MAT 305: Mathematical Computing

Lecture 2: 2-D Graphing in Sage

John Perry

University of Southern Mississippi

Fall 2009

Outline

You should be in worksheet mode to repeat the examples.

Outline

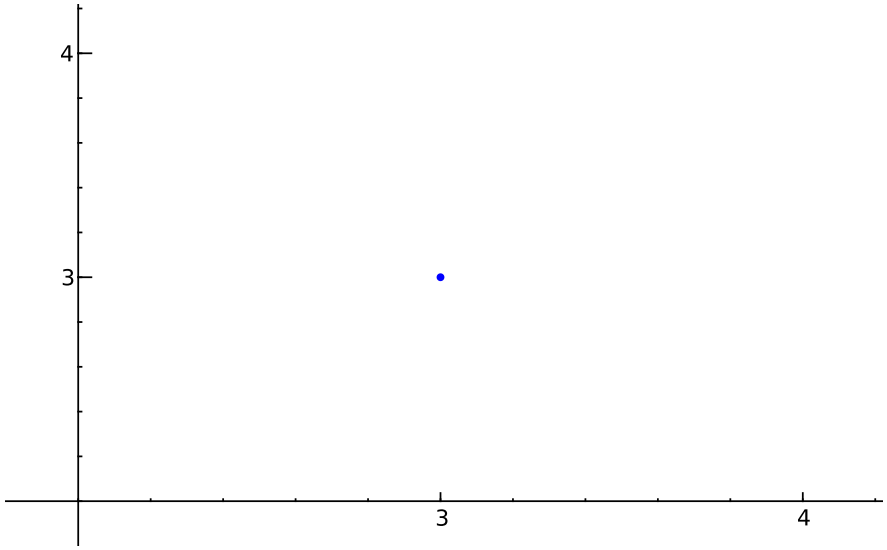
The `point()` command

`point(`(x_0, y_0), *options*`)` where

- (x_0, y_0) is a Python *tuple* `(x,y)`
- *options* include
  - `pointsize`: size of point, default size is 10
  - `rgbcolor`: rgb tuple
    - default color is blue
    - some names allowed in quotes: `'red'`, `'black'`, etc.
    - more on this later

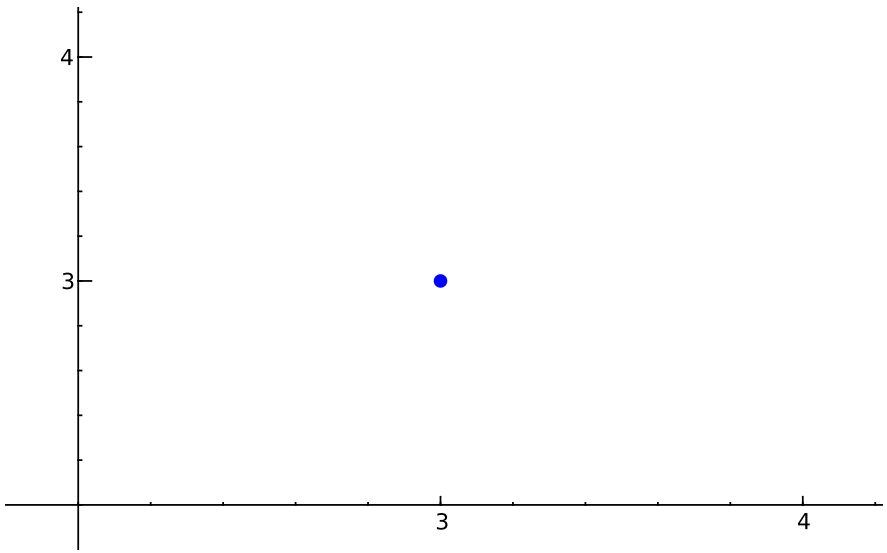
Example

```
sage: point((3,3))
```



Example

```
sage: point((3,3),pointsize=30)
```



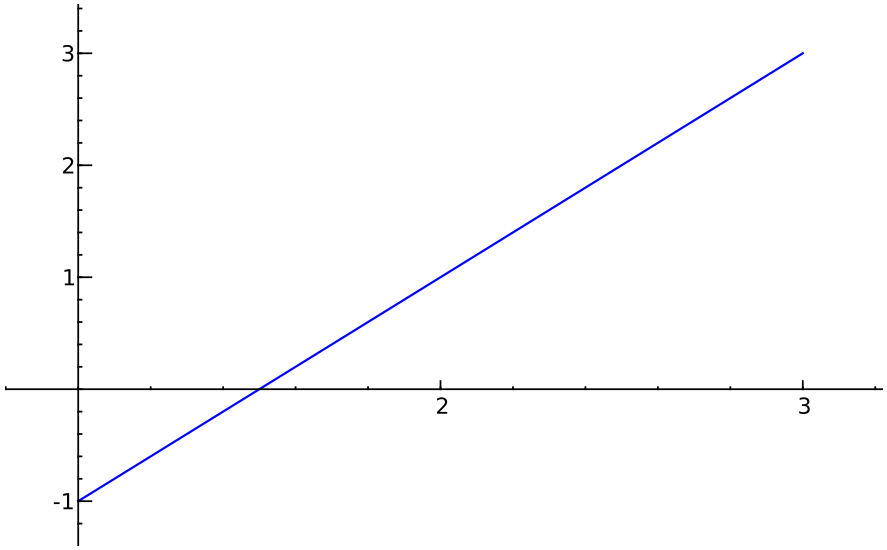
The `line()` command

`line(`[(x_1, y_1), (x_2, y_2)]`,` *options*`)` where

- (x_1, y_1) and (x_2, y_2) are Python *tuples*
- *options* include
  - `thickness` of curve (default is 1)
  - `linestyle`: `'-'` (solid), `'--'` (dashed), `'-.'` (dash-dot), `':'` (dots), steps
  - `rgbcolor`

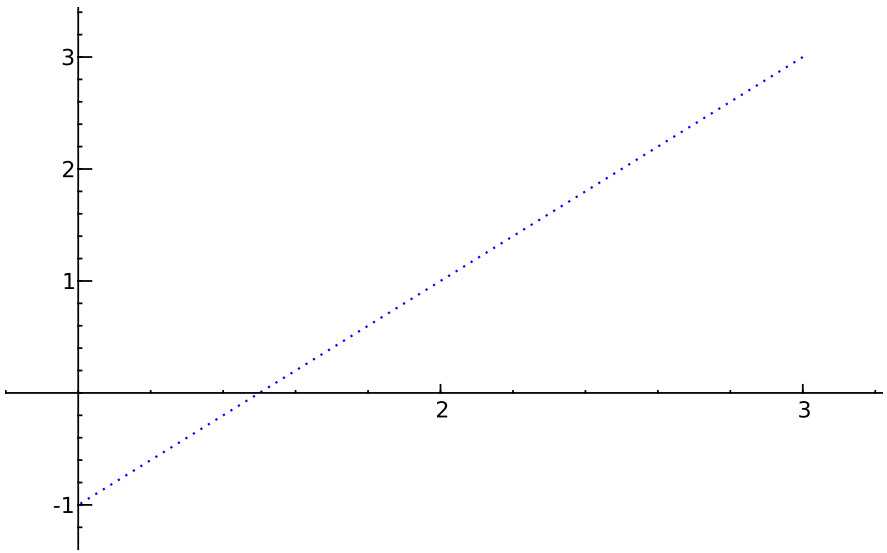
Example

```
sage: line([(3,3),(1,-1)])
```



Example

```
sage: line([(3,3),(1,-1)],linestyle=':')
```



The `polygon()` command

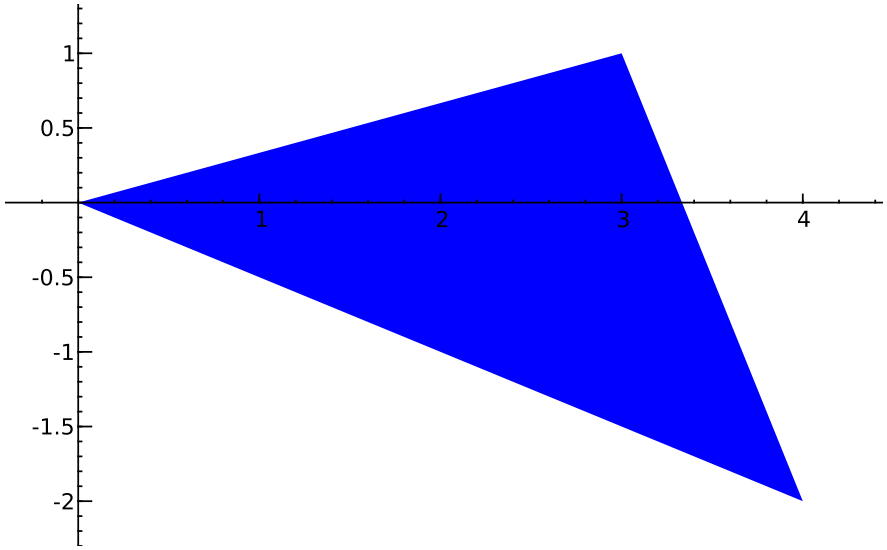
`polygon([`(x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n)`]`, *options*`)` where

- (x_i, y_i) is a Python *tuple* representing a point of the polygon
- *options* include
  - `thickness` of lines (default is 1)
  - `alpha`: how transparent the line is
    - value from `0` to 1
    - `0`: invisible; 1 opaque
  - `rgbcolor`

The polygon will be filled. Don't want a filled polygon? combine lines instead. See below.

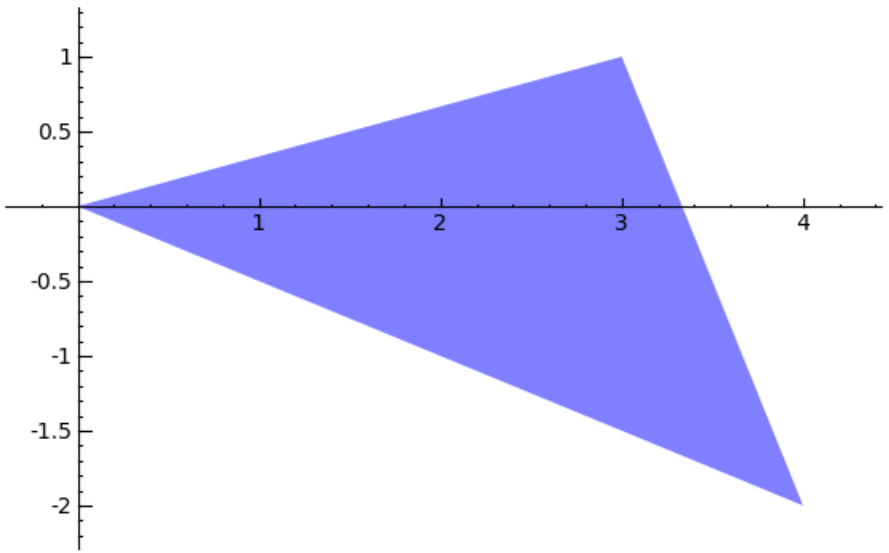
Example

```
sage: polygon([(0,0),(3,1),(4,-2)])
```



Example

```
sage: polygon([(0,0),(3,1),(4,-2)],alpha=0.5)
```



The `text()` command

`text(`*message*`, (`x_0`, `y_0`), `*options*`)` where

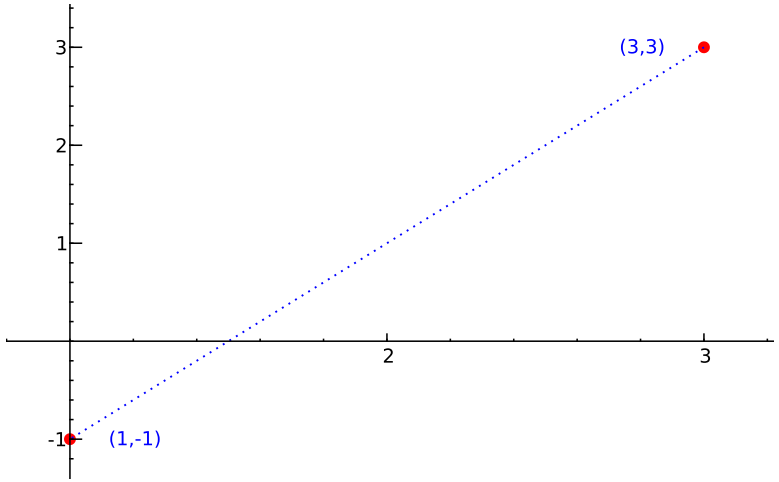
- *message* can be a number, function, or string
- the text is centered over (x_0, y_0)
- *options* include
  - `fontsize` controls the size of the text (default is 10)
  - `rgbcolor`

Combine plot objects with +

- Store graphics objects in memory using expressions
- Addition combines simple objects into complex objects

Example

```
sage:  point1 = point((3,3),pointsize=30,
                      rgbcolor='red')
sage:  point2 = point((1,-1),pointsize=30,
                      rgbcolor='red')
sage:  my_line = line([(3,3),(1,-1)],linestyle=':')
sage:  my_label1 = text('(3,3)',(2.8,3))
sage:  my_label2 = text('(1,-1)',(1.2,-1))
sage:  point1 + point2 + my_line + my_label1 +
my_label2
```



Outline

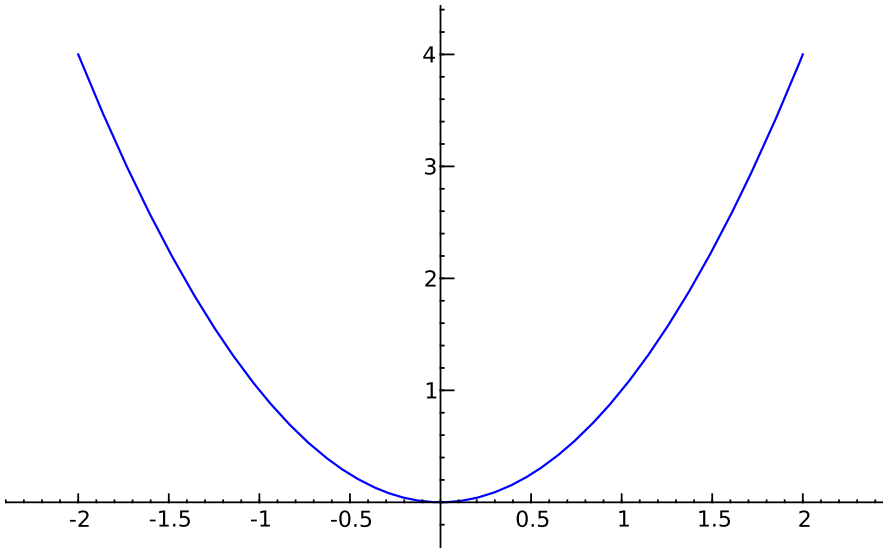
The `plot()` command

`plot(`$f(x)$, *options*`)` where

- $f(x)$ is an expression in a defined variable x
- *options* include
  - `xmin`, `xmax` (*no* `ymin`, `ymax` options in `plot()`)
  - `plot_points`: minimal number of plot points
  - `fill`: to `axis`, `min` y value, `max` y value, a number c, or a function $g(x)$
  - `fillcolor`
  - `rgbcolor`
  - `thickness`
  - `linestyle`

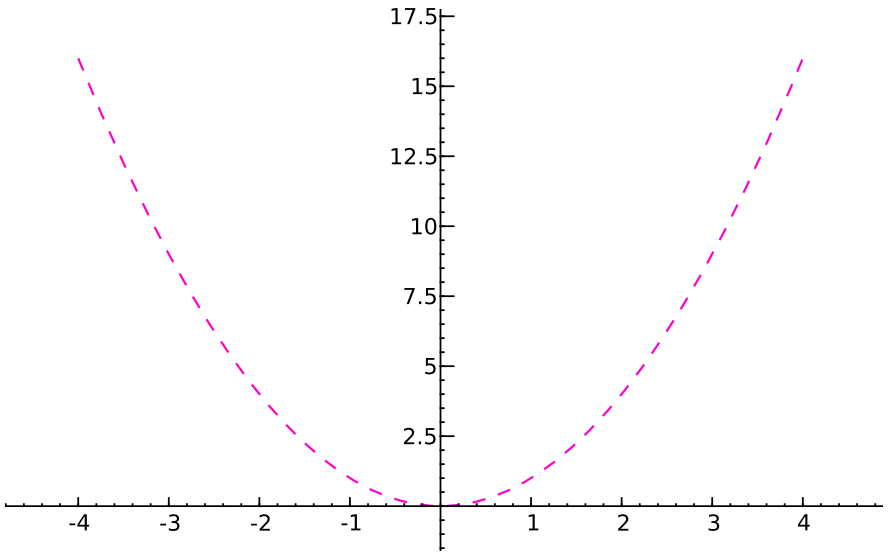
Basic example

```
sage: plot(x**2, xmin=-2,xmax=2)
```



Experiment with options!

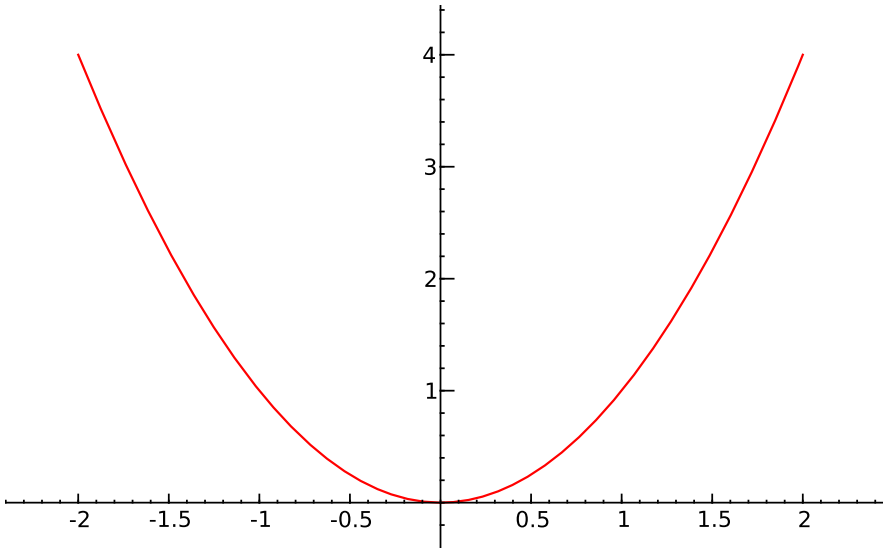
```
sage:  plot(x**2, xmin=-4, xmax=4,
            linestyle="--", rgbcolor=(1,0,0.8))
```



RGB colors

Specify colors using RGB tuples.

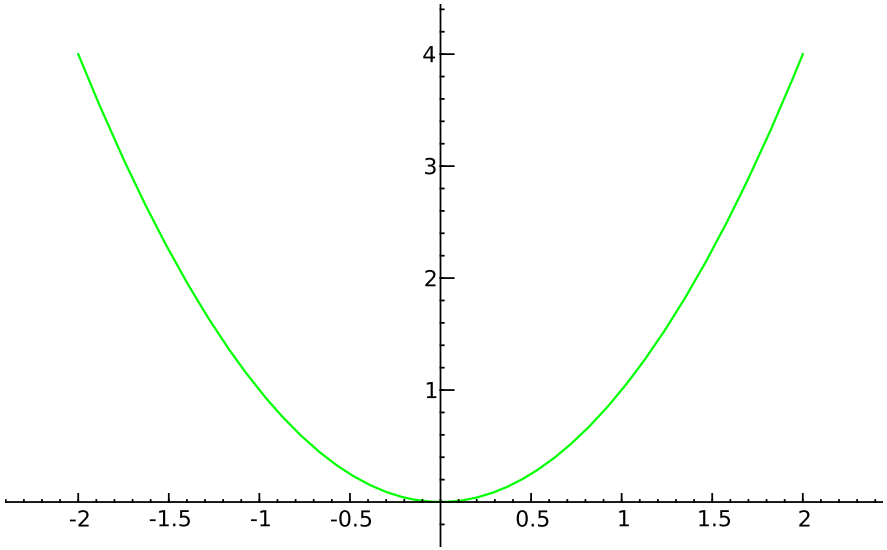
- Red, Green, Blue: primary colors of light
- Pure red: (1,0,0)



RGB colors

Specify colors using RGB tuples.

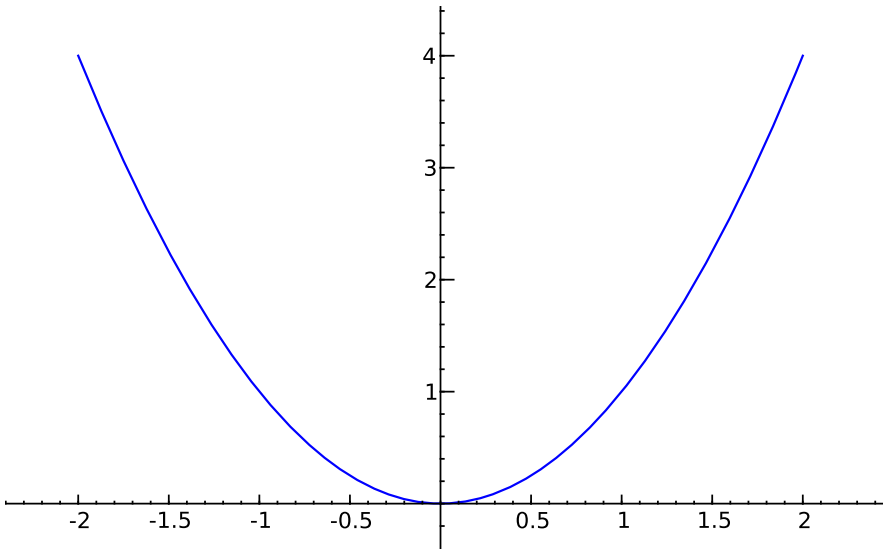
- Red, Green, Blue: primary colors of light
- Pure green: (0,1,0)



RGB colors

Specify colors using RGB tuples.

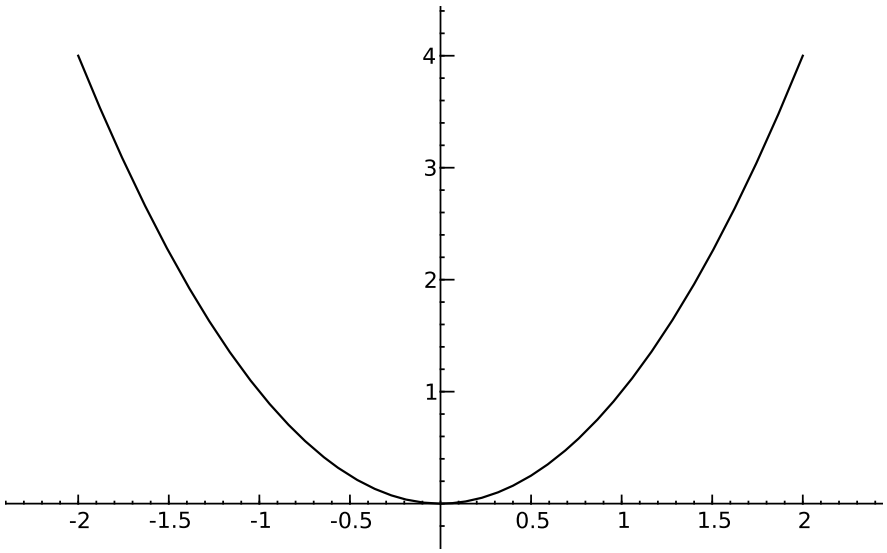
- Red, Green, Blue: primary colors of light
- Pure blue: (0,0,1)



RGB colors

Specify colors using RGB tuples.

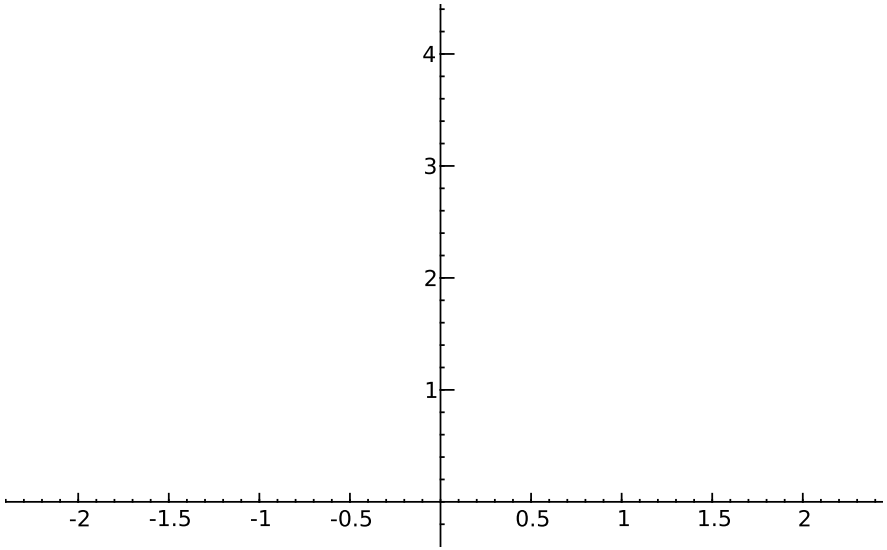
- Red, Green, Blue: primary colors of light
- Black is the absence of color: (0,0,0)



RGB colors

Specify colors using RGB tuples.

- Red, Green, Blue: primary colors of light
- White is the presence of all colors: (1,1,1)

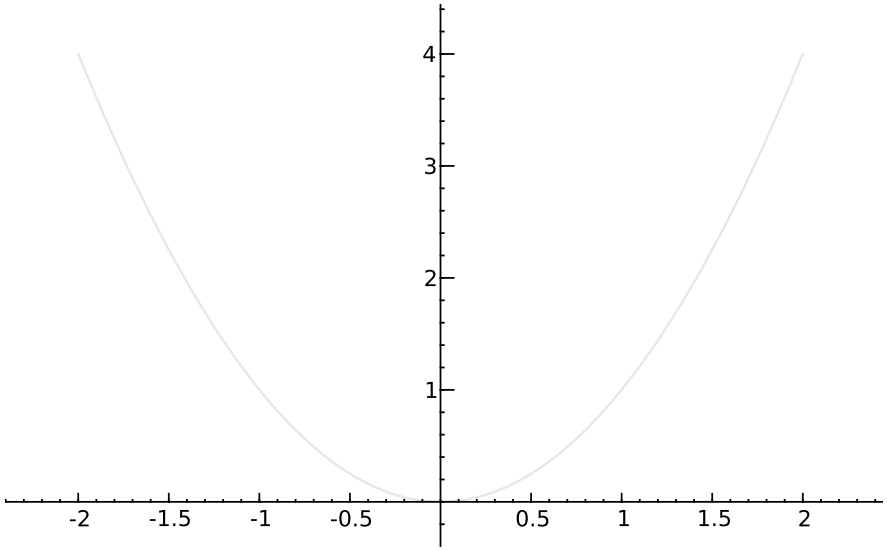


(oops!)

RGB colors

Specify colors using RGB tuples.

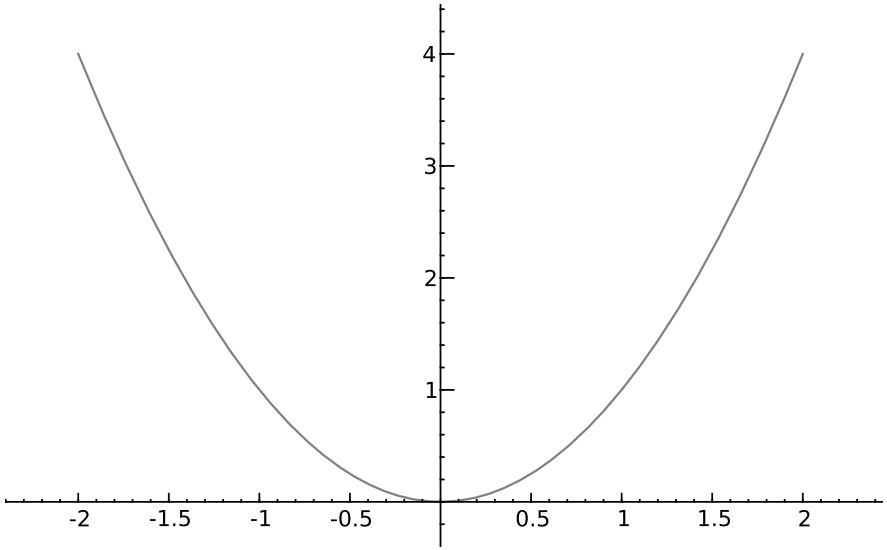
- Red, Green, Blue: primary colors of light
- Gray is an even mixture of the colors: (0.9,0.9,0.9)



RGB colors

Specify colors using RGB tuples.

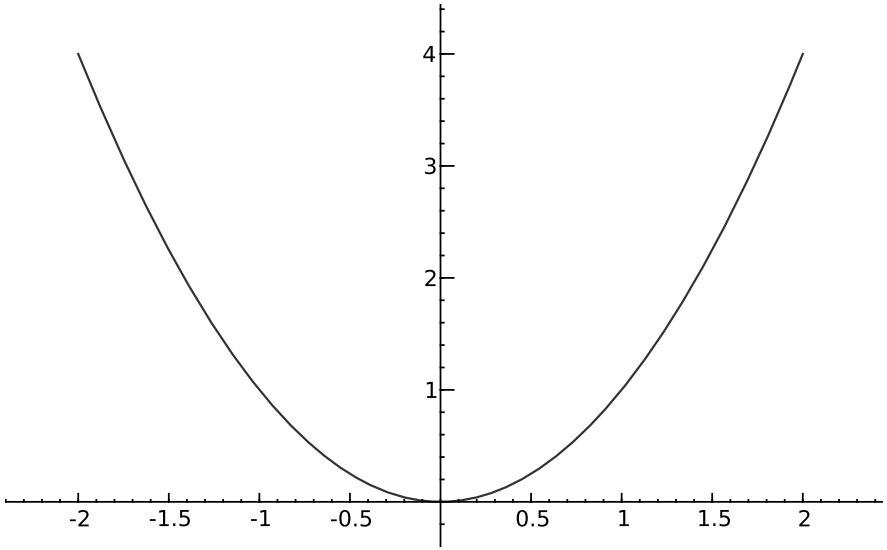
- Red, Green, Blue: primary colors of light
- Gray is an even mixture of the colors: (0.5,0.5,0.5)



RGB colors

Specify colors using RGB tuples.

- Red, Green, Blue: primary colors of light
- Gray is an even mixture of the colors: (0.2,0.2,0.2)



RGB colors

Specify colors using RGB tuples.

- Red, Green, Blue: primary colors of light
- What colors do these tuples represent?
  - (0.8,0.6,0.2)
  - (0.9,0.9,0)
  - (0.3,0.8,0.9)

RGB colors

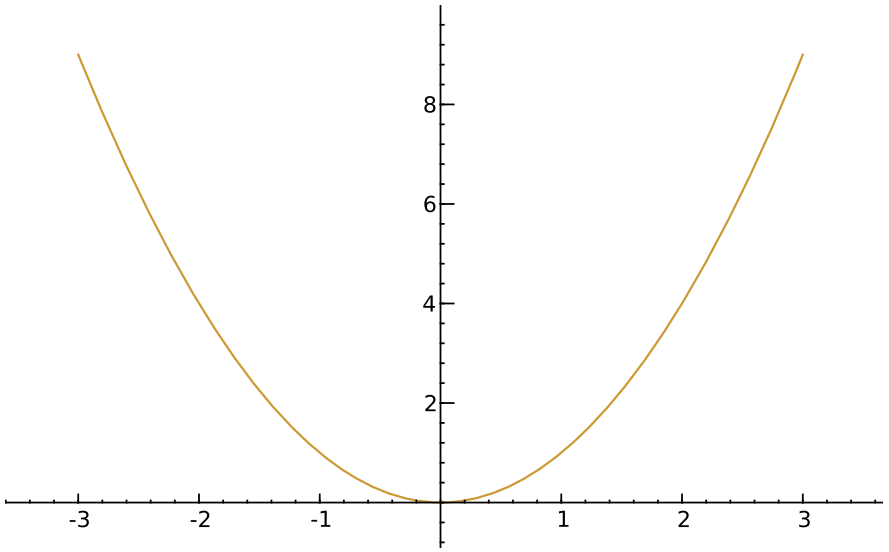
Specify colors using RGB tuples.

- Red, Green, Blue: primary colors of light
- What colors do these tuples represent?
  - (0.8,0.6,0.2) brown
  - (0.9,0.9,0) yellow
  - (0.3,0.8,0.9) blue-green

Remember colors

- Use expressions to remember colors

```
sage:  brown = (0.8,0.6,0.2)
sage:  plot(x**2,xmin=-3,xmax=3,color=brown)
```



Outline

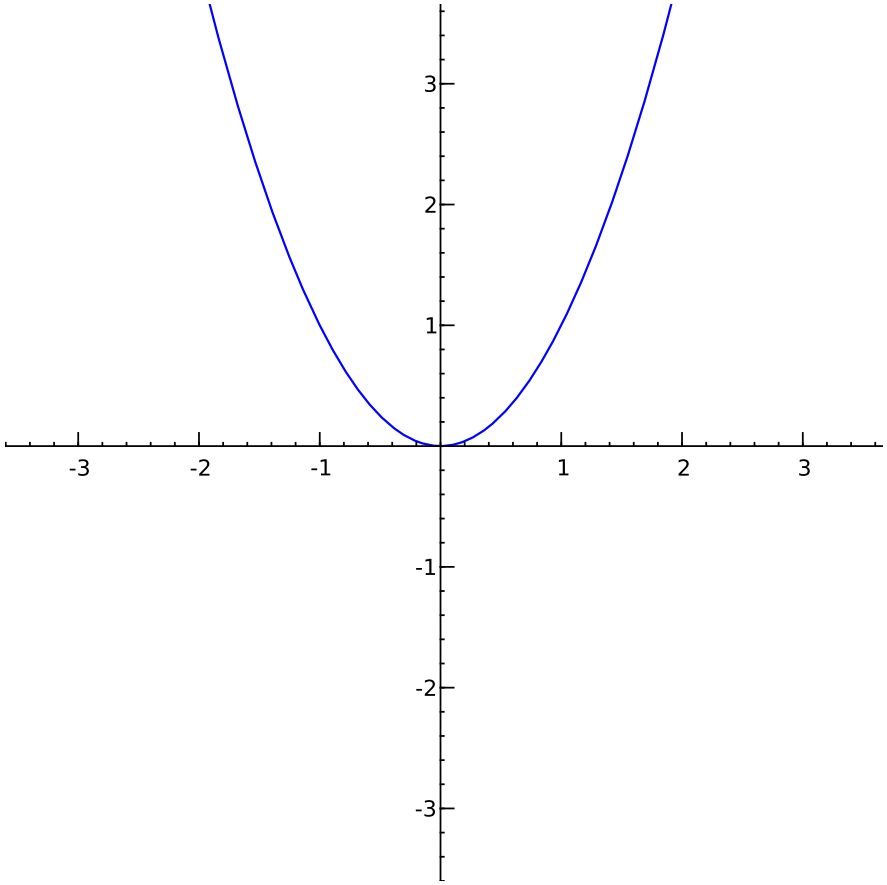
The `show()` command

`show(`*plot object*, *options*`)` where

- *plot object* is any object generated by a plot
- *options* include
  - `aspect_ratio`: ratio of the width x values to the height of y values (1 makes a "square" graph)
  - `xmin, xmax`
  - `ymin, ymax`

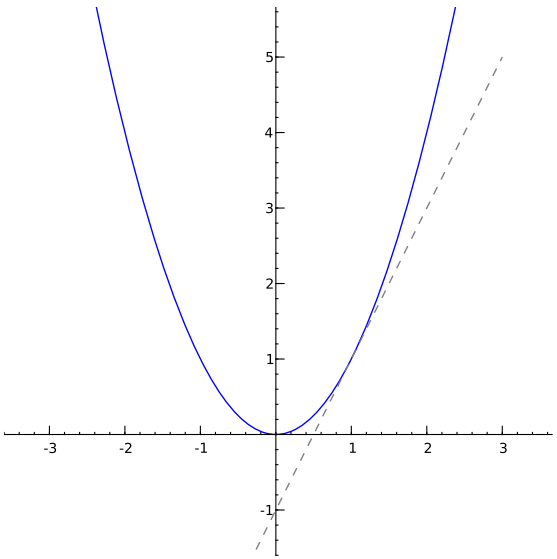
Example

```
sage: myplot = plot(x**2,xmin=-3,xmax=3)
sage: show(myplot,ymin=-3,ymax=3,aspect_ratio=1)
```



Combine plots

```
sage:  par_plot = plot(x**2,xmin=-3,xmax=3)
sage:  tan_plot = plot(2*x-1,xmin=-3,xmax=3,
                       color='gray',linestyle='--')
sage:  com_plot = par_plot + tan_plot
sage:  show(com_plot,ymin=-1,ymax=5,aspect_ratio=1)
```



The `animate()` command

`animate([`*plot1*`,`*plot2*`,...],` *options*`)` where

- *plot1*, *plot2*, ... are graphics objects
  - each object constitutes one frame of the animation
- *options* include
  - `xmin`, `xmax`, `ymin`, `ymax`, `aspect_ratio`

show animation using `show()` command, with options:

- `delay` in hundredths of a second (default is 20)
- `iterations` (default is `0`, which means forever)
- cannot specify `xmin`, `xmax`, `ymin`, `ymax`, `aspect_ratio` in `show()` when showing an animation; specify in `animate()` instead

Example

```
sage: par_plot = plot(x**2,xmin=-3,xmax=3,
          thickness=2,color=(0,0,0))
sage: tan_plot = plot(2*x-1,xmin=-3,xmax=3,thickness=2)
sage: com_plot = par_plot + tan_plot
sage: pink = (1.0,0.5,0.5)
sage: sec1_plot = plot(1,xmin=-3,xmax=3,
          color=pink,linestyle='--')
sage: sec2_plot = plot(1/2*x+1/2,xmin=-3,xmax=3,
          color=pink,linestyle='--')
sage: sec3_plot = plot(x,xmin=-3,xmax=3,
          color=pink,linestyle='--')
sage: sec4_plot = plot(3/2*x-1/2,xmin=-3,xmax=3,
          color=pink,linestyle='--')
sage: my_anim = animate([
          com_plot+sec1_plot, com_plot+sec2_plot,
          com_plot+sec3_plot, com_plot+sec4_plot,
          com_plot
      ])
sage: show(my_anim)
```

Notes on `xmin`, `xmax`

- In `plot()`,
  `xmin` and `xmax` indicate x values to *compute*.
- In `show()` and `animate()`,
  `xmin` and `xmax` indicate x values to *display*.
- `plot(x^2,xmin=-3,xmax=3)` computes points on the interval $[-3, 3]$
- `show(my_plot,xmin=-1,xmax=1)` displays $x \in [-1, 1]$, *regardless of the x values computed in* `my_plot`

Outline

Implicit plots

- Implicit plots are handled by a Python package called `matplotlib`
- Things behave differently than usual plots
  - look different
  - different options

The `implicit_plot()` command

`implicit_plot(`$f(x,y)$, `(x,` *xmin*, *xmax*`)`, `(y,` *ymin*, *ymax*`)`, *options*`)` where

- $f(x,y)$ is a function of x and y
  - graphs $f(x,y) = 0$
  - don't forget to define y as a variable
- *options* include
  - `plot_points`: number of points to plot in each direction of grid
  - `fill`: fill the region $f(x,y) < 0$ (use value `True` or `False`)
  - `cmap`: next slide

Color maps

For technical reasons, choice of color is restricted to bizarre names instead of rgb tuples. Try

- `'Reds_r'`
- `'Blues_r'`
- `'Oranges_r'`
- `'Greens_r'`
- `'Greys_r'`
- `'Purples_r'`

To see all possible names, type

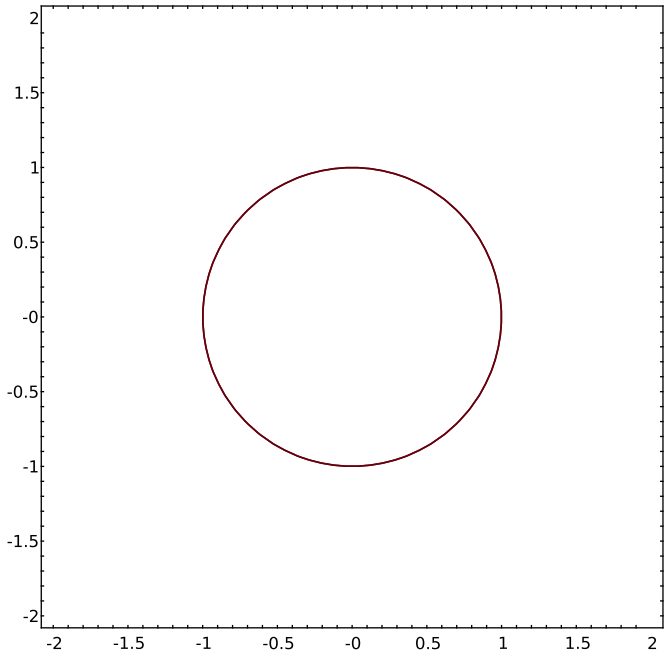
```
sage: matplotlib.cm.datad.keys()
```

Example

To plot a circle, rewrite the equation as $f(x,y)=0$:

$$x^2+y^2=1 \quad\Longrightarrow\quad x^2+y^2-1=0 \quad\Longrightarrow\quad f(x,y)=x^2+y^2-1$$

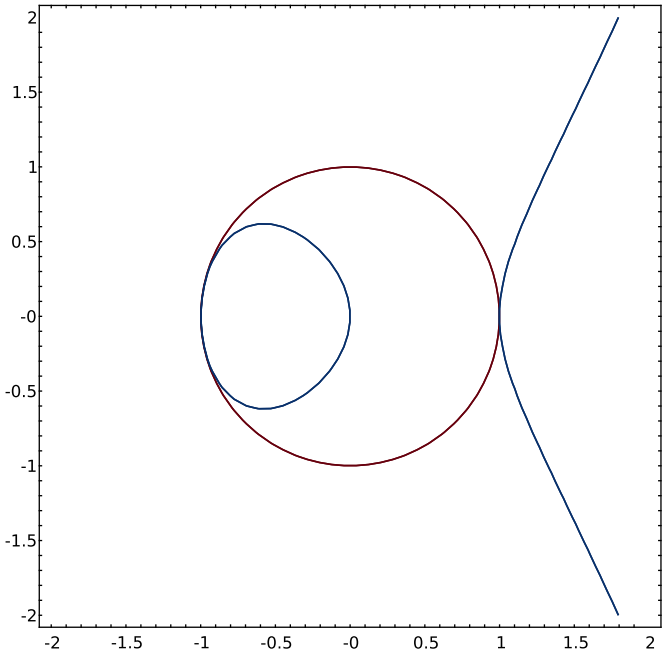
```
sage:  icircle = implicit_plot(x**2 + y**2-1,
                 (x,-2,2),(y,-2,2),cmap='Reds_r')
sage:  show(icircle,aspect_ratio=1)
```



Combining

Can combine implicit plots:

```
sage:  ielliptic = implicit_plot(y**2-x**3+x,
                 (x,-2,2),(y,-2,2),cmap='Blues_r')
sage:  show(icircle+ielliptic,aspect_ratio=1)
```



Animating

```
sage:  elliptic1 = implicit_plot(y**2-x**3+x,
                 (-2,2),(-2,2),cmap='Blues_r',plot_points=100)
sage:  elliptic2 = implicit_plot(y**2-x**3+0.75*x,
                 (-2,2),(-2,2),cmap='Blues_r',plot_points=100)
sage:  elliptic3 = implicit_plot(y**2-x**3+0.5*x,
                 (-2,2),(-2,2),cmap='Blues_r',plot_points=100)
sage:  elliptic4 = implicit_plot(y**2-x**3+0.25*x,
                 (-2,2),(-2,2),cmap='Blues_r',plot_points=100)
sage:  elliptic5 = implicit_plot(y**2-x**3+0.1*x,
                 (-2,2),(-2,2),cmap='Blues_r',plot_points=100)
sage:  elliptic6 = implicit_plot(y**2-x**3,
                 (-2,2),(-2,2),cmap='Blues_r',plot_points=100)
sage:  my_anim = animate([elliptic1, elliptic2, elliptic3,
                 elliptic4, elliptic5, elliptic6],
                 aspect_ratio=1)
sage:  show(my_anim)
```

Outline

Parametric equations

Form:

$$\begin{cases} x(t) & = \dots \\ y(t) & = \dots \end{cases}, \qquad t \in \left[t_{\text{min}}, t_{\text{max}}\right]$$

Example

A Bezier curve with control points (x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3) is defined by

$$\begin{cases} x(t) & = x_0(1-t)^3 + x_1 t(1-t)^2 + x_2 t^2(1-t) + x_3 t^3 \\ y(t) & = y_0(1-t)^3 + y_1 t(1-t)^2 + y_2 t^2(1-t) + y_3 t^3 \end{cases}, \; t \in [0, 1].$$

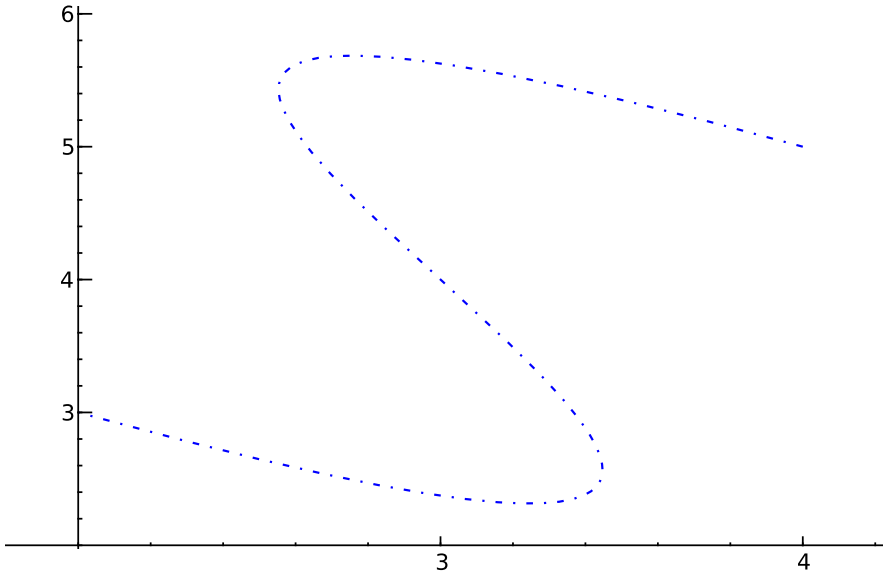
The `parametric_plot()` command

`parametric_plot(`$(x(t),y(t))$`,(tmin,tmax), plot options)`
`where`

- $x(t)$ and $y(t)$ are functions of t
- don't forget to define t as a variable
- usual plot options apply

Example Bezier Curve

```
sage:  parametric_plot(
           (2*t**3 + 6*3*t**2*(1 - t)
                + 0*3*t*(1 - t)**2 + 4*(1 - t)**3,
           3*t**3 + 0*3*t**2*(1 - t)
                + 8*3*t*(1 - t)**2 + 5*(1 - t)**3),
       (0,1),linestyle='-.')
```



Polar plots

Form: radius r a function of angle θ

Example

A limaçon has the form

$$r = 1 + c \sin \theta.$$

It is difficult to describe this implicitly.

The `polar_plot()` command

`polar_plot(`$r(x)$, *options*`)` where

- $r(x)$ is a *polar* function of x
  - x stands in for θ
  - can define a variable θ if you really want, but…
- usual plot options apply

Example limaçon

```
sage:  lim1 = polar_plot(1+2.5*sin(x),xmin=0,xmax=2*pi)
sage:  lim2 = polar_plot(1+1.7*sin(x),xmin=0,xmax=2*pi)
sage:  lim3 = polar_plot(1+sin(x),xmin=0,xmax=2*pi)
sage:  lim4 = polar_plot(1+0.7*sin(x),xmin=0,xmax=2*pi)
sage:  lim5 = polar_plot(1+0.5*sin(x),xmin=0,xmax=2*pi)
sage:  lim6 = polar_plot(1+0*sin(x),xmin=0,xmax=2*pi)
sage:  my_anim = animate([lim1,lim2,lim3,
                          lim4,lim5,lim6],
                 aspect_ratio=1,xmin=-2,xmax=2,
                 ymin=-1,ymax=4)
sage:  show(my_anim)
```

Outline

Summary

- Sage offers many commands for plotting 2-D objects
  - points, lines
  - functions
  - equations: implicit, parametric, polar
- Most options work for all objects
- Combine objects by "adding" them together
- Animate using a list of objects
- `implicit_plot()` is a bit odd