

latextool

DR. YIHSIANG LIOW (DECEMBER 16, 2013)

Contents

1	Notes and API and TODOs	2
2	Test mylist	8
3	execute	11
4	console	13
5	Python	16
6	verbatim	19
7	Text Color	30
8	table	31
9	Plot	40
10	Grid	41
11	tabrect	44
12	Scaling	45
13	Line	46
14	circle	58
15	rect	63
16	array	69
17	RectContainer	72

18 Snipped Array	89
19 Singly Linked List	93
20 Doubly linked list	96
21 positions	98
22 tree	100
23 tree2	104
24 graph	105
25 chunkedarray	110
26 2D Array	118
27 bend	124
28 arc	127
29 shell	129
30 Frame	132
31 Test Verbatim Spacing	133
32 Automata	134
33 2d vector diagram	144
34 Line Graph	145
35 TEST ISOCELES TRIANGLE	149

long verbatim:

[illegible]

Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
DR. YIHSIANG LIOW
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak
Testing pagebreak

ISSUE: The label in the rect is centered. The problem is the font for a character with greater height will appear lower than another. So for an array of rects of character, the base line of the labels are not the same.

ISSUE: The boundary line are drawn within the bounding box. So joining up rects would mean the common line will look thicker. DONE. See Rect2.

ISSUE:

- Joining rects in a uniform way to get hor and ver arrays and 2d arrays. Allows for containers with finite area, infinite horizontally, infinite vertically.
- Where to put first rect? What's the API? For hor array that grows to the right, only need top-left corner or bottom-left.or center-left. Maybe just specify the absolute corners of the first rect?
- Allow each element of array to be any shape. Use BaseNode of course.

CLEAN UP OF PYTHON-LATEX DRAWING LIBRARY

`latextool.py``latextool_basic.py``automata``courses/book/graph.py -----> projects/latextool/latextool.py``courses/ciss362-automata/resources/dfa/dfa-backup.py``courses/ciss362-automata/resources/dfa/dfa.py``courses/ciss362-automata/resources/dfa/graph/graph.py``projects/automata/dfa/dfa.py``linked list``courses/math325/n/latex_ds.py``projects/test-latex/xxx/latex_ds.py``array sorting``2d graphs:``courses/math325/n/makegraph/plot.py`

=====

`automata`

1d shapes: line or things made up of lines (broom?...) They have

- color
- style
- starting tip: dot, arrow (and arrow style)
- ending tip: dot, arrow (and arrow style)

2d shapes: rect, circle, They have

- boundary is a line. So it has boundary color and boundary style

- interior: color
- minipage

File: testmylist.tex

2 Test mylist

For comparison, this is enumerate:

- Line 1.
- line 2.
- line 3.

After enumerate.

This is with paragraphing within items:

- Line 1. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test.

Next paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

- line 2.
- line 3.

A new paragraph after mylist.

`mylist` environment: Vertical spacing between items is set to 0. Also vertical and horizontal paragraph spacing in an item is set to 0.

Testing mylist:

- line 1
- line 2
- line 3

After mylist. Notice that there is no vertical spacing between items. However there is still vertical spacing before and after the mylist.

This is with paragraphing within items:

- Line 1. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test.
Next paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
- line 2.
- line 3.

A new paragraph after above environment. The next environment is in a new paragraph.

- line 1
- line 2
- line 3

`tightlist` environment: This is the same as `mylist` except that the vertical spacing before and after the environment is removed. This does not work quite right if the environment is in a paragraph of its own – there will be a little extra vertical spacing before the environment.

Testing `tightlist`:

- line 1
- line 2
- line 3

After environment.

This is with paragraphing within items:

- Line 1. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test. This is a test.
Next paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
- line 2.
- line 3.

A new paragraph after above environment. The next environment is in a new paragraph.

- line 1
- line 2
- line 3

File: execute.tex

3 execute

The `execute` (in `latexbasic_tool.py`) function executes a python string as a program and the stdout is inserted into the latex file:

```
\begin{python}
from latextool_basic import execute
execute(r"""
print "hello world"
""")
\end{python}
```

Here's the result:

hello world

Note that there's some extra whitespace before the hello world. TODO: Need to figure a way to remove the whitespace.

The following does the same except that it first prints the python source in a framed verbatim box:

```
\begin{python}
from latextool_basic import execute
execute(r"""
print "hello world"
""", print_source=True)
\end{python}
```

If `debug` is set to `True`, there are three outputs: the python source, the stderr, and the stdout, all in framed verbatim boxes.

```
\begin{python}
from latextool_basic import execute
execute(r"""
print "hello world"
""", debug=True)
\end{python}
```

If there's an error in the python source, the output will be the same as if `debug` is set to `True`.

TEST: debug=True

PYTHON ERROR. See source, stderr, stdout below

```
print 'hello world'
```

```
hello world
```

TEST: debug=False

```
hello world
```

TEST: print source=True, debug=False

```
print 'hello world'
```

```
hello world
```

TEST: Error in source

PYTHON ERROR. See source, stderr, stdout below

```
print hello world
```

```
File "ppnnfwys.tmp.py", line 1
    print hello world
      ^
```

```
SyntaxError: invalid syntax
```

NOTE: console function is now the same as the verbatim function.

There a console latex environment and a console function in `latextool_basic`.

```
hello world
```

```
hello world
```

Here's a python environment that prints hello world, prints a console function return value, prints hello world:

```
hello world
```

Note that the spacing is different from the above.

```
from latextool_basic import console
print console("hello world")
```

```
hello world
```

DR. YIHSIANG LIOW yliow@ccis.edu

```
from latextool_basic import console
print console(r'''
line 1 ... hello world hello world hello world
line 2 ... hello world hello world hello world hello world hello world hell
o world hello world hello world hello world hello world hello world hello w
orld hello world hello world hello world hello world hello world hello worl
d hello world hello world hello world
'''.strip())
```

```
line 1 ... hello world hello world hello world
line 2 ... hello world hello world hello world hello world hello world hell
o world hello world hello world hello world hello world hello world hello w
orld hello world hello world hello world hello world hello world hello worl
d hello world hello world hello world
```

You can control the window width:

```
from latextool_basic import console
print console(r'''
line 1 ... hello world hello world hello world
line 2 ... hello world hello world hello world hello world hello world hell
o world hello world hello world hello world hello world hello world hello w
orld hello world hello world hello world hello world hello world hello worl
d hello world hello world hello world
'''.strip(), width=40)
```

```
line 1 ... hello world hello world hello
world
line 2 ... hello world hello world hello
world hello world hello world hello wor
ld hello world hello world hello world h
ello world hello world hello world hello
world hello world hello world hello wor
ld hello world hello world hello world h
ello world hello world
```

You can specify the `wrapmarker` that is used to indicate a line wrap:

```
from latextool_basic import console
print console(r'''
line 1 ... hello world hello world hello world
line 2 ... hello world hello world hello world hello world hello world hell
o world hello world hello world hello world hello world hello world hello w
orld hello world hello world hello world hello world hello world hello worl
d hello world hello world hello world
'''.strip(), width=40, wrapmarker='---> ')
```

```
line 1 ... hello world hello world hello
---> world
line 2 ... hello world hello world hello
---> world hello world hello world hello wor
---> ld hello world hello world hello world h
---> ello world hello world hello world hello
---> world hello world hello world hello wor
---> ld hello world hello world hello world h
---> ello world hello world
```

TODO: Fix non-pagebreak.

How to insert → symbol into verbatim area?

File: python.tex

5 Python

There are vertical spacing issues with the python environment. Vertical spacing before and after the python environment has been removed from the python environment in `mypython.tex`.

TEST 1. Python environment with console function call. Line before python environment.

hello world

Line after the python environment.

TEST 2. One python environment with two console function calls. Line before python environment.

hello world 1

hello world 2

Line after the python environment. ISSUE: Note that the two frames are joined up. For the time being do not put two console environment immediately next to each other.

TEST 3. One python environment with two console function calls with one blank line between the console function calls. Line before python environment.

```
hello world 1
```

```
hello world 2
```

Line after the python environment. ISSUE: Note that the two frames are joined up. For the time being do not put two console environment immediately next to each other.

TEST 4. Two python environments with one console function call each. Line before python environment.

```
hello world 1
```

Line between the python environments.

```
hello world 2
```

Line after the second python environment.

TEST 5. This is the console environment in latex (no python).

```
hello world 1
```

Line after.

TEST 6. 1 console environment in latex (no python).

```
hello world 1
```

Line between.

```
hello world 2
```

Line after.

TEST 7. 2 console environments in latex next to each other.

```
hello world 1
```

```
hello world 2
```

Line after.

TEST 8. 1 verbatim environment in latex (no python).

```
hello world 1
```

Line after.

TEST 9. This is 2 Verbatim environments in latex (no python).

```
hello world 1
```

Line between.

```
hello world 2
```

Line after.

TEST 10. 2 Verbatim environments next to each.

```
hello world 1
```

```
hello world 2
```

Line after.

File: verbatim.tex

6 verbatim

The `verbatim` function allows you to create verbatim environments. Executing tex commands in the environment can be easier.

Example:

```
from latextool_basic import verbatim
print verbatim("hello world\nlorem ipsum")
```

```
hello world
lorem ipsum
```

You can have no frame:

```
from latextool_basic import verbatim
print verbatim("hello world\nlorem ipsum", frame=None)
```

```
hello world
lorem ipsum
```

You have line numbers on the left:

```
from latextool_basic import verbatim
print verbatim("hello world\nlorem ipsum", numbers='left')
```

```
1 hello world
2 lorem ipsum
```

You have line numbers on the right:

```
from latextool_basic import verbatim
print verbatim("hello world\nlorem ipsum", numbers='right')
```

```
hello world
lorem ipsum
```

1
2

You can escape to a tex command:

```
t = r"""
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
"""
from latextool_basic import verbatim
print verbatim(t.strip(), command=['redtext', 'iostream'])
```

```
#include <redtext>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
```

You can specify a list of words for each tex command:

```
t = r"""
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
"""
from latextool_basic import verbatim
print verbatim(t.strip(), command=['redtext', ['iostream', 'std']])
```

```
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
```

You can use regular expressions:

```
t = r"""
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}"""
from latextool_basic import verbatim
print verbatim(t.strip(), command=['redtext', '"[a-z ]*"'])
```

```
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
```

WARNING. Since search is by regular expression, be careful to escape metacharacters:

```
from latextool_basic import verbatim
t = r"""
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
"""

print verbatim(t.strip(), command=['textbox', r'main\(\)'])
```

```
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
```


You can also specify which part of the text should be escaped by specifying 3 numbers: the line number, the starting column number, and the ending column number.

```
t = r"""
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
"""
from latextool_basic import verbatim
print verbatim(t.strip(), command=['underline', [4, 4, 13]])
```

```
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    return 0;
}
```

You can also specify a linenumber and a text:

```
t = r"""
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    std::cout << "hello world" << std::endl;
    std::cout << "hello world" << std::endl;
    return 0;
}
"""
from latextool_basic import verbatim
print verbatim(t.strip(), command=['textred', [5, 'std::cout']])
```

```
#include <iostream>

int main()
{
    std::cout << "hello world" << std::endl;
    std::cout << "hello world" << std::endl;
    std::cout << "hello world" << std::endl;
    return 0;
}
```

You can have a list of tex commands:

```
from latextool_basic import verbatim
print verbatim("the answer is 42",
               commands=[
                   ['underline', 'the'],
                   ['textbox', ['answer', '42']],
               ])
)
```

the answer is 42

You can also use the contents of a file like this:

```
from latextool_basic import verbatim
print verbatim(filename='helloworld.cpp')
```

`verbatim` is not exactly a verbatim function: there's line wrap. By default the width is set to 75. You can set the width to 40:

```
from latextool_basic import verbatim
print verbatim("abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde", width=10)
```

```
abcdeabcde
abcdeabcde
abcdeabcde
abcdeabcde
abcde
```

You can also specify a string to indicate line wrap like this:

```
from latextool_basic import verbatim
print verbatim("abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcde", width=10,
               wrapmarker='---> ')
```

```
abcdeabcde
---> abcdeabcde
---> abcdeabcde
---> abcdeabcde
---> abcde
```

TEST 1. Test vertical spacing before and after verbatim(). Before.

```
hello world 1  
hello world 2
```

After.

TEST 2. Test vertical spacing between two verbatim() in 1 python environment. Before.

```
frame 1  
hello world 1  
hello world 2
```

```
frame 2  
hello world 1  
hello world 2
```

After.

TEST 3. Same as before but print a blank line between the 2 verbatim()s. Before.

```
frame 1  
hello world 1  
hello world 2
```

```
frame 2  
hello world 1  
hello world 2
```

After.

TEST 4. Test vertical spacing between 2 python environments, each containing 1 verbatim(), no blank line between the python environments. Before.

```
frame 1  
hello world 1  
hello world 2
```

```
frame 2  
hello world 1  
hello world 2
```

After.

TEST 5. For user input in verbatim environment. Before.

```
Please enter x: 42
Please enter y and z: 43 44
Please enter login: jdoe
```

After.

File: textcolor.tex

7 Text Color

Testing `\textred`: hello world ... $x = 1 \dots \int x \, dx = \frac{1}{2}x^2 + C$

$$\sin \theta = 0$$

Black again.

Testing `\verb!\textred!`:

`\textred{hello world ... $x = 1$... $\int x \, dx = \frac{1}{2}x^2 + C$}`

`\[`

`\sin \theta = 0`

`\]`

`\]`

Black again.

Testing `\textblue`: hello world ... $x = 1 \dots \int x \, dx = \frac{1}{2}x^2 + C$

$$\sin \theta = 0$$

Black again.

Testing `\textwhite`:

Black again.

Using `\underline`/`\textbox` with `\textwhite`:

$$\int x \, dx = \frac{1}{2}x^2 + C$$

$$\int x \, dx =$$

$$\int x \, dx = \text{\textbox{100pt}{100pt}}$$

File: table.tex

8 table

```
from latextool_basic import table
print table({0:0, 1:1, 2:4, 3:9, 4:16})
```

0	0
1	1
2	4
3	9
4	16

```
from latextool_basic import table
print table([(1, 2, 3),
             (2, 2, 5),
             (5, 2),
             (-1, 2, 6),
             (1, 2, 3),
             ])
```

1	2	3
2	2	5
5	2	
-1	2	6
1	2	3


```

from latextool_basic import table
print table([(1, 2, 3),
            (2, 2, 5),
            (5, 2),
            (-1, 2, 6),
            (1, 2, 3),
            ],
            col_headings = ['$x$', '$y$', '$z$'])

```

x	y	z
1	2	3
2	2	5
5	2	
-1	2	6
1	2	3

```

from latextool_basic import table
print table([('0', 2, 5, 8),
            ('2', '2', 3, 6),
            ('5', '2', '2', 3),
            ('8', '2', '2', '2')
            ],
            col_headings = ['0', '2', '5', '8'],
            row_headings = ['0', '2', '5', '8'])

```

	0	2	5	8
0		2	5	8
2			3	6
5				3
8				

```
from latextool_basic import table
print table([(0, 2, 5, 8),
            (0, 0, 3, 6),
            (0, 0, 0, 3),
            (0, 0, 0, 0)
            ],
            col_headings = ['0', '2', '5', '8'],
            row_headings = ['0', '2', '5', '8'],
            topleft_heading = r'$\Delta X$',
            )
```

ΔX	0	2	5	8
0		2	5	8
2			3	6
5				3
8				

Too few row headings:

```
from latextool_basic import table
print table([(' ', 2, 5, 8),
            (' ', ' ', 3, 6),
            (' ', ' ', ' ', 3),
            (' ', ' ', ' ', ' ')
            ],
            col_headings = ['0', '2', '5', '8'],
            row_headings = ['0', '2'],
            topleft_heading = r'$\Delta X$',
            )
```

ΔX	0	2	5	8
0		2	5	8
2			3	6
				3

Too few col headings

```
from latextool_basic import table
print table([(' ', 2, 5, 8),
            (' ', ' ', 3, 6),
            (' ', ' ', ' ', 3),
            (' ', ' ', ' ', ' ')
            ],
            col_headings = ['0', '2'],
            row_headings = ['0', '2', '5', '8'],
            topleft_heading = r'$\Delta X$',
            )
```

ΔX	0	2		
0		2	5	8
2			3	6
5				3
8				

Too few rows of data

```
from latextool_basic import table
print table([(' ', 2, 5, 8),
            (' ', ' ', 3, 6),
            ],
            col_headings = ['0', '2', '5', '8'],
            row_headings = ['0', '2', '5', '8'],
            topleft_heading = r'$\Delta X$',
            )
```

ΔX	0	2	5	8
0		2	5	8
2			3	6
5				
8				

Too few rows of data

```
from latextool_basic import table
print table([(' ', 2, 5, 8),
            (' ', ' ', 3, 6),
            ],
            col_headings = ['0', '2', '5', '8'],
            row_headings = ['0', '2', '5', '8'],
            topleft_heading = r'$\Delta X$',
            )
```

ΔX	0	2	5	8
0		2	5	8
2			3	6
5				
8				

Column headings only (no row headings). Note that topleft heading does not show.

```
from latextool_basic import table
print table([(' ', 2, 5, ),
            (' ', ' ', 3, ),
            (' ', ' ', ' ', ),
            (' ', ' ', ' ', )
            ],
            col_headings = ['0', '2', '5', '8'],
            topleft_heading = r'$\Delta X$',
            )
```

0	2	5	8
	2	5	
		3	

Row headings only (no column headings). Note that topleft heading does not show.

```
from latextool_basic import table
print table([( ' ', 2, 5, ),
             ( ' ', ' ', 3, ),
             ( ' ', ' ', ' ', ),
             ( ' ', ' ', ' ', )
            ],
            row_headings = ['0', '2', '5', '8'],
            topleft_heading = r'$\Delta X$',
            )
```

0		2	5
2			3
5			
8			

File: plot.tex

9 Plot

All drawing go into a Plot object. The following should be in a python environment:

```
from latextool_basic import Plot, Circle
p = Plot()
p += Circle(x=1, y=1, r=1)
print p
```

or run in the execute function in a python environment:

```
from latextool_basic import execute

s = r"""
from latextool_basic import Plot, Circle
p = Plot()
p += Circle(x=1, y=1, r=1)
print p
"""
execute(s)
```

A centered tikzpicture environment is then inserted.

You can also add a pgf/tikz string to plot:

```
from latextool_basic import Plot, Circle
p = Plot()
p += Circle(x=1, y=1, r=1)
p.add("\draw ...")
print p
```

You can create and store a pdf image in the tmp/ subdirectory. See next section.

File: grid.tex

10 Grid

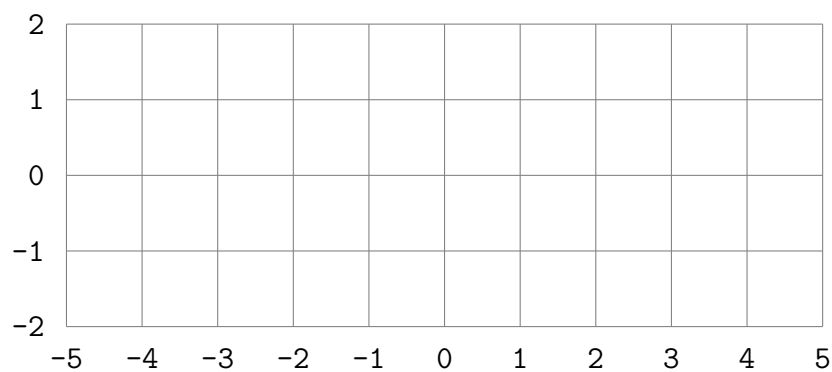
Plot with default grid.

```
from latextool_basic import *
p = Plot()
p += Grid()
print p
```

0 .
0

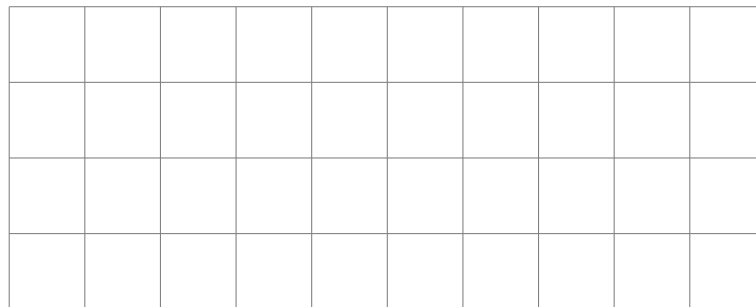
With object grid and bounding box:

```
from latextool_basic import *
p = Plot()
p += Grid(x0=-5, y0=-2, x1=5, y1=2)
print p
```



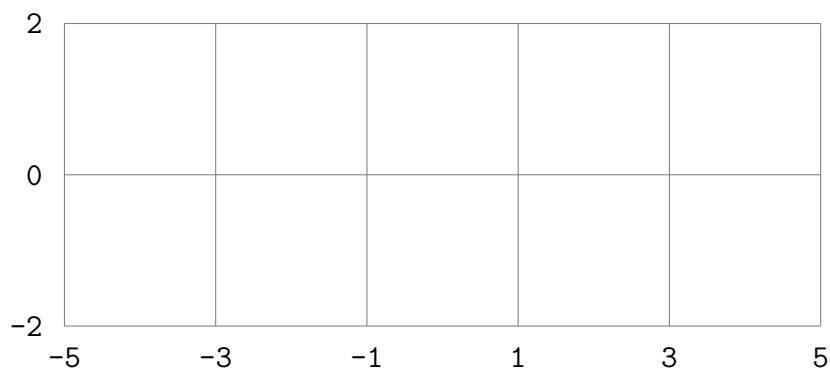
With label_axis set to false.

```
from latextool_basic import *  
p = Plot()  
p += Grid(x0=-5, y0=-2, x1=5, y1=2, label_axes=False)  
print p
```



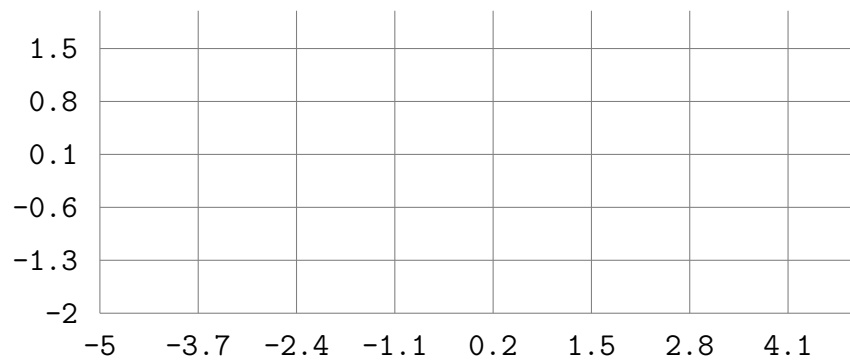
Example. $dx = 2$, $dy = 2$.

```
from latextool_basic import *  
p = Plot()  
p += Grid(x0=-5, y0=-2, x1=5, y1=2, dx=2, dy=2)  
print p
```



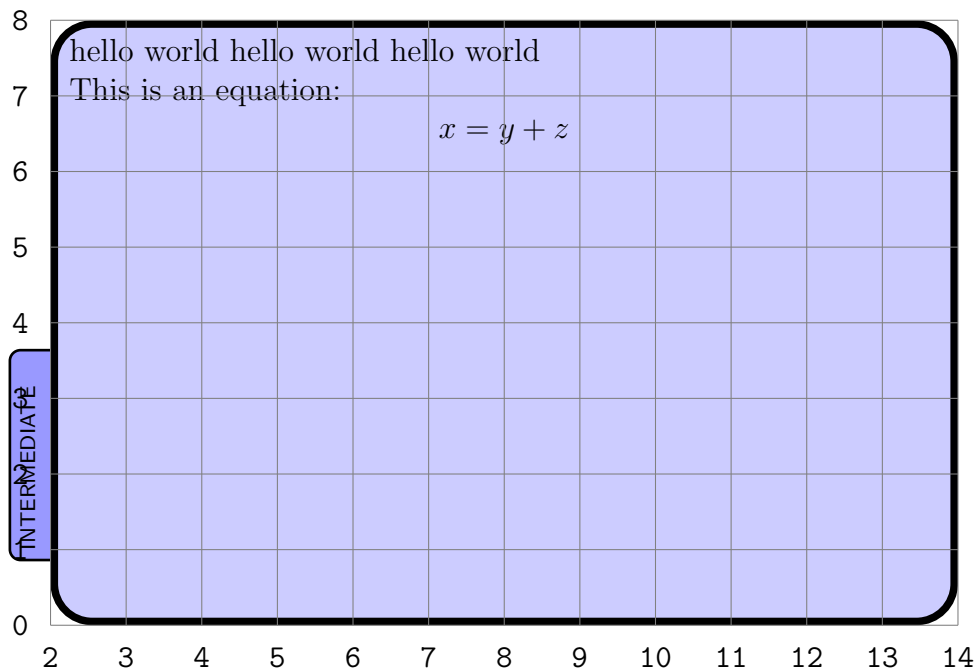
Example. $dx = 1.3$, $dy = 0.7$.

```
from latextool_basic import *  
p = Plot()  
p += Grid(x0=-5, y0=-2, x1=5, y1=2, dx=1.3, dy=0.7)  
print p
```



File: tabrect.tex

11 tabrect



```

from latextool_basic import *
p = Plot()
minipage = r'''
hello world hello world hello world

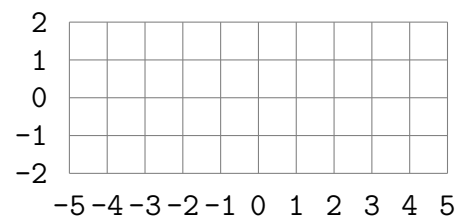
This is an equation:
\[
x = y + z
\]
'''
p.add(tabrect, x0=2, y0=0, x1=14, y1=8, background='blue!20',
      innersep=0.5, linewidth=0.1, s=minipage)
p += Grid()
print p

```

File: scaling.tex

12 Scaling

```
from latextool_basic import *  
p = Plot(scale=0.5)  
p.add(grid, x0=-5, y0=-2, x1=5, y1=2)  
print p
```



tmp/0cd209905ab75415b1e99c9ca01a5c1b.pdf

File: line.tex

13 Line

API:

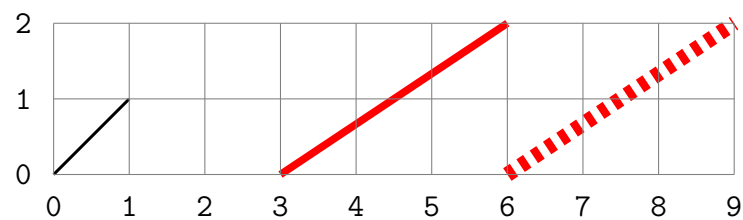
```
from latextool_basic import Line, verbatim
print verbatim(Line.__doc__)
```

```
Line(
    x0=0, y0=0, x1=0, y1=0,
    points=None,
    linecolor='black',
    linewidth='',
    linestyle='',
    startstyle='',
    endstyle='',
    r=0,                                # radius of dot at start or end of line
)

TODO: Path class
```

TEST 1. Test linecolor, linewidth, startstyle, endstyle

```
from latextool_basic import *
p = Plot()
p += Line(x0=0, y0=0, x1=1, y1=1)
p += Line(x0=3, y0=0, x1=6, y1=2, linecolor='red', linewidth=0.1)
p += Line(x0=6, y0=0, x1=9, y1=2, linecolor='red', linewidth=0.2,
          linestyle='dashed')
p += Grid()
print p
```



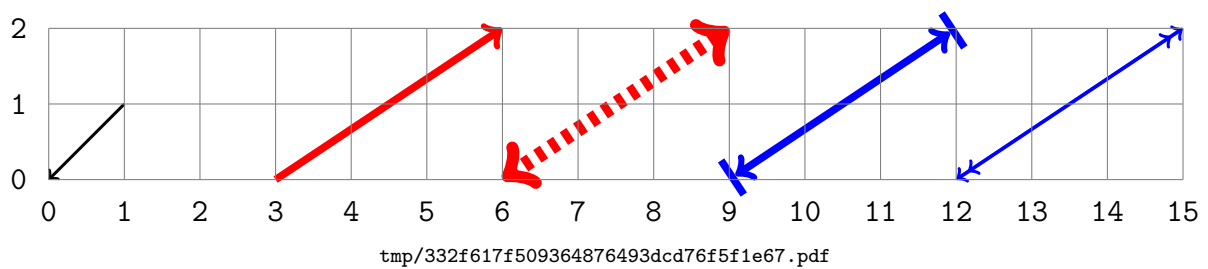
tmp/7d58b9267620a06511c3637f196482d4.pdf

TEST 2.

```

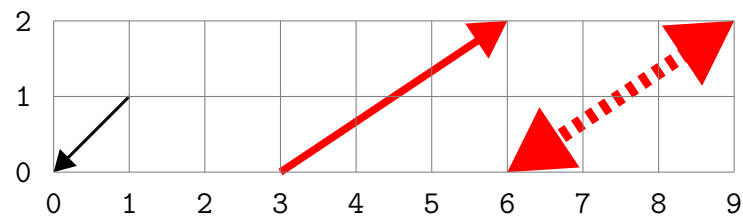
from latextool_basic import *
p = Plot()
p += Line(x0=0, y0=0, x1=1, y1=1, startstyle='>')
p += Line(x0=3, y0=0, x1=6, y1=2, linecolor='red', linewidth=0.1,
          endstyle='>')
p += Line(x0=6, y0=0, x1=9, y1=2, linecolor='red', linewidth=0.2,
          linestyle='dashed',
          startstyle='>', endstyle='>')
p += Line(x0=9, y0=0, x1=12, y1=2, linecolor='blue', linewidth=0.1,
          startstyle='>|', endstyle='>|')
p += Line(x0=12, y0=0, x1=15, y1=2, linecolor='blue', linewidth=0.05,
          startstyle='>>', endstyle='>>')
p += Grid()
print p

```



TEST 3. Test `arrowstyle=triangle`.

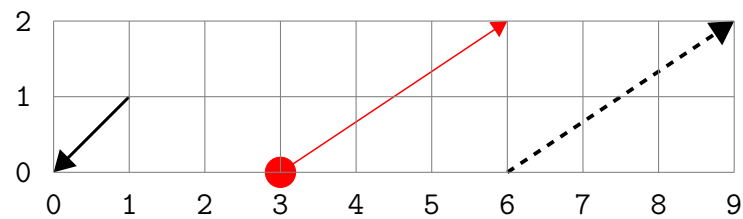
```
from latextool_basic import *
p = Plot()
p += Line(x0=0, y0=0, x1=1, y1=1, startstyle='->', arrowstyle='triangle')
p += Line(x0=3, y0=0, x1=6, y1=2, linecolor='red', linewidth=0.1,
          endstyle='->', arrowstyle='triangle')
p += Line(x0=6, y0=0, x1=9, y1=2, linecolor='red', linewidth=0.2,
          linestyle='dashed',
          startstyle='->', endstyle='->',
          arrowstyle='triangle')
p += Grid()
print p
```



tmp/d258d1ad69868e042ab3d90a3d5d1baf.pdf

TEST 4: Test `startstyle='dot'`.

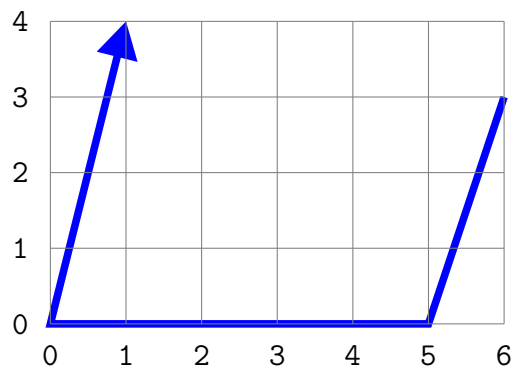
```
from latextool_basic import *
p = Plot()
p += Line(x0=0, y0=0, x1=1, y1=1, startstyle='->',
          arrowstyle='triangle', endstyle='dot')
p += Line(x0=3, y0=0, x1=6, y1=2, linecolor='red', linewidth=0.02,
          endstyle='->', arrowstyle='triangle',
          startstyle='dot', r=0.2)
p += Line(x0=6, y0=0, x1=9, y1=2, linecolor='black', linewidth=0.05,
          linestyle='dashed', startstyle='dot',
          endstyle='->', arrowstyle='triangle')
p += Grid()
print p
```



tmp/39c7db097d3c58286d19cff4c9e8d86f.pdf

TEST 5: Test points:

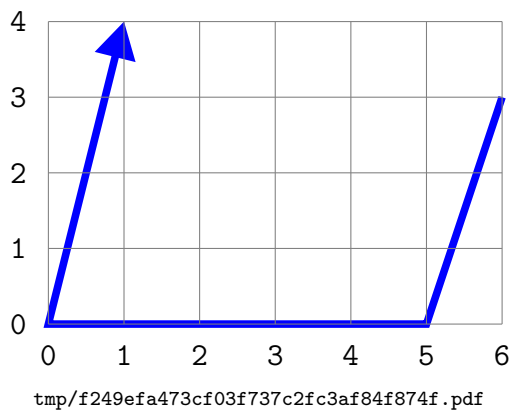
```
from latextool_basic import *
p = Plot()
p += Line(points=[(1,4), (0,0), (5,0), (6,3)],
          linecolor='blue', linewidth=0.1,
          startstyle='->', arrowstyle='triangle', endstyle='dot')
p += Grid()
print p
```



tmp/f249efa473cf03f737c2fc3af84f874f.pdf

Test 6. Testing spacing.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

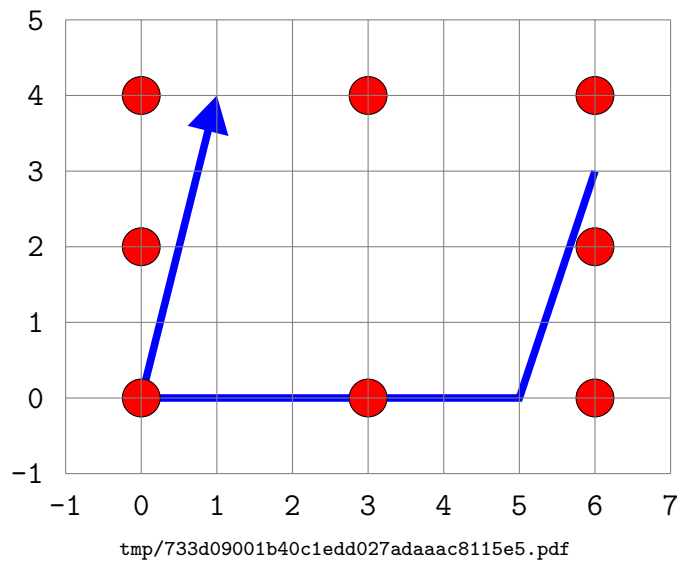
TEST 7. Test boundary points

```

from latextool_basic import *
p = Plot()

aline = Line(points=[(1,4), (0,0), (5,0), (6,3)],
              linecolor='blue', linewidth=0.1,
              startstyle='->', arrowstyle='triangle', endstyle='dot')
p += aline
p += Circle(center=aline.top(), r=0.25, background='red')
p += Circle(center=aline.bottom(), r=0.25, background='red')
p += Circle(center=aline.left(), r=0.25, background='red')
p += Circle(center=aline.right(), r=0.25, background='red')
p += Circle(center=aline.topleft(), r=0.25, background='red')
p += Circle(center=aline.topright(), r=0.25, background='red')
p += Circle(center=aline.bottomleft(), r=0.25, background='red')
p += Circle(center=aline.bottomright(), r=0.25, background='red')
p += Grid()
print p

```

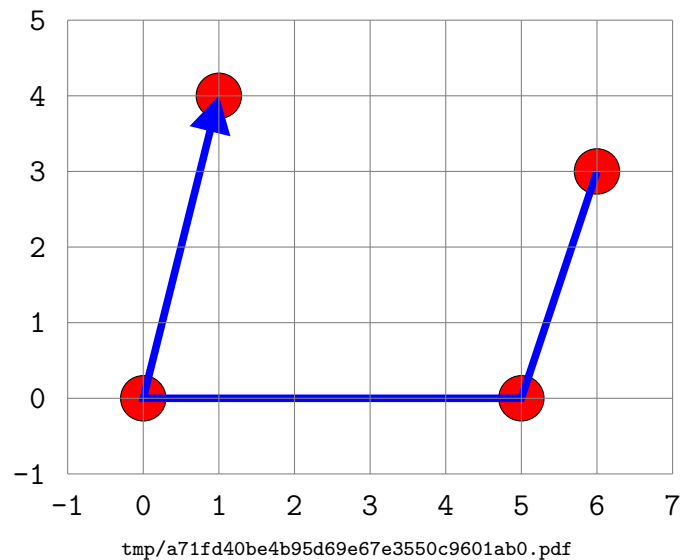


TEST 8. Test points

```
from latextool_basic import *
p = Plot()

aline = Line(points=[(1,4), (0,0), (5,0), (6,3)],
              linecolor='blue', linewidth=0.1,
              startstyle='->', arrowstyle='triangle', endstyle='dot')
for point in aline.points:
    p += Circle(center=point, r=0.3, background='red')

p += aline
p += Grid()
print p
```

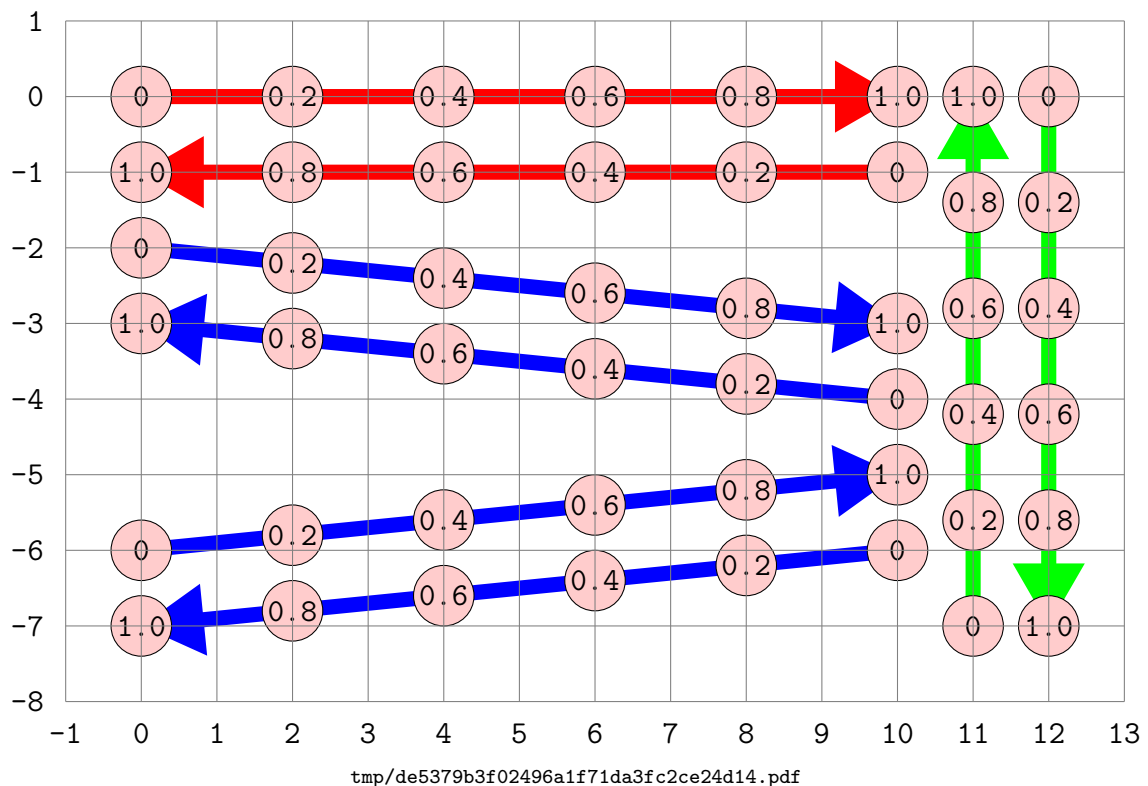


TEST 9. Test midpoint for 1 segment.

```

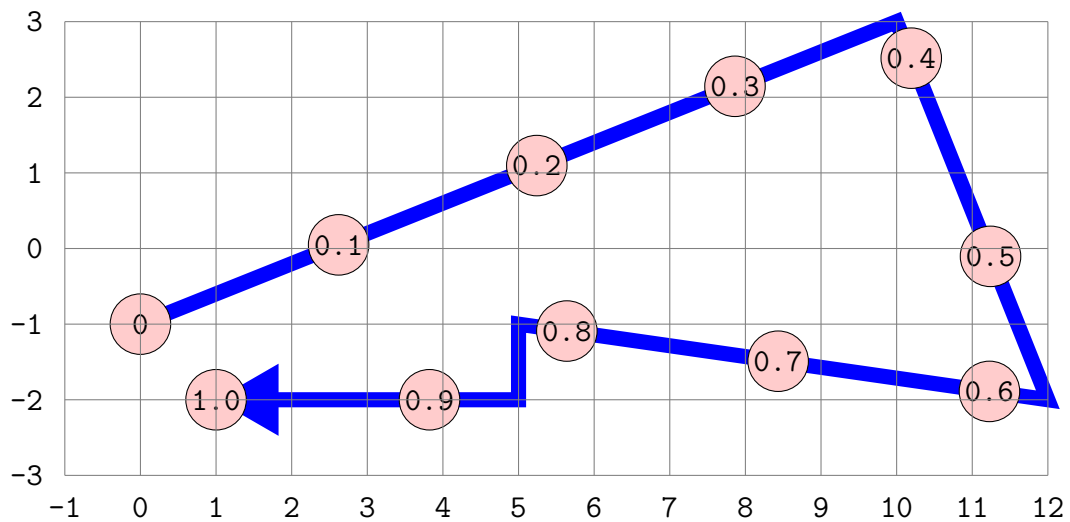
from latextool_basic import *
p = Plot()
def linewidthpoints(p, points, color1, color2):
    aline = Line(points=points, linecolor=color1, linewidth=0.2,
                 startstyle='dot', arrowstyle='triangle', endstyle='->')
    p += aline
    for i in [0, 0.2, 0.4, 0.6, 0.8, 1.0]:
        p += Circle(center=aline.midpoint(ratio=i), r=0.4,
                    background=color2, label=r'\texttt{%s}' % i)
linewidthpoints(p, [(0, 0), (10, 0)], 'red', 'red!20')
linewidthpoints(p, [(10, -1), (0, -1)], 'red', 'red!20')
linewidthpoints(p, [(0, -2), (10, -3)], 'blue', 'red!20')
linewidthpoints(p, [(10, -4), (0, -3)], 'blue', 'red!20')
linewidthpoints(p, [(0, -6), (10, -5)], 'blue', 'red!20')
linewidthpoints(p, [(10, -6), (0, -7)], 'blue', 'red!20')
linewidthpoints(p, [(11, -7), (11, 0)], 'green', 'red!20')
linewidthpoints(p, [(12, 0), (12, -7)], 'green', 'red!20')
p += Grid()
print p

```



TEST 10. Test midpoint for 1 segment.

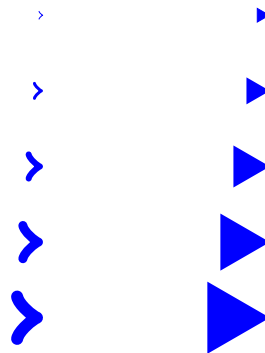
```
from latextool_basic import *
p = Plot()
points = [(0,-1), (10,3), (12,-2), (5,-1), (5,-2), (1,-2)]
aline = Line(points=points, linecolor='blue', linewidth=0.2,
             startstyle='dot', arrowstyle='triangle', endstyle='->')
p += aline
for i in [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]:
    p += Circle(center=aline.midpoint(ratio=i), r=0.4,
               background='red!20', label=r'\texttt{\%s}' % i)
p += Grid()
print p
```



tmp/ef5379327ea88357d1ab1aad3abcba72.pdf

TEST 11. Test short line with arrow tip (basically to test arrow tip).

```
from latextool_basic import *
p = Plot()
for i, x in enumerate([0.2, 0.15, 0.1, 0.05, 0.01]):
    p += Line(points=[(1-x, i), (1, i)], linecolor='blue', linewidth=x,
               endstyle='->')
    p += Line(points=[(4-x, i), (4, i)], linecolor='blue', linewidth=x,
               endstyle='->', arrowstyle='triangle')
#p += Grid()
print p
```



tmp/3a6cd516bd77420c403217a0077bb72e.pdf

For arrows with line width x , it seems enough to have a length of x .

File: circle.tex

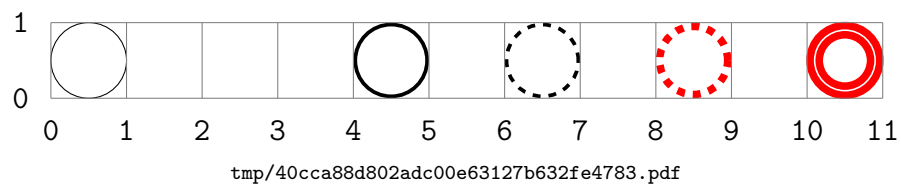
14 circle

Boundary:

```

from latextool_basic import *
p = Plot()
p += Circle(x=0.5, y=0.5, r=0.5)
p += Circle(x=2.5, y=0.5, r=0.5, linewidth=0)
p += Circle(x=4.5, y=0.5, r=0.5, linewidth=0.05)
p += Circle(x=6.5, y=0.5, r=0.5, linewidth=0.05, linestyle='dashed')
p += Circle(x=8.5, y=0.5, r=0.5, linewidth=0.1, linestyle='dashed',
            linecolor='red')
p += Circle(x=10.5, y=0.5, r=0.5, linewidth=0.1, linestyle='double',
            linecolor='red')
p += Grid()
print p

```

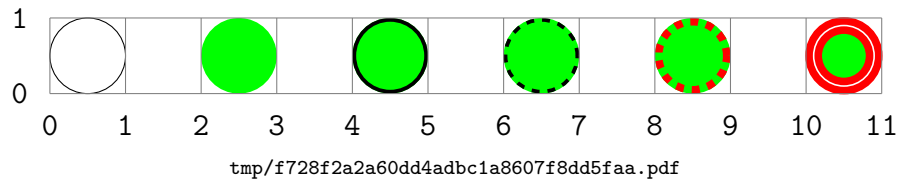


Background:

```

from latextool_basic import *
p = Plot()
p += Circle(x=0.5, y=0.5, r=0.5)
p += Circle(x=2.5, y=0.5, r=0.5, linewidth=0, background='green')
p += Circle(x=4.5, y=0.5, r=0.5, linewidth=0.05, background='green')
p += Circle(x=6.5, y=0.5, r=0.5, linewidth=0.05, linestyle='dashed',
            background='green')
p += Circle(x=8.5, y=0.5, r=0.5, linewidth=0.1, linestyle='dashed',
            linecolor='red', background='green')
p += Circle(x=10.5, y=0.5, r=0.5, linewidth=0.1, linestyle='double',
            linecolor='red', background='green')
p += Grid()
print p

```

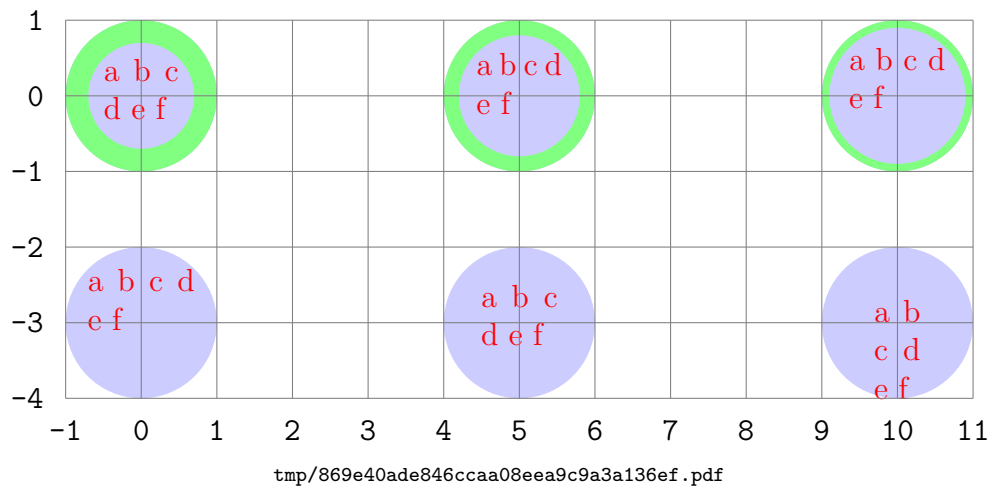


Label:

```

from latextool_basic import *
p = Plot()
s = 'a b c d e f'
p += Circle(x=0, y=0, r=1, linecolor='green!50!white', linewidth=0.3,
            background='blue!20!white', foreground='red', innersep=0, s=s)
p += Circle(x=5, y=0, r=1, linecolor='green!50!white', linewidth=0.2,
            background='blue!20!white', foreground='red', innersep=0, s=s)
p += Circle(x=10, y=0, r=1, linecolor='green!50!white', linewidth=0.1,
            background='blue!20!white', foreground='red', innersep=0, s=s)
p += Circle(x=0, y=-3, r=1, linecolor='green!50!white', linewidth=0,
            background='blue!20!white', foreground='red', innersep=0, s=s)
p += Circle(x=5, y=-3, r=1, linecolor='green!50!white', linewidth=0,
            background='blue!20!white', foreground='red',
            innersep=0.2, s=s)
p += Circle(x=10, y=-3, r=1, linecolor='green!50!white', linewidth=0,
            background='blue!20!white', foreground='red',
            innersep=0.4, s=s)
p += Grid()
print p

```

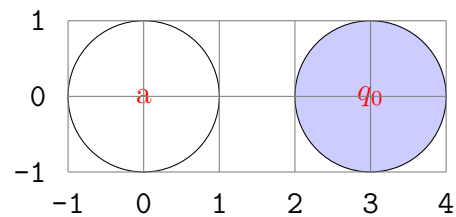


Foreground:

```

from latextool_basic import *
p = Plot()
p += Circle(x=0, y=0, r=1, foreground='red', label='a')
p += Circle(x=3, y=0, r=1, background='blue!20!white',
            foreground='red', label='$q_0$')
p += Grid()
print p

```



tmp/139205171631562f54a8b67ce058bc02.pdf

Center and boundary of Circle object:

```

s='1 2 3 4 5 6 7 8 9 10 11'
b1 = 'blue!50!white'
b2 = 'red!50!white'

from latextool_basic import *
p = Plot()

c1 = Circle(x=0, y=0, r=1, background=b1, s=s)
c2 = Circle(center=c1.center(), r=0.5, background=b2)
p += c1; p += c2

c1 = Circle(x=3, y=0, r=1, background=b1, s=s)
c2 = Circle(center=c1.left(), r=0.5, background=b2)
p += c1; p += c2

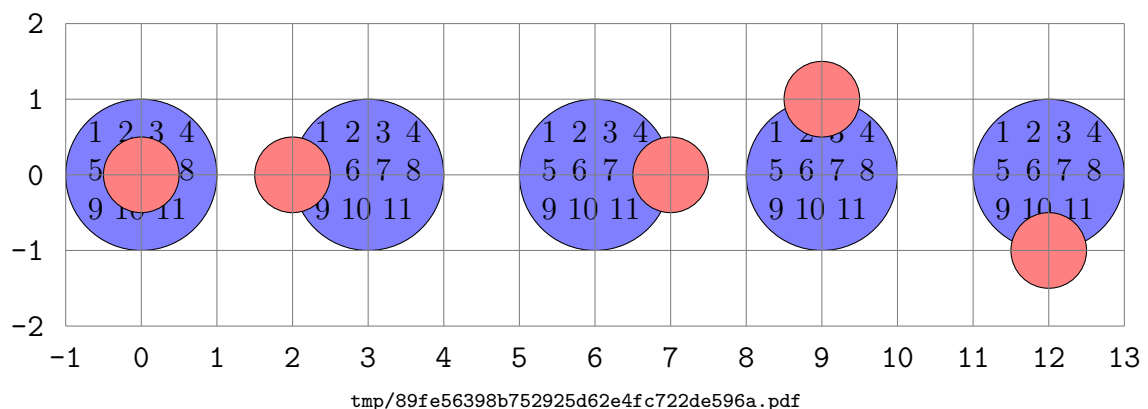
c1 = Circle(x=6, y=0, r=1, background=b1, s=s)
c2 = Circle(center=c1.right(), r=0.5, background=b2)
p += c1; p += c2

c1 = Circle(x=9, y=0, r=1, background=b1, s=s)
c2 = Circle(center=c1.top(), r=0.5, background=b2)
p += c1; p += c2

c1 = Circle(x=12, y=0, r=1, background=b1, s=s)
c2 = Circle(center=c1.bottom(), r=0.5, background=b2)
p += c1; p += c2

p += Grid()
print p

```

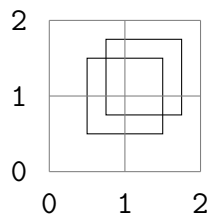


File: rect.tex

15 rect

TEST: Background is '', i.e., totally transparent.

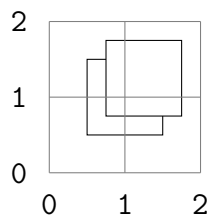
```
from latextool_basic import *
p = Plot()
p += Rect(x0=0.5, y0=0.5, x1=1.5, y1=1.5)
p += Rect(x0=0.75, y0=0.75, x1=1.75, y1=1.75)
p += Grid()
print p
```



tmp/52f83a12d7cb72f506ac1cce11855fe3.pdf

TEST: Background is white

```
from latextool_basic import *
p = Plot()
p += Rect(x0=0.5, y0=0.5, x1=1.5, y1=1.5, background='white')
p += Rect(x0=0.75, y0=0.75, x1=1.75, y1=1.75, background='white')
p += Grid()
print p
```



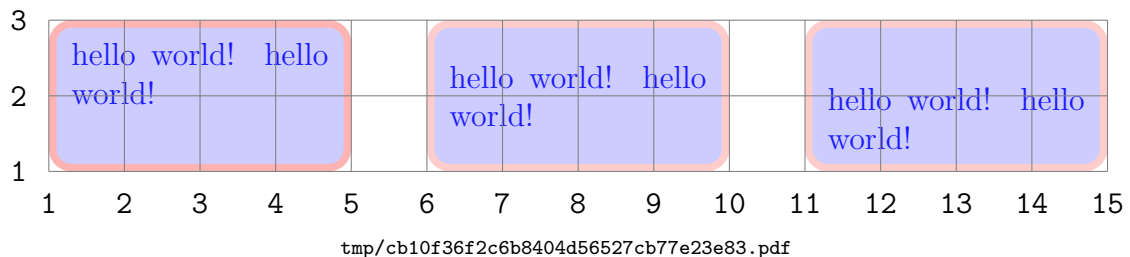
tmp/3830f16bc2c72b9450fd4ea98db65f24.pdf

TEST border, background, foreground:


```

from latextool_basic import *
p = Plot()
p += Rect(x0=1, y0=1, x1=5, y1=3,
          radius=0.25, innersep=0.25,
          linecolor='red!30!white', linewidth=0.1,
          background='blue!20!white!', foreground='blue!90!white',
          s='hello world! hello world!', align='t')
p += Rect(x0=6, y0=1, x1=10, y1=3,
          radius=0.25, innersep=0.25,
          linecolor='red!20!white', linewidth=0.1,
          background='blue!20!white!', foreground='blue!90!white',
          s='hello world! hello world!', align='c')
p += Rect(x0=11, y0=1, x1=15, y1=3,
          radius=0.25, innersep=0.25,
          linecolor='red!20!white', linewidth=0.1,
          background='blue!20!white!', foreground='blue!90!white',
          s='hello world! hello world!', align='b')
p += Grid()
print p

```



TEST border, background, foreground:

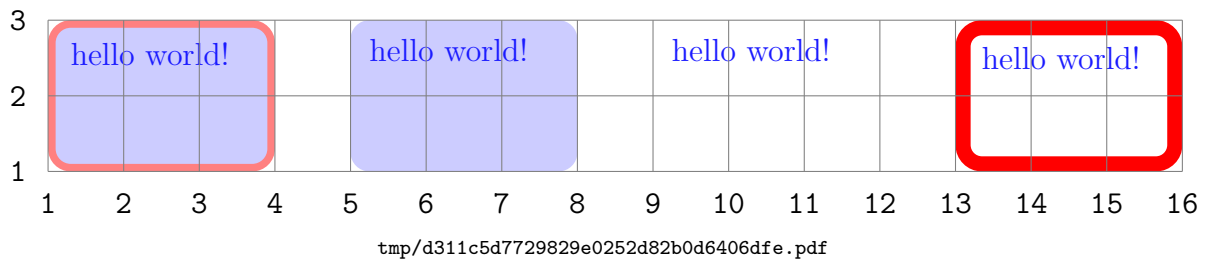
```

from latextool_basic import *
p = Plot(radius=0.25, innersep=0.25,
          linecolor='red!50!white', linewidth=0.1,
          background='blue!20!white', foreground='blue!90!white',
          align='t')
p += RectAdaptor(env=p.env, x0= 1, y0=1, x1= 4, y1=3, s='hello world!')
p.env['linewidth'] = 0
p += RectAdaptor(env=p.env, x0= 5, y0=1, x1= 8, y1=3, s='hello world!')
p.env['background'] = ''
p += RectAdaptor(env=p.env, x0= 9, y0=1, x1= 12, y1=3, s='hello world!')

p.env['background'] = ''
p.env['linecolor'] = 'red'
p.env['linewidth'] = 0.2
p += RectAdaptor(env=p.env, x0= 13, y0=1, x1= 16, y1=3, s='hello world!')

p += Grid()
print p

```

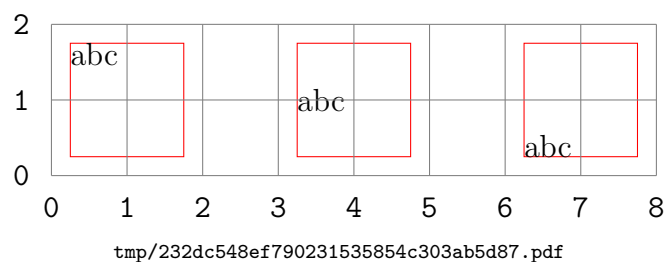


TEST: Align

```

from latextool_basic import *
p = Plot()
p += Rect(x0=0.25, y0=0.25, x1=1.75, y1=1.75, s='abc', align='t',
          linecolor='red', foreground='black')
p += Rect(x0=3.25, y0=0.25, x1=4.75, y1=1.75, s='abc', align='c',
          linecolor='red')
p += Rect(x0=6.25, y0=0.25, x1=7.75, y1=1.75, s='abc', align='b',
          linecolor='red')
p += Grid()
print p

```

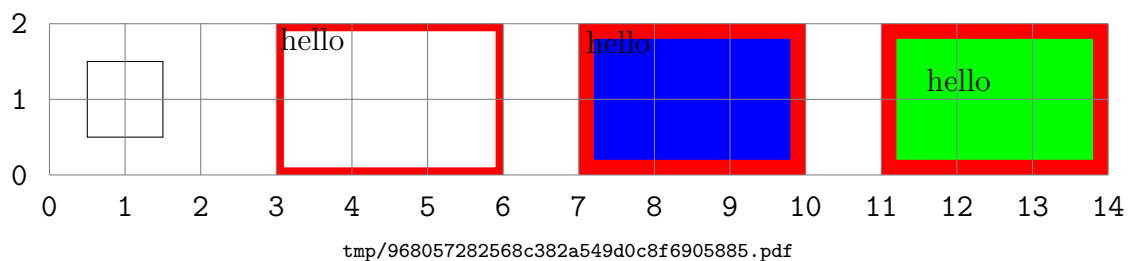


As above but using objects:

```

from latextool_basic import *
p = Plot()
p += Rect(x0=0.5, y0=0.5, x1=1.5, y1=1.5)
p += Rect(x0=3, y0=0, x1=6, y1=2, linecolor='red', linewidth=0.1, s='hello'
)
p += Rect(x0=7, y0=0, x1=10, y1=2, linecolor='red', background='blue',
          linewidth=0.2, s='hello')
p += Rect(x0=11, y0=0, x1=14, y1=2, linecolor='red', linewidth=0.2,
          background='green',
          innersep=0.5, s='hello')
p += Grid()
print p

```

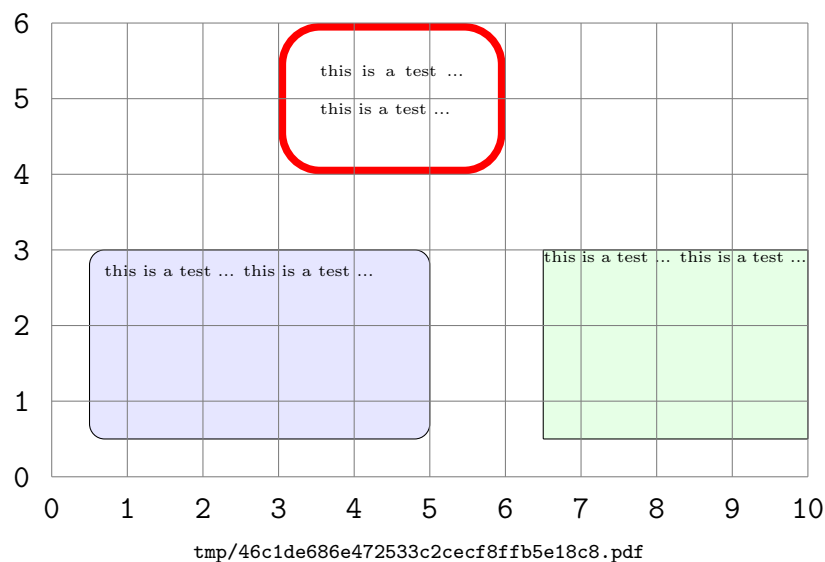


rect with rounded corners: (radius of corner is set to innersep by default):

```

from latextool_basic import *
p = Plot()
p += Rect(x0=3, y0=4, x1=6, y1=6, linecolor='red', linewidth=0.1,
          innersep=0.5, radius=0.5,
          s=r'\tiny this is a test ... this is a test ...')
p += Rect(x0=0.5, y0=0.5, x1=5, y1=3,
          background='blue!10!white',
          innersep=0.2, radius=0.2,
          s=r'\tiny this is a test ... this is a test ...')
p += Rect(x0=6.5, y0=0.5, x1=10, y1=3,
          background='green!10!white',
          innersep=0.01, radius=0.01,
          s=r'\tiny this is a test ... this is a test ...')
p += Grid()
print p

```

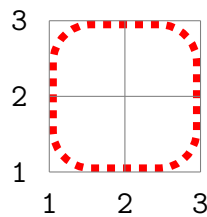


line style:

```

from latextool_basic import *
p = Plot()
p += Rect(x0=1, y0=1, x1=3, y1=3,
          linewidth=0.1, linecolor='red',
          radius=0.5, linestyle='dashed')
p += Grid()
print p

```



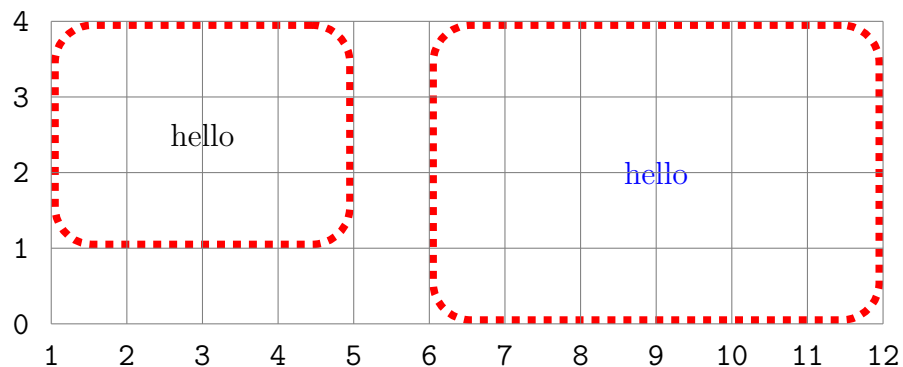
tmp/e3d3876073dc28cd4f56e77645c13985.pdf

label:

```

from latextool_basic import *
p = Plot()
p += Rect(x0=1, y0=1, x1=5, y1=4,
          linewidth=0.1, linecolor='red',
          radius=0.5, linestyle='dashed', label='hello')
p += Rect(x0=6, y0=0, x1=12, y1=4,
          linewidth=0.1, linecolor='red',
          foreground='blue',
          radius=0.5, linestyle='dashed', label='hello')
p += Grid()
print p

```

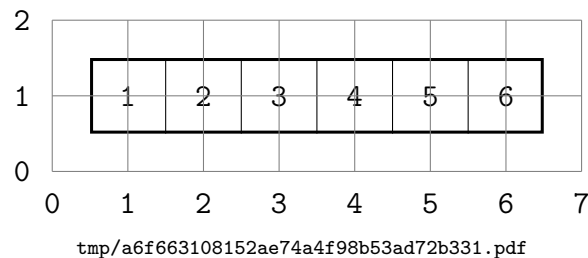


tmp/4ea3539e225df454128f29e9e067651c.pdf

File: array.tex

16 array

```
from latextool_basic import Plot, grid, array
p = Plot()
s = array(x0=0.5, y0=0.5, width=1, height=1, xs=[1,2,3,4,5,6])
p.add(s)
p.add(grid, x0=0, y0=0, x1=7, y1=2)
print p
```



```
from latextool_basic import Plot, array
p = Plot()
s = array(x0=0.5, y0=0.5, width=1, height=0.6, xs=[1,2,3,4,5,6])
p.add(s)
print p
```

1	2	3	4	5	6
---	---	---	---	---	---

tmp/bdffb063efd919ffd955e52c27a8c352.pdf

Array of characters:

```

from latextool_basic import Plot, array
p = Plot()
s = array(x0=0.5, y0=0.5, width=1, height=0.6,
          xs=['h','e','l','l','o'])
p.add(s)
print p

```

'h'	'e'	'l'	'l'	'o'
-----	-----	-----	-----	-----

tmp/c43092afc22cca6d4b4c2e4e8247197c.pdf

array with arraylinewidth

```

from latextool_basic import Plot, array
p = Plot()
s = array(x0=0.5, y0=0.5, width=1, height=0.6,
          xs=[1,2,3,4,5,6],
          arraylinewidth=0.2)
p.add(s)
print p

```

1	2	3	4	5	6
---	---	---	---	---	---

tmp/14feeb74f183d9a523a277b89a294469.pdf

array with celllinewidth

```

from latextool_basic import Plot, array
p = Plot()
s = array(x0=0.5, y0=0.5, width=1, height=0.6,
          xs=[1,2,3,4,5,6],
          celllinewidth=0.1)
p.add(s)
print p

```

1	2	3	4	5	6
---	---	---	---	---	---

tmp/90bebe142d87b27aa9a31ced66f3d13f.pdf

new array2

```
from latextool_basic import Plot, array2
p = Plot()
s = array2(x0=0.5, y0=0.5, width=1, height=0.6,
           vs=[1,2,3,4,5,6],
           celllinewidth=0.04)
p.add(s)
print p
```

1	2	3	4	5	6
---	---	---	---	---	---

tmp/5136b13ea4abf9754cd5468d85c9aa7e.pdf

TODO: Array variable name to left of array? Variables in general? Align variables and values?

File: rectcontainer.tex

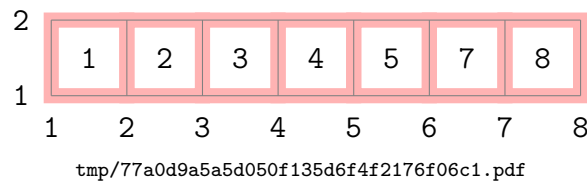
17 RectContainer

Horizontal array:

```
from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1)
for x in '1234578':
    c += Rect2(x0=0, y0=0, x1=1, y1=1,
               linewidth=0.2, linecolor='red!30!white',
               label=r'\texttt{%s}' % x)

p += c
p += Grid()
print p
```



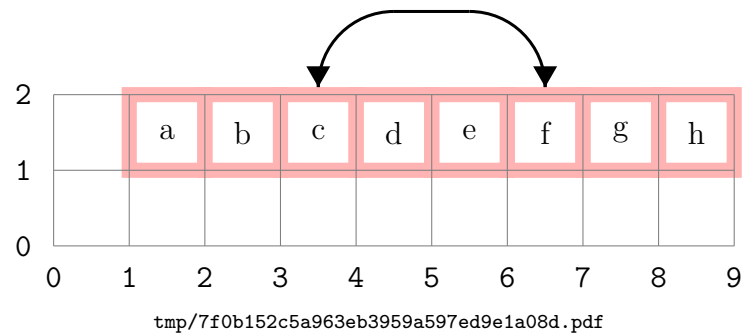
Using []:

```

from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1)
for x in 'abcdefgh':
    c += Rect2(x0=0, y0=0, x1=1, y1=1,
               linewidth=0.2, linecolor='red!30!white',
               label=x)
p += c
p += bend(c[2].top(), c[5].top())
p += Grid()
print p

```



Left-to-right, align bottom:

```

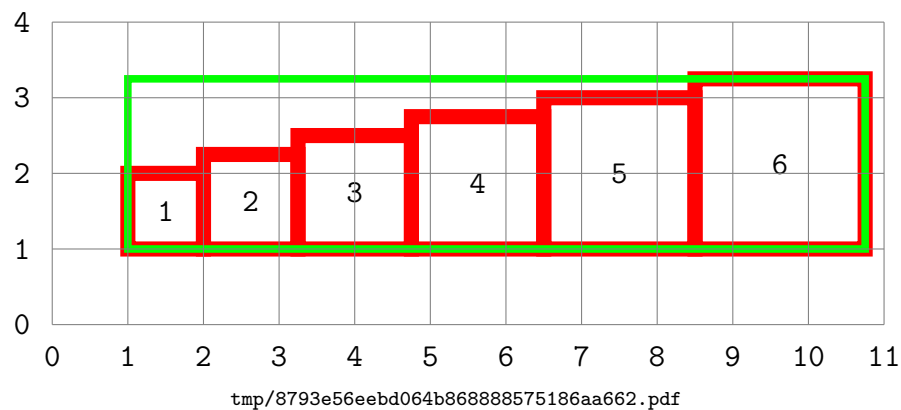
from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1)
for i, x in enumerate([1,2,3,4,5,6]):
    c += Rect2(x0=0, y0=0, x1=1+i/4.0, y1=1+i/4.0, linecolor='red', linewidth=0.2,
               label=r'\texttt{%s}}' % x)
c.layout()
p += c

# boundary of container
p += Rect2(x0=c.x0, y0=c.y0, x1=c.x1, y1=c.y1, linewidth=0.1, linecolor='green')

p += Grid()
print p

```



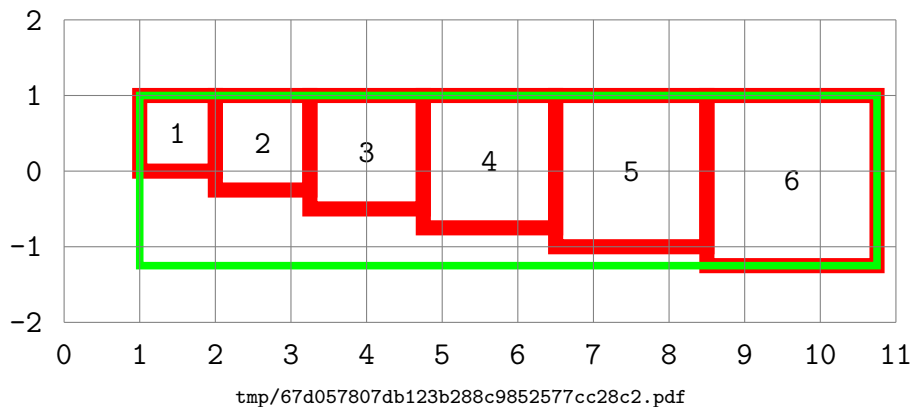
Align top, left-to-right:

```
from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1, align='top')
for i, x in enumerate([1,2,3,4,5,6]):
    c += Rect2(x0=0, y0=0, x1=1+i/4.0, y1=1+i/4.0, linecolor='red', linewidth=0.2,
               label=r'\texttt{%s}}' % x)
p += c

# boundary of container
p += Rect2(x0=c.x0, y0=c.y0, x1=c.x1, y1=c.y1, linewidth=0.1, linecolor='green')

p += Grid()
print p
```



top-to-bottom, align left:

```

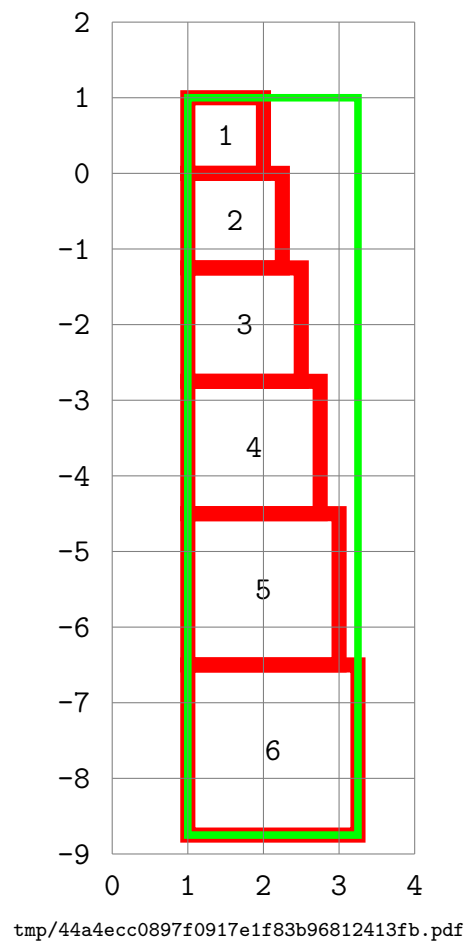
from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1, align='left', direction='top-to-bottom')
for i, x in enumerate([1,2,3,4,5,6]):
    c += Rect2(x0=0, y0=0, x1=1+i/4.0, y1=1+i/4.0, linecolor='red', linewidth=0.2,
               label=r'\texttt{%s}' % x)
p += c

# boundary of container
p += Rect2(x0=c.x0, y0=c.y0, x1=c.x1, y1=c.y1, linewidth=0.1, linecolor='green')

p += Grid()
print p

```



top-to-bottom, align right:


```

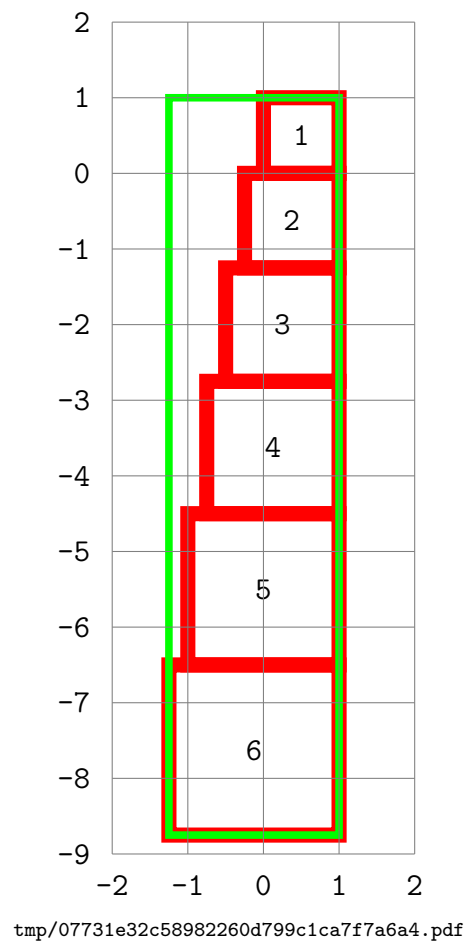
from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1, align='right', direction='top-to-bottom')
for i, x in enumerate([1,2,3,4,5,6]):
    c += Rect2(x0=0, y0=0, x1=1+i/4.0, y1=1+i/4.0, linecolor='red', linewidth=0.2,
               label=r'\texttt{%s}' % x)
p += c

# boundary of container
p += Rect2(x0=c.x0, y0=c.y0, x1=c.x1, y1=c.y1, linewidth=0.1, linecolor='green')

p += Grid()
print p

```



2-d stack

```

from latextool_basic import *
p = Plot()

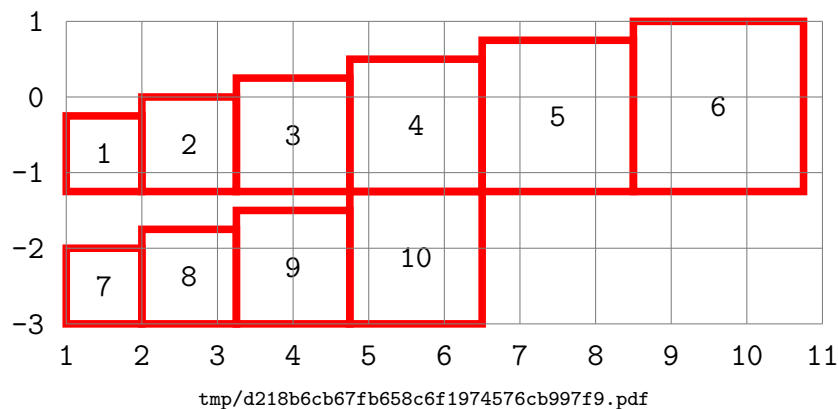
c0 = RectContainer(x=1, y=1, align='bottom', direction='left-to-right')
for i, x in enumerate([1,2,3,4,5,6]):
    c0 += Rect2(x0=0, y0=0, x1=1+i/4.0, y1=1+i/4.0, linecolor='red', linewidth=0.1,
                label=r'\texttt{%s}' % x)

c1 = RectContainer(x=1, y=1, align='bottom', direction='left-to-right')
for i, x in enumerate([7,8,9,10]):
    c1 += Rect2(x0=0, y0=0, x1=1+i/4.0, y1=1+i/4.0, linecolor='red', linewidth=0.1,
                label=r'\texttt{%s}' % x)

C = RectContainer(x=1, y=1, align='left', direction='top-to-bottom')
C += c0; c0.x = c0.x0; c0.y = c0.y0; c0.layout()
C += c1; c1.x = c1.x0; c1.y = c1.y0; c1.layout()

p += C;
p += Grid()
print p

```



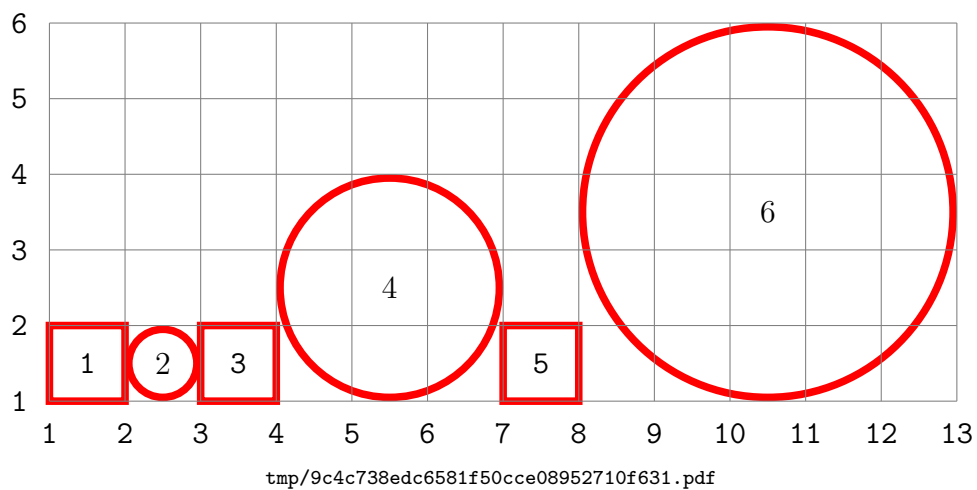
```

from latextool_basic import *
p = Plot()

c = RectContainer(x=1, y=1)
for i, x in enumerate([1,2,3,4,5,6]):
    if i % 2 == 0:
        c += Rect2(x0=0, y0=0, x1=1, y1=1, linecolor='red', linewidth=0.1,
                    label=r'\texttt{%s}' % x)
    else:
        c += Circle(x=0.5, y=0.5, r=0.5 + i/2, label=x, linecolor='red', li
newwidth=0.1)

p += c
p += Grid()
print p

```



Vertical array and pointers

```

L = 0.1 # line width
from latextool_basic import *
p = Plot()

v = RectContainer(x=0, y=0, align='left', direction='top-to-bottom')
for i in ['', '', '', '']:
    v += Rect2(0,0,1,1,s=i, linewidth=0.1, linecolor='red')

p += v

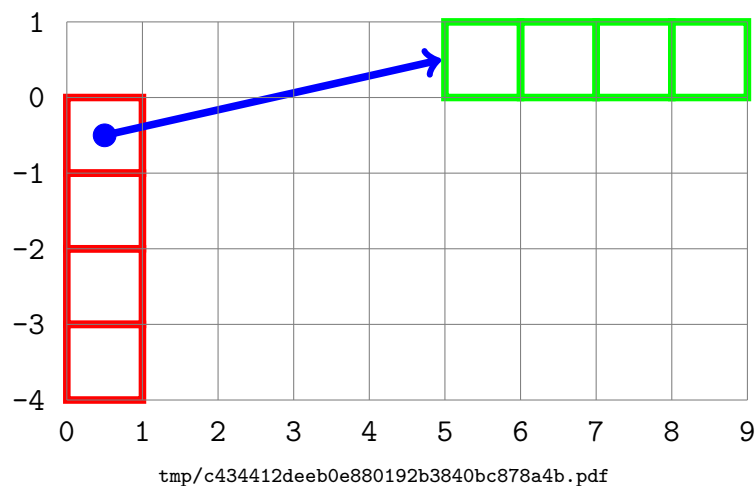
h = RectContainer(x=5, y=0, align='bottom', direction='left-to-right')
for i in ['', '', '', '']:
    h += Rect2(0,0,1,1,s=i, linewidth=0.1, linecolor='green')

p += h

p += Line(x0=v[0].centerx(), y0=v[0].centery(),
          x1=h[0].leftx(), y1=h[0].centery(),
          linewidth=0.1, linecolor='blue',
          endstyle='->', startstyle='dot', r=0.01)

p += Grid()
print p

```



DFA/NFA:

```

from latextool_basic import *
p = Plot()

h = RectContainer(x=0, y=0, align='bottom', direction='left-to-right')
for i in ['a','b','a','a','$\sqcup$', '$\sqcup$', '$\sqcup$',]:
    h += Rect2(0,0,1,1,
               label= r'\texttt{%s}' % i,
               linewidth=0.1, linecolor='red')
p += h

p += Line(x0=h[-1].x1, y0=h[-1].y1, x1=h[-1].x1+0.5, y1=h[-1].y1,
          linewidth=0.1, linecolor='red')

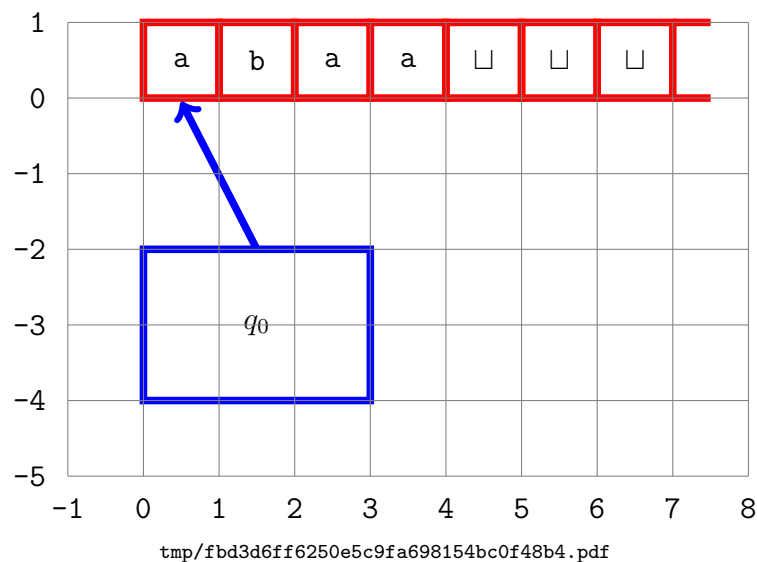
p += Line(x0=h[-1].x1, y0=h[-1].y0, x1=h[-1].x1+0.5, y1=h[-1].y0,
          linewidth=0.1, linecolor='red')

dfa = Rect2(0,-4,3,-2, linewidth=0.1, linecolor='blue', label='$q_0$')
p += dfa

p += Line(x0=dfa.centerx(), y0=dfa.y1,
          x1=h[0].centerx(), y1=h[0].bottomy(),
          linewidth=0.1, linecolor='blue', endstyle='->')

p += Grid()
print p

```



PDA:

```

from latextool_basic import *
p = Plot()
h = RectContainer(x=0, y=0, align='bottom', direction='left-to-right')
for i in ['a', 'b', 'a', 'a', '$\sqcup$', '$\sqcup$', '$\sqcup$',]:
    h += Rect2(0,0,1,1,
               label= r'\texttt{%s}' % i,
               linewidth=0.1, linecolor='red')
p += h

p += Line(x0=h[-1].x1, y0=h[-1].y1, x1=h[-1].x1+0.5, y1=h[-1].y1,
          linewidth=0.1, linecolor='red')
p += Line(x0=h[-1].x1, y0=h[-1].y0, x1=h[-1].x1+0.5, y1=h[-1].y0,
          linewidth=0.1, linecolor='red')

pda = Rect2(0,-4,3,-2, linewidth=0.1, linecolor='blue', label='$q_0$')
p += pda

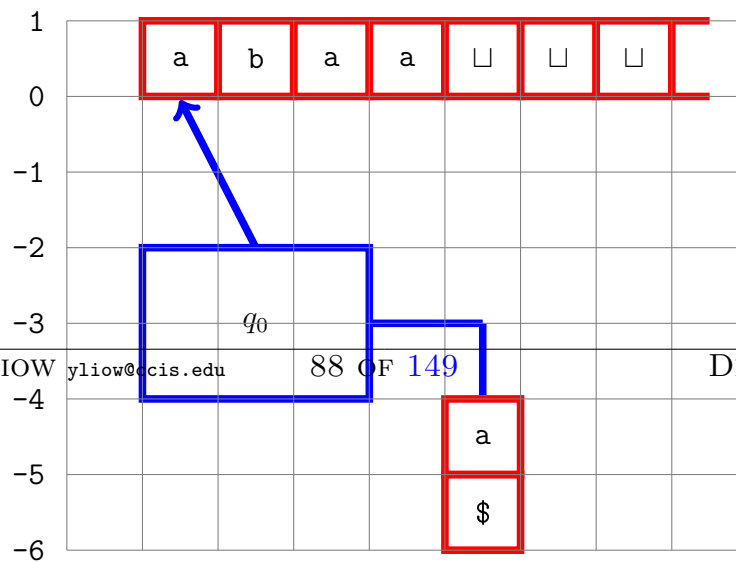
p += Line(x0=pda.centerx(), y0=pda.y1,
          x1=h[0].centerx(), y1=h[0].bottomy(),
          linewidth=0.1, linecolor='blue', endstyle='->')

v = RectContainer(x=4, y=-4, align='left', direction='top-to-bottom')
for i in ['a', '\$',]:
    v += Rect2(0,0,1,1,
               label= r'\texttt{%s}' % i, linewidth=0.1, linecolor='red')
p += v

p += Line(x0=pda.x1,y0=pda.centery(), x1=v[0].centerx(), y1=pda.centery(),
          linewidth=0.1, linecolor='blue')
p += Line(x0=v[0].centerx(),y0=pda.centery(),x1=v[0].centerx(),y1=v[0].topy(),
          linewidth=0.1, linecolor='blue')

p += Grid()
print p

```



File: `snipped-array.tex`

18 Snipped Array

Snipped array (snip in the end):

```
from latextool_basic import *
p = Plot()

c = SnippedArray(x=1, y=1, xs=[1, 2, 3, '...'])
p += c
print p
```

1	2	3	...
---	---	---	-----

tmp/f894ecc88eae39feb6ec9081c2a31fca.pdf

Snipped array (snip in the end):

```
from latextool_basic import *
p = Plot()

c = SnippedArray(x=1, y=1, xs=[1, 2, 3, '...', 4, 5, 6, '...', 7, 8])
p += c
print p
```

1	2	3	...	4	5	6	...	7	8
---	---	---	-----	---	---	---	-----	---	---

tmp/49d25ef8eac1a059032482bef8c02e9d.pdf

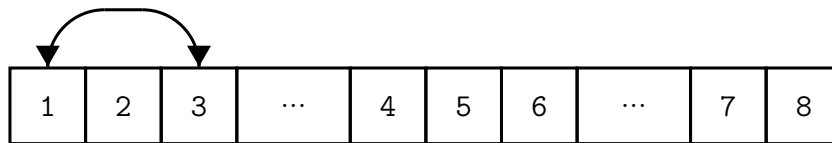
Snipped array (multiple snips):

```

from latextool_basic import *
p = Plot()

c = SnippedArray(x=1, y=1, xs=[1, 2, 3, '...', 4, 5, 6, '...', 7, 8])
p += c
p += bend(c[0].top(), c[2].top(), dy=0.75)
print p

```



tmp/c8263ab8f063d09f6be8a0a1304a9f9f.pdf

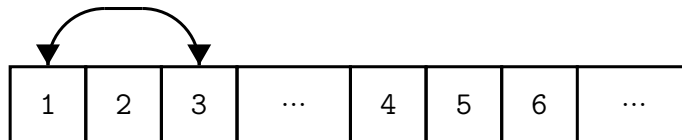
Snipped array (snip at the end):

```

from latextool_basic import *
p = Plot()

c = SnippedArray(x=1, y=1, xs=[1, 2, 3, '...', 4, 5, 6, '...'])
p += c
p += bend(c[0].top(), c[2].top(), dy=0.75)
print p

```



tmp/93925e6ca4fa223ee053aa3ac96a2da9.pdf

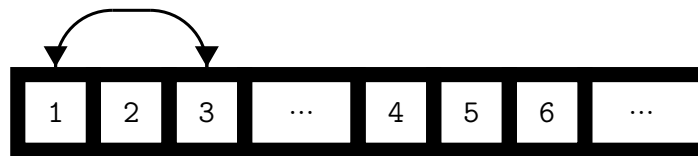
Snipped array (snip at the end, with linewidth):

```

from latextool_basic import *
p = Plot()

c = SnippedArray(x=1, y=1, xs=[1, 2, 3, '...', 4, 5, 6, '...'], linewidth=0.2)
p += c
p += bend(c[0].top(), c[2].top(), dy=0.75)
print p

```



tmp/39845a191b4675ba053f476824e03f1c.pdf

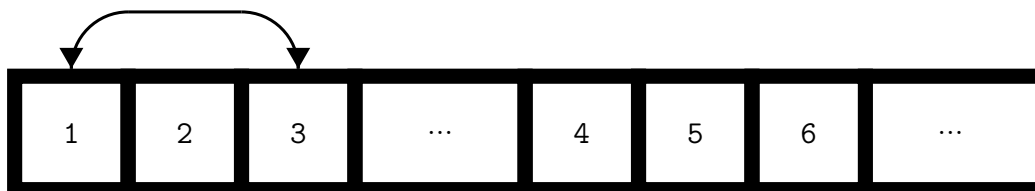
Snipped array (width and height):

```

from latextool_basic import *
p = Plot()

c = SnippedArray(x=1, y=1, xs=[1, 2, 3, '...', 4, 5, 6, '...'],
    width=1.5, height=1.5, linewidth=0.2)
p += c
p += bend(c[0].top(), c[2].top(), dy=0.75)
print p

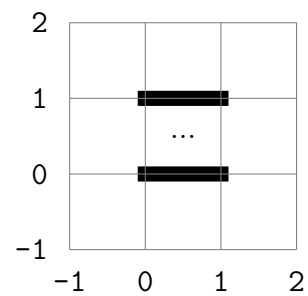
```



tmp/9d1f87ef92b28017c5fe584f27e9c087.pdf

Snipped array (snip at the end, with linewidth):

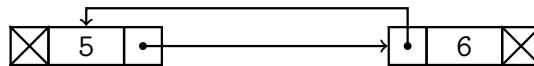
```
from latextool_basic import *  
p = Plot()  
  
c = Rect2NoLeftRight(x0=0, y0=0, x1=1, y1=1, linewidth=0.2, label='...')  
p += c  
p += Grid(-1, -1, 2, 2)  
print p
```



tmp/9d8487de695462bccc689cc1dbc31e2d.pdf

File: singlylinkedlist.tex

19 Singly Linked List



File: doublylinkedlist.tex

20 Doubly linked list

PYTHON ERROR. See source, stderr, stdout below

```

from latextool_basic import *
class DLNodeRect:
    L = 0.05 # linewidth
    def __init__(self, x0=0, y0=0, w=0, h=0, label='',
                  prev=None, next=None):
        L = DLNodeRect.L
        self.x0, self.y0 = x0, y0
        self.w, self.h = w, h
        self.label = label
        self.prev, self.next = prev, next
        c = RectContainer(x0=x0, y0=y0)
        c += Rect2(x0=0, y0=0, x1=w, y1=h, linewidth=L, label='')
        c += Rect2(x0=0, y0=0, x1=w, y1=h, linewidth=L, label=label)
        c += Rect2(x0=0, y0=0, x1=w, y1=h, linewidth=L, label='')
        self.c = c
    def next_rect(self): return self.c[2]
    def key_rect(self): return self.c[1]
    def prev_rect(self): return self.c[0]
    def __str__(self):
        L = DLNode.L
        s = str(self.c)
        if self.next != None:
            a = self.next_rect().center()
            b = self.next.prev_rect().left()
            s += line(points=[a, b], linewidth=L,
                      endstyle='->', startstyle='dot', arrowstyle='triangle
')

        else: # self.next is None
            r = self.next_rect()
            s += line(points=[r.topleft(), r.bottomright()], linewidth=L)
            s += line(points=[r.topright(), r.bottomleft()], linewidth=L)
        if self.prev != None:
            a = self.prev_rect().center()
            b = self.prev.key_rect().bottom()
            c = a[0], a[1] - 1
            d = b[0], c[1]
            s += line(points=[a, c, d, b], linewidth=L,
                      endstyle='->', startstyle='dot', arrowstyle='triangle
')

        else: # self.prev is None
            r = self.prev_rect()
            s += line(points=[r.topleft(), r.bottomright()], linewidth=L)
            s += line(points=[r.topright(), r.bottomleft()], linewidth=L)
        return s

class DLNode:

```

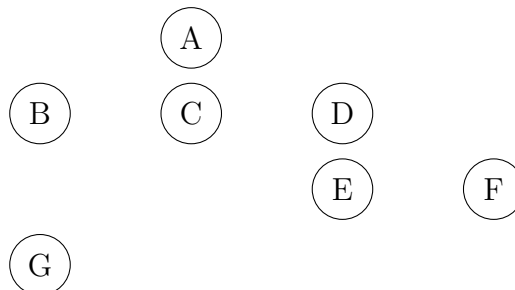

File: position.tex

21 positions

The `positions` function computes positions.

```
from latextool_basic import positions
d = positions(r'''
  A
B C D
  E F
G
''')

from latextool_basic import Plot, Circle
p = Plot()
for label, pos in d.items():
    x, y = pos
    p += Circle(x=x, y=y, r=0.4, label=label)
print p
```



tmp/f5442629d66cfbd227df35856c093582.pdf

You can scale the x and y axes:

```

from latextool_basic import positions
d = positions(r'''
  A
B C D
  E F
G
''', xscale=2.0, yscale=0.5)

from latextool_basic import Plot, circle
p = Plot()
for label, pos in d.items():
    x, y = pos
    p.add(circle, x=x, y=y, r=0.25, label=label)
print p

```



tmp/9a22e760080015821205d97b2bb3ddae.pdf

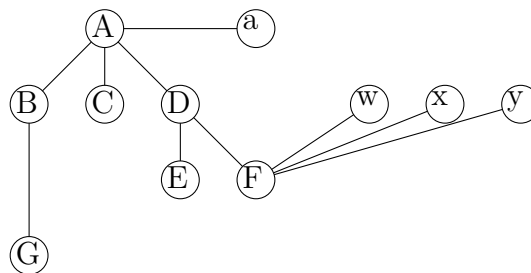
File: tree.tex

22 tree

```
from latextool_basic import positions
d = positions(r"""
  A   a
B C D   w x y
      E F
G
""", xscale=0.5)

edges = {'A': ['B', 'C', 'D', 'a'],
        'B': ['G'],
        'D': ['E', 'F'],
        'F': ['w', 'x', 'y'],
        }

from latextool_basic import Plot, tree
p = Plot()
p.add(tree(d, edges=edges))
print p
```



tmp/4728a320501beeace354f62e6ff7620e.pdf

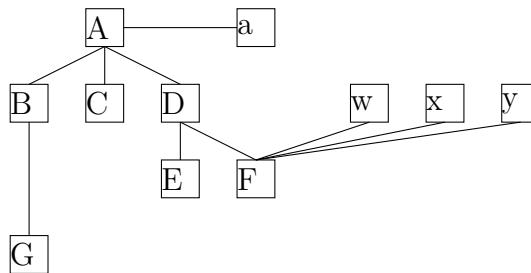
```

from latextool_basic import positions
d = positions(r"""
  A   a
B C D   w x y
    E F
G
""", xscale=0.5)

edges = {'A': ['B', 'C', 'D', 'a'],
        'B': ['G'],
        'D': ['E', 'F'],
        'F': ['w', 'x', 'y'],
        }

from latextool_basic import Plot, tree
p = Plot()
p.add(tree(d, edges=edges, node_shape='rect'))
print p

```



tmp/b74cfb72d43bef4c72655e49da14d3fd.pdf

```

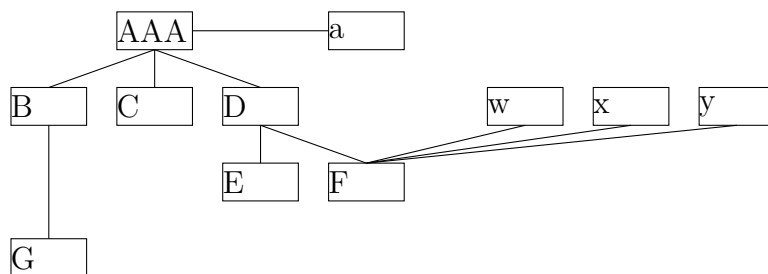
from latextool_basic import positions
d = positions(r"""
  A   a
B C D   w x y
  E F
G
""", xscale=0.7)

edges = {'A': ['B', 'C', 'D', 'a'],
        'B': ['G'],
        'D': ['E', 'F'],
        'F': ['w', 'x', 'y'],
        }

from latextool_basic import Plot, tree
p = Plot()
p.add(tree(d,
          width=1, height=0.5,
          node_label={'A': 'AAA'},
          node_shape='rect',
          edges=edges))

print p

```



tmp/110bed55994ceb027689b0a8b1f0fd75.pdf

Auto adjust node position

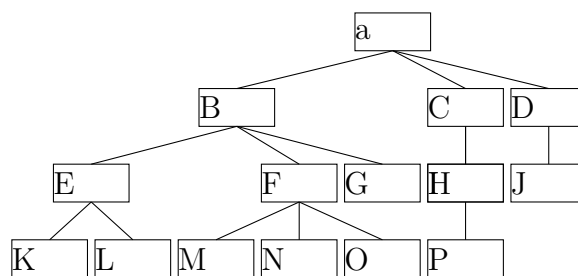
```

from latextool_basic import positions
d = positions(r"""
    A
  B  C  D
E F G H  I J
K L M N O P
""", xscale=0.7)

edges = {'A': ['B', 'C', 'D'], 'B': ['E', 'F', 'G'],
        'C': ['H', 'I'],      'D': ['J'],
        'E': ['K', 'L'],      'F': ['M', 'N', 'O'],
        'I': ['P'],
        }

from latextool_basic import Plot, tree
p = Plot()
p.add(tree(d,
          hor_sep=0.1,
          width=1, height=0.5,
          node_label={'A': 'a'},
          node_shape='rect',
          edges=edges,
          autoadjust=True))
print p

```



tmp/52113effd1e1320f52be67de39a73d61.pdf

File: tree2.tex

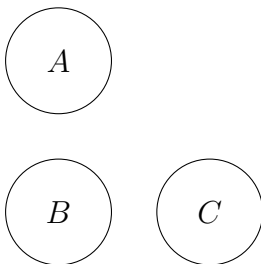
23 tree2

vertices: ['A', 'B', 'C', 'D']

File: graph.tex

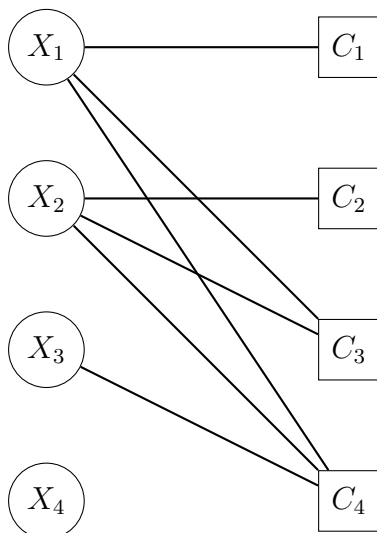
24 graph

```
\begin{python}
from latextool_basic import graph
print graph(layout="""
A
B C
""")
\end{python}
```



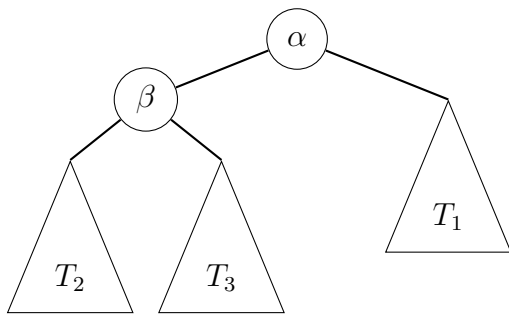
nodes and labels:

```
\begin{python}
from latextool_basic import graph
print graph(layout='''
    X1  C1
    X2  C2
    X3  C3
    X4  C4
''',
minimum_size='8mm',
edges='X1-C1,X2-C2,X1-C3,X2-C3,X1-C4,X2-C4,X3dashedC4',
X1='label=$X_1$',
X2='label=$X_2$',
X3='label=$X_3$',
X4='label=$X_4$',
C1='shape=rectangle, label=$C_1$',
C2='shape=rectangle, label=$C_2$',
C3='shape=rectangle, label=$C_3$',
C4='shape=rectangle, label=$C_4$',
)
\end{python}
```



Isoceles triangle as subtree:

```
graph(layout='''
    A
  B   C
D E
''',
minimum_size='8mm',
edges='A-B,A-C',
A=r'label=$\alpha$',
B=r'label=$\beta$',
C=r'shape=tree, minimum height=2cm, label=$T_1$',
D=r'shape=tree, minimum height=2cm, label=$T_2$',
E=r'shape=tree, minimum height=2cm, label=$T_3$',
)
```

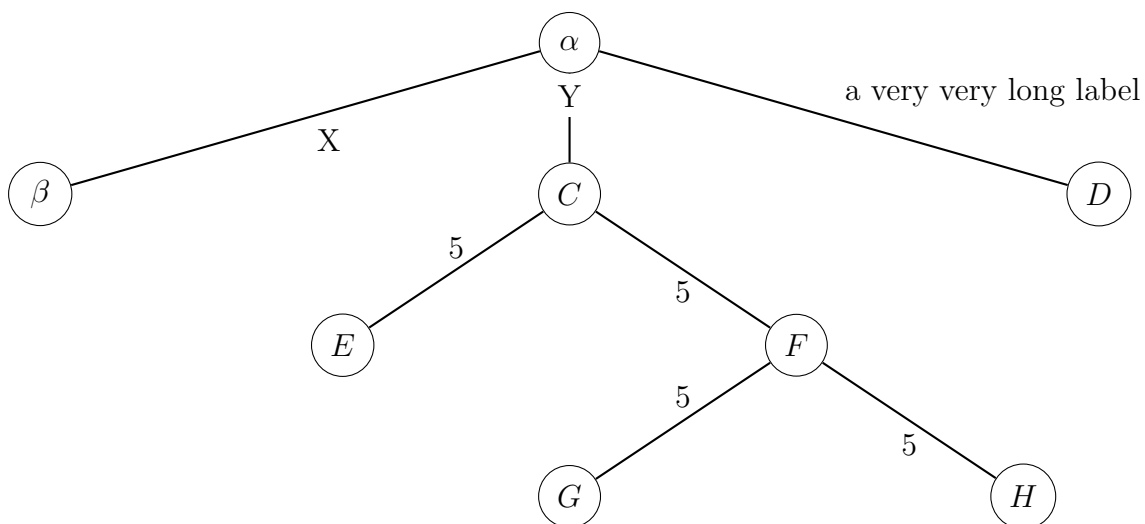


Edge labels:

```

from latextool_basic import graph
print graph(layout='''
      A
    B   C   D
      E   F
      G   H
      I
''',
minimum_size='8mm',
edges='A-B,A-C,A-D,C-E,C-F,F-G,F-H,G-I',
A=r'label=$\alpha$',
B=r'label=$\beta$',
edge_label={('A','B':{'label':'X'},
              ('A','C':{'label':'Y', 'style':'pos=0.25,fill=white'},
              ('A','D':{'label':'a very very long label'},
              ('C','E':{'label':'5', 'style':'pos=0.5,above'},
              ('C','F':{'label':'5', 'style':'pos=0.5,below'},
              ('F','G':{'label':'5', 'style':'pos=0.5,above,inner sep=1mm,circle'},
              ('F','H':{'label':'5', 'style':'pos=0.5,below,inner sep=1mm,circle'},
              ('G','I':{'label':'abcdef', 'style':'sloped'},
            },
)

```



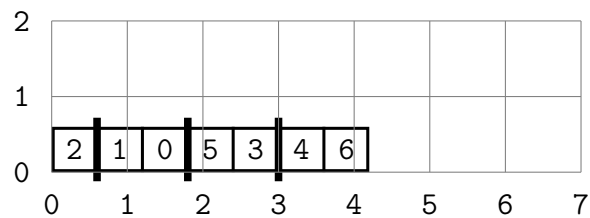
<http://tex.stackexchange.com/questions/86386/edge-nodes-auto-and-node-distance>

File: chunkedarray.tex

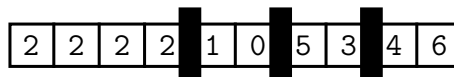
25 chunkedarray

This is for list of lists.

```
from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[2],[1,0],[5,3],[4,6]])
p.add(s)
p.add(grid, x0=0, y0=0, x1=7, y1=2)
print p
```



```
from latextool_basic import Plot, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  pipewidth=0.3,
                  pipeheight=1,
                  arr=[[2,2,2,2],[1,0],[5,3],[4,6]])
p.add(s)
print p
```



One empty list in front:

```
from latextool_basic import Plot, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[ ], [1,0], [3,5], [4,6]])
p.add(s)
print p
```

	1	0	3	5	4	6
--	---	---	---	---	---	---

2 empty list in front:

```
from latextool_basic import Plot, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=1,
                  cellheight=0.6,
                  arr=[[ ], [ ], [1,0], [3,5], [4,6]])
p.add(s)
print p
```

		1	0	3	5	4	6
--	--	---	---	---	---	---	---

3 empty list in front:

```
from latextool_basic import Plot, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=1.6,
                  cellheight=0.6,
                  arr=[[ ], [ ], [ ], [1,0], [3,5], [4,6]])
p.add(s)
print p
```

			1	0	3	5	4	6
--	--	--	---	---	---	---	---	---

One empty list in the middle:


```

from latextool_basic import Plot, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[2],[1,0],[],[4,6]])
p.add(s)
print p

```

2	1	0		4	6
---	---	---	--	---	---

1 empty list at the end:

```

from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[2,3],[1,0],[4,5],[[]]])
p.add(s)
print p

```

2	3	1	0	4	5
---	---	---	---	---	---

2 empty lists at the end:

```

from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[2,3],[1,0],[4,5],[[]],[[]]])
p.add(s)
print p

```

2	3	1	0	4	5
---	---	---	---	---	---

2 empty lists at the end and in front:

```

from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[ ], [ ], [2,3], [1,0], [4,5], [ ], [ ]])
p.add(s)
print p

```

2	3	1	0	4	5
---	---	---	---	---	---

3 empty lists in front:

```

from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[ ], [ ], [ ], [2,3], [1,0], [4,5], [ ], [ ], [ ]])
p.add(s)
print p

```

2	3	1	0	4	5
---	---	---	---	---	---

Two empty lists in the middle:

```

from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=1.2,
                  cellheight=0.6,
                  arr=[[2,3], [6,7], [ ], [ ], [1,0], [4,5]])
p.add(s)
print p

```

2	3	6	7	1	0	4	5
---	---	---	---	---	---	---	---

3 empty lists in the middle:

```

from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=1.2,
                  cellheight=0.6,
                  arr=[[2,3],[6,7],[ ],[ ],[ ],[1,0],[4,5]])
p.add(s)
print p

```

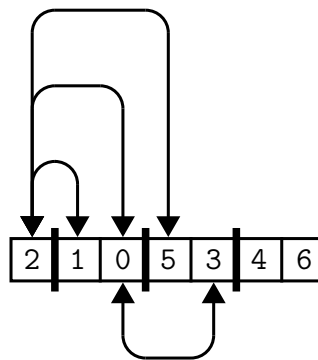


Swaps:

```

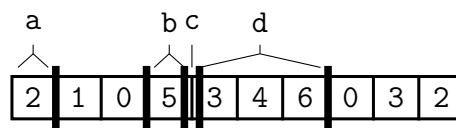
from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                  cellheight=0.6,
                  arr=[[2],[1,0],[5,3],[4,6]],
                  swaps=[(0,1,1),(0,2,2),(0,3,3),(2,4,-1)])
p.add(s)
print p

```



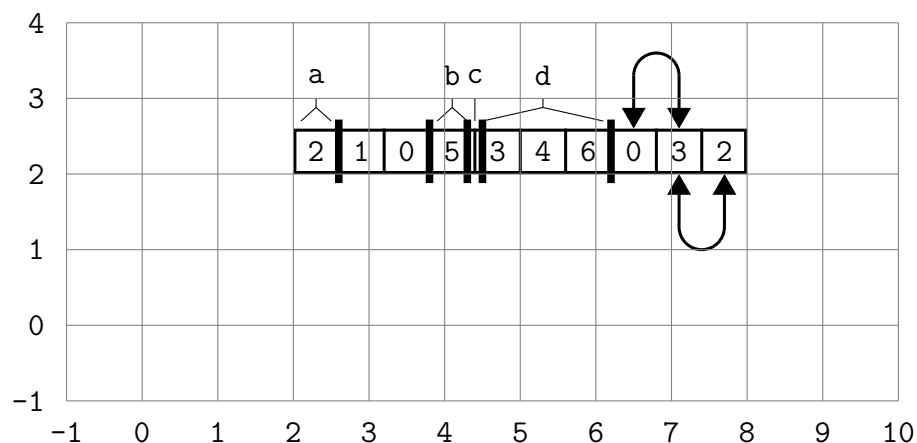
Labels for chunks

```
from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                 cellheight=0.6,
                 arr=[[2],[1,0],[5],[ ],[3,4,6],[0,3,2]],
                 chunklabels=['a', ' ', 'b', 'c','d'])
p.add(s)
print p
```



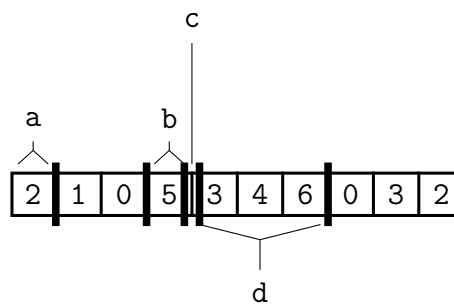
Labels for chunks with x,y:

```
from latextool_basic import Plot, chunkedarray, Grid
p = Plot()
s = chunkedarray(x=2, y=2,
                 cellwidth=0.6,
                 cellheight=0.6,
                 arr=[[2],[1,0],[5],[ ],[3,4,6],[0,3,2]],
                 swaps=[(7,8,1),(8,9,-1)],
                 chunklabels=['a', ' ', 'b', 'c','d'])
p.add(s)
p += Grid(x0=-1,y0=-1,x1=10,y1=4)
print p
```



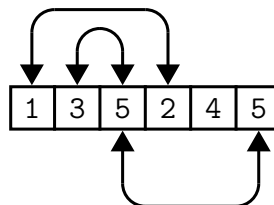
Labels with heights:

```
from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                 cellheight=0.6,
                 arr=[[2],[1,0],[5],[ ],[3,4,6],[0,3,2]],
                 chunklabels=['a', ' ', 'b', ('c',2),('d',-1)])
p.add(s)
print p
```



Using chunkedarray as array with swaps:

```
from latextool_basic import Plot, grid, chunkedarray
p = Plot()
s = chunkedarray(cellwidth=0.6,
                 cellheight=0.6,
                 arr=[[1,3,5,2,4,5]],
                 swaps=[(0,3,1), (1,2,0.75), (2,5,-1)])
)
p.add(s)
print p
```



TODO: unify API for chunked and regulars.

TODO: avoid collision of arrow tips and edge crossing.

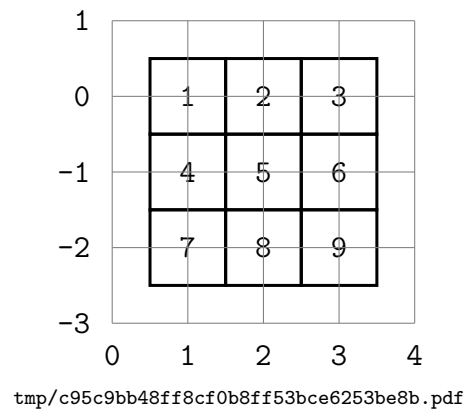
TODO: label arrow

TODO: label array

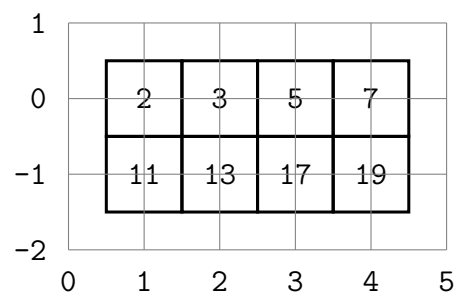
File: array2d.tex

26 2D Array

```
from latextool_basic import *  
p = Plot()  
  
c = Array2d(x=0.5, y=0.5, xs=[[1,2,3],[4,5,6],[7,8,9]])  
p += c  
p += Grid()  
print p
```

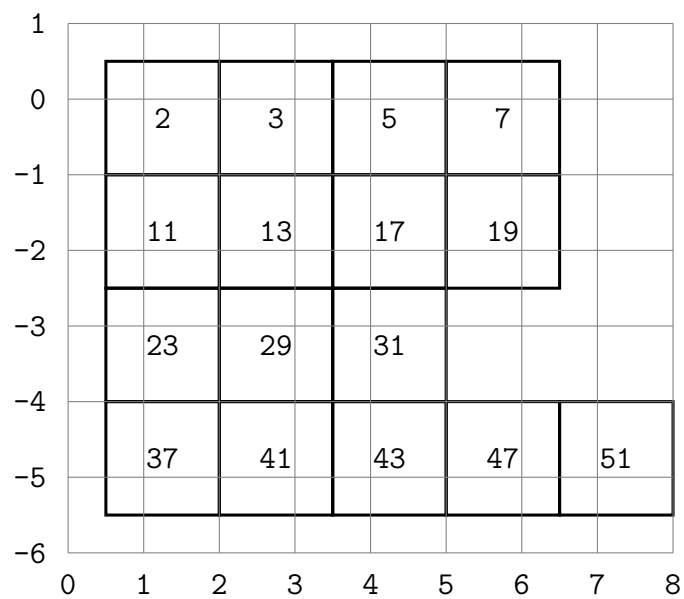


```
from latextool_basic import *  
p = Plot()  
a = Array2d(0.5, 0.5, [[2, 3, 5, 7], [11,13,17,19]])  
  
p += a  
p += Grid()  
print p
```




```
from latextool_basic import *
p = Plot()
a = Array2d(0.5, 0.5, width=1.5, height=1.5, xs=[[2, 3, 5, 7], [11,13,17,19]
], [23, 29, 31], [37,41,43,47,51]])

p += a
p += Grid()
print p
```



```

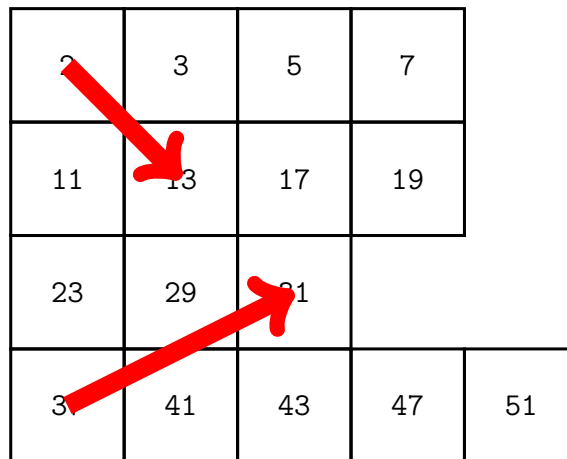
from latextool_basic import *
p = Plot()
a = Array2d(0, 0, width=1.5, height=1.5, xs=[[2, 3, 5, 7], [11,13,17,19], [
23, 29, 31], [37,41,43,47,51]])

def l((r0,c0), (r1,c1)):
    return Line(a[r0][c0].centerx(), a[r0][c0].centery(),
                a[r1][c1].centerx(), a[r1][c1].centery(),
                linecolor='red', linewidth=0.25, endstyle='->')

p += a
p += l((0,0),(1,1))
p += l((3,0),(2,2))

print p

```

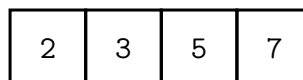


Test Array2d drawing 1d array:

```

from latextool_basic import *
p = Plot()
p += Array2d(0, 0, width=1, height=1, xs=[[2, 3, 5, 7]])
print p

```



The split operation in mergesort:

```

from latextool_basic import *
p = Plot()

def arr(x, y, xs):
    return Array2d(x, y, width=1, height=1, xs=xs)

def string(x, y, s):
    return Rect2(x0=x, y0=y, x1=x, y1=y, linewidth=0, label = s)

def _line(x0, y0, x1, y1):
    gap = 0.1
    y0 -= gap
    y1 += gap
    return Line(x0, y0, x1, y1, endstyle='->', linewidth=0.05)

xs = [2, 5, 3, 7]; left = xs[:len(xs)/2]; right = xs[len(xs)/2:]
common_len = max(len(left), len(right))

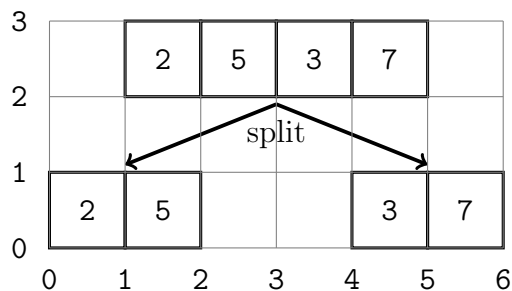
x = 1; y = 3
a = arr(x, y, xs=[xs])
midx = a.bottom()[0]

b = arr(midx - 1 - common_len, y - 2, xs=[left])
c = arr(midx + 1, y - 2, xs=[right])
d = string(midx, 1.5, "split")
p += a; p += b; p += c; p += d

x0, y0 = a.bottom()
x1, y1 = midx - 1 - common_len/2.0, y0 - 1
p += _line(x0, y0, x1, y1)
x1, y1 = midx + 1 + common_len/2.0, y0 - 1
p += _line(x0, y0, x1, y1)

p += Grid()
print p

```

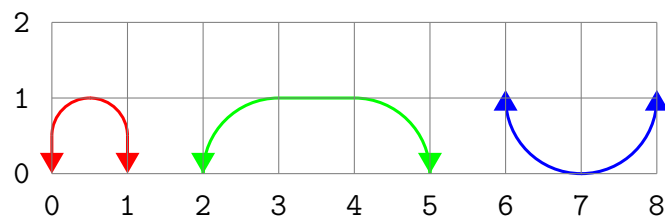


File: bend.tex

27 bend

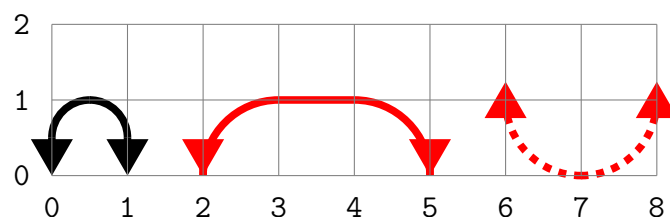
Basically used for drawing swaps in array sorting.

```
from latextool_basic import Plot, Grid, bend
p = Plot()
p += bend([0,0], [1, 0], dy=1, linecolor='red')
p += bend([2,0], [5, 0], dy=1, linecolor='green')
p += bend([6,1], [8, 1], dy=-1, linecolor='blue')
p += Grid(x0=0, y0=0, x1=8, y1=2)
print p
```



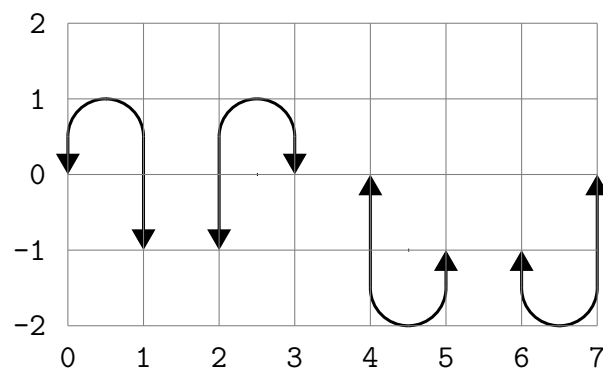
You can specify linewidth, linecolor, linestyle:

```
from latextool_basic import Plot, Grid, bend
p = Plot()
p += bend([0,0], [1, 0], dy=1, linewidth=0.1)
p += bend([2,0], [5, 0], dy=1, linewidth=0.1, linecolor='red')
p += bend([6,1], [8, 1], dy=-1, linewidth=0.1, linecolor='red',
          linestyle='dashed')
p += Grid(x0=0, y0=0, x1=8, y1=2)
print p
```



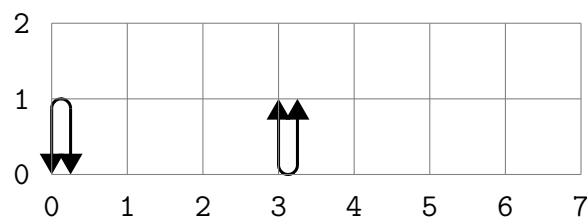
The y values can be different:

```
from latextool_basic import Plot, Grid, bend
p = Plot()
p += bend([0,0], [1, -1], dy=1)
p += bend([2,-1], [3, 0], dy=1)
p += bend([4,0], [5, -1], dy=-1)
p += bend([6,-1], [7, 0], dy=-1)
p += Grid(x0=0, y0=-2, x1=7, y1=2)
print p
```



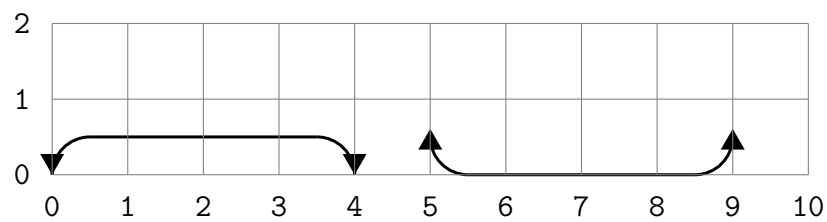
default radius (dy too large):

```
from latextool_basic import Plot, Grid, bend
p = Plot()
s0 = bend([0,0], [0.25,0], dy=1)
s1 = bend([3,1], [3.25,1], dy=-1)
p.add(s0)
p.add(s1)
p += Grid(x0=0, y0=0, x1=7, y1=2)
print p
```



default radius (dx too large):

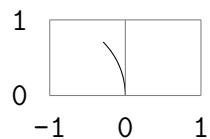
```
from latextool_basic import Plot, Grid, bend
p = Plot()
s0 = bend([0,0], [4,0], dy=0.5)
s1 = bend([5,0.5], [9,0.5], dy=-0.5)
p.add(s0)
p.add(s1)
p += Grid(x0=0, y0=0, x1=10, y1=2)
print p
```



File: arc.tex

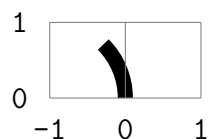
28 arc

```
from latextool_basic import Plot, Grid, arc
p = Plot()
p += arc(x=0, y=0, r=1, angle0=0, angle1=45)
p += Grid(x0=-1, y0=0, x1=1, y1=1)
print p
```



Set linewidth:

```
from latextool_basic import Plot, Grid, arc
p = Plot()
p += arc(x=0, y=0, r=1, angle0=0, angle1=45, linewidth=0.2)
p += Grid(x0=-1, y0=0, x1=1, y1=1)
print p
```

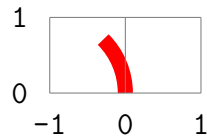


Set color:


```

from latextool_basic import Plot, Grid, arc
p = Plot()
p += arc(x=0, y=0, r=1, angle0=0, angle1=45,
         linewidth=0.2, linecolor='red')
p += Grid(x0=-1, y0=0, x1=1, y1=1)
print p

```

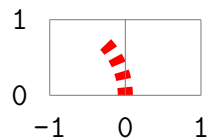


Set linestyle:

```

from latextool_basic import Plot, Grid, arc
p = Plot()
p += arc(x=0, y=0, r=1, angle0=0, angle1=45,
         linewidth=0.2, linecolor='red', linestyle='dashed')
p += Grid(x0=-1, y0=0, x1=1, y1=1)
print p

```

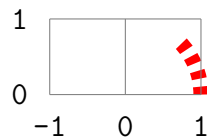


Change starting point of arc:

```

from latextool_basic import Plot, Grid, arc
p = Plot()
p += arc(x=1, y=0, r=1, angle0=0, angle1=45,
         linewidth=0.2, linecolor='red', linestyle='dashed')
p += Grid(x0=-1, y0=0, x1=1, y1=1)
print p

```



File: shell.tex

29 shell

The following are 3 ways to draw a framed verbatim environment.

METHOD 1. The following is a console latex environment:

```
[student@localhost latextool] pwd
home/student
```

Line after the console environment. Here are two such:

```
[student@localhost latextool] pwd
home/student
```

```
[student@localhost latextool] pwd
home/student
```

METHOD 2. The following is a python environment that prints a shell function call:

```
[student@localhost doc] pwd
home/student
```

Line after python environment with shell function call. Here are two such:

```
[student@localhost doc] pwd
home/student
```

```
[student@localhost doc] pwd
home/student
```

METHOD 3. The following is a python environment that prints an execute function call:

```
[student@localhost doc] pwd
home/student
```

Line after python environment with execute function call. Here are two such:

```
[student@localhost doc] pwd
home/student
```

```
[student@localhost doc] pwd  
home/student
```

Notice the vertical spacing before and after the framed window is smaller for 1. The verbatim spacing before and after the framed window for 2 and 3 are the same.

METHOD 4. The following is a python environment that prints an execute function call:

```
from latextool_basic import shell  
print shell('pwd')
```

```
[student@localhost doc] pwd  
home/student
```

Line after python environment with execute function call.

Another example:

```
from latextool_basic import shell  
print shell('ls -la c*.py')
```

```
[student@localhost doc] ls -la c*.py  
ls: cannot access c*.py: No such file or directory
```

Line after python environment with execute function call.

Execute command in another directory

```
from latextool_basic import shell  
print shell('pwd; ls -la .b*', dir='/home/student')
```

```
[student@localhost ~] pwd; ls -la .b*  
/home/student  
-rw-----. 1 student student 8629 Dec 16 09:28 .bash_history  
-rw-r--r--. 1 student student  18 Jun 22  2010 .bash_logout  
-rw-r--r--. 1 student student 176 Jan 25  2012 .bash_profile  
-rw-r--r--. 1 student student 176 Jun 22  2010 .bash_profile~  
-rw-r--r-- 1 student student 1047 Mar 19  2013 .bashrc  
-rw-r--r-- 1 student student 1032 Feb 19  2013 .bashrc~
```

no execution

```
from latextool_basic import shell
print shell('ls -la', execute=False)
```

```
[student@localhost doc] ls -la
```

Executing with error:

PYTHON ERROR. See source, stderr, stdout below

```
print x
```

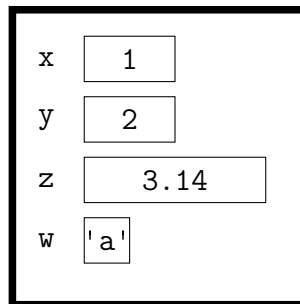
```
Traceback (most recent call last):
  File "poyzdxcl.tmp.py", line 1, in <module>
    print x
NameError: name 'x' is not defined
```

File: frame.tex

30 Frame

```
from latextool_basic import *
p = Plot()
env = [('x', 1), ('y', 2), ('z', 3.14), ('w', 'a')]
p.add(frame(env, top="main()"))
print p
```

main()



x	1
y	2
z	3.14
w	'a'

tmp/de9ccc511094d770e786a5aea648bcb4.pdf

File: test-verbatim-spacing.tex

31 Test Verbatim Spacing

First here's the vertical spacing between paragraphs. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

TEST 1. Text, console environment, text:

hello world

Line after.

TEST 2. Text, console environment, console environment, text:

hello world

hello world

Line after. Probably won't be used. Probably too much spacing between the two environments.

TEST 3. Text, python environment with console function, text:

hello world

Line after.

File: automata.tex

32 Automata

The `automata` function (in the `latextool_basic` module) spits out latex code for state diagrams which can then be modified by hand.

Here are some examples.

If you execute the following python code:

```
print automata(layout="""
    A
    """,
    A="initial|label=$q''_0$",
    )
```

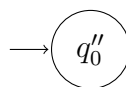
you will get this latex code as output:

```
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial] (A) at ( 4, 0) {$q''_0$};

\path[->]

;
\end{tikzpicture}
\end{center}
```

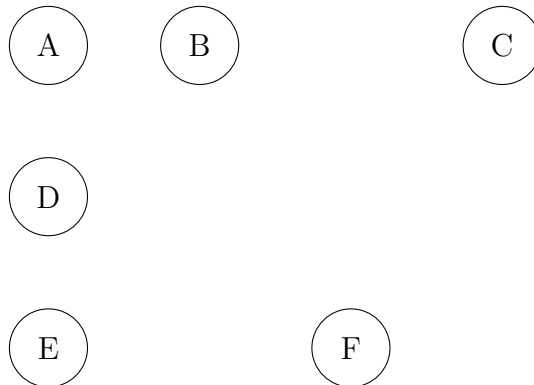
When you paste the above output into a latex document, you will get this picture:



The `layout` parameter is a string for placing the nodes. When you execute this:

```
from latextool_basic import *
automata(layout="""
A B    C
D
E    F
""")
```

you get this output:

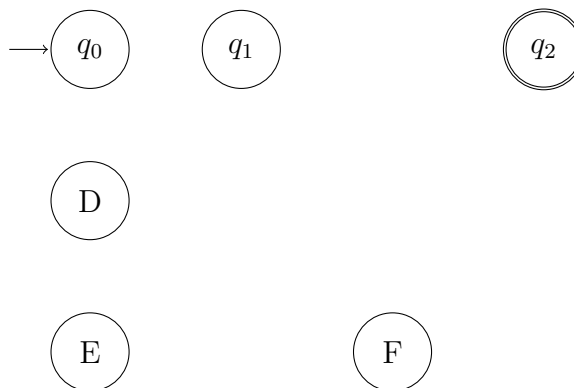


You can then put decorate each node.

```

from latextool_basic import *
print automata(layout="""
A B   C
D
E   F
""",
A = "initial|label=$q_0$",
B = "label=$q_1$",
C = "accept|label=$q_2$"
)
  
```

gives this picture



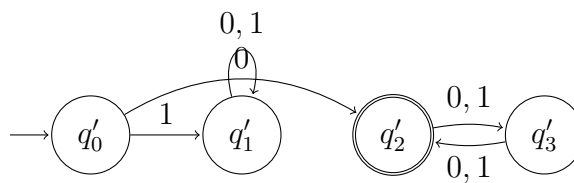
For the rest of the examples, I will just give you the python code and the corresponding picture.

DFA for “starts with 0 and has odd length”:


```

from latextool_basic import *
print automata(layout=""
A D B C
"",
edges="A,$0$,B|B,$0,1$,C|C,$0,1$,B|A,$1$,D|D,$0,1$,D",
A="initial|label=$q'_0$",
B="accept|label=$q'_2$",
C="label=$q'_3$",
D="label=$q'_1$",
)

```



Sipser 1.7c

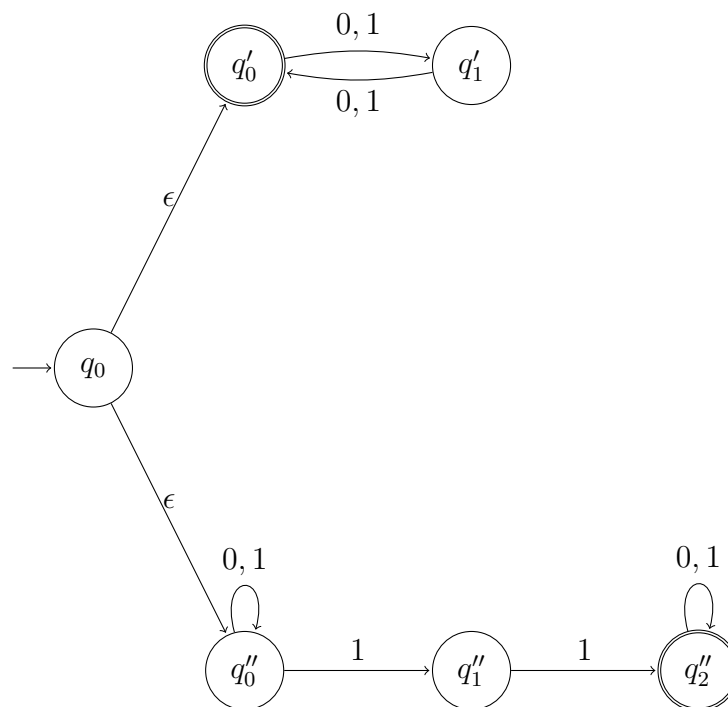
```

from latextool_basic import *
print automata(layout="""
    B  D

A

    C  E  F
""",
edges="A,$\epsilon$,B|B,$0,1$,D|D,$0,1$,B|A,$\epsilon$,C|C,$1$,E|E,$1$,F|C,$0,1$,C|F,$0,1$,F",
A="initial|label=$q_0$",
B="accept|label=$q'_0$",
C="label=$q''_0$",
D="label=$q'_1$",
E="label=$q''_1$",
F="label=$q''_2$|accept",
)

```



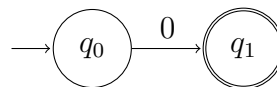
Sipser 1.7d

```

from latextool_basic import *

print automata(layout="""
A B
""",
edges="A,$0$,B",
A="initial|label=$q_0$",
B="accept|label=$q_1$",
)

```

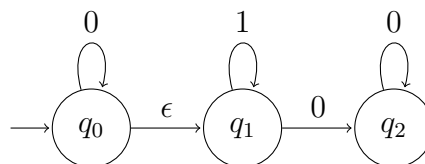


Sipser 1.7e

```

from latextool_basic import *
print automata(layout="""
A B C
""",
edges="A,$\epsilon$,B|A,$0$,A|B,$1$,B|B,$0$,C|C,$0$,C",
A="initial|label=$q_0$",
B="label=$q_1$",
C="label=$q_2$",
)

```

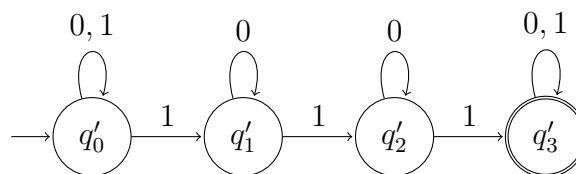


NFA for at least three 1s:

```

from latextool_basic import *
print automata(layout="""
A B C D
""",
edges="A,1,B|B,1,C|C,1,D|A,$0,1$,A|D,$0,1$,D|B,0,B|C,0,C",
A="initial|label=$q'_0$",
B="label=$q'_1$",
C="label=$q'_2$",
D="accept|label=$q'_3$",
)

```

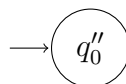


NFA for \emptyset :

```

from latextool_basic import *
print automata(layout="""
A
""",
A="initial|label=$q''_0$",
)

```

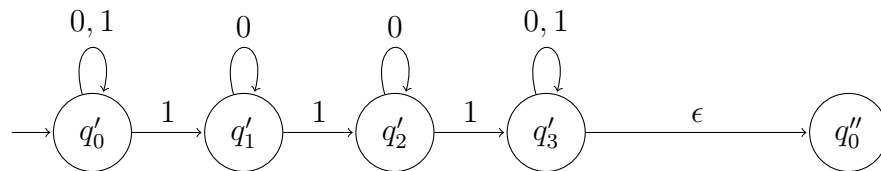


NFA for concat of “contains at least 3 1s” and \emptyset :

```

from latextool_basic import *
print automata(layout="""
A B C D   E
""",
edges="A,1,B|B,1,C|C,1,D|A,$0,1$,A|D,$0,1$,D|B,0,B|C,0,C|D,$\epsilon$,E",
A="initial|label=$q'_0$",
B="label=$q'_1$",
C="label=$q'_2$",
D="label=$q'_3$",
E="label=$q''_0$"
)

```

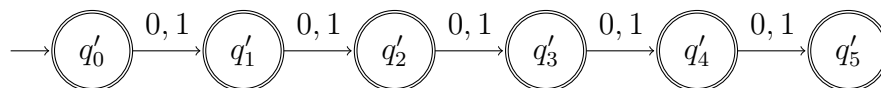


NFA for concat of “length ≤ 5 ” and “every odd position is a 1”:

```

from latextool_basic import *
print automata(layout="""
A B C D E F
""",
edges="A,$0,1$,B|B,$0,1$,C|C,$0,1$,D|D,$0,1$,E|E,$0,1$,F",
A="accept|initial|label=$q'_0$",
B="accept|label=$q'_1$",
C="accept|label=$q'_2$",
D="accept|label=$q'_3$",
E="accept|label=$q'_4$",
F="accept|label=$q'_5$",
)

```

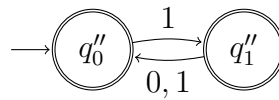


NFA for “every odd position is a 1”:

```

from latextool_basic import *
print automata(layout=""
A B
"",
edges="A,$1$,B|B,$0,1$,A",
A="accept|initial|label=$q''_0$",
B="accept|label=$q''_1$",
)

```

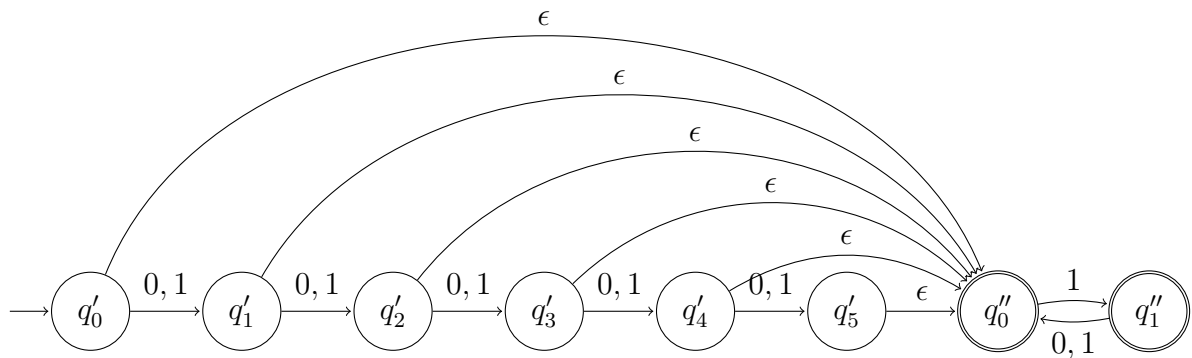


NFA for concat of “length ≤ 5 ” and “every odd position is a 1”:

```

from latextool_basic import *
print automata(layout="""
A B C D E F G H
""",
edges="A,$0,1$,B|B,$0,1$,C|C,$0,1$,D|D,$0,1$,E|E,$0,1$,F"+\
"|F,$\epsilon$,G"+\
"|A,$\epsilon$,G"+\
"|B,$\epsilon$,G"+\
"|C,$\epsilon$,G"+\
"|D,$\epsilon$,G"+\
"|E,$\epsilon$,G"+\
"|G,$1$,H|H,$0,1$,G",
A="initial|label=$q'_0$",
B="label=$q'_1$",
C="label=$q'_2$",
D="label=$q'_3$",
E="label=$q'_4$",
F="label=$q'_5$",
G="accept|label=$q''_0$",
H="accept|label=$q''_1$",
)

```

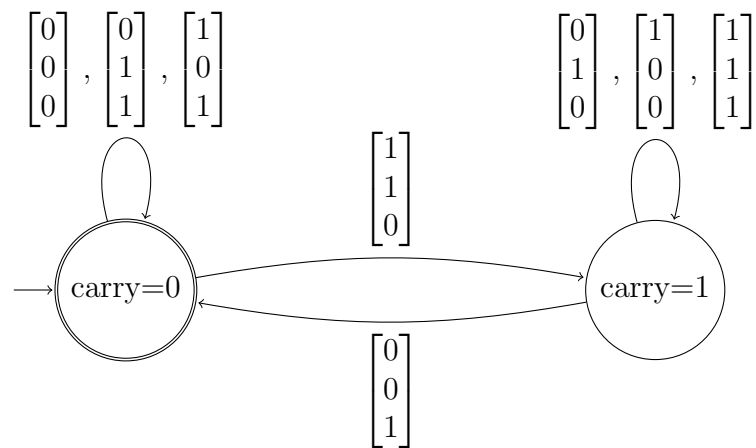


NFA with vectors as Σ :

```

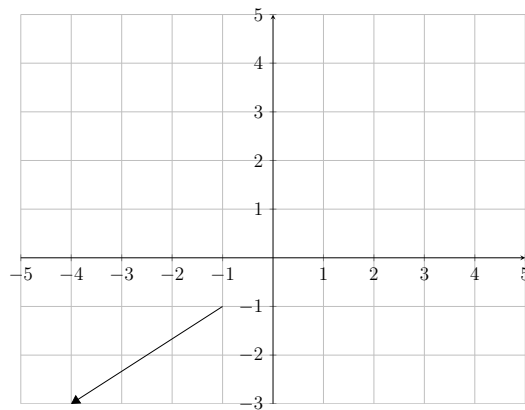
from latextool_basic import *
print automata(layout="""
A      B
""",
edges=r"A,$\begin{bmatrix}0\\0\\0\end{bmatrix},\begin{bmatrix}0\\1\\1\end{bmatrix}\end{bmatrix},\begin{bmatrix}1\\0\\1\end{bmatrix}$,A"+\
r"|A,$\begin{bmatrix}1\\1\\0\end{bmatrix}$,B"+\
r"|B,$\begin{bmatrix}0\\1\\0\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix}\end{bmatrix},\begin{bmatrix}1\\1\\1\end{bmatrix}\end{bmatrix}$,B"+\
r"|B,$\begin{bmatrix}0\\0\\1\end{bmatrix}$,A"+\
r""",
A="initial|accept|label=carry=0",
B="label=carry=1",
)

```



File: vec2d.tex

33 2d vector diagram



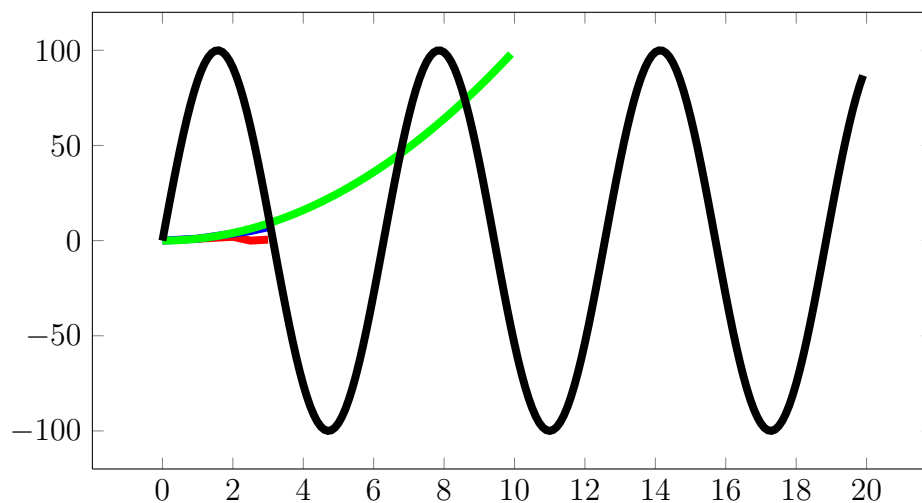
ANSWER:

File: line-graph.tex

34 Line Graph

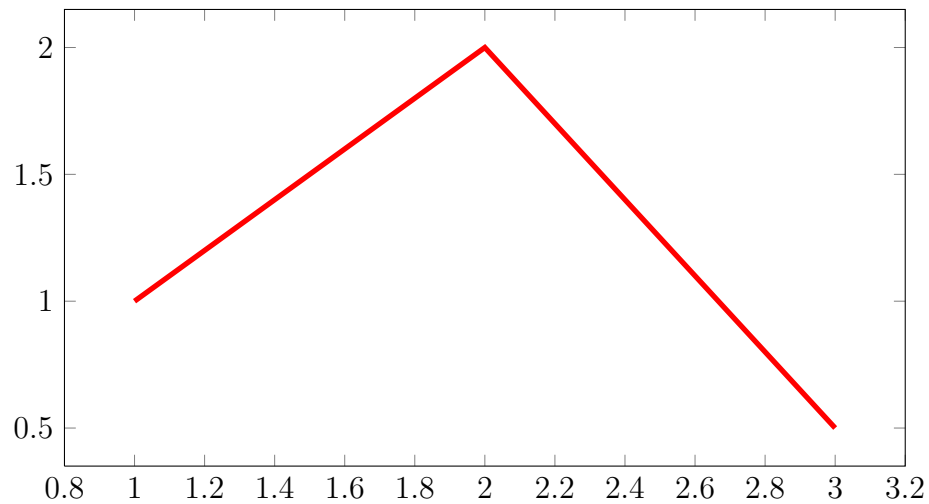
```
from math import sin
from latextool_basic import *
plot = FunctionPlot()

data = ((1, 1), (2, 2), (2.5, 0), (3, 0.5))
data2 = ((0, 0), (1, 1), (2.5, 5), (3, 7))
data3 = [(x/10.0, (x/10.0)**2) for x in range(0, 100)]
data4 = [(x/10.0, 100 * sin(x/10.0)) for x in range(0, 200)]
plot.add(data, line_width='3', color='red')
plot.add(data2, line_width='3', color='blue')
plot.add(data3, line_width='3', color='green')
plot.add(data4, line_width='3', color='black')
print plot
```



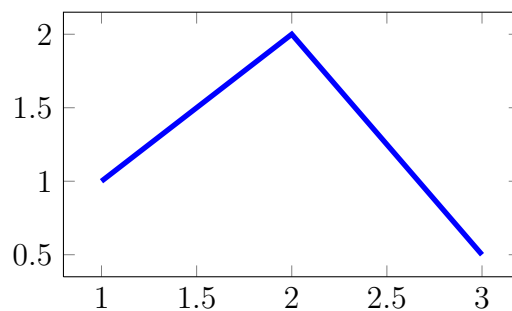
Line width and color:

```
from latextool_basic import *  
plot = FunctionPlot()  
plot.add(((1, 1), (2, 2), (3, 0.5))), line_width='2', color='red')  
print plot
```



Width and height:

```
from latextool_basic import *  
plot = FunctionPlot(width="3in", height="2in")  
plot.add(((1, 1), (2, 2), (3, 0.5))), line_width='2', color='blue')  
print plot
```

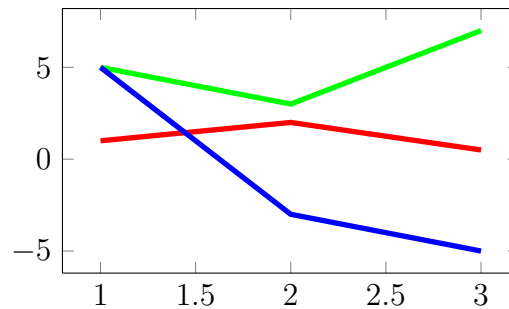


Multiple lines:

```

from latextool_basic import *
plot = FunctionPlot(width="3in", height="2in")
plot.add(((1, 1), (2, 2), (3, 0.5)), line_width='2', color='red')
plot.add(((1, 5), (2, 3), (3, 7)), line_width='2', color='green')
plot.add(((1, 5), (2, -3), (3, -5)), line_width='2', color='blue')
print plot

```

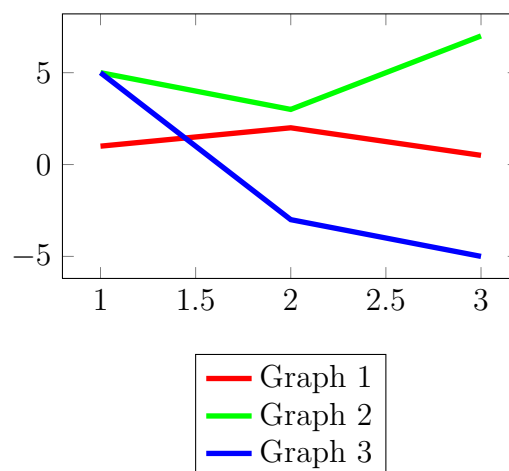


Legend:

```

from latextool_basic import *
plot = FunctionPlot(width="3in", height="2in")
plot.add(((1, 1), (2, 2), (3, 0.5)), line_width='2', color='red', legend='Graph 1')
plot.add(((1, 5), (2, 3), (3, 7)), line_width='2', color='green', legend='Graph 2')
plot.add(((1, 5), (2, -3), (3, -5)), line_width='2', color='blue', legend='Graph 3')
print plot

```



Legend:

```
from math import sin
domain = [x / 10.0 for x in range(-100, 101)]
y_sinx = [(x, 5 * sin(x)) for x in domain]
y_x = [(x, x) for x in domain]

from latextool_basic import *
plot = FunctionPlot(width="6in", height="3in")
plot.add(y_sinx, line_width='2', color='red', legend=r'$y = 5 \sin(x)$')
plot.add(y_x, line_width='2', color='green', legend='$y = x$')
print plot
```

