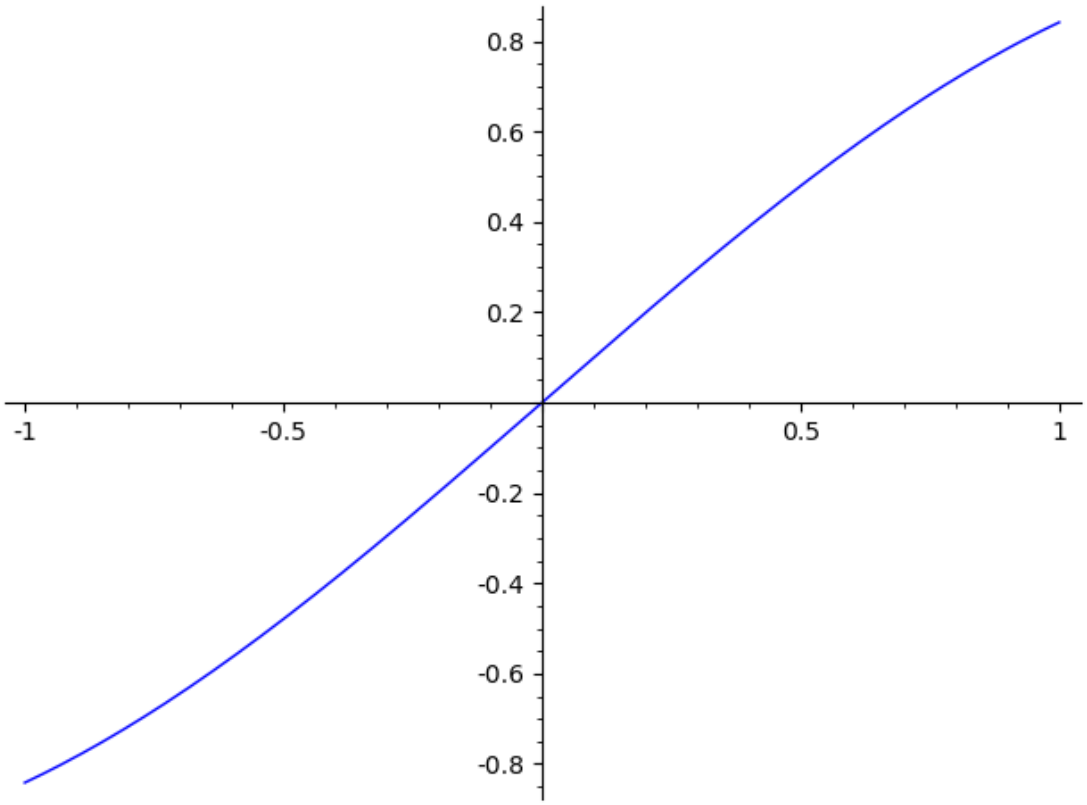
# Untitled2

September 13, 2023

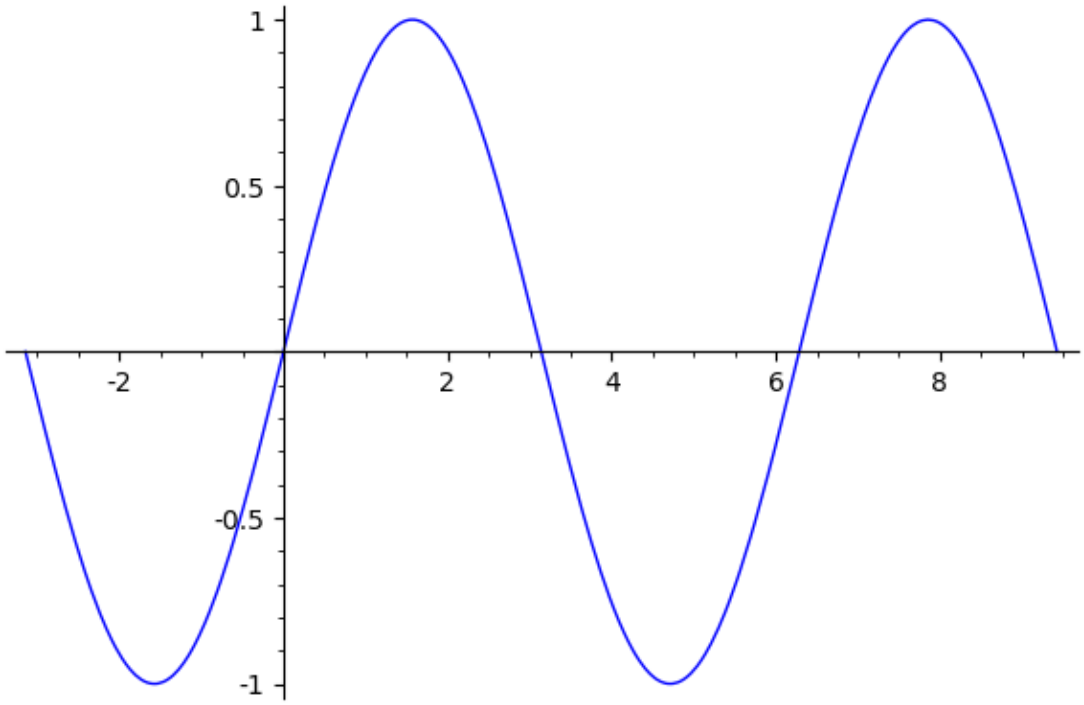
[1]:
```
# MATH 1110H-B F01 Lab 2023-09-12
#
# Wherein we ring the changes on the several plotting commands in
# SageMath,  without getting into the fancy stuff like labelling
# points,  curves,  or axes.

plot(sin(x))  # The basic plot command,  which plots the function
              # for x between -1 and 1 by default.  Note that the
              # scale on the vertical and horizontal axes is not
              # quite the same.
```

[1]:



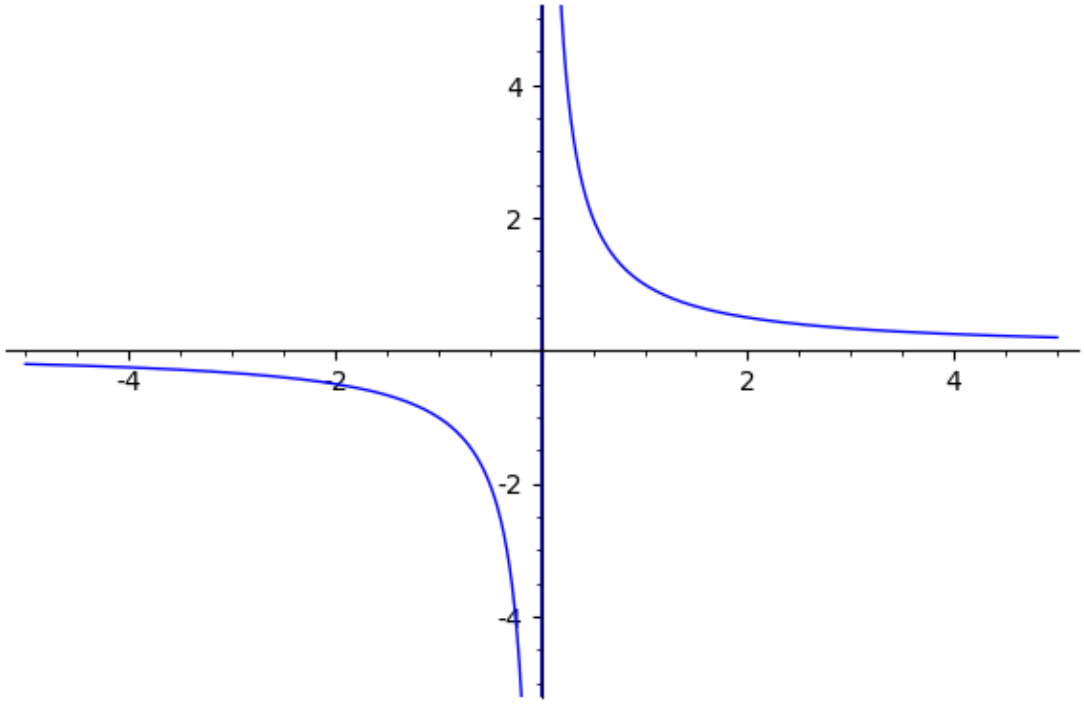
[2]:
```
plot(sin(x),-pi,3*pi)  # The basic plot command with the minimum and
                       # maximum values of x specified.  Note that
                       # one must specify multiplication - 3pi will
                       # give you an error - and that famous constant
                       # is named pi in SageMath.
```

[2]:



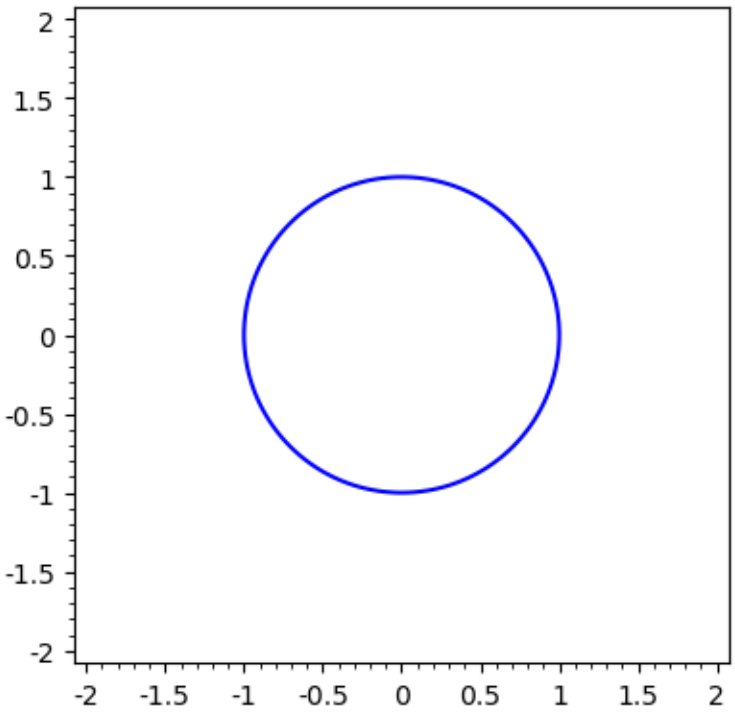

[3]:
```
plot(1/x,-5,5,ymin=-5,ymax=5)  # If you are graphing a function that
                               # gets very large,  you can restrict
                               # how much SageMath shows vertically
                               # by specifying minimum and maximum
                               # values for y.
```

[3]:



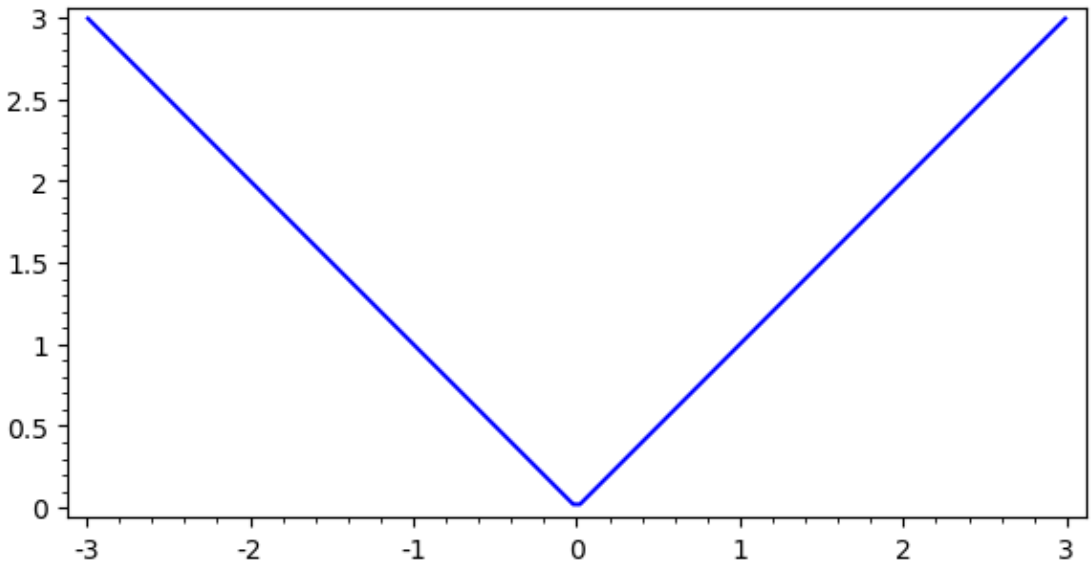

[4]:
```
var("y")  # If you want anything other than x to be considered as a
          # variable,  you need to specify as such.
implicit_plot(x^2 + y^2 == 1,(x,-2,2),(y,-2,2))  # Graphing curves
          # defined implicitly by an equation has its own command.
          # Note the use of == for equality in the given equation
          # and the need to specify the range for each variable
          # separately.  (With a format a little different from how
          # one does it in the basic plot command.
```

[4]:



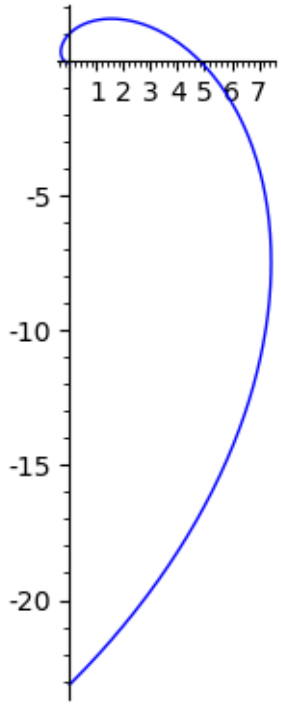

[5]:
```
implicit_plot(y==abs(x),(x,-3,3),(y,0,3))  # Note the use of abs()
                                           # for the absolute value
                                           # function.
```

[5]:



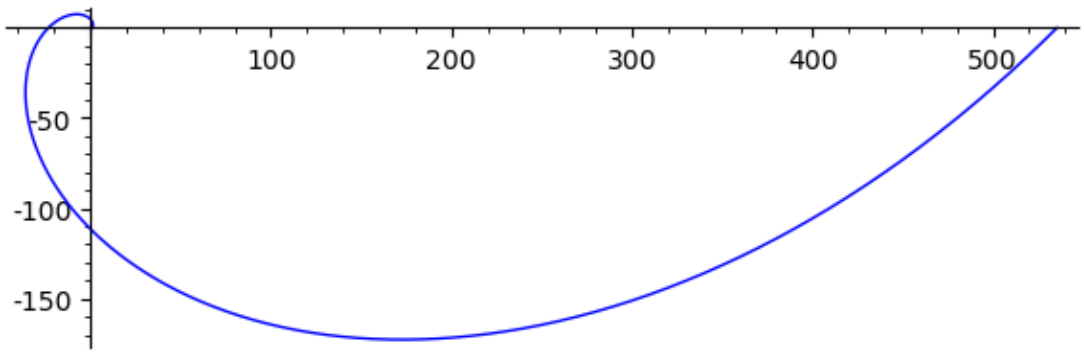
[6]:
```
var("t")  # We're going to use t as a variable,  so we need to tell
          # SageMath this before actually using it as such.
parametric_plot((e^t*sin(t),e^t*cos(t)),(t,-pi,pi))  # This is the
          # specialized command for plotting parametric curves,  in
          # which the x and y coordinates are controlled by a third
          # variable (the parameter),  i.e. x = f(t) and y = g(t)
          # for some functions f(t) and g(t).  (See Section 10.4 of
          # the textbook.)  Note that the x and y coordinates are
          # specified in an ordered pair and that the range of t to
          # be used is given in the same format as ranges in the
          # implicit_plot command are.
```

[6]:



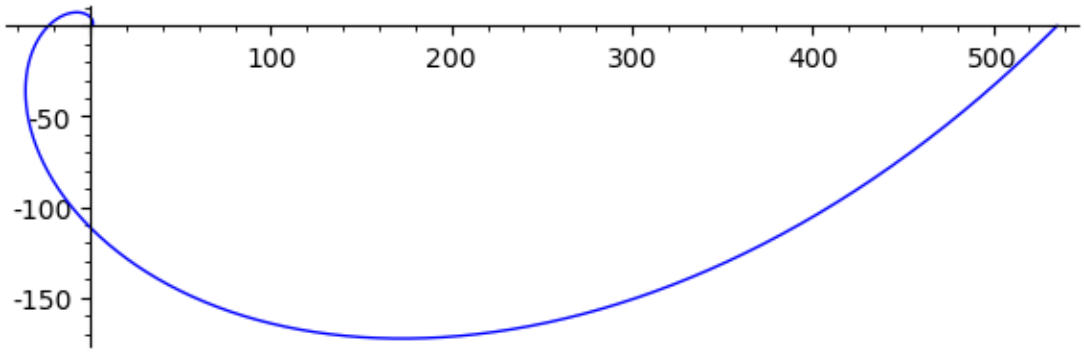

[7]:
```
var("theta")  # We're going to use theta as a variable,  so...
polar_plot(e^theta,(theta,0,2*pi))  # This is the specialized
          # command for plotting in polar coordinates.  Theta
          # gives the direction of a point,  i.e. the angle that
          # the line joining the origin to the point makes with
          # p theositive x-axis,  measured counterclockwise,
          # and r = f(theta) gives the distance the point is
          # from the origin.  (See Section 10.1 of the textbook.)
          # Note that the range of theta to be used is given in
          # the same format as ranges in the implicit_plot and
          # parametric_plot commands are.
```

[7]:




[8]:
```
f = e^theta  # If you want to give a function a name,  you can do
             # so.  Useful if you want to use it repeatedly and do
             # want to keep writing it out.  Note the use of = to
             # assign the definition of the function to its name.
polar_plot(f,(theta,0,2*pi))
```

[8]:




[9]:
```
  #  And that's all for this time! :-)
```