

$\text{T}_{\text{E}}\text{X}(\text{r})$ evolution

$\text{T}_{\text{E}}\text{X}$



$\epsilon\text{-T}_{\text{E}}\text{X}$



pdf $\text{T}_{\text{E}}\text{X}$



lua $\text{T}_{\text{E}}\text{X}$ = Lua + $\text{T}_{\text{E}}\text{X}$ + METAPOST

luatex lunatic

And Now for Something Completely Different
-- Monty Python , 1972

Completely Different ? Not really:

METAPOST is an example of "embedding" an interpreter into lua_T_EX at compilation time: see `luaopen_mplib(L)` in `source/texk/web2c/luatexdir/lua/luastuff.c`

```
void luainterpreter(void)
```

```
lua_State *L;
L = lua_newstate(my_luaalloc, NULL);
:
/* our own libraries */
luaopen_ff(L);
luaopen_pdf(L);
:
luaopen_tex(L);
luaopen_mplib(L);
```

So hosting Python it's not a new idea:
it can be something like

`luaopen_pythonlib(L) ...`

(of course it must works under all OS actually supported by lua_T_EX)

We are interested to find the smallest set of patches of luatex codebase, and we want to avoid:

1. constrains of any sort to development team
2. massive code modications
3. radical modication of building process.

Item 1. and 2. already say to us that
`luaopen_pythonlib(L)` is not a good solution ...

Lunatic Python:

"Lunatic Python is a two-way bridge between Python and Lua, allowing these languages to intercommunicate. Being two-way means that it allows Lua inside Python, Python inside Lua, Lua inside Python inside Lua, Python inside Lua inside Python, and so on.

The bridging mechanism consists of creating the missing interpreter state inside the host interpreter. That is, when you run the bridging system inside Python, a Lua interpreter is created; when you run the system inside Lua, a Python interpreter is created."

<http://labix.org/lunatic-python>

Gustavo Niemeyer's "personal laboratory"

Installation is simple:
download Lunatic python (rev. 7) with bazaar

```
$> bzz branch lp:lunatic-python  
(see https://code.launchpad.net/~niemeyer  
/lunatic-python/trunk)
```

and modify 4 lines in setup.py to match luatex dis-
tro.

```
1c1  
< #!/usr/bin/python  
---  
> #!/opt/luatex/luatex-lunatic/bin/python  
14,16c14,16  
< LUALIBS = ["lua5.1"]  
< LUALIBDIR = []  
< LUAINCDir = glob.glob("/usr/include/lua*")  
---  
> LUALIBS = ["lua51"]  
> LUALIBDIR = ['/opt/luatex/luatex-lunatic/luatex/build/texk/web2c']  
> LUAINCDir = glob.glob("../luatex/source/texk/web2c/luatexdir/lua51*")  
48a49  
>
```

It looks promising: fews modifications to luatex code.

In source/texk/web2c/luatexdir/lua51/loadlib.c
69c69

```
< void *lib=dlopen(path, RTLD_NOW);
```

```
---
```

```
> void *lib=dlopen(path, RTLD_NOW|RTLD_GLOBAL);
```

- RTLD_GLOBAL

The symbols defined by this library will be made available for symbol resolution of subsequently loaded libraries.

"We are loading the Python interpreter as a shared object, and the Python interpreter may load its own external modules which are compiled as shared objects as well, and these will want to link back to the symbols in the Python interpreter."

Modify loadlib.c is not enough: reference manual
manual/luatexref-t.pdf says:

"Dynamic loading of .so and .dll les is disabled on all platforms."

So we must enable it linking luatex with libdl.so:
other 2 files to modify.

```
##### source/texk/web2c/luatexdir/am/liblua51.am #####
```

```
12c12
```

```
< liblua51_a_CPPFLAGS += -DLUA_USE_POSIX
```

```
---
```

```
> liblua51_a_CPPFLAGS += -DLUA_USE_LINUX
```

```
##### source/texk/web2c/Makefile.in
```

```
#####
```

```
98c98
```

```
< @MINGW32_FALSE@am__append_14 = -DLUA_USE_POSIX
```

```
---
```

```
> @MINGW32_FALSE@am__append_14 = -DLUA_USE_LINUX
```

```
1674c1674
```

```
< $(CXXLINK) $(luatex_OBJECTS) $(luatex_LDADD)
```

```
$(LIBS)
```

```
---
```

```
> $(CXXLINK) $(luatex_OBJECTS) $(luatex_LDADD)
```

```
$(LIBS) -Wl,-E -uLuaL_openlibs -fvisibility=hidden -fvisibility-inlines-hidden -ldl
```

Let's look at last line: things start to complicate ...

We must ensure that `dlopen` symbol in `libdl.so` is visible to `luatex`, so `Lua loadlib` works:
this is done passing

```
-Wl,-E -uluaL_openlibs -ldl
```

options to the linker
(hard to discover `-uluaL_openlibs` opt.)

What about

```
-fvisibility=hidden -fvisibility-inlines-hidden
```

options ?

To avoid symbols collision: more exactly, a collision between a symbol of an external shared library and a symbol of luatex .

Example: luatex see symbol `png_memcpy_check` because it's coded in `libpng.so` linked at compilation time; it's in `source/libs/libpng/libpng-1.2.38/pngmem.c`

We have symbols collision if luatex lunatic load (at runtime, by python) a shared object that refers `png_memcpy_check` of another `libpng.so` ; otherwise no.

So symbols collision is a **runtime error**.

The solution is dirty:

- "hide all symbols" (simple):
just patch the build.sh script of luatex sources
by adding
28a29,36
> CFLAGS="-g -O2 -Wno-write-strings
-fvisibility=hidden"
> CXXFLAGS="\$CFLAGS
-fvisibility-inlines-hidden"
> export CFLAGS
> export CXXFLAGS
- "unhide" the symbols that are necessary to see
dlopen of libdl.so (not-so-simple)

"Unhide" is a matter of change from

```
gcc -DHAVE_CONFIG_H -I. -I../.../source/texk/web2c
-I../... -I/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/texk
-I/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/source/texk
-I../.../source/texk/web2c/luatexdir/lua51 -DLUA_USE_LINUX -g -O2
-Wno-write-strings
-fvisibility=hidden
-Wdeclaration-after-statement
-MT liblua51_a-lapi.o -MD -MP -MF .deps/liblua51_a-lapi.Tpo -c -o
liblua51_a-lapi.o `test -f 'luatexdir/lua51/lapi.c' || echo
'../.../source/texk/web2c/'`luatexdir/lua51/lapi.c
```

to

```
gcc -DHAVE_CONFIG_H -I. -I../.../source/texk/web2c
-I../... -I/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/texk
-I/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/source/texk
-I../.../source/texk/web2c/luatexdir/lua51 -DLUA_USE_LINUX -g -O2
-Wno-write-strings
-Wdeclaration-after-statement
-MT liblua51_a-lapi.o
-MD -MP -MF .deps/liblua51_a-lapi.Tpo -c -o liblua51_a-lapi.o `test -f
'luatexdir/lua51/lapi.c' || echo
'../.../source/texk/web2c/'`luatexdir/lua51/lapi.c
```

(about 30 lines) ,

then "hide" and recompile luatex:

```
/bin/bash ./libtool --tag=CXX --mode=link ./CXXLD.sh -g -O2
-Wno-write-strings -fvisibility=hidden -fvisibility-inlines-hidden -o
luatex luatex-luatex.o libluatex.a libff.a libluamisc.a libzzip.a
libluasocket.a liblua51.a
/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/libs/libpng/libpng.a
/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/libs/zlib/libz.a
/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/libs/xpdf/libxpdf.a
/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/libs/
obsdcompat/libopenbsd-compat.a libmd5.a libmplib.a lib/lib.a
/opt/luatex/luatex-lunatic/luatex-snapshot-0.42.0/build/texk/kpathsea/libkpath-
sea.la
-lm -Wl,-E -uluaL_openlibs -fvisibility=hidden
-fvisibility-inlines-hidden -ldl
```

It's a manual operation, about ~20 min. of editing.
(I have collected all these stuff in `trick.sh`)

After that,
we are ready to build Lunatic python and load python
interpreter with a wrapper python.lua like this:

```
loaded = false
func = package.loadlib(
"/opt/luatex/luatex-lunatic/lib/python2.6/
site-packages/python.so",
"luaopen_python")
  if func then
    func()
    return
  end
if not loaded then
  error("unable to find python module")
end
```

and test test.tex

```
% test.tex
\directlua{require "python";sys = python.import("sys");
tex.print(tostring(sys.version_info))}
\bye
$> luatex --fmt=plain --output-format=pdf test.tex
```

First conclusion:

1. constrains of any sort to development team ✓
2. massive code modifications ✓
3. radical modification of building process ✗

And Now for Something Completely Different ...

.....Intermezzo: Why do we do it ?

luaT_EX + ConT_EXt-mkiv is already a powerful tool for publishing content !

Because we gain

- +power with Lua loadlib (remember, it's enabled in luatex lunatic...)
- ++power with python in lua (Python has more bindings than Lua...)

It looks attractive expecially with MetaT_EX

But we lose

- **portability** (I have no idea about Windows[®]; and what about differents Linux distros ?)
- **stability** (Python 2.6.1, 2.7alpha..., 3.1.1, 3.2alpha...and it's only about python)

.....and why I didn't done it!

A python interpreter does not "complete" in any sense luatex, because Lua is a perfect choice: it's small, stable, and OS-aware (it' based on "C" operating system).

In the end, an existing python binding is attractive if is stable, rich, complete and mature respect to an existing lua binding, or if there is not a lua binding (and you don't want to build one by yourself)

Gallery

```
import sys
import fontforge
class simpledraw(object):
    def __init__(self,font_file):
        self.font = fontforge.open(font_file)
    def getcurve(self,letter):
        self.glname = letter
        res_Array = []
        res = dict()
        try :
            g = self.font[letter]
        except Exception ,e :
            res['err'] = str(e)
            res_Array.append(res)
```

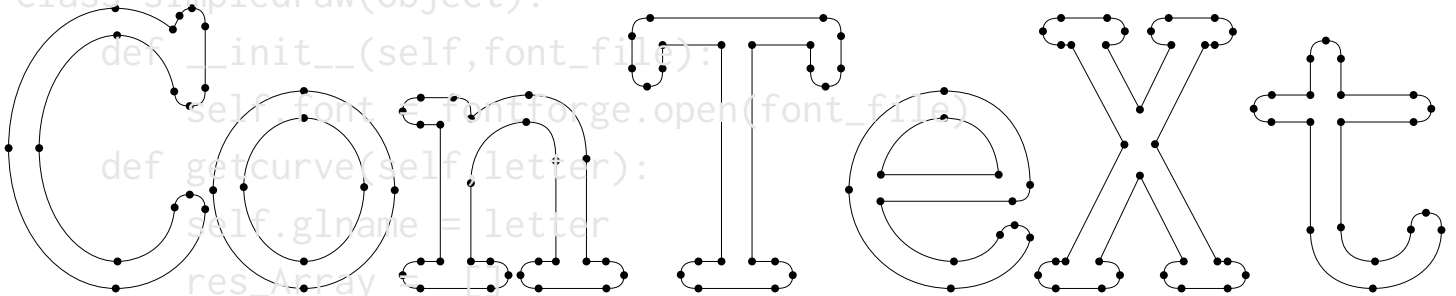


Image Processing:ImageMagick®

```
%% here some setups ...
```

```
%%
```

```
%%
```

```
%% lua layer
```

```
%%
```

```
\startluacode
```

```
function testimagemagick(box,t)
```

```
    local w, h, d, f
```

```
    local res = 118.110 -- 300 dpi
```

```
    local opacity = 25
```

```
    local sigma = 15
```

```
    local x = 10
```

```
    local y = 10
```

```
    w = math.floor((tex.wd[box]/65536 )
```

```
        /72.27*2.54*res)
```

```
    h = math.floor(((tex.ht[box]/65536)+
```

```
        (tex.dp[box]/65536))
```

```
        /72.27*2.54*res)
```

```
    f = string.format("%s.png",t)
```

```
--
```

```
-- Call python interpreter
```

```
--
```

```
require("python")
```

```
pmw = python.import("PythonMagickWand")
```

```
wand = pmw.NewMagickWand()
```

```
background = pmw.NewPixelWand(0)
```

```
pmw.MagickNewImage(wand,w,h,background)
```

```
pmw.MagickSetImageResolution(wand,res,res)
```

```
pmw.MagickSetImageUnits(wand,
```

```
    pmw.PixelsPerCentimeterResolution)
```

```
pmw.MagickShadowImage(wand,opacity,sigma,x,y)
```

```
pmw.MagickWriteImage(wand ,f)
```

```
end
```

```
\stopluacode
```

```
%%
```

```
%% Python code
```

```
%%
```

```
%% import PythonMagickWand as pmw
```

```
%% pmw.MagickWandGenesis()
```

```
%% wand = pmw.NewMagickWand()
```

```
%% background = pmw.NewPixelWand(0)
```

```
%% pmw.MagickNewImage(wand,200,200,background)
```

```
%% pmw.MagickSetImageResolution(wand,118.110,118.110)
```

```
%% pmw.MagickSetImageUnits(wand,
```

```
    pmw.PixelsPerCentimeterResolution)
```

```
%% pmw.MagickShadowImage(wand,90,3,2,2)
```

```
%% pmw.MagickWriteImage(wand,"out.png")
```

```

%% TeX layer
%%
\def\testimagemagick[#1]{%
\getparameters[Imgk][#1]%
\ctxlua{%
    testimagemagick(\csname Imgkbox\endcsname,
        "\csname Imgkfilename\endcsname")}%
}
%% ConTeXt layer
%%
\newcount\shdw
\long\def\startShadowtext#1\stopShadowtext{%
\bggroup%
\setbox0=\vbox{#1}%
\testimagemagick[box=0,
    filename={shd-\the\shdw}]{%
\defineoverlay[backg]{%
    [{\externalfigure[shd-\the\shdw.png]}]}%
\framed[background=backg,
    frame=off,offset=4pt]{\box0}%
\global\advance\shdw by 1%
\egroup}%
%%
%% Main program
\starttext
\startTEXpage%
\startShadowtext%
\input tufte
\stopShadowtext%
\stopTEXpage
\stoptext

```



We thrive in information-thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeon-hole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsise, winnow the wheat from the chaff and separate the sheep from the goats.

Image Processing:PIL

```
\startluacode
function testPIL(imageorig,imagesepia)
  require("python")
  PIL_Image = python.import("PIL.Image")
  PIL_ImageOps = python.import("PIL.ImageOps")
  python.execute([[
def make_linear_ramp(white):
  ramp = []
  r, g, b = white
  for i in range(255):
    ramp.extend((r*i/255, g*i/255, b*i/255))
  return ramp
]])
-- make sepia ramp
-- (tweak color as necessary)
sepia = python.eval("make_linear_ramp((255,
240, 192))")
im = PIL_Image.open(imageorig)
-- convert to grayscale
if not(im.mode == "L")
  then
    im = im.convert("L")
  end
end

-- optional: apply contrast
-- enhancement here, e.g.
im = PIL_ImageOps.autocontrast(im)
-- apply sepia palette
im.putpalette(sepia)
-- convert back to RGB for JPEG/PNG
im = im.convert("RGB")
im.save(imagesepia)
end
\stopluacode
%%
%% ConTeXt layer
\def\SepiaImage#1#2{%
\ctxlua{testPIL("#1","#2")}%
\startcombination[1*2]
{\externalfigure[#1][width=512pt]}{\ss
Orig.}
{\externalfigure[#2][width=512pt]}{\ss
Sepia}
\stopcombination
}
```

```
%%  
%% Main program  
\starttext  
\startTEXpage  
\SepiaImage{lenna.png}{lenna-sepia.png}  
\stopTEXpage  
\stoptext
```



Orig.



Sepia

Language adapter:Postscript

```
\startluacode
function testgs(epsin,pdfout)
  require("python")
  gsmodule = python.import("testgs")
  ghost = gsmodule.gs()
  ghost.appendargs('-q')
  ghost.appendargs('-dNOPAUSE')
  ghost.appendargs('-dEPSCrop')
  ghost.appendargs('-sDEVICE=pdfwrite')
  ghost.InFile = epsin
  ghost.OutFile = pdfout
  ghost.run()
end
\stopluacode

\def\epstopdf#1#2{\ctxlua{testgs("#1","#2")}}
\def\EPSfigure[#1]{%lazy way to load eps
\epstopdf{#1.eps}{#1.pdf}%
\externalfigure[#1.pdf]%
}
\starttext
\startTEXpage
{\EPSfigure[golfer]}
{\ss golfer.eps}
\stopTEXpage
\stoptext
```



golfer.eps

testgs.py "ad hoc" binding for ghostscript.

Language adapter:Postscript cont'ed .

```
\startluacode
function epstopdf(epsin,pdfout)..end
function barcode(text,type,options,savefile)
  require("python")
  gsmodule = python.import("testgs")
  barcode_string =
    string.format("%%!\\n100 100 moveto (%s) (%s)
                  %s barcode showpage",
                  text,options,type)
  psfile = string.format("%s.ps",savefile)
  epsfile = string.format("%s.eps",savefile)
  pdffile = string.format("%s.pdf",savefile)
  temp = io.open(psfile,'w')
  print(psfile)
  temp:write(tostring(barcode_string),"\\n")
  temp:flush()
  io.close(temp)
  ghost = gsmodule.gs()
  ghost.rawappendargs('-q')
  ghost.rawappendargs('-dNOPAUSE')
  ghost.rawappendargs('-sDEVICE=epswrite')
  ghost.rawappendargs(
    string.format('-sOutputFile=%s',epsfile))
  ghost.rawappendargs('barcode.ps')
  ghost.InFile= psfile
  ghost.run()
end
\stopluacode

%%
%% ConTeXt layer
\def\epstopdf#1#2{\ctxlua{epstopdf("#1","#2")}}
\def\EPSfigure[#1]{%lazy way to load eps
\epstopdf{#1.eps}{#1.pdf}%
\externalfigure[#1.pdf]%
}

\def\PutBarcode[#1]{%
\getparameters[bc][#1]%
\ctxlua{barcode("\csname bctext\endcsname",
               "\csname bctype\endcsname",
               "\csname bcoptions\endcsname",
               "\csname bcsavefile\endcsname"
)}}%
\expanded{\EPSfigure[\csname bcsavefile\end-
csname]}%
}
```

```

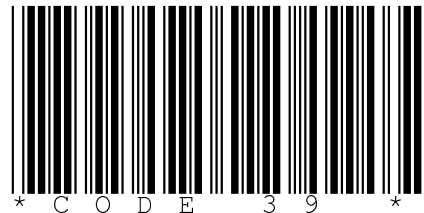
\starttext
\startTEXpage
{\PutBarcode[text={CODE 39},type={code39},
               options={includecheck includetext},
               savefile={TEMP1}]]\\

{\ss code39}
\blank
{\PutBarcode[text={CONTEXT},type={code93},
               options={includecheck includetext},
               savefile={TEMP2}]]\\

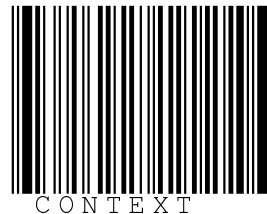
{\ss code93}
\blank
{\PutBarcode[text={977147396801},type={ean13},
               options={includetext},
               savefile={TEMP3}]]\\

{\ss ean13}
\stopTEXpage
\stoptext

```



code39



code93



ean13

they look good, but once printed they are ugly ...

Scientific & math extension: Sagemath

```
\startluacode
function test_ode(graphout)
  require("python")
  pg = python.globals()
  SAGESTUB = python.import("sagestub")
  sage = SAGESTUB.sage
  python.execute([[
def f_1(t,y,params):
  return[y[1],
        -y[0]-params[0]*y[1]*(y[0]**2-1)]
]])
python.execute([[
def j_1(t,y,params):
  return [ [0,1.0],
           [-2.0*params[0]*y[0]*y[1]-1.0,
            -params[0]*(y[0]*y[0]-1.0)], [0,0]
]
]])

T=sage.gsl.ode.ode_solver()
T.algorithm="rk8pd"
f_1 = pg.f_1
j_1 = pg.j_1
pg.T=T
python.execute("T.function=f_1")
T.jacobian=j_1

python.execute("T.ode_solve(y_0=[1,0],
                          t_span=[0,100],
                          params=[10],num_points=1000)")
python.execute(string.format(
  "T.plot_solution(filename='%s')",
  graphout ))
end
\stopluacode
```

```

\def\TestODE#1{%
\ctxlua{test_ode("#1")}%
\startcombination[1*2]
{%
\ vbox{\ hsize=8cm
Consider solving the Van der pol oscillator
 $x''(t) + ux'(t)(x(t)^2 - 1) + x(t) = 0$ 
between  $t=0$  and  $t= 100$ .
As a first order system it is
 $x'=y$ 
 $y'=-x+uy(1-x^2)$ 
Let us take  $u=10$  and use
initial conditions  $(x,y)=(1,0)$  and use the
\emphsl{\ hbox{runge-kutta} \ hbox{prince-dormand}}
algorithm.
}%
}{\ss \ }
{\externalfigure[#1][width=9cm]}{\ss Result for
1000 points}
\stopcombination}

\starttext
\startTEXpage
\TestODE{ode1.pdf}
\stopTEXpage
\stoptext

```

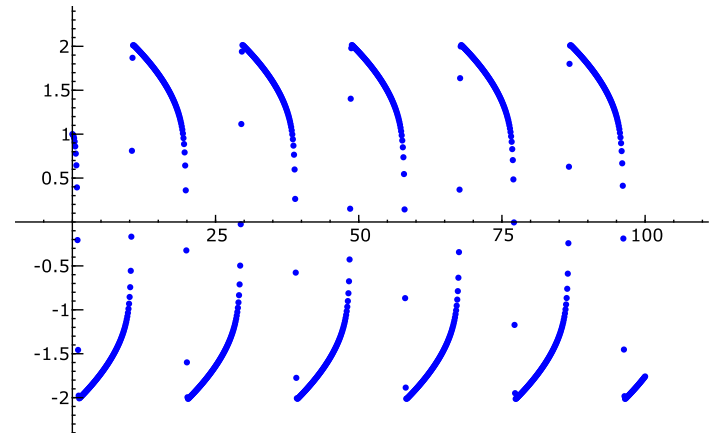
Consider solving the Van der Pol oscillator
 $x''(t) + ux'(t)(x(t)^2 - 1) + x(t) = 0$
between $t = 0$ and $t = 100$.

As a first order system it is

$$x' = y$$

$$y' = -x + uy(1 - x^2)$$

Let us take $u = 10$ and use initial conditions $(x,y) = (1,0)$ and use the *Runge-Kutta Prince-Dormand* algorithm.



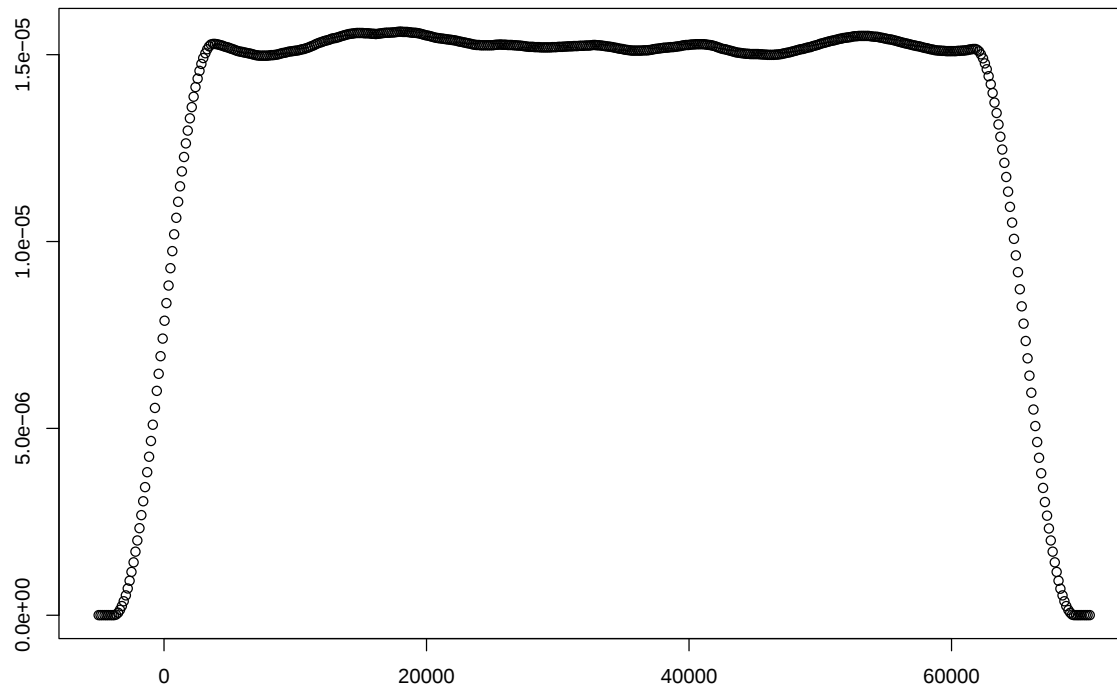
Result for 1000 points

Scientific & math extension: R

```
# test-R
import rpy2.robjects as robjects
import rpy2.rinterface as rinterface
class density(object):
    def __init__(self,samples,outputpdf,w,h,kernel):
        self.samples = samples
        self.outputpdf= outputpdf
        self.kernel = kernel
        self.width=w
        self.height=h
    def run(self):
        r = robjects.r
        data = [int(k.strip())
                for k in
                file(self.samples,'r').readlines()]
        x = robjects.IntVector(data)
        r.pdf(file=self.outputpdf,
              width=self.width,
              height=self.height)
        z = r.density(x,kernel=self.kernel)
        r.plot(z[0],z[1],xlab='',ylab='')
        r['dev.off']()
if __name__ == '__main__' :
    dens =
        density('u-random-int','test-001.pdf',
                10,7,'o')
    dens.run()
```

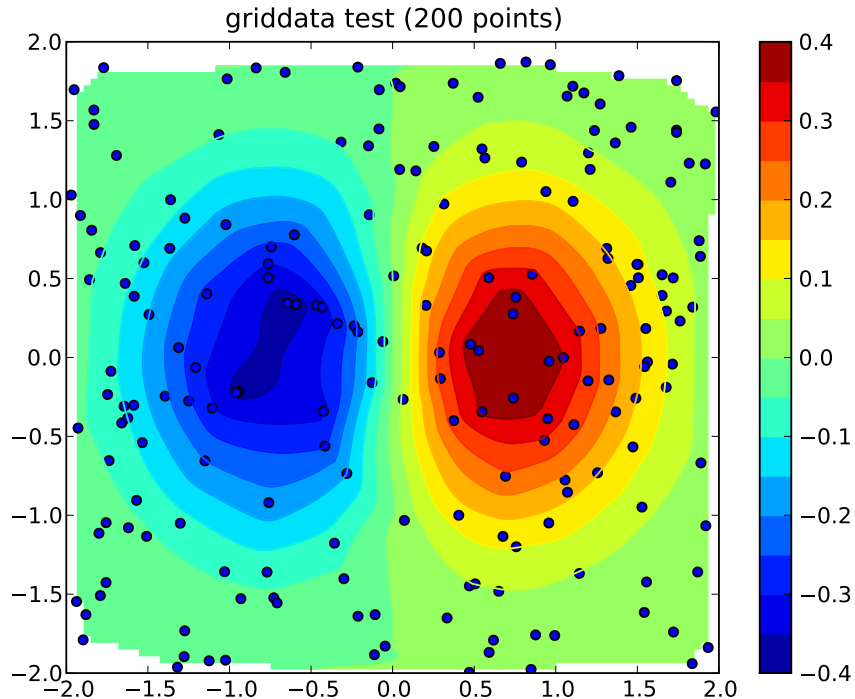
```
%% luatex
\startluacode
function testR(samples,outputpdf,w,h,kernel)
    require("python")
    pyR = python.import("test-R")
    dens =
        pyR.density(samples,outputpdf,w,h,kernel)
    dens.run()
end
\stopluacode
%% ConTeXt Layer
\def\plotdensity[#1]{%
\getparameters[R][#1]%
\expanded{\ctxlua{testR("\Rsamples",
"\Routputpdf","\Rwidth","\Rheight","\Rkernel")}}}%
% Main
\starttext
\startTEXpage
\plotdensity[samples={u-random-int},kernel={o},
outputpdf={test-001.pdf},width={10},height={7}]
\startcombination[1*2]
{\vbox{\hsize=400bp
This is a density plot of around {\tt 100000}
random numbers between $0$ and $2^{16}-1$
generated from {\tt \hbox{/dev/urandom}}}}{\}
{\externalfigure[test-001.pdf]
[width={400bp}]}{\}
\stopcombination
\stopTEXpage
\stoptext
```

This is a density plot of around 100 000 random numbers between 0 and $2^{16} - 1$ generated from `/dev/urandom`



It's worth to noting that rpy2 actually is included in Sage too.
GuitMeeting 2009

Scientific & math extension: scipy

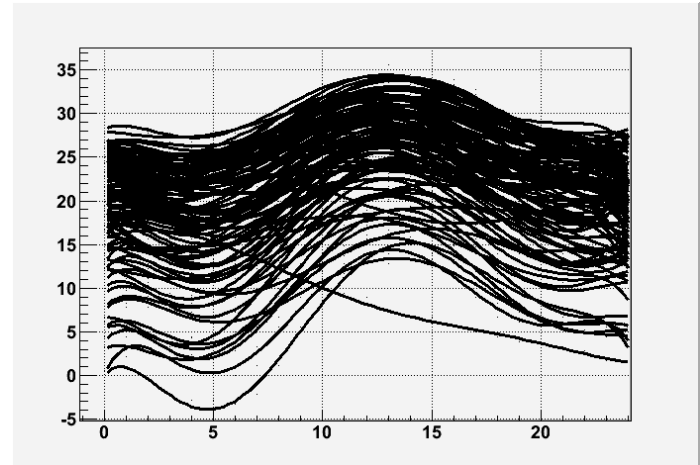


Scientific & math extension: ROOT

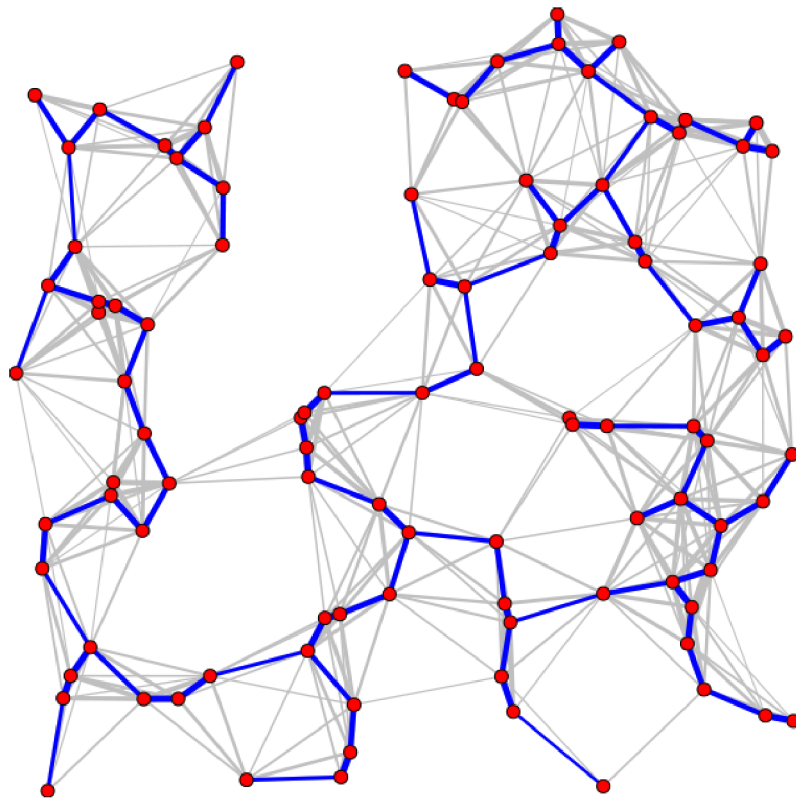
```
#
#test-ROOT1
#
from ROOT import TCanvas,
    TGraph ,TGraphErrors,TMultiGraph
from ROOT import gROOT
from math import sin
from array import array
def run(filename):
    c1 = TCanvas("c1","multigraph",200,10,700,500)
    c1.SetGrid()
    mg = TMultiGraph()
    n = 24;
    x = array('d',range(24))
    data = file('data').readlines()
    for line in data:
        line = line.strip()
        y = array('d',
            [float(d) for d in line.split()])
        gr = TGraph(n,x,y)
        gr.Fit("pol6","q")
        mg.Add(gr)
    mg.Draw("ap")
    c1.Update()
    c1.Print(filename)
```

Scientific & math extension: ROOT

```
%%  
%% ConTeXt  
\startluacode  
function test_ROOT(filename)  
  require("python")  
  test = python.import('test-ROOT1')  
  test.run(filename)  
end  
\stopluacode  
\starttext  
\startTEXpage  
\ctxlua{test_ROOT("data.pdf")}  
\rotate[rotation=90]{\externalfigure[data.pdf]}  
\stopTEXpage  
\stoptext
```

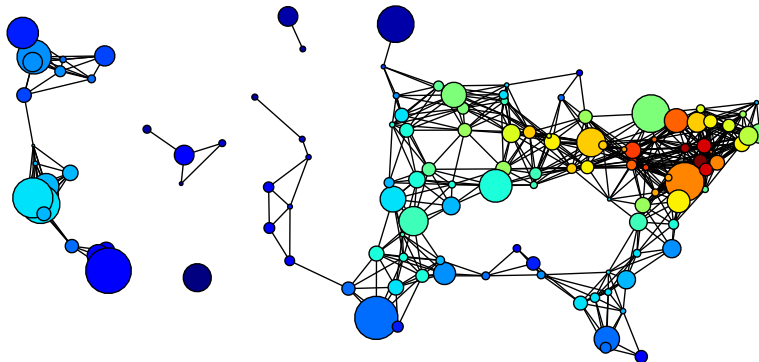


Graph: igraph



Minimum spanning tree
GuitMeeting 2009

Graph: networkx



`miles_graph()` returns an undirected graph over the 128 US cities from the datafile `miles_dat.txt`. The cities each have location and population data. The edges are labeled with the distance between the two cities.

This example is described in Section 1.1 in Knuth's book [1,2].

References.

- [1] Donald E. Knuth,
"The Stanford GraphBase: A Platform for Combinatorial Computing",
ACM Press, New York, 1993.
- [2] <http://www-cs-faculty.stanford.edu/~knuth/sgb.html>

Database: Oracle Berkeley DB XML

```
\usetypescriptfile[type-gentium]
\usetypescript[gentium]
\setupbodyfont[gentium,10pt]
\setuppapersize[A5][A5]
\setuplayout[height=middle,
topspace=1cm,header={2\lineheight},
footer=0pt,backspace=1cm,margin=1cm,
width=middle]
%%
%% DB XML
%%
\startluacode
function testdbxml(title,preamble,
                    postamble,filename)
    require("python")
    pg = python.globals()
    wikiversity =
        python.import("wikidbxml_queryTxn")
    wikiversity.writers(title,preamble,
                        postamble,filename)
end
\stopluacode
```

```
%%
%% ConTeXt
%%
\def\testdbxml[#1]{%
\getparameters[dbxml][#1]%
\ctxlua{%
testdbxml("\csname dbxmltitle\endcsname",
          "\csname dbxmlpreamble\endcsname",
          "\csname dbxmlpostamble\endcsname",
          "\csname dbxmlfilename\endcsname")}%
\input \csname dbxmlfilename\endcsname %
}
\starttext
\testdbxml[title={Primary mathematics/Numbers},
          preamble={},
          postamble={},
          filename={testres.tex}]
\stoptext
```

1. from 'MediaWiki-format' to XML-DocBook (more exactly docbook RELAX NG grammar 4.4)
2. from XML-DocBook to ConT_EXt-mkiv

1 Primary mathematics/Numbers

1.1 Primary mathematics/Numbers

1.1.1 Teaching Number

This page is for teachers or home-schoolers. It is about teaching the basic concepts and conventions of simple number.

1.1.1.1 Developing a sound concept of number

Children typically learn about numbers at a very young age by learning the sequence of words, "one, two, three, four, five" etc. Usually, in chanting this in conjunction with pointing at a set of toys, or mounting a flight of steps for example. Typically, 'mistakes' are made. Toys or steps are missed or counted twice, or a mistake is made in the chanted sequence. Very often, from these sorts of activities, and from informal matching activities, a child's concept of number and counting emerges as their mistakes are corrected. However, here, at the very foundation of numerical concepts, children are often left to 'put it all together' themselves, and some start off on a shaky foundation. Number concepts can be deliberately developed by suitable activities. The first one of these is object matching.

1.1.2 Matching Activities

As opposed to the typical counting activity children are first exposed to, matching sets of objects gives a firm foundation for the concept of number and numerical relationships. It is very important that matching should be a physical activity that children can relate to and build on.

Typical activities would be a toy's tea-party. With a set of (say) four toy characters, each toy has a place to sit. Each toy has a cup, maybe a saucer, a plate etc. Without even mentioning 'four', we can talk with the child about 'the right number' of cups, of plates etc. We can talk about 'too many' or 'not enough'. Here, we are talking about number and important number relations without even mentioning which number we are talking about! Only after a lot of activities of this type should we talk about specific numbers and the idea of number in the abstract.

1.1.3 Number and Numerals

Teachers should print these numbers or show the children these numbers. Ideally, the numbers should be handled by the student. There are a number of ways to achieve this: cut out numerals from heavy cardstock, shape them with clay together, purchase wooden numerals or give them sandpaper numerals to trace. Simultaneously, show the definitions of these numbers as containers or discrete quantities (using boxes and small balls, eg. 1 ball, 2 balls, etc. Note that 0 means "no balls"). This should take some time to learn thoroughly (depending on the student).

0 1 2 3 4 5 6 7 8 9

1.1.4 Place Value

The Next step is to learn the place value of numbers.

It is probably true that if you are reading this page you know that after 9 comes 10 (and you usually call it ten) but this would not be true if you would belong to another culture.

Take for example the Maya Culture where there are not the ten symbols above but twenty symbols.

cfr <http://www.michiel.nl/maya/math.html>

Imagine that instead of using 10 symbols one uses only 2 symbols. For example 0 and 1

Here is how the system will be created:

Binary	0	1	10	11	100	101	110	111	1000	...
Decimal	0	1	2	3	4	5	6	7	8	...

Or if one uses the symbols A and B one gets:

Binary	A	B	BA	BB	BAA	BAB	BBA	BBB	BAAA	...
Decimal	0	1	2	3	4	5	6	7	8	...

This may give you enough information to figure the place value idea of any number system.

For example what if you used 3 symbols instead of 2 (say 0,1,2).

Trinary	0	1	2	10	11	12	20	21	22	100	...
Decimal	0	1	2	3	4	5	6	7	8	9	...

If you're into computers, the HEXADECIMAL (Base 16) or Hex for short, number system will be of interest to you. This system uses 4 binary digits at a time to represent numbers from 0 to 15 (decimal). This allows for a more convenient way to express numbers the way computers think - that we can understand. So now we need 16 symbols instead of 2, 3, or 10. So we use 0123456789ABCDEF.

Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000	...
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	...

1.1.5 Resources for Early Math

15 Fun Ways and Easy Games for Young Learners Math: Reproducible, Easy-to-Play Learning Games That Help Kids Build Essential Math Skills, by Susan Julio, Scholastic Professional, 2001.

Enie Meenie Miney Math: Math Play for You and Your Preschooler, by Linda Allison, Little Brown & Co., 1993.

Marshmallow Math: Early Math for Toddlers, Preschoolers and Primary School Children, by Trevor Schindeler, Trafford, 2002.

Number Wonder: Teaching Basic Math Concepts to Preschoolers, by Deborah Saathoff and Jane Jarrell, Holman Bible, 1999.

cfr Category:School of Mathematics

cfr Category:Pages moved from Wikibooks

cfr Category:Primary education

Next in Primary School Mathematics:

cfr http://en.wikiversity.org/wiki/Primary_mathematics:Adding_numbers

Database: sqlite

```
%% some setups here ..
```

```
%%
```

```
%% sqlite
```

```
%%
```

```
\startluacode
```

```
function listtitles(title)
```

```
    require("python")
```

```
    pg = python.globals()
```

```
    wikiversity =
```

```
        python.import("wikidbxml_queryTxn")
```

```
    r = wikiversity.querycategory(title)
```

```
    local j = 0
```

```
    local res = r[j] or {}
```

```
    while res do
```

```
        local d =
```

```
            string.format("%s\\par",
```

```
                string.gsub(tostring(res),'_',' '))
```

```
            tex.sprint(tex.ctxcatcodes,d)
```

```
            j = j+1
```

```
            res = r[j]
```

```
        end
```

```
    end
```

```
\stopluacode
```

```
%%
```

```
%% sqlite
```

```
%%
```

```
\startluacode
```

```
function simplereports(title)
```

```
    require("python")
```

```
    pg = python.globals()
```

```
    wikiversity =
```

```
        python.import("wikidbxml_queryTxn")
```

```
    r = wikiversity.simplereports(title)
```

```
    local j = tonumber(r)
```

```
    for v = 0,j-1 do
```

```
        local d =
```

```
            string.format("\\input reps\\%04d ",v)
```

```
            tex.sprint(tex.ctxcatcodes,d)
```

```
        end
```

```
        print( j )
```

```
    end
```

```
\stopluacode
```

```
%%
```

```
%% ConTeXt
```

```
%%
```

```
\starttext
```

```
{\bfb Query for 'geometr':}
```

```
\ctxlua{listtitles("geometr")}%
```

```
\ctxlua{simplereports("geometr")}%
```

```
\stoptext
```

Query for 'geometr': Geometric algebra
Geometry
Introductory Algebra and Geometry
Orbital geometry
Coordinate Geometry

1 Geometric algebra

1.1 Geometric algebra

This category is for articles listed under "Geometric algebra"
cfr Category:Geometry
cfr de:Kategorie:Algebraische Geometrie

2 Geometry

2.1 Geometry

This subdivision is dedicated to bridging the gap between the mathematical layperson and the student who is ready to learn calculus and higher mathematics or to take on any other endeavour that requires an understanding of basic algebra and (at least Euclidean) geometry.

2.1.1 Subdivision news

2.1.2 Departments

2.1.3 Active participants

The histories of Wikiversity pages indicate who the active participants are. If you are an active participant in this subdivision, you can list your name here (this can help small subdivisions grow and the participants communicate better; for large subdivisions a list of active participants is not needed). Please remember: if you have an

cfr <http://en.wikiversity.org/w/index.php?title=Special:Userlogin&type=signup>

cfr Category:Geometry
cfr Category:introductions

cfr \#
cfr \#

In Cartesian or Analytic Geometry we will learn how to represent points, lines, and planes using the Cartesian Coordinate System, also called Rectangular Coordinate System. This can be applied to solve a broad range of problems from geometry to algebra and it will be very useful later on Calculus.

2.1.4 Cartesian Coordinate System

The foundations of Analytic Geometry lie in the search for describing geometric shapes by using algebraic equations. One of the most important mathematicians that helped to accomplish this task was René Descartes for whom the name is given to this exciting subject of mathematics.

2.1.4.1 The Coordinate System

For a coordinate system to be useful we want to give to each point an attribute that help to distinguish and relate different points. In the Cartesian system we do that by describing a point using the intersection of two(2D Coordinates) or more(Higher Dimensional Coordinates) lines. Therefore a point is represented as $P(x_1, x_2, x_3, \dots, x_n)$ in "n" dimensions.

2.1.5 Licensing:

"Geometry is the only science that it hath pleased God hitherto to bestow on mankind."-Thomas Hobbes

This department of the Olympiad Mathematics course focuses on problem-solving based on circles and vectors, thus generalizing to Coordinate Geometry. Our major focus is on Rectangular (Cartesian) Coordinates, although the course does touch upon Polar coordinates.

The first section is based on the geometric study of circles. Although not based on pure analytical geometry, it uses Apollonius-style reference lines in addition to Theorems on Tangents, Areas, etc.

The second section is devoted to Vector Analysis, covering problem-solving from Lattices and Affine Geometry to Linear Algebra of Vectors

Third section, focusing on locus problems, is all about conic sections and other curves in the Cartesian plane.

2.1.6 Textbooks

2.1.7 Practice Questions

2.1.8 Related WebApps

2.1.9 Chapters

2.1.10 Active participants

The histories of Wikiversity pages indicate who the active participants are. If you are an active participant in this division, you can list your name here (this can help small divisions grow and the participants communicate better; for large divisions a list of active participants is not needed).

#REDIRECT

cfr \#

POST YOUR QUESTIONS ON THIS TALKPAGE AND GET ANSWERS FROM OTHER VIEWERS. ALSO WE CAN DISCUSS SOLUTIONS AND DISAGREEMENTS.

2.1.11 Congruency

Two things are congruent if they are the same. Okay, onto next topic... Wait a second; we didn't define that very well! That's okay. Congruency is a thing that is defined for everything that is used in geometry. For example, congruency can be defined for peanut butter jars

Two jars of peanut butter are congruent if and only if they have jars of the same shape, are filled to the same height with peanut butter, and the peanut butter tastes the same for both jars.

You could do a similar thing for gumballs, but that wouldn't be relevant to geometry, unlike peanut butter jars. Every element of geometry has its own definition of congruence.

This is the symbol of congruence.

Now for some congruency definitions!

Two line segments are congruent if and only if they have equal lengths.

Two angles are congruent if and only if their measures of the angles are equal.

Two points, lines, rays, or planes have no congruency definition.

Actually, those three definitions will do for now!

cfr Category:Geometry

cfr \#

2.1.12 Conditionals

If it is sunny, then I can play outside. If there are clouds, then it will rain soon. If you add sodium to water, then you will create an explosion. If a statement contains if and then, then it is called a conditional.

A conditional is simply an if/then statement. If you accept the part after the "if", also called the hypothesis, then you must accept the statement after the "then", also known as the conclusion. So, all conditionals are in the form

If hypothesis, then conclusion

You may recall using these statements in science class with the scientific method. However, remember that only the statement after the "if" is called the hypothesis. The whole statement is just called the conditional. Remember, if you accept the hypothesis of a conditional to be true, then the conclusion must also be true. Now, take a look at some of the old postulates. You'll notice that they can be reworded into conditionals. For example, the postulate which says Through any two points there is only one line can be read as if there are two points, then there is a unique line through the points. Yes, that is awkward wording, but it shows that postulates can be written as conditionals.

If there are three collinear points A, B, and C, and B is between A and C, then $AB + BC = AC$. If there are three points, then there is at least one plane through all three points. If there is a line, then there are at least two points on that line.

There are also some new conditionals which we will introduce in 3, 2, 1...

2.1.13 Conditionals which have traveled from algebra to geometry

The following conditionals are always true. It will make sense if, while you read them, you shout, "These are an insult to my intelligence!" for in fact, that's what they'll seem like.

The first one is called the reflexive property.

If a is a number, then $a=a$.

The second one is called the symmetric property.

If $a=b$, then $b=a$.

The third one is called the transitive property.

If $a=b$ and $b=c$, then $a=c$.

Now, explanations of the conditionals. The first one says any number equals itself. Strangely enough, this incredibly obvious statement will be a common sight when you write proofs. The second one says that equations can be reversed. In other words the statement $x=4$ is equivalent to the statement $4=x$. The third statement says that if two numbers are equal to the same number, then they are equal. Thus, if $x=4$ and $y=4$, then $x=y$.

The next four conditionals are very similar to each other.

The first one is called the addition property of equality.

If $a=b$ and $c=d$, then $a+c=b+d$.

The second one is called the subtraction property of equality.

If $a=b$ and $c=d$, then $a-c=b-d$.

The third one is called the multiplication property of equality. (Are you seeing the pattern?)

If $a=b$ and $c=d$, then $ac=bd$.

The fourth one is called the division property of equality.

If $a=b$ and $c=d$ and neither c nor d equal 0, then $a/c=b/d$.

Now, all of these properties are very similar, as you probably can tell; for example, if $x=2$ and $y=3$, then

You may notice that the division property of equality has an extra part in its hypothesis that isn't in any other property of equality. (Remember the name hypothesis? If hypothesis, then conclusion. If the hypothesis is accepted as true, then the conclusion must be accepted as true as well.) That is because division is undefined when the denominator is 0. $1/0$ doesn't exist! Neither does $4/0$, $-200/0$, or $0/0$. The extra part of the hypothesis is to prevent division by 0.

The final conditional we will look at today is known as the substitution property, and it is incredibly useful in proofs.

If two values are equal, then they may substitute for each other.

For example, suppose we know $x=y$, and that $x=2+4$. Then we also know by the substitution property that $y=2+4$, since x and y can substitute for each other. It is very important that you remember the names of these conditionals. The four "properties of equality" should be easy to remember, since they are named for the arithmetic operation that they have in their property. The substitution property also should be easy to remember, since you substitute values. That just leaves the transitive, symmetric, and reflexive property. You can remember the Transitive property if you remember that it uses Three values. If you remember lines of symmetry, then you can say that the symmetric property says that equations have lines of symmetry right down their equals sign. As for reflexive? Well, you'll just have to remember it. Also note that you can use other properties from algebra, for example $a(b+c)=ab+ac$ by the distributive property.

2.1.14 Proof

Given:

Statements	Reasons
1.	1. Given
2. $AC=BD$	2. Definition of congruent segments
3. $AB+BC=AC$; $BC+CD=BD$	3. Segment addition postulate
4. $AB+BC=BC+CD$	4. Substitution property
5. $BC=BC$	5. Reflexive property
6. $AB=CD$	6. Subtraction property of equality
7.	7. Definition of congruent segments

2.1.15 See also

cfr Category:Geometry

cfr Category:Mathematics

Hello, and welcome to Wikiversity's online geometry course!

2.1.16 Syllabus/Curriculum

2.1.17 Lessons

2.1.18 Homework assignments

2.1.19 Tests, quizzes, and exams

cfr Category:Mathematics
 cfr Category:Geometry
 cfr es:Geometria
 cfr fr:Département:Géométrie
 cfr it:Materia:Geometria
 cfr \#
 cfr Category:Mathematics
 cfr de:Kategorie:Geometries
 cfr fr:Catégorie:Géométrie
 cfr it:Categoria:Geometria
 cfr pt:Categoria:Geometria
 cfr \#
 cfr http://de.wikibooks.org/wiki/Geometry/Chapter_5

2.1.20 Triangle Congruency

2.1.20.1 Triangle Congruency Postulates

1) The SSS Triangle Congruency Postulate

Postulate: Every SSS (Side-Side-Side) correspondence is a congruence.

2) The SAS Triangle Congruency Postulate

Postulate: Every SAS (Side-Angle-Side) correspondence is a congruence.

Note: It is important that the angle is included; that is, it is adjacent to the two congruent sides.

3) The ASA Triangle Congruency Postulate

Postulate: Every ASA (Angle-Side-Angle) correspondence is a congruence.

Note: It is important that the side is included; that is, it is adjacent to the two congruent angles.

2.1.21 Congruency vs. Similarity

Congruency is, simply put, when two things have the same size and the same shape. Similarity, however, has only part of that definition: two things are similar if they have the same shape, but not necessarily the same size. Any two circles, for example, are similar, because they have the same shape, but one circle can be huge and another small. The same goes for all squares and equilateral triangles.

2.1.22 Triangle Similarity

2.1.22.1 Triangle Similarity Theorems

1) The AA Triangle Similarity Theorem

Theorem: Every AA (Angle-Angle) correspondence is a similarity.

2) The SSS Triangle Similarity Theorem

Welcome to the Lesson of Trilaterometry

Trilaterometry or Triangle Geometry refers to the aspect of plane geometry which deals with triangles, which are the the most basic polygon with only 3 (minimum possible) sides. Its importance can be justified by the fact that all polygons can cut into triangles with all its vertices being vertices of original polygon.

2.1.23 Theorems

2.1.23.1 Area of a Triangle

Although simple, this formula is only useful if the height can be readily found. For example, the surveyor of a triangular field measures the length of each side, and can find the area from his results without having to construct a 'height' with the Heron's Formula. The shape of the triangle is determined by the lengths of the sides alone. Therefore the area s also can be derived from the lengths of the sides. By Heron's formula:

where $s = \frac{1}{2}(a + b + c)$ is the semiperimeter, or half of the triangle's perimeter.

Three equivalent ways of writing Heron's formula are

Heron's formula is a special case of Brahmagupta's Theorem.

2.1.24 Intercept Theorem

The intercept theorem is an important theorem in elementary geometry about the ratios of various line segments, that are created if 2 intersecting lines are intercepted by a pair of parallels. It is equivalent to the theorem about ratios in similar triangles. Traditionally it is attributed to Greek mathematician Thales, which is the reason why it is named theorem of Thales in some languages.

Theorem and Proof:

2.1.25 Pythagorean Theorem

Proof 1:

Let ABC represent a right triangle, with the right angle located at C, as shown on the figure. We draw the altitude from point C, and call H its intersection with the side AB. The new triangle ACH is similar to our triangle ABC, because they both have a right angle (by definition of the altitude), and they share the angle at A, meaning that the third angle will be the same in both triangles as well. By a similar reasoning, the triangle CBH is also similar to ABC. The similarities lead to the two ratios.:

As

Proof 2 (as given in Baudhayana Sulbasutra):

The A-side angle and B-side angle of each of these triangles are complementary angles, so each of the angles of the blue area in the middle is a right angle, making this area a square with side length C. The area of this square is C^2 . Thus the area of everything together is given by:

2.1.26 Angle Bisector Theorem

Consider a triangle ABC. Let the angle bisector of angle A intersect side BC at a point D. The angle bisector theorem states that the ratio of the length of the line segment BD to the length of segment DC is equal to the ratio of the length of side AB to the length of side AC

The generalized angle bisector theorem states that if D lies on BC, then

This reduces to the previous version if AD is the bisector of BAC.

Proof of generalization:

Let B1 be the base of altitude in the triangle ABD through B and let C1 be the base of altitude in the triangle ACD through C. Then, it is also true that both the angles DB1B and DC1C are right, while the angles B1DB and C1DC are congruent if D lies on the segment BC and they are identical otherwise, so the triangles DB1B and DC1C are similar (AAA), which implies that

2.1.27 Apollonius' Theorem

It states that given a triangle ABC, if D is any point on BC such that it divides BC in the ratio $m:n$ (or

special cases of the theorem:

In simpler words, in any triangle

2.1.28 Ceva's Theorem

2.1.29 Examples

2.1.30 Resources

2.1.30.1 Textbooks

2.1.31 Practice Questions

2.1.32 Related WebApps

2.1.33 Active participants

The histories of Wikiversity pages indicate who the active participants are. If you are an active participant in this division, you can list your name here (this can help small divisions grow and the participants communicate better; for large divisions a list of active participants is not needed).

If it is sunny, then I can play outside. If there are clouds, then it will rain soon. If you add sodium to water, then you will create an explosion. If a statement contains if and then, then it is called a conditional.

A conditional is simply an if/then statement. If you accept the part after the "if", also called the hypothesis, then you must accept the statement after the "then", also known as the conclusion. So, all conditionals are in the form

if hypothesis, then conclusion

You may recall using these statements in science class with the scientific method. However, remember that only the statement after the "if" is called the hypothesis. The whole statement is just called the conditional. Remember, if you accept the hypothesis of a conditional to be true, then the conclusion must also be true. Now, take a look at some of the old postulates. You'll notice that they can be reworded into conditionals. For example, the postulate which says Through any two points there is only one line can be read as if there are two points, then there is a unique line through the points. Yes, I know that is awkward wording, but it shows that postulates can be written as conditionals.

If there are three collinear points A, B, and C, and B is between A and C, then $AB+BC=AC$. If there are three points, then there is at least one plane through all three points. If there is a line, then there are at least two points on that line.

There are also some new conditionals which we will introduce in 3, 2, 1...

2.1.34 Conditionals which have traveled from algebra to geometry

The following conditionals are always true. It will make sense if, while you read them, you shout, "These are an insult to my intelligence!" for in fact, that's what they'll seem like.

The first one is called the reflexive property.

If a is a number, then $a=a$.

The second one is called the symmetric property.

If $a=b$, then $b=a$.

The third one is called the transitive property.

If $a=b$ and $b=c$, then $a=c$.

Now, explanations of the conditionals. The first one says any number equals itself. Strangely enough, this incredibly obvious statement will be a common sight when you write proofs. The second one says that equations can be reversed, AKA the statement $x+4$ is equivalent to the statement $4+x$. The third statement says that if two numbers are equal to the same number, then they are equal, AKA if $x+4$ and $y+4$, then $x=y$.

The next four conditionals are very similar to each other.

The first one is called the addition property of equality.

If $a=b$ and $c=d$, then $a+c=b+d$.

The second one is called the subtraction property of equality.

If $a=b$ and $c=d$, then $a-c=b-d$.

The third one is called the multiplication property of equality. (Are you seeing the pattern?)

If $a=b$ and $c=d$, then $ac=bd$.

The fourth one is called the division property of equality.

If $a=b$ and $c=d$ and neither c nor d equal 0, then $a/c=b/d$.

Now, all of these properties are very similar, as you probably can tell; for example, if $x=2$ and $y=3$, then

You may notice that the division property of equality has an extra part in its hypothesis that isn't in any other property of equality. (Remember the name hypothesis? If hypothesis, then conclusion. If the hypothesis is accepted as true, then the conclusion must be accepted as true as well.) That is because division is undefined when the denominator is 0. $1/0$ doesn't exist! Neither does $4/0$, $-200/0$, or $0/0$. The extra part of the hypothesis is to prevent division by 0.

The final conditional we will look at today is known as the substitution property, and it is incredibly useful in proofs.

luatex lunatic: that's all folks.

http://wiki.contextgarden.net/User:Luigi.scarso/luatex_lunatic

The lunatic is on the grass
The lunatic is on the grass
Remembering games and daisy chains and laughs
Got to keep the loonies on the path
-- Brain Damage,
The dark side of the Moon,
Pink Floyd 1970
Mr. luaT_EX hosts a Python,
and become a bit lunatic
-- Anonymous

$\text{T}_{\text{E}}\text{X}$

