

# Some frequently and not so frequently asked questions about R<sub>X</sub><sup>1</sup>

(January 20, 2004)

Gert-Ludwig Ingold  
<gert.ingold@physik.uni-augsburg.de>

## Contents

<b>1</b>	<b>General aspects of R<sub>X</sub></b>	<b>3</b>
1.1	Where do I get the latest version of R <sub>X</sub> ?	3
1.2	How can I determine the version of R <sub>X</sub> running on my machine?	3
1.3	Does R <sub>X</sub> run under my favorite operating system?	3
1.4	Under which versions of Python will R <sub>X</sub> run?	3
1.5	Where can I get help if my question is not answered in this FAQ?	3
<b>2</b>	<b>Python</b>	<b>3</b>
2.1	What is Python?	3
2.2	Where can I learn more about Python?	4
2.3	What do I need to import in order to use R <sub>X</sub> ?	4
2.4	What is a raw string and why should I know about it when using R <sub>X</sub> ?	4
<b>3</b>	<b>Plotting of graphs</b>	<b>4</b>
3.1	General aspects	4
3.1.1	How do I generate a graph from data as simply as possible?	4
3.1.2	How do I generate a graph of a function as simply as possible?	5
3.1.3	How can I stack graphs?	5
3.2	Axis properties	6
3.2.1	How do I specify the tick increment?	6
3.2.2	How do I get rid of the zero line?	6
3.3	Data properties	6
3.3.1	How do I choose the symbol?	6
3.3.2	How do I choose the color of the symbols?	6
3.3.3	How do I choose the line style?	7
<b>4</b>	<b>T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X</b>	<b>7</b>
4.1	General aspects	7
4.1.1	What is T <sub>E</sub> X/L <sub>A</sub> T <sub>E</sub> X and why do I need it?	7
4.1.2	I don't know anything about T <sub>E</sub> X and L <sub>A</sub> T <sub>E</sub> X. Where can I read something about it?	7
4.1.3	What is CTAN?	8
4.1.4	Is there support for ConT <sub>E</sub> Xt?	8
4.2	T <sub>E</sub> X and L <sub>A</sub> T <sub>E</sub> X commands useful for R <sub>X</sub>	8
4.2.1	How do I get a specific symbol with T <sub>E</sub> X or L <sub>A</sub> T <sub>E</sub> X?	8
4.3	T <sub>E</sub> X and L <sub>A</sub> T <sub>E</sub> X errors	8

---

<sup>1</sup>This version of the FAQ is intended for use with R<sub>X</sub> version 0.5.

4.3.1	Undefined control sequence <code>\usepackage</code>	8
4.3.2	Undefined control sequence <code>\frac</code>	8
4.3.3	Missing <code>\$</code> inserted	8
4.3.4	Font shape <code>'OT1/xyz/m/n'</code> undefined	9
4.3.5	File ... is not available or not readable	9
4.3.6	No information for font <code>'cmr10'</code> found in font mapping file	9
4.4	Fonts	10
4.4.1	I want to use a font other than computer modern roman	10
4.4.2	Can I use a TrueType font with $\LaTeX$ ?	10

## Acknowledgements

The following persons have in one way or the other contributed to the answers given in this FAQ:  
 Jörg Lehmann, Michael Schindler, André Wobst.

# 1 General aspects of PyX

## 1.1 Where do I get the latest version of PyX?

The current release of PyX (as well as older ones) is freely available from <http://pyx.sourceforge.net> where also a CVS repository with the latest patches can be found. Possibly older versions of PyX are also available as package for various Linux distributions: see, for instance, <http://packages.debian.org/testing/python/python-pyx.html> for information on the PyX package in Debian GNU/Linux, <http://packages.gentoo.org/ebuilds/?pyx-0.3.1> for a Gentoo Linux ebuild, and [http://www.suse.de/en/private/products/suse\\_linux/i386/packages\\_professional/python-pyx.html](http://www.suse.de/en/private/products/suse_linux/i386/packages_professional/python-pyx.html) for the PyX package in the SUSE LINUX professional distribution.

## 1.2 How can I determine the version of PyX running on my machine?

Start a python session (usually by typing python at the system prompt) and then type the following two commands (>>> is the python prompt)

```
>>> import pyx
>>> pyx.__version__
```

## 1.3 Does PyX run under my favorite operating system?

Yes, if you have installed Python ([↑2.1](#)) and T<sub>E</sub>X ([↑4.1.1](#)). Both are available for a large variety of operating systems so chances are pretty good that you will get PyX to work on your system.

## 1.4 Under which versions of Python will PyX run?

PyX is supposed to work with Python 2.0 and above. However, most of the development takes place under the current production version of Python (2.3.3 by the time of this writing) and thus PyX is better tested with this version. On the other hand, the examples and tests are verified to run with all Python versions 2.x. PyX will not work with Python 1.x due to missing language features.

The version of your Python interpreter can be determined by calling it with the option -V. Alternatively, you can simply start the interpreter and take a look at the startup message. Note that there may be different versions of Python installed on your system at the same time. The default Python version need not be the same for all users.

## 1.5 Where can I get help if my question is not answered in this FAQ?

The PyX sources contain a reference manual and a set of examples demonstrating various features of PyX. If the feature you are looking for is among them, using the appropriate part of the example code or adapting it for your purposes may help.

There is also a user discussion list about PyX which you can subscribe to at <http://lists.sourceforge.net/lists/listinfo/pyx-user>. The archive of the discussion list is available at [http://sourceforge.net/mailarchive/forum.php?forum\\_id=23700](http://sourceforge.net/mailarchive/forum.php?forum_id=23700).

Finally, it might be worth checking <http://pyx.sourceforge.net/pyxfaq.pdf> for an updated version of this FAQ.

# 2 Python

## 2.1 What is Python?

From [www.python.org](http://www.python.org):

Python is an *interpreted, interactive, object-oriented* programming language. It is often compared to Tcl, Perl, Scheme or Java.

Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface.

The Python implementation is portable: it runs on many brands of UNIX, on Windows, OS/2, Mac, Amiga, and many other platforms. If your favorite system isn't listed here, it may still be supported, if there's a C compiler for it. Ask around on [news:comp.lang.python](http://news.comp.lang.python) — or just try compiling Python yourself.

The Python implementation is **copyrighted** but **freely usable and distributable, even for commercial use**.

## 2.2 Where can I learn more about Python?

The place to start is [www.python.org](http://www.python.org) where you will find plenty of information on Python including tutorials.

## 2.3 What do I need to import in order to use $\text{\R}\text{\X}$ ?

It is recommended to begin your Python code with

```
from pyx import *
```

when using  $\text{\R}\text{\X}$ . This allows you for example to write simply `graph.graphxy` instead of `pyx.graph.graphxy`. The following modules will be loaded: `box`, `canvas`, `color`, `connector`, `data`, `deco`, `epsfile`, `graph`, `path`, `style`, `trafo`, `tex`, `text`, and `unit`.

For convenience, you might import specific objects of a module like in

```
from graph import graphxy
```

which allows you to write `graphxy()` instead of `graph.graphxy()`.

All code segments in this document assume that the import line mentioned in the first code snippet is present.

## 2.4 What is a raw string and why should I know about it when using $\text{\R}\text{\X}$ ?

The backslash serves in standard Python strings to start an escape sequence. For example `\n` corresponds to a newline character. On the other hand,  $\text{\TeX}$  and  $\text{\LaTeX}$ , which do the typesetting in  $\text{\R}\text{\X}$ , use the backslash to indicate the start of a command. In order to avoid the standard interpretation, the string should be marked as a raw string by prepending it by an `r` like in

```
c.text(0, 0, r"$\alpha\beta\gamma$")
```

# 3 Plotting of graphs

## 3.1 General aspects

### 3.1.1 How do I generate a graph from data as simply as possible?

Suppose that you have a data file `x.dat` containing values for  $x$  and  $y$  in two columns. Then the following code will do the job

```

from pyx import *

g = graph.graphxy(width=10)
g.plot(graph.data("x.dat", x=1, y=2))
g.writetofile("x")

```

graphxy creates a canvas (called `g` in this example) onto which the graph will be drawn and it sets the default behavior including the axis. There is, however, no default value for the width of the graph. In `plot` you have to specify the name of the data file and the columns from which the data should be taken. Finally, `writetofile` will generate the postscript file `x.eps` which you can view or print.

A minimal example is also provided in the R<sub>X</sub> distribution as `examples/graphs/minimal.py`.

### 3.1.2 How do I generate a graph of a function as simply as possible?

The following example will draw a parabola:

```

from pyx import *

g = graph.graphxy(width=10,
                  x=graph.linaxis(min=-2, max=2)
                  )

g.plot(graph.function("y=x**2"))

g.writetofile("x")

```

Most of the code has been explained in [↑3.1.1](#). The main difference is that here you need to specify minimum and maximum for the  $x$ -axis so that R<sub>X</sub> knows in which range to evaluate the function.

See [↑3.2.2](#) for an explanation of how one can suppress the zero lines.

Another, slightly more complex, example is also provided in the R<sub>X</sub> distribution as `examples/graphs/piaxis.py`.

### 3.1.3 How can I stack graphs?

R<sub>X</sub> always needs a canvas to draw on. One possibility therefore consists in creating a new canvas with

```
c = canvas.canvas()
```

and inserting the graphs into this canvas with `c.insert(...)`. Here, `...` has to be replaced by the name of the graph. Alternatively, the canvas created with `graph.graphxy` for one of the graphs can be used to insert the other graphs even if they will be positioned outside the first graph.

The second issue to address is positioning of the graphs. By specifying `xpos` and `ypos` when calling `graphxy`, you can define the position of a graph. Later on, the position and size of a graph `g` can be referred to as `g.xpos`, `g.ypos`, `g.width`, and `g.height` even if for example the height has never been specified explicitly but is only defined by a R<sub>X</sub> default.

The following example shows how to put graph `gupper` above graph `glower` on a canvas `c`:

```

from pyx import *
from graph import graphxy

c = canvas.canvas()

glower = graphxy(width=10)
glower.plot(...)
c.insert(glower)

```

```

gupper = graphxy(width=10, ypos=glower.ypos+glower.height+2)
gupper.plot(...)

c.insert(gupper)
c.writetofile(...)

```

where ... has to be replaced by the appropriate information like data and symbol specifications and the name of the output file. Here, `c.insert` is used to actually insert the subcanvasses for the graphs into the main canvas `c` and `c.writetofile` in the last line requests to write the contents of this canvas to a file.

## 3.2 Axis properties

### 3.2.1 How do I specify the tick increment?

In the partition of a linear axis, the increments associated with ticks, subticks etc. can be specified as argument of `linpart`. In the following example, ticks will be drawn at even values while subticks will be drawn at all integers:

```

tg = graph.graphxy(width=10,
                   x=graph.linaxis(min=1, max=10,
                                   part=graph.linpart(tickdist=[2,1]))
                   )

```

### 3.2.2 How do I get rid of the zero line?

The `axispainter` takes an argument `zeropathattrs` which defaults to an empty list. Setting the `axispainter` for the appropriate axis to `None` will remove the zero line like in this example:

```

myaxispainter = graph.axispainter(zeropathattrs=None)

g = graph.graphxy(width=10,
                  x=graph.linaxis(min=-5, max=5, painter=myaxispainter),
                  y=graph.linaxis(min=-5, max=5)
                  )

```

This will keep the horizontal zero line but discard the vertical one.

## 3.3 Data properties

### 3.3.1 How do I choose the symbol?

Suppose a graph called `g` has been initialized, e.g. by using `graph.graphxy`. Then, data and the style of their representation in the graph are defined by calling `g.plot` like in the following example in which filled circles are requested:

```

g.plot(graph.data("test.dat"),
       graph.symbol(graph.symbol.circle, symbolattrs=[deco.filled])
       )

```

### 3.3.2 How do I choose the color of the symbols?

Colors are not properties of the symbol as such and can therefore not be specified in `symbolattrs` directly. The color is rather related to the plotting of the symbol as defined by `deco.stroked` or `deco.filled`. With

```
graph.symbol(graph.symbol.circle,
             symbolattrs=[deco.stroked([color.rgb.red]),
                          deco.filled([color.rgb.green])])
)
```

you will obtain a circle filled in green with a red borderline.

### 3.3.3 How do I choose the line style?

If you do not want to use symbols, you can set the line style as in this example

```
g.plot(graph.data("test.dat"),
      graph.line(lineattrs=[style.linewidth.Thin])
)
```

where the linewidth is set.

If you also want to use symbols, you can set the line attributes together with the symbols. Extending the example in 3.3.1), you could use

```
g.plot(graph.data("test.dat"),
      graph.symbol(graph.symbol.circle,
                  symbolattrs=[deco.filled],
                  lineattrs=[style.linewidth.Thin, style.linestyle.dashed]
                )
)
```

to set the linewidth and to choose dashed lines.

## 4 T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X

### 4.1 General aspects

#### 4.1.1 What is T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X and why do I need it?

T<sub>E</sub>X is a high quality typesetting system developed by Donald E. Knuth which is available for a wide variety of operating systems. L<sup>A</sup>T<sub>E</sub>X is a macro package originally developed by Leslie Lamport which makes life with T<sub>E</sub>X easier, in particular for complex typesetting tasks. The current version of L<sup>A</sup>T<sub>E</sub>X is referred to as L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and offers e.g. improved font selection as compared to the older L<sup>A</sup>T<sub>E</sub>X 2.09 which should no longer be used.

All typesetting tasks in R<sub>X</sub> are finally handed over to T<sub>E</sub>X (which is the default) or L<sup>A</sup>T<sub>E</sub>X, so that R<sub>X</sub> cannot do without it. On the other hand, the capabilities of T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X can be used for complex tasks where both graphics and typesetting are needed.

#### 4.1.2 I don't know anything about T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. Where can I read something about it?

Take a look at CTAN (↑4.1.3) where in [CTAN:info](#) you may be able to find some useful information. There exists for example “A Gentle Introduction to T<sub>E</sub>X” by M. Doob ([CTAN:gentle/gentle.pdf](#)) and “The Not So Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>” ([CTAN:info/lshort/english/lshort.pdf](#)) by T. Oetiker et al. The latter has been translated into a variety of languages among them korean (which you will not be able to read unless you have appropriate fonts installed) and mongolian.

Of course, it is likely that these documents will go way beyond what you will need for generating graphics with R<sub>X</sub> so you don't have to read all of it (unless you want to use T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X for typesetting which can be highly recommended).

There exists also a number of FAQs on T<sub>E</sub>X at [CTAN:help](#).

### 4.1.3 What is CTAN?

CTAN is the Comprehensive TeX Archive Network where you will find almost everything related to TeX and friends. The main CTAN servers are [tug.ctan.org](http://tug.ctan.org), [dante.ctan.org](http://dante.ctan.org), and [cam.ctan.org](http://cam.ctan.org). A list of FTP mirrors can be found at [CTAN:CTAN.sites](http://CTAN:CTAN.sites).

In this FAQ, CTAN: refers to the root of an anonymous ftp CTAN tree, e.g. <ftp://ctan.tug.org/tex-archive/>, <ftp://ftp.dante.de/tex-archive/>, and <ftp://ftp.tex.ac.uk/tex-archive/>. The links to CTAN in this document point to one of these servers but you might consider using a FTP mirror closer to you in order to reduce traffic load.

### 4.1.4 Is there support for ConTeXt?

No, and as far as I know there no plans to provide it in the near future. Given the close ties between ConTeXt and MetaPost, ConTeXt users probably prefer to stick with the latter anyway.

## 4.2 TeX and LaTeX commands useful for R<sub>X</sub>

### 4.2.1 How do I get a specific symbol with TeX or LaTeX?

A list of mathematical symbols together with the appropriate command name can be found at [CTAN:info/symbols/math/symbols.ps](http://CTAN:info/symbols/math/symbols.ps). A comprehensive list containing more than 2500 symbols for use with LaTeX can be obtained from [CTAN:info/symbols/comprehensive/symbols-a4.pdf](http://CTAN:info/symbols/comprehensive/symbols-a4.pdf). In some cases it might be necessary to install fonts or packages available from CTAN (↑4.1.3).

## 4.3 TeX and LaTeX errors

### 4.3.1 Undefined control sequence \usepackage

The command `\usepackage` is specific to LaTeX. Since by default R<sub>X</sub> uses TeX, you have to specify the correct mode:

```
text.set(mode="latex")
```

### 4.3.2 Undefined control sequence \frac

The command `\frac` is only available in LaTeX. In TeX you should use `{a\over b}` in math mode to produce  $\frac{a}{b}$ . As an alternative you may ask for the LaTeX mode as explained in 4.3.1.

### 4.3.3 Missing \$ inserted

You have specified TeX- or LaTeX-code which is only valid in math mode. Typical examples are greek symbols, sub- and superscripts or fractions.

On the R<sub>X</sub> level, you can specify math mode for the whole string by using `text.mathmode` as in

```
c.text(0, 0, r"\alpha", text.mathmode)
```

Keep also in mind that the standard Python interpretation of the backslash as introducing escape sequences needs to be prevented ↑2.4.

On the TeX/LaTeX level you should enclose the commands requiring math mode in `$`'s. As an example, `$\alpha_i^j$` will produce  $\alpha_i^j$ . This allows you to specify math mode also for substrings. There exist other ways to specify math mode in TeX and LaTeX which are particularly useful for more complex typesetting tasks. To learn more about it, you should consult the documentation ↑4.1.2.

#### 4.3.4 Font shape ‘OT1/xyz/m/n’ undefined

You have asked to use font xyz which is not available. Make sure that you have this font available in Type1 format, i.e. there should be a file xyz.pfb somewhere. If your T<sub>E</sub>X system is TDS compliant (TDS=T<sub>E</sub>X directory structure, cf. [CTAN:tds/draft-standard/tds/tds.pdf](#)) you should take a look at the subdirectories of TEXMF/fonts/type1.

#### 4.3.5 File ... is not available or not readable

Such an error message might already occur when running the example file hello.py included in the R<sub>X</sub> distribution. Usually, the error occurs due to an overly restrictive umask setting applied when unpacking the tar.gz sources. This may render the file mentioned in the error message unreadable because the python distutil installation package doesn’t change the file permissions back to readable for everyone.

If the file exists, the problem can be solved by changing the permissions to allow read access.

#### 4.3.6 No information for font ‘cmr10’ found in font mapping file

Such an error message can already be encountered by simply running the example file hello.py included in the R<sub>X</sub> distribution. The likely reason is that the T<sub>E</sub>X system does not find the cmr fonts in Type1 format. R<sub>X</sub> depends on these fonts as it does not work with the traditional pk fonts which are stored as bitmaps.

Therefore, the first thing to make sure is that the cmr Type1 fonts are installed. In some T<sub>E</sub>X installations, the command `kpsewhich cmr10.pfb` will return the appropriate path if the cmr fonts exist in the binary Type1 format (extension pfb) required by R<sub>X</sub>. If the command does not work but the TeX system is TDS compliant (↑4.3.4), a look should be taken at TEXMF/fonts/type1/bluesky/cm where TEXMF is the root of the texmf tree.

If the Type1 fonts do not exist on the system, they may be obtained from the CTAN ↑4.1.3 at [CTAN:fonts/cm/ps-type1/bluesky](#). See the README for information about who produced these fonts and why they are freely available.

If the Type1 fonts exist, the next step is to take a look at `psfonts.map`. There may be several files with this name on the system, so it is important to find out which one TeX is actually using. `kpsewhich psfonts.map` might give this information.

The most likely problem is that this file does not contain a line telling TeX what to do if it encounters a request for font cmr10, i.e. the following line may be missing

```
cmr10          CMR10          <cmr10.pfb
```

It is probable that the required lines (in practice, you do not just need cmr10) are found in a file named `psfonts.cmz` which resides in TEXMF/dvips/bluesky.

One solution is to instruct R<sub>X</sub> to read additional map files like `psfonts.cmz` or `psfonts.amz`. This can be achieved by modifying the appropriate pyxrc file which is either the systemwide `/etc/pyxrc` or `.pyxrc` in the user’s home directory. Here, the names of the map files to be read by R<sub>X</sub> should be appended separated by whitespaces like in the following example:

```
fontmaps = psfonts.map psfonts.cmz psfonts.amz
```

Alternatively, the contents of `psfonts.cmz` can be inserted into `psfonts.map`. Probably, `psfonts.map` recommends not to do this by hand. In this case the instructions given in the file should be followed. Otherwise, `psfonts.cmz` should be copied into `psfonts.map` while keeping a backup of the old `psfonts.map` just in case. After these changes, R<sub>X</sub> most likely will be happy. When inserting `psfonts.cmz` into `psfonts.map` it may be a good idea to include `psfonts.amz` as well. `psfonts.amz` contains information about some more fonts which might be needed at some point. Making these changes of `psfonts.map` will imply that the T<sub>E</sub>X system will use the cmr fonts in Type1 format instead of pk format which is actually not a bad thing, in particular if `latex / dvips / ps2pdf` is used to generate PDF output. With fonts in pk

format this will look ugly and using Type1 fonts solves this problem as well. When pdf $\text{\LaTeX}$  is used to create PDF files, Type1 fonts will be used anyway.

## 4.4 Fonts

### 4.4.1 I want to use a font other than computer modern roman

As long as you have a font in Type1 format available, this should be no problem (even though it may cost you some time to set up things properly).

In the simplest case, your  $\text{\LaTeX}$  system contains everything needed. Including the following line into your code will probably work

```
text.set(mode="latex")
text.preamble(r"\usepackage{mathptmx}")
```

and give you Times as roman font.

If you own one of the more common commercial fonts, take a look at [CTAN:fonts](#) and its subdirectories as well as at the web page <http://home.vr-web.de/was/fonts.html> of Walter Schmidt. It is not unlikely that somebody has already done most of the work for you and created the files needed for the font to work properly with  $\text{\LaTeX}$ . But remember: we are talking about commercial fonts here, so do not expect to find the fonts themselves for free.

If none of these cases applies, you should spend some time reading manuals about font installation, e.g. [CTAN:macros/latex/doc/fntguide.pdf](#) (of course, I do not expect font wizards to read the last few lines).

### 4.4.2 Can I use a TrueType font with $\text{\LaTeX}$ ?

Not directly as  $\text{\LaTeX}$  only knows how to handle Type1 fonts (although it is possible to get  $\text{\LaTeX}$  to work with TrueType fonts). However, you may use `ttf2pt1` (from <http://ttf2pt1.sourceforge.net>) to convert a TrueType font into a Type1 font which you then install in your  $\text{\TeX}$  system [↑4.4.1](#). You will lose hinting information in the conversion process but this should not really matter on output devices with not too low resolution.