  - **Solve[2x==4,x]** → **{{x→2}}**
- **Solve** can find several solutions:
  - **EQ = 2 x^2 == 4;**
  - **Solve[EQ,x]** → **{{x → -√2},{x → √2}}**
  - **EQ /. %** → **{True,True}**
- Several equations can be solved simultaneously with '**&**':

```
Solve[ 2x+y==10 && x-y==2,{x,y}]
```

- **Reduce[…]** is similar to **Solve[…]**
  - **Solve [Sin[x]==0,x]** → $\{\{x \to 2\pi c_1 \text{ if } c_1 \in \mathbb{Z}\}, \{x \to \pi + 2\pi c_1 \text{ if } c_1 \in \mathbb{Z}\}\}$
  - **Reduce[Sin[x]==0,x]** → $c_1 \in \mathbb{Z}$ && $(x == 2\pi c_1 \,||\, x == \pi + 2\pi c_1)$
- Minima can be found with **Minimize[]** or **FindMinimum[]**

Solving Equations

- Note: Equations must have a '**==**' sign, '**>**',… or similar!!
- The result of **`Solve[]`** (not for **`Reduce[]`** ...) is always a *list* of replacement rules (even if there is only one result)
- To assign the result to an expression/variable, use **`/.`**:



Reduce

- Reduce is more general. It can also solve inequalities:

```
In[215]:= Solve[2 x > 1, x]
Out[215]= {{}}

In[216]:= Reduce[2 x > 1, x]
Out[216]= x > 1/2
```

- Often the domain must be given:

```
In[211]:= Reduce[Sin[x] > 1/2, x]
Out[211]= Reduce[Sin[x] > 1/2, x]

In[212]:= Reduce[Sin[x] > 1/2, x, Reals]
Out[212]= C[1] ∈ Integers && 1/6 (π + 12 π C[1]) < x < 1/6 (5 π + 12 π C[1])
```

Some More Details

- Reduce *can* just 'simplify' expressions (without argument!)
- `Reduce[3x+6y==7 && 7x-y==4]`

  → `y == 37/45 && x == 31/45`

- If the solution is not unique, the variables to eliminate must be specified:

- `Solve[3x+48y+5z/8==12 && x-8y-z/8==4, {x,z}]`

  → `{{x -> 4 - y, z -> -72 y}}`

- `Solve[3x+48y+5z/8==12 && x-8y-z/8==4, {y,z}]`

  → `{{y -> 4 - x, z -> 72 (-4 + x)}}`

(‘Issues’ with Solve)

- I encountered this subtle ‘problem’:

```
In[55]:= $Assumptions = True;
In[56]:= Solve[R S == 100, S]
Out[56]= {{S → 100/R}}
In[57]:= $Assumptions = S > 0;
In[58]:= Solve[R S == 100, S]
Out[58]= {}
In[59]:= $Assumptions = S > 0 && R > 0;
In[60]:= Solve[R S == 100, S]
Out[60]= {{S → 100/R}}
```

Tabula Rasa

Obviously, this trivial equation can be solved

I often constrain variables to reasonable ranges to allow for instance $\sqrt{S^2}$ -> S

Now, no solution is found! This is because R COULD be negative and then S would be negative, contradicting the assumption…

If we make sure R>0, then the solution is found again…

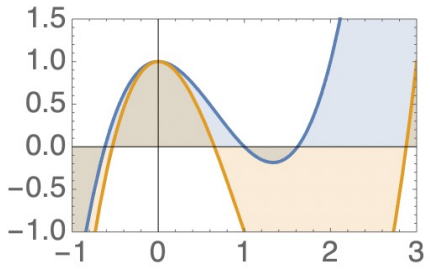
(Roots)

- For polynomial eqns, solution is often expressed as a ‘root’
- This can be expanded with **`ToRadicals[]`**

```
In[39]:= f1[x_] = 1 - 2 x^2 + x^3; f2[x_] = 1 - 3 x^2 + x^3; (* two very similar functions *);

In[40]:= Plot[{f1[x], f2[x]}, {x, -1, 3}] (* Both have 3 real roots *)
```

Out[40]=




```
In[42]:= s1 = Solve[f1[x] == 0, x] (* here we get normal expressions *)
```

Out[42]= $\left\{\{x \to 1\}, \left\{x \to \frac{1}{2}\left(1 - \sqrt{5}\right)\right\}, \left\{x \to \frac{1}{2}\left(1 + \sqrt{5}\right)\right\}\right\}$

```
In[43]:= s2 = Solve[f2[x] == 0, x] (* here we get a 'Root' object *)
```

Out[43]= $\left\{\left\{x \to \text{Root}[-0.532\ldots]\right\}, \left\{x \to \text{Root}[0.653\ldots]\right\}, \left\{x \to \text{Root}[2.88\ldots]\right\}\right\}$

```
In[72]:= x /. ToRadicals[s2]
```

Out[72]= $\left\{1 - \frac{1}{2}\left(1 - i\sqrt{3}\right)\left(\frac{1}{2}\left(1 + i\sqrt{3}\right)\right)^{1/3} - \left(\frac{1}{2}\left(1 + i\sqrt{3}\right)\right)^{2/3}, 1 - \frac{1 - i\sqrt{3}}{2^{2/3}\left(1 + i\sqrt{3}\right)^{1/3}} - \frac{\left(1 + i\sqrt{3}\right.}{2 \times 2^{1/}}\right.$

```
FullSimplify@Im@First@# (* check that first 'complicated' solution is real *)

Out[73]= 0
```

Exercise 3: Solving Equations

- 3 friends Andy, Bob, Conny have different ages:
  1. Conny is 2 years older than Andy
  2. Conny is twice as old as Bob
  3. Together they are 38 years old
- How old are they?
  - Solve the problem in one line of code…

- Replace the first condition by
  1. Conny is older than Andy
- Can you **`Solve[]`** now? Try **`Solve[..,{A,B}]`**
- Try **`Reduce[..., {A,B}]!`**
- What do you learn about Conny's age?

- Add 3 condition that the ages are >0 and **`Reduce`** again

Exercise 4

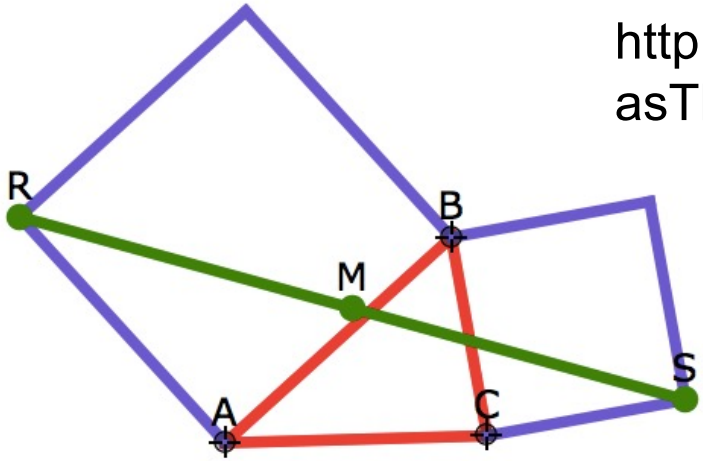
- Calculate the `Sum[…]` of $1/n^2$ for n=1…∞
- Let the sum run only until 100
- Extract π from this approximation. How large is the error ?

- Make a list containing Sin[α] for α = 0...π in steps of $\pi/4$

- Make a list of the first 10 numbers (1…10)
- Find a way to square them all (Hint: The solution is not far…)

Exercise 5: Bottema's Theorem

- One of the many relationships in geometry is shown here:

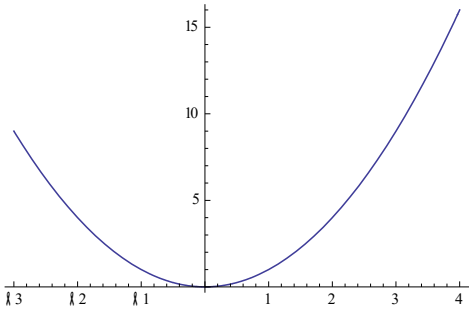


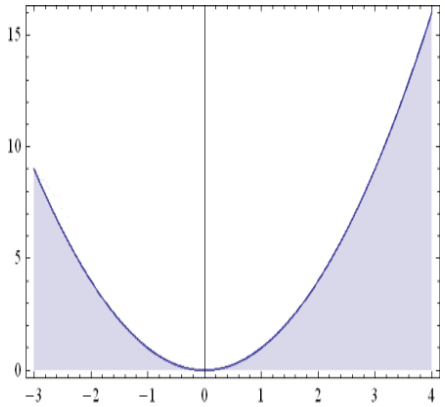
http://demonstrations.wolfram.com/BottemasTheorem/

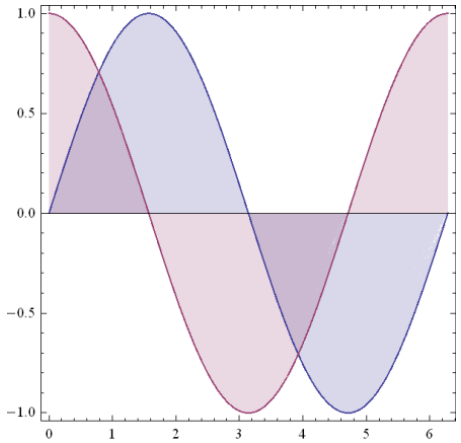
"Take an arbitrary triangle ABC. Create squares on edges AB and BC. The centre of the line connecting the 'lower' corners of the squares is **independent** of B."

- Prove the theorem!
  - Let points A,B,C be *3D* vectors **`A={ax,ay,0}`**,… with z=0
  - Get R,S with the cross product **`Cross[...,{0,0,1}]`**
  - Calculate M. Show that it is independent of B
  - If A={0,0} and C={1,0}: Where is M?

Simple Plotting







- `Plot[expression,{var,start,stop},parameters…]`
  - `Plot[x^2, {x,-3,4}]`
- Parameters can be used to change the display:
  - `Frame->True`
  - `Filling->Axis`
  - `ImageSize->300, AspectRatio->2`
  - `PlotRange->{ymin,ymax}` or `PlotRange->{{xmin,xmax},{ymin,ymax}}`
  - `Plot[x^2,{x,-3,4},Frame->True, Filling->Axis, ImageSize->300]`
- Several expressions can be plotted at once:
  - `Plot[{Sin[x],Cos[x]},{x,0,2π}]`
- The options can be set globally:
  - `Options[Plot]` `(* show the options *)`
  - `SetOptions[Plot,{…}];` `(* set them forever *)`

Exercise 6: Simple Plotting

- Define function `f1[k_,x_]:=Sin[k x]/x`;

- Plot f1 for k=1 for x from 0 to 2π.
  - Play with Options, e.g.
    `Frame->True, Filling->Axis, ImageSize->300`
  - Try to find some more options in the help

- Now use the list {1,2} for k to get multiple plots
  - How can you get the same result when you plot directly
    `Plot[Sin(k x)/.k->{…},{…}]`
    (Using a replacement rule for k)

- Plot sin(x) and cos(x) in the same plot. Hint: Use a list!

Common Mistake

- `f[k_,x_] := Sin[k x]` defines a function with **two** parameters!
  - f alone (with no arguments) is NOT defined

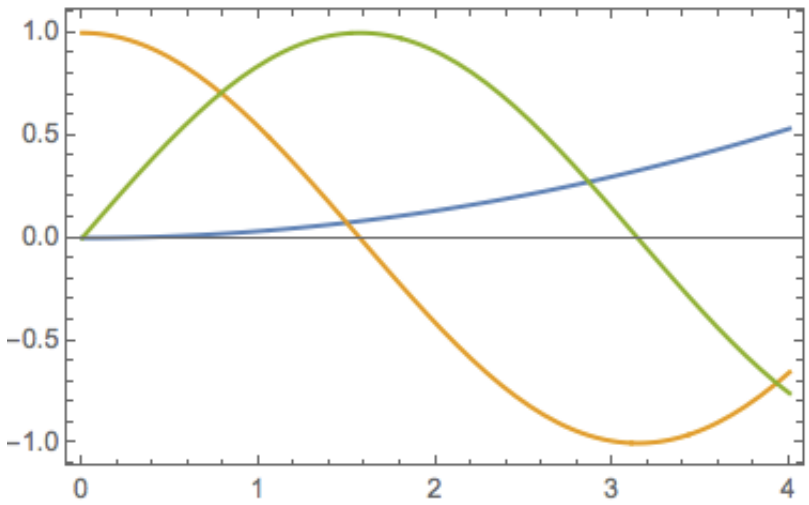
- You can do
  - `f1[k_,x_]:=Sin[k x]`
  - `Plot[f1[1,x], {x, 1, 2}]`

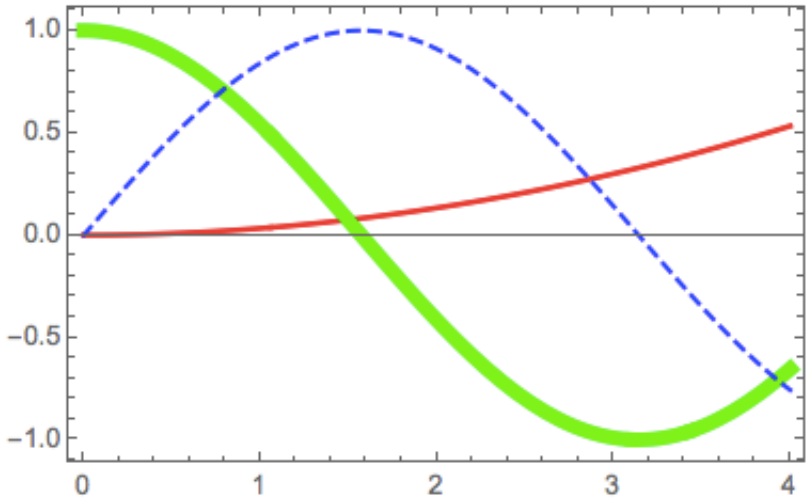
- Or
  - `f2:= Sin[k x]`
  - `Plot[f2 /. k -> 1, {x, 1, 2}]`

- But NOT (with f1)
  - `Plot[f1 /. k -> 1, {x, 1, 2}]`

(Plotting Several Curves)

- A *list of functions* automatically plots with different colours:
- `Plot[ {x^2/30, Cos[x], Sin[x]}, {x,0,4} ]`





- The way the individual curves are displayed can be changed with the `PlotStyle->` command. For instance:
-
```
Plot[{x^2/30, Cos[x], Sin[x]}, {x,0,4},
  PlotStyle -> {{Thick, Red}, {Thickness[0.02],
  Green}, {Dashed, Blue}}
]
```

(Plotting: Why do curves look the same?)

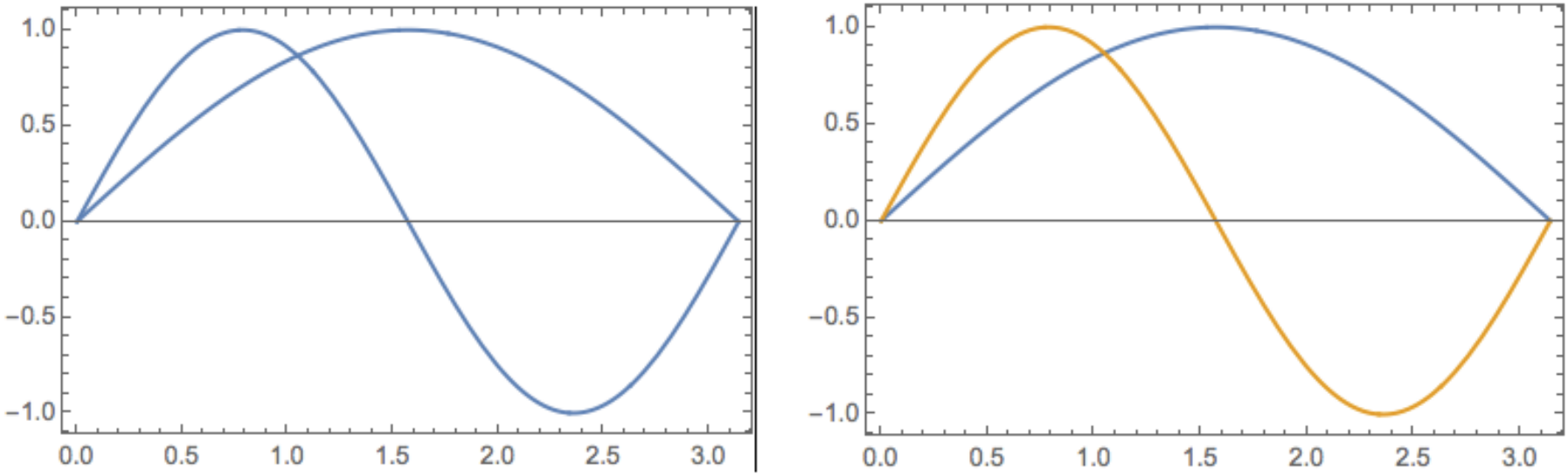
- Sometimes, curves look the same:

```
Plot[Sin[k x] /. k -> {1, 2}, {x,0,π}]
```

or

```
Plot[Table[Sin[k x], {k, 1, 2}], {x,0,π}]
```

gives:



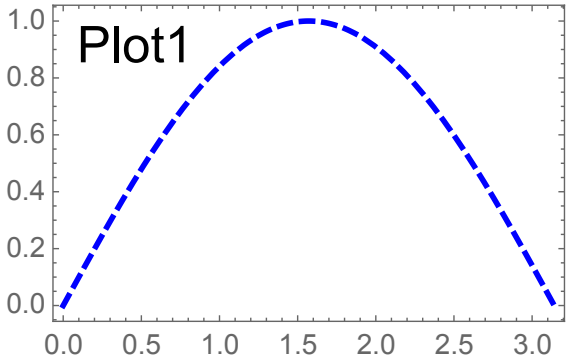
- The reason is that the (function) expression is not *Evaluated* and is considered as a 'group' of curves. Evaluation can be forced (here using prefix notation):

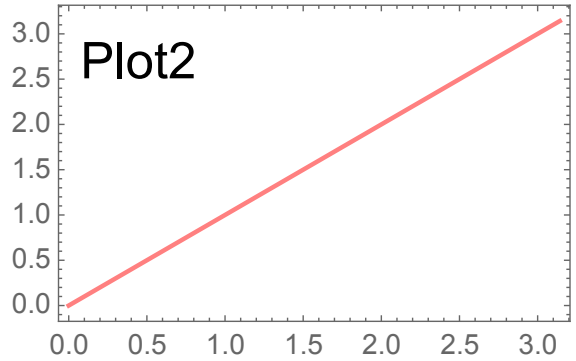
```
Plot[Evaluate@Table[Sin[k x], {k, 1, 2}], {x,0,π}]
```

(Merging Several Plots)

- Plots can be stored

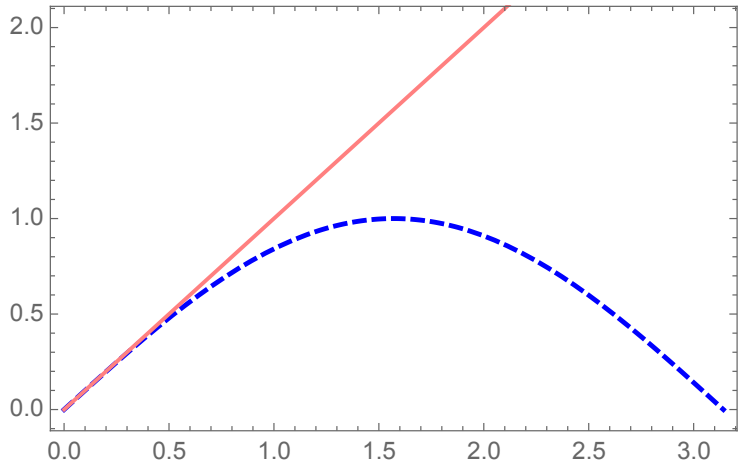
```
Plot1 = Plot[Sin[x], {x,0,π},
  PlotStyle -> {Thick, Blue, Dashed}]
Plot2 = Plot[x, {x,0,π}, PlotStyle -> Pink]
```





and combined in one image using `Show[]`:

```
Show[Plot1, Plot2,
PlotRange -> {0, 2}]
```

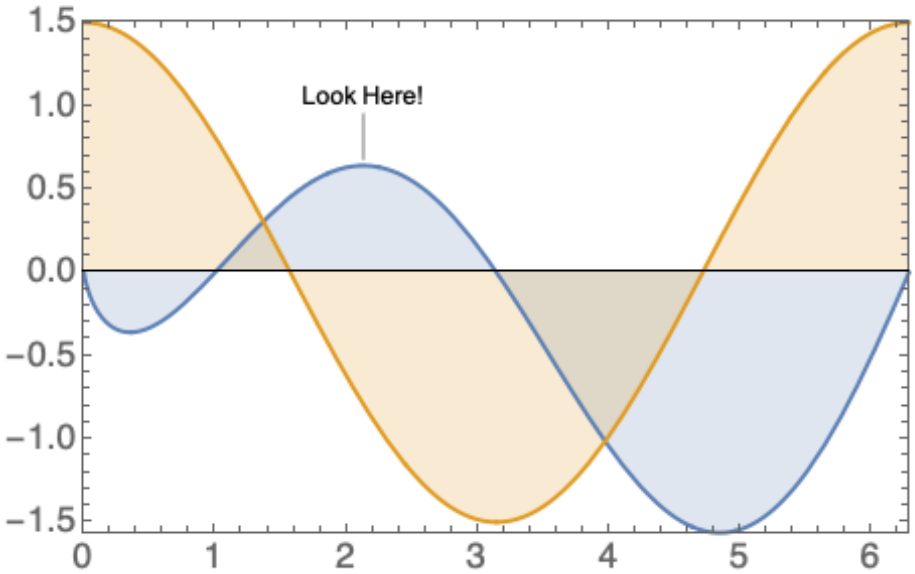


(Useful: Adding Plot Labels)

- `Callout[object,text]` allows adding a legend:

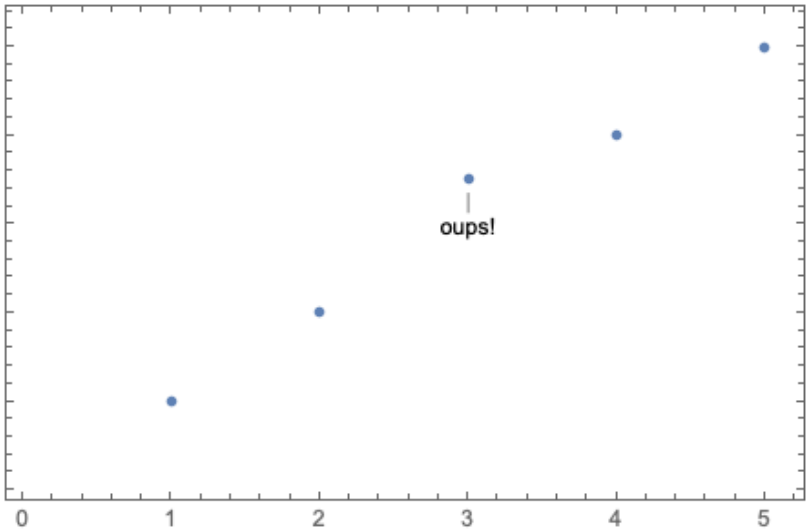
```
In[190]:= Plot[{Callout[Log[x] Sin[x], "Look Here!", Above], 1.5 Cos[x]}, {x, 0, 2 π}]
```

Out[190]=




```
In[201]:= ListPlot[
  {1, 2, Callout[3.5, "oups!", Below], 4, 5}
  , Frame → True
]
```

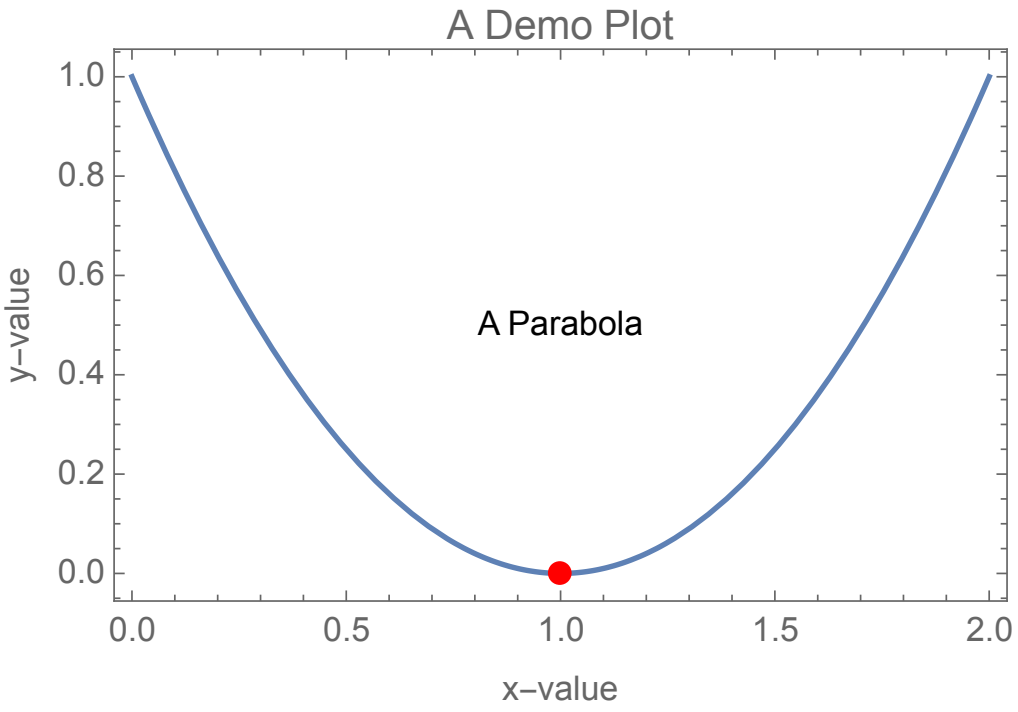
Out[201]=



(Adding Extra Stuff to Plots)

- To add stuff, you can **`Show[]`** the plot with other graphics objects *or* add the 'stuff' in an **`Epilog->`** command:

```
Plot[(x-1)^2, {x,0,2}
   ,PlotLabel -> "A Demo Plot"
   ,FrameLabel -> {"x-value", "y-value"}
   ,Epilog -> {Text["A Parabola", {1, 0.5}],
    Red, PointSize[Large], Point[{1, 0}]}
]
```



Exercise 7: Solving Equations

- Define two functions `f1[x_]=3x+2` and `f2[x_]=2(x-2)^2`
- Plot them (x=0…6)
- Find the intersections. Assign the result to `Sol`
- Try

```
pp = {x, f1[x]} /. Sol
```

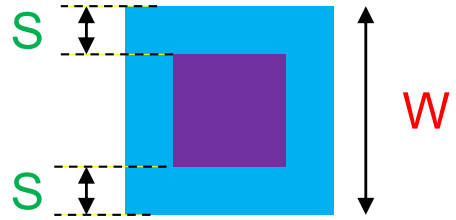
- What happens?
- Get the numerical value and compare to your plot

- Add the points `Point[pp]` to your plot with the option `Epilog -> {PointSize[0.03], Point[pp]}`

- You may try for instance $f_1=x+4$ and $f_2= 3x^3 - 2x^2 - x + 5$

Exercise 8: A simple real world application

- In my group, we design silicon chips which contain photo diodes. We assume these are square.
  - A pixel has side width W which we can chose freely
  - The (square) sensitive photo diode is smaller by (fixed) S=3 µm on all sides
  - During production, rare defects with a (fixed) density of $\alpha \sim 10^{-5}$ (defects per μm^2) can 'kill' the photo diode, when it is 'hit'.



- When we use
  - many small diodes, we lose a lot of active area because of the edges, but the defects 'kills' only small areas
  - few large diodes, we lose a large area for each defect
- What is the optimal diode size W, maximizing *active* area?
- Hints:
  - Write down pixel area and active area as a function of W (and S)
  - For a total device area A, how many pixels Npix do you need?
  - What is the probability that one pixel is 'killed'. How many of the Npix pixels survive?
  - What is the total active area A_{active}? what is the 'fill factor, i.e. ff = A_{active} / A
  - Plot ff[W]. Find (numerically) the maximum with **`NMaximize[]`**.

(Exercise 9: Quadratic Equation and More)

1. Quadratic Equation
   - Solve the quadratic equation a x^2 + b x + c == 0
   - Assign the solutions to two variables x1 and x2
   - Check that x1 and x2 are indeed solutions

2. Try to solve a general polynominal of 3rd, 4th degree

3. Generate 10 Polynomials of the form $1+x^n$ in a list.
   - Try to `Factor[…]` them. Which of them are 'prime'?

4. Generate a random polynominal of 5^{th} order
   - Generate random numbers between -1..1 with `RandomReal[]`
   - Use `Sum[...,{..}]` to generate the polynominal (use help!)
   - Plot it. (For fun, plot several random polynominals in one plot)
   - Solve it numerically using `NSolve[]`
   - Eliminate all complex solutions with assumption x $\in$ Reals

(Exercise 10: Maximal Area of a Rectangle)

- Which Rectangle of periphery P has largest area A?
  - We treat a rectangle with sides a,b, periphery P and area A

1. Find a for given P, b (from P == 2 a + 2 b)
2. Inject this value into the equation for A
3. **`Maximize[]`** the area

- Is this a square ?

- Plot the area for P=40 as a function of b

Now Let's do 'Real' Mathematics: Derivatives

- Mathematica can do symbolic derivatives
  - `D[expression, var]` (* first derivative *)
  - `D[expression, {var,n}]` (* n-th derivative *)
  - `f'[x]` (* can use ' *)

```
D[3 x^2 + 5 x, x]
```

```
5 + 6 x
```

```
D[Log[Sin[x + y]], y]
```

```
Cot[x + y]
```

```
f[x_] = (Cos[x] Sin[5 x])^2; f'[x]
```

```
10 Cos[x]^2 Cos[5 x] Sin[5 x] - 2 Cos[x] Sin[x] Sin[5 x]^2
```

```
Table[D[Sin[x], {x, n}], {n, 0, 4}]
```

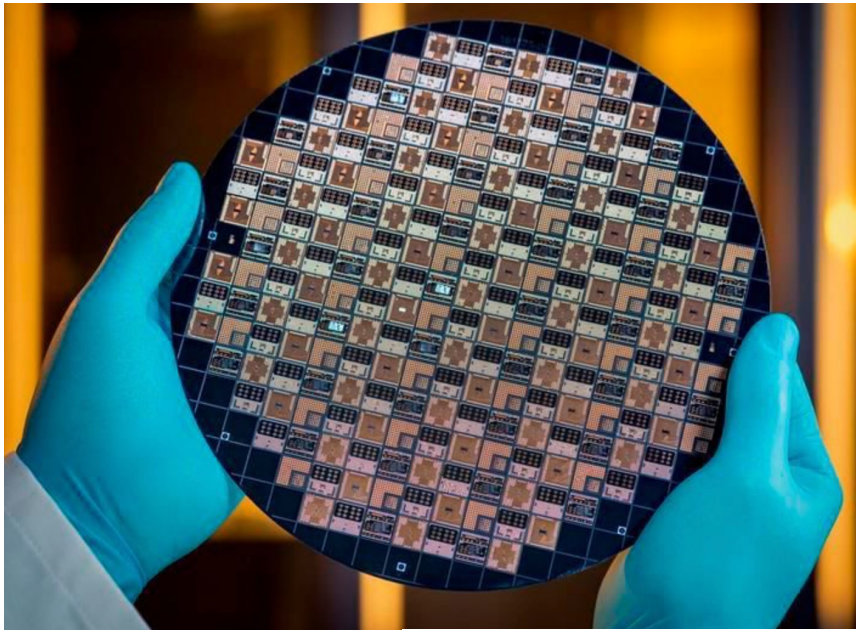
```
{Sin[x], Cos[x], -Sin[x], -Cos[x], Sin[x]}
```

```
D[Sin[x]^10, {x, 4}]
```

```
5040 Cos[x]^4 Sin[x]^6 - 4680 Cos[x]^2 Sin[x]^8 + 280 Sin[x]^10
```

Exercise 11: A Real World Problem

- A silicon wafer of 200 mm diameter contains two types of microchips:
  - Chips 'A' with a size of 3x3 mm^2
  - Chips 'B' with 5x5 mm^2



Fraunhofer IZM | Volker Mai

- The relative amount of chips can be chosen by you. We assume that chips 'A' cover an *area* fraction α of the wafer ($0 \le \alpha \le 1$).
- The vendor produces $N_W=12$ wafers . This 'batch' is split in two types:
  - A fraction β ($0 \le \beta \le 1$), i.e. $\beta \times N_W$ wafers, is produced such that chips 'A' can be used
  - The remaining wafers are used for chips 'B'.
- You need 10000 chips 'A', not more.
- How do you chose α and β so that you get *as many chips* 'B' as possible?

Exercise 11: Hints

- Best collect all parameters in one replacement list
  PAR = {WaferArea→…, AreaA→...,...}

- What is the area of one wafer?
- How much area is available for chips A/B ?
- How many chips A/B do you get per wafer? How many in total?

- You get the number of chips NA(α,β) and NB(α,β) as a function of α,β.
- Solve NA==10000 for β, yielding βsol.
- Plot NB for this βsol as a function of α. Is there a maximum?
- Find the maximum by setting the derivative of the above function to zero.
- For this αmax, what is βmax?

- What is the numerical value (i.e. how many wafer of type 'A' should you order)?
- How many Chips B do you get?

Integrals

- Indefinite Integrals
  - `Integrate[expression, var]`
- Definite Integrals
  - `Integrate[expression, {var,start,stop}]`

```
Integrate[Log[x], x]
```

$$-x + x\,\text{Log}[x]$$

```
Integrate[1/(x^3 + 1), x] // Simplify
```

$$\frac{1}{6}\left(2\sqrt{3}\,\text{ArcTan}\left[\frac{-1+2x}{\sqrt{3}}\right] + 2\,\text{Log}[1+x] - \text{Log}\left[1-x+x^2\right]\right)$$

```
Integrate[Exp[-c x^2], {x, -∞, ∞}]
```

$$\text{ConditionalExpression}\left[\frac{\sqrt{\pi}}{\sqrt{c}}, \text{Re}[c] > 0\right]$$

```
{∫_0^π Sin[x] dx, Integrate[Cos[x], {x, 0, π}]}
```

$$\{2, 0\}$$

Other Analysis Stuff

- Limits (also for x→∞)
  - `Limit[Sin[x]/x,x->0]` → `1`
- Sums
  - `Sum[k^2, {k,1,n}]` → `1/6 n (1+n) (1+2 n)`
  - `Sum[1/k^2, {k,1,∞}]` → $\pi^2/4$

$$\text{Limit}\left[\frac{\text{Sin}[x]}{x}, x \to 0\right]$$

```
1
```

$$\text{Limit}\left[\frac{\text{Sin}[x+\epsilon]-\text{Sin}[x]}{\epsilon}, \epsilon \to 0\right]$$

```
Cos[x]
```

$$\text{ArcTan}[\infty]$$

$$\frac{\pi}{2}$$

$$\text{Limit}\left[\frac{\text{Cosh}[x]}{\text{Sinh}[x]}, x \to \infty\right]$$

```
1
```

$$\text{Sum}[k^2, \{k, 1, n\}]$$

$$\frac{1}{6} n (1+n) (1+2n)$$

$$\text{Sum}[\text{Table}[1/k^n, \{n, 1, 4\}], \{k, 1, \infty\}]$$

Sum::div : Sum does not converge. ≫

$$\left\{\sum_{k=1}^{\infty} \frac{1}{k}, \frac{\pi^2}{6}, \text{Zeta}[3], \frac{\pi^4}{90}\right\}$$

$$\text{Sum}\left[\frac{x^n}{n!}, \{n, 0, \infty\}\right]$$

$$e^x$$

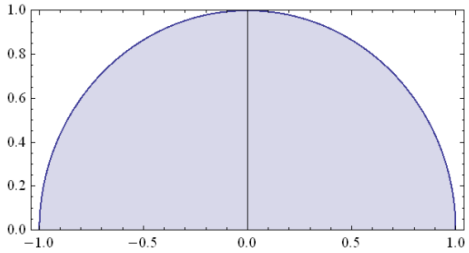
Exercise 12: Integral

- Consider the function `f[x_]=A/(1+x^2)`
- Plot f
- Find `A` so that the function is normalized, i.e. its integral over all x is 1
- Plot f and its integral ('Stammfunktion')

- Generalize a bit by introducing a parameter α:
  `g[x_]=A/(1+ax^2)`

Exercise 13: Area of a circle

- Write down the relation EQ between x, y, r for a circle
- **Solve** this for y (at given x, r) and define a function `y[x,r]`
- Plot a half circle (x from –r to r)
  - Make sure the plot is properly scaled



- Calculate the area by integration from –r to r (this takes surprisingly long..)
- Simplify the result by using r>0

- Now first calculate the general integral 'Integral y[x,r] dx'
  - Plot how the area increases as x increases from –1 to 1 (for r=1)

  - Try to set r=x.
    Get rid of the indefinite expression with **`Limit[…]`**
  - Is the result as expected?
  - Use the option **`Direction`** in **`Limit[…]`**

(Exercise 14: Some More Plotting & Stuff)

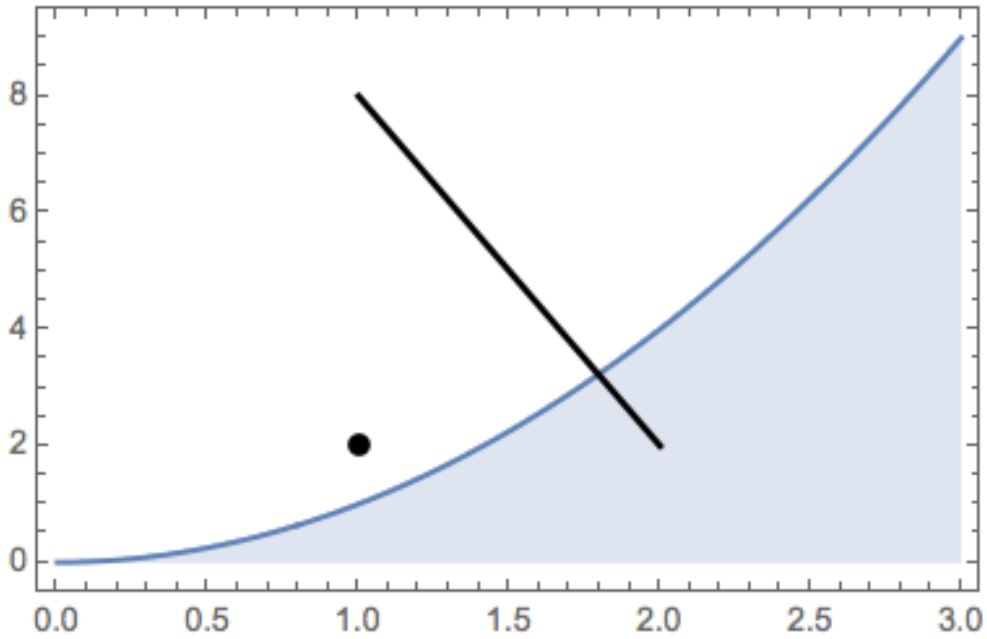
- Define the function **`f[x_]=Sin[a x] Exp[-λ x]`**
  - Use Assumptions **`a>0`** and **`λ>0`**
- Try Solve/Reduce to find **`a`** such that the first zero is at x=1…
  - Also try to prove that the a you chose leads to f[x]=0 for integer value of x
- Plot the function with this **`a`** for some values of λ

- You may know that a saw tooth Function can be decomposed (Fourier transformed) into functions **`Sin[kx]`** with coefficients **`(-1)^k / k`**
  - Define the k-th such component as **`g[x,k]`**
  - Plot these base functions with increasing k (use **`Table[..]`**)
  - Define the sum of the first k components f[x,k] (use **`Sum[..]`**)
  - Plot some sums with increasing number of overtones to show how the saw tooth function is approximated

ADVANCED TOPICS

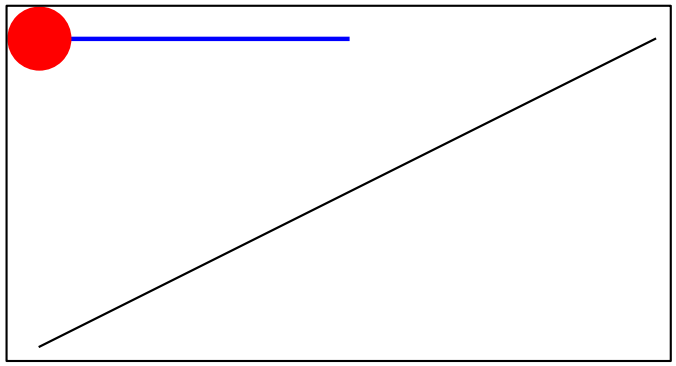
Graphics (see 'Adding extra stuff to plots')

- Result of `Plot[…]` is a ***graphic*** object:
  - `Plot1 = Plot[x^2,{x,0,3}];`
- There are other graphic objects:
  - `APOINT = Graphics[Point[{1,2}]];`
  - `ALINE  = Graphics[Line[{{2,2},{1,8}}]];`
- They can be plotted together:
  - `Show[Plot1,APOINT,ALINE];`

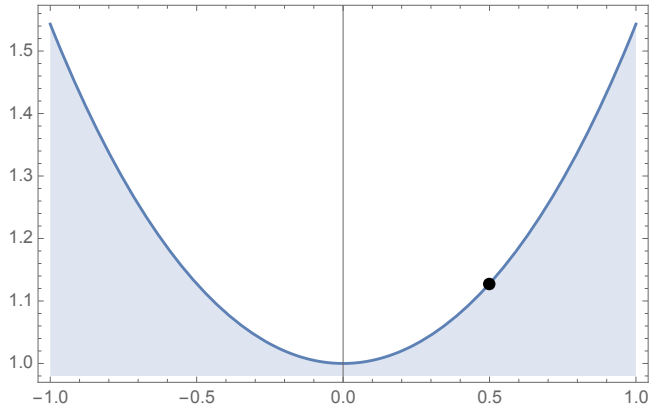


Graphics

- Graphics objects can be merged in list (with `{ .. }`):
  - `Graphics[{Line[{{1, 1}, {3, 2}}], Thick, Blue, Line[{{2, 2}, {1, 2}}], PointSize[0.1], Red, Point[{1, 2}]}]`

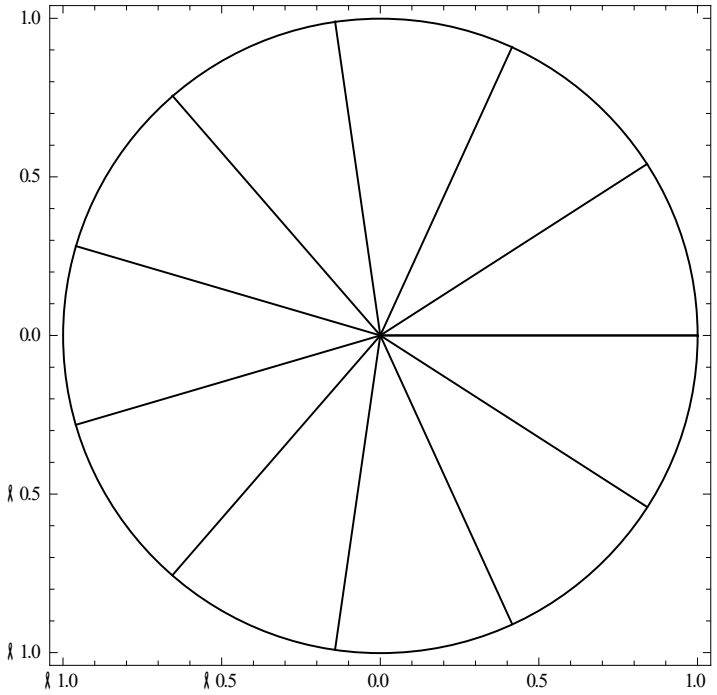


- To directly include graphics in a `Plot[]`, you an use `Epilog:`
  - `Plot[Cosh[x], {x, -1, 1}, Epilog -> { PointSize[Large] , Point[{0.5, Cosh[0.5]}] } ]`



Exercise 15: Drawing a Wheel

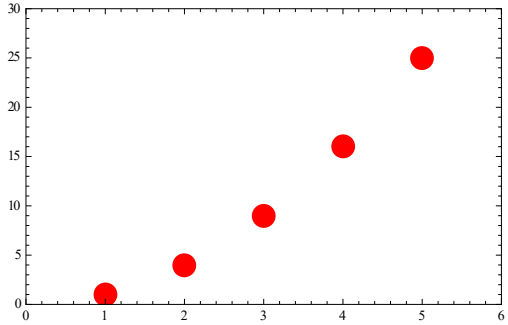
- Draw the following Wheel:



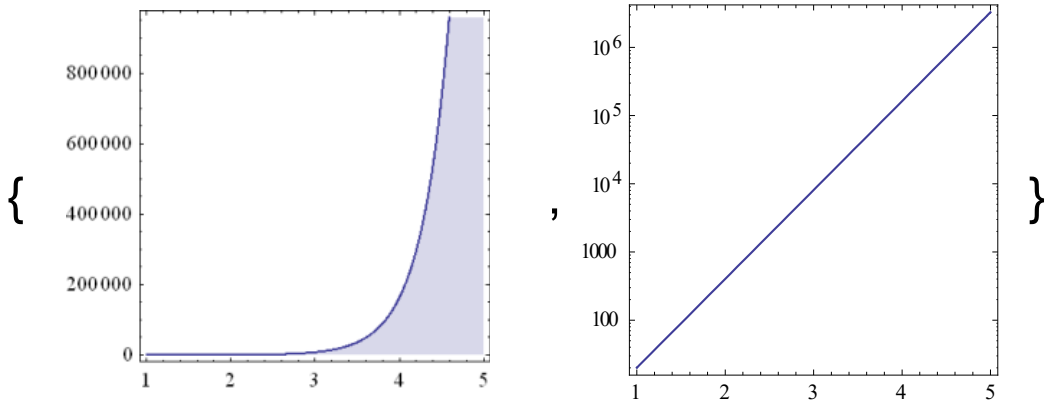
- Hint: you get one spoke by
  `Line[{{0,0}, {Cos[2 π α], Sin[2 π α]}}]`
- Make a **`Table[]`** of such spokes for α from 0 to 1 in 11 steps
- **`Show[Graphics[{…the spokes…,…the circle…}]]`**

More 2D Plotting

- Plot data sets (lists): `ListPlot[list]`
  - ```
    ListPlot[Table[i^2,{i,1,5}],
      PlotRange->{{0,6},{0,30}},
      PlotStyle->{PointSize[0.05], Red}
    ]
    ```

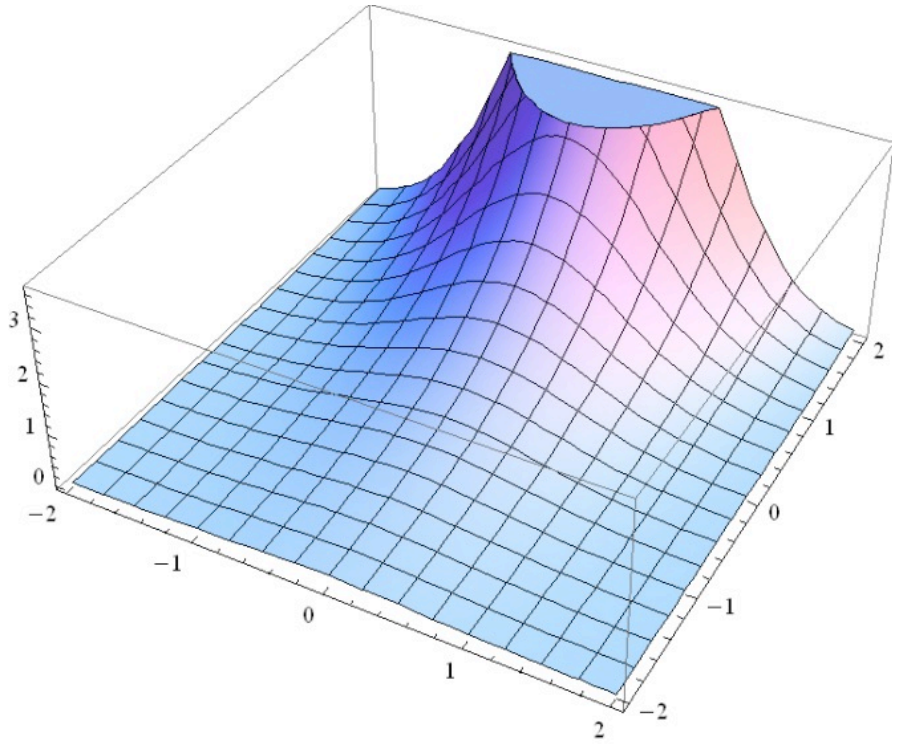


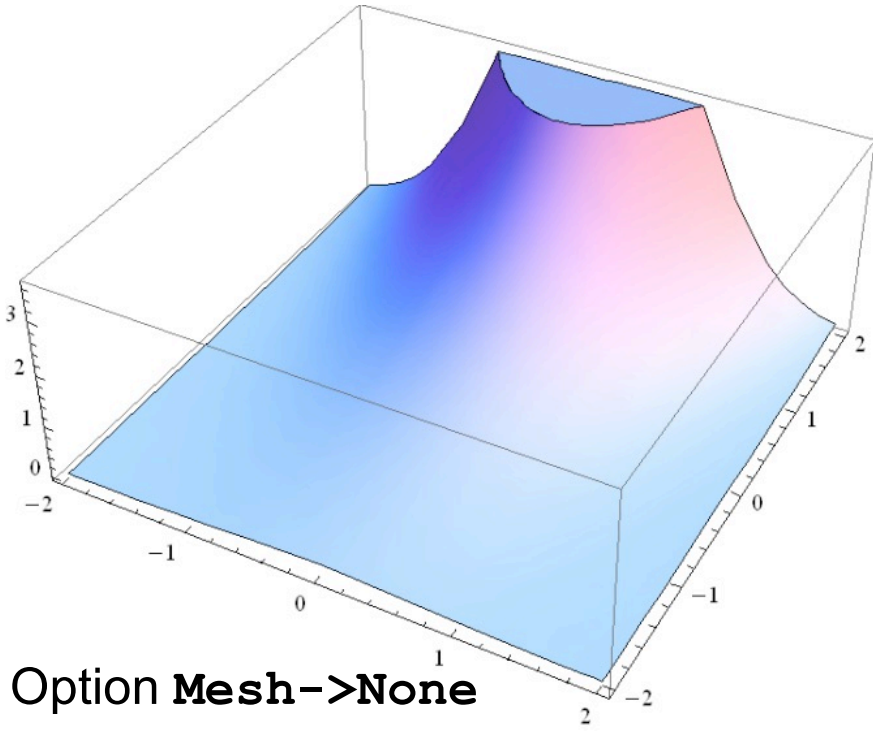
- Axis scaling: `LogPlot[…], LogLogPlot[…]`
  - ```
    {Plot[Exp[3x],{x,0,5},
    ImageSize->200,Frame->True,AspectRatio->1],
    LogPlot[Exp[3x],{x,0,5},
    ImageSize->200,Frame->True,AspectRatio->1]
    }
    ```



3D Graphs

- Many ways to show surfaces, …
- For instance
  - `Plot3D[f[x,y],{x,-2,2},{y,-2,2},…options…]`





Option **`Mesh->None`**

- View position can be changed with the mouse.
- Many options (see help)

Data & Image File IO

- Can save plot results to file:
  - `SetDirectory["C:/…"]`
    `or`
    `SetDirectory[$HomeDirectory <> "/xxx/xxx"]`
    `or`
    `SetDirectory[NotebookDirectory[] <> "subdir"]`
  - `PLOT = …`
  - `Export["file.pdf",PLOT]  (* or .jpg, .gif,… *)`
- Can also read data from file:
  - `A = Import["file"];`
- You can specify data types, skip lines, …:
  - `Import["filename", "csv", HeaderLines -> 1];`

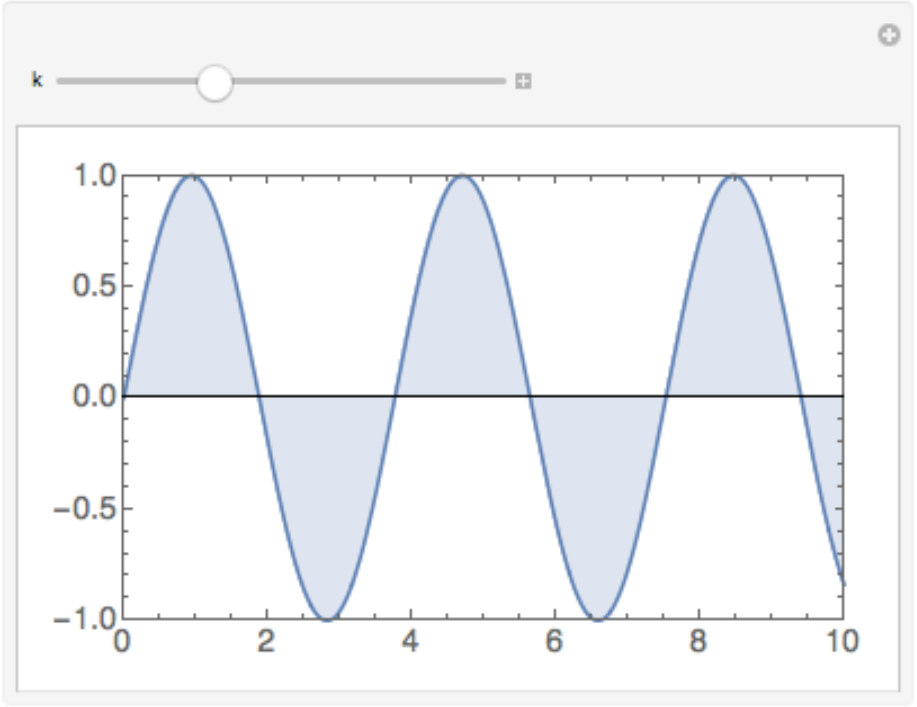
(Magic: `Dynamic[]`)

- The expression `Dynamic[x]` will be updated whenever x changes!
  - **Everywhere** on the notebook, i.e. also above the place where you change x!
- Try `Slider[Dynamic[x]]` and then assign a value (0…1) to…
- Or Make a second, identical slider and use it….
- What is `Dynamic[x]`

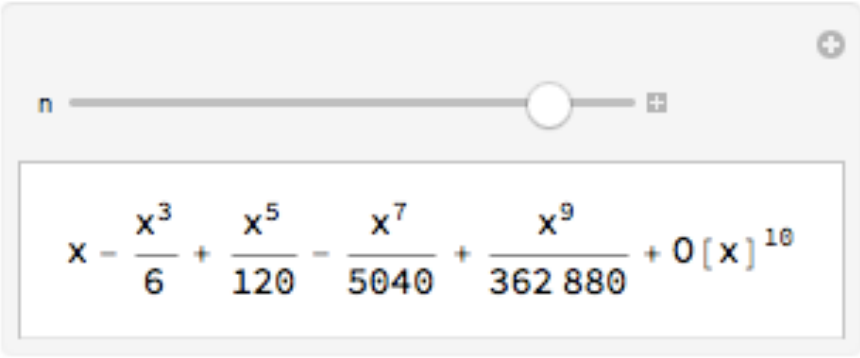
Manipulate: Direct User Interaction

- Manipulate is used for interactive animations:

```
Manipulate[
 Plot[Sin[k x], {x, 0, 10}]
 , {k, 1, 3}
]
```

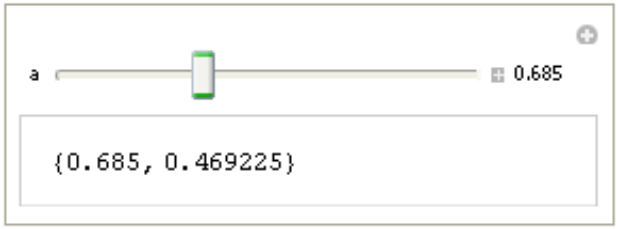



```
Manipulate[
 Series[Sin[x], {x, 0, n}]
 , {n, 1, 10, 1}
]
```

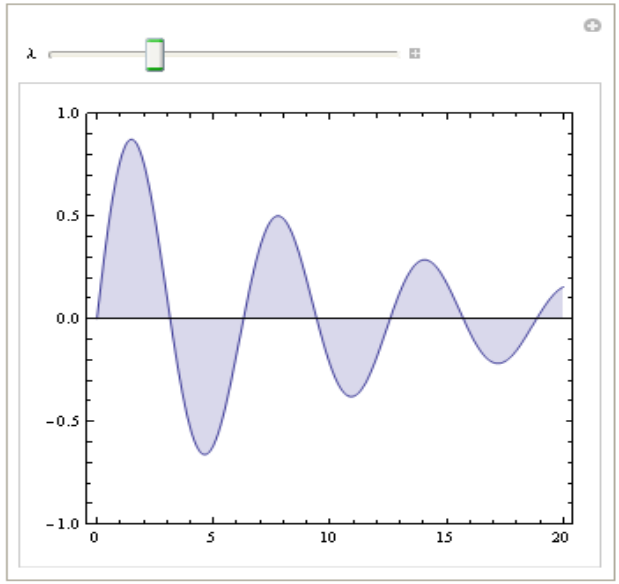


Manipulate: Syntax

- Syntax:
  - **Manipulate[**
    … some commands …
    ,{ {variable,default,"caption"}, min, max,
    Appearance->"Labeled"}
    ]
- For instance:
  - **Manipulate[**
    {a,a²},
    {{a,1,"a"},0,2,
    Appearance->"Labeled"}
    ]

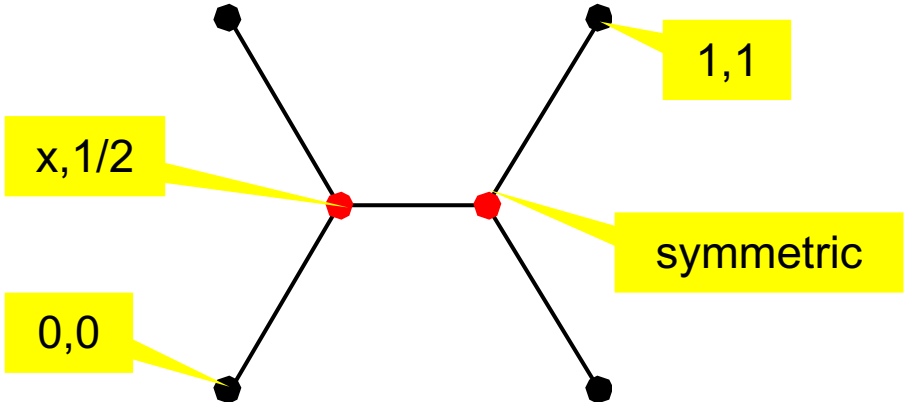


  - **Manipulate[**
    Plot[Exp[-λ x] Sin[x],{x,0,20},
    PlotRange->{-1,1}],
    {λ,0,0.3}
    ]



Exercise 16: Minimizing a Mesh

- We want to connect the two red points to four corners with minimal total distance ('soap films').
  - The black corners are at 0/0,1/0,0/1,1/1, the left red dot is at x/0.5

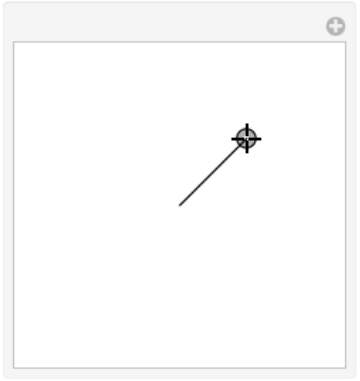


  - Calculate the total distance as a function of x
  - Plot it for x = 0…0.5
  - For which x is it minimal ? Plot that point into the diagram!

  - At which angles do the lines join at the red points? (half angle?)
  - Define 4 corner points and a list corners=Point[{p0,p1,p2,p3}];
  - Make a graphic which shows the above figure (without text).

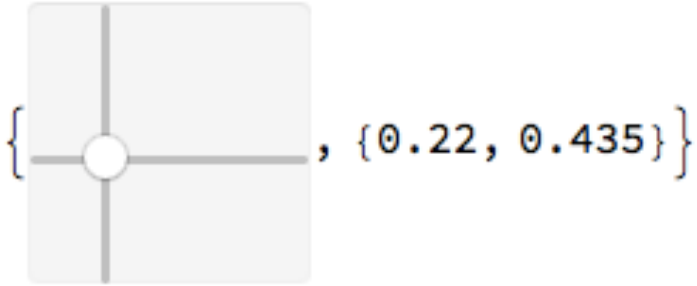
(Manipulate: Advanced Controls)

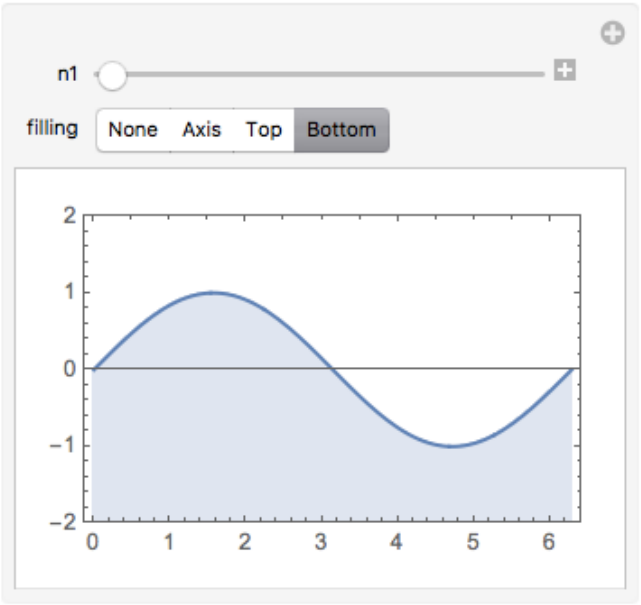
- There are many different types of control objects: Button, Checkboxes, Sliders, 2D Locators, 2D Sliders,.. :



```
Manipulate[
   Graphics[Line[{{0, 0}, p}],PlotRange->2],
 {{p, {1, 1}}, Locator}
 ]
```

```
{Slider2D[Dynamic[r]], Dynamic[r]}
```






```
Manipulate[
   Plot[Sin[n1 x], {x,0,π}, Filling->filling, PlotRange->2]
 , {n1, 1, 20}
 , {filling, {None, Axis, Top, Bottom}}
 ]
```

Programming, Local Variables

- Programming with usual command (`If[], For[],..`) is possible
- Local Variables can be defined using
  `Module[{variable}, code]`

```
S1[NN] := Module[{i, sum = 0},
  For[i = 0, i < NN + 1, i++, sum += i];
  sum]
```

```
S1[10]
```

```
S1[10]
```

(Advanced: Compile)

- To speed up calculation, routines can be compiled:

```
In[9]:= S1[NN_] := Module[{i, sum = 0},
           For[i = 0, i < NN + 1, i++, sum += i];
           sum
          ];

In[10]:= S1[1 000 000] // Timing

Out[10]= {1.31566, 500 000 500 000}

In[11]:= S2 = Compile[{NN}, Module[{i, sum = 0},
           For[i = 0, i < NN + 1, i++, sum += i];
           sum
          ]];

In[12]:= S2[1 000 000] // Timing

Out[12]= {0.112252, 500 000 500 000}
```

Define a Function which calculates the sum from 1…NN

Run it and measure time: **1.3 s**

Now **Compile** it

Compiled version runs faster: **0.11 s**

What Else?

- Differential Equations (analytical, numerical)

```
DSolve[f'[x]^2 == f[x], f[x], x]
```

$$\left\{\left\{f[x] \to \frac{1}{4}\left(x^2 - 2\,x\,C[1] + C[1]^2\right)\right\}, \left\{f[x] \to \frac{1}{4}\left(x^2 + 2\,x\,C[1] + C[1]^2\right)\right\}\right\}$$

```
DSolve[Sin[x] f'[x] == f[x], f[x], x]
```

$$\left\{\left\{f[x] \to C[1]\,\text{Tan}\left[\frac{x}{2}\right]\right\}\right\}$$

- Dynamical Variables
- Machine Learning
- Image Processing
- Graph Theory
- ….

Common Mistakes

- Miss-spelled function name
  - `simplify[Expression]`
- Expression has already been assigned
  - Observe the display colour!
  - Check with `?Expression`
  - Clear with `Clear[Expression]`
- Variables are not separated in the input:
  - `y = ax` is different from `y = a x`
- No '_' after variable in function definition `f[x] := x^2`
- Mismatch in number of arguments:
  - `A=Sin[x]; Plot[A[x], {x, 0, 3}]` -> no result
  - `A[x_]=Sin[x]; Plot[A, {x, 0, 3}` -> no result
- Subtle: Assignment contains command

```
In[35]:= x = {1, 2} // MatrixForm
Out[35]//MatrixForm=
(1)
(2)

In[36]:= 3 x
Out[36]= 3 (1)
           (2)
```

(init.m)

- You can execute a set of instructions at the start of mathematica (to be more precise: start of the kernel)
- The commands are stored (in normal mathematica syntax) in the file **init.m**
- You can get the file location by the command
  - FileNameJoin[{$UserBaseDirectory,"Kernel","init.m"}]
- To restart the kernel (note that this will clear ALL definitions), use
  - Quit[]
- You can load init.m (like any other package) directly by
  - << init` (use correct hyphen…)

MORE EXCERCISES
… IF YOU HAVE FUN…

‘Adventskalender’

- Since some years, the DPG research centre MATHEON publishes a mathematical ‘Adventskalender’
- Solve the following exercise (from http://www.mathekalender.de)

Aufgabe:

Beim Einpacken von Weinflaschen in viereckige Geschenkkartons fällt dem Weihnachtsmann auf, dass er jedesmal eine Menge Luft mit einpackt. Und außerdem verbrauchen diese Ecken sicher viel zu viel vom teuren Karton. „Wenn ich die Ecken abschneide und durch eine gerade Verbindung ersetze, kann ich die Verpackung aus weniger Karton falten“, überlegt er. Flugs macht er sich ans Werk und umhüllt die nächste Flasche mit einem achteckigen Karton. Zufrieden packt er weiter, bis ihm nach einer Weile auffällt, dass immer noch zu viel Karton verbraucht wird. „Wenn ich die Ecken wieder abschneide, sollte ich noch weniger Karton nehmen können“, sagt er sich und falzt eine sechzehneckige Verpackung. Nun packt ihn der Ehrgeiz. Schnell ist ihm klar, dass kreisförmige (zylindrische) Verpackungen optimal wären, nur leider lässt sich der Karton nicht gut biegen. Aber immerhin kann er mehr und mehr Ecken falzen. Er hört mit der Eckenverdoppelung erst auf, als sein Verpackungsumfang um nicht mehr als ein Tausendstel länger ist als bei einer zylindrischen Umhüllung.

Wie viele Ecken hat seine Verpackung dann?

- Define a Function L[N] which gives the length of the regular polygon with N corners and (inner) radius 1. Plot the function
- Check when the function reaches $1.001 \times 2\pi$
- Find the numerical solution with **`FindRoot`** (not **`NSolve`**)

Particle Absorption

- Particles penetrating a material are absorbed according to an exponential extinction law f(x) = A Exp[-α x]
- Find A so that the total number of absorbed particles is normalized to 1

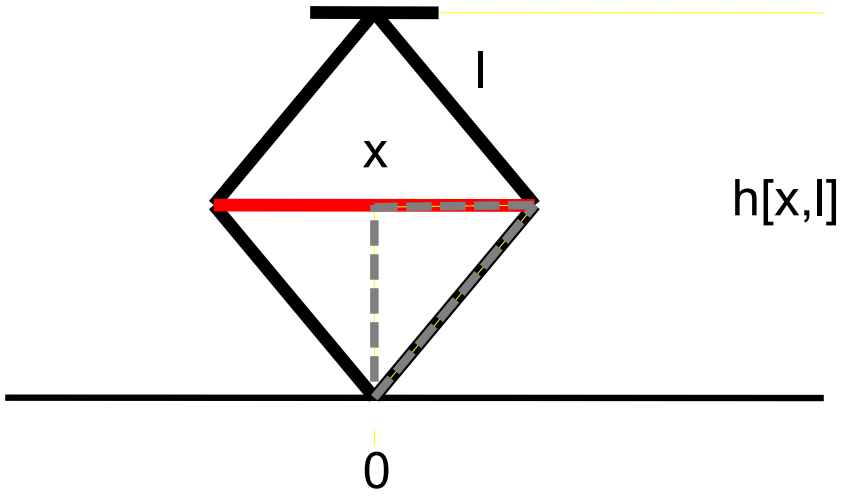
- Our absorber has a total thickness T.
- We want to divide it in two layers (0..T1, T1..T) so that an equal amount of particles are absorbed in both layers.

- Calculate T1 as a function of α (and T)
- Plot your result

- What T1 do you find for T=1 and α=1?

Car Lifter

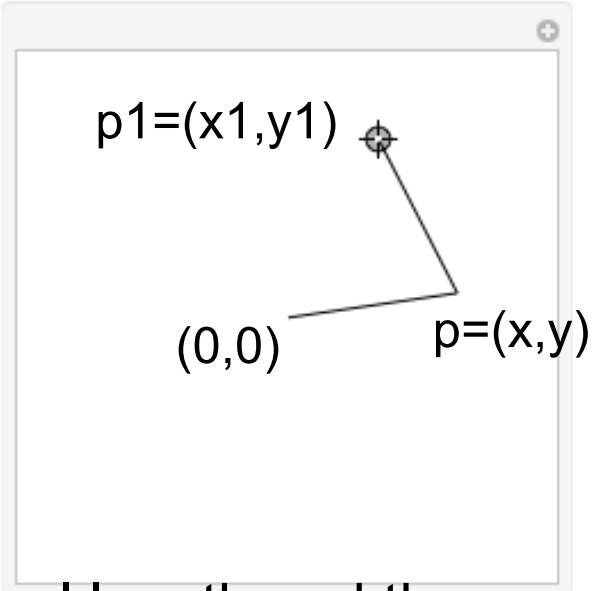
- A car lifter is composed of a (black) frame with 4 edges of length l and of a (red) drive of variable length x (see figure).



- Calculate the height h as a function of x. (You can solve Pythagoras's equation for the gray triangle for x)
- Plot this (what is the range of x?)
- Plot the vertical force created by a constant force on x as a function function of x (hint: you need a derivative!)
- If you want: Make a graphics and try Manipulate…

Equal Trace Lengths

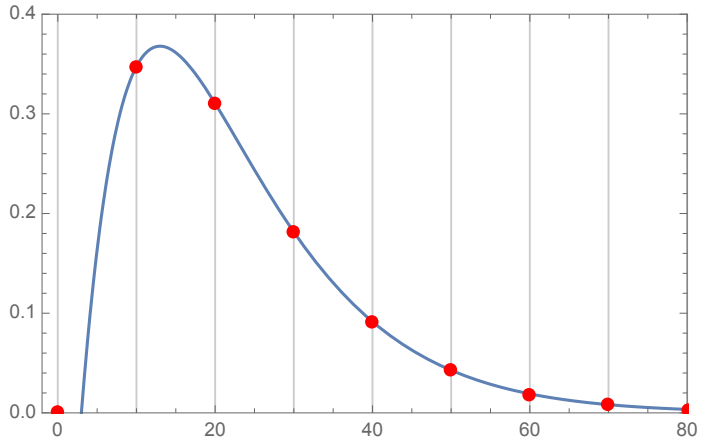
- On a 2D detector, we want to connect an amplifier at (x,y)=0 to a pixel at a position p1=(x1,y1).
- We want to keep the trace length constant (same capacitance)
- For simplicity, we use 2 straight lines of equal length with bend at p=(x,y)



- Limit all coordinates (x,y,x1,y1) to >0
- Define points p1 and p
- Calculate the length of the 2 straight parts as their `Norm[]`
- Find (x,y) such that both parts have equal length and the sum is some constant K>0.
- Assign the result to a point `Presult[x,y,K]`
- Try some special cases. Plot.
- Use Manipulate with a 'Locator' (see help!)

Pulse Sampling (Real World Problem)

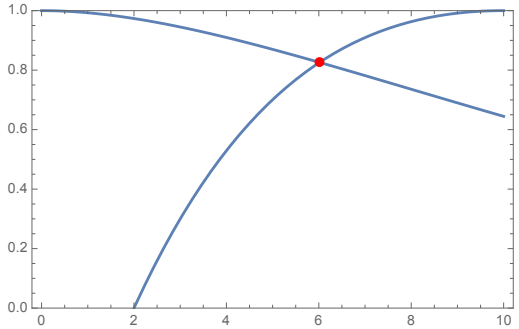
- A detector delivers pulses with pulse shape $f(t) = t/\tau\ e^{-t/\tau}$ starting at *random* times
- An ADC samples this signal at times $0, \Delta, 2\Delta, \ldots$



- We want to know the peak amplitude. This is often measured too small by the random sampling.
- We want to know the average error (for all possible sampling times) as a function of the pulse time constant τ (for a given Δ=1)

Pulse Sampling (cont.)

- Define the function f[t, τ, A] (A is an amplitude factor)
- Plot it for τ = 10 and A = 1
- Find out at which time the peak is located (set the derivative to zero and solve)
- What is the peak amplitude? Find A such that the peak amp. is 1.
- Define a new function f1[t, τ] which has peak amplitude 1
- Consider the samples around the peak. One sample shall be at a time x behind the peak, the other is Δ before. Show the amplitudes of the two samples in one plot (for x ranging from 0…Δ). Use for instance τ=8 and Δ=10.



- Where (at which x) do the two curves cross?
- We obviously take the larger sample as the amplitude estimate. We want to know the average amplitude (averaging over all x): Integrate the first part up to the crossing and the second part up to Δ and divide by Δ.
- What is the result for τ=8?