Sage3-Vorlesung-Live

```
def fib(n):
    if n <2:
        return 1
    else:
        return fib(n-1)+fib(n-2)
```

```
fib(20)
```

```
10946
```

```
%time
fib(34)
```

```
9227465
CPU time: 8.76 s,  Wall time: 8.81 s
```

```
@cached_function
def fib(n):
    if n <2:
        return 1
    else:
        return fib(n-1)+fib(n-2)
```

```
%time
fib(34)
```

```
9227465
CPU time: 0.00 s,  Wall time: 0.00 s
```

```
fib(50)
```

```
20365011074
```

```
fib.get_cache()
```

```
__main__:2: DeprecationWarning: The .get_cache() method is
deprecated, use the .cache attribute instead.
See http://trac.sagemath.org/19694 for details.
{((12,), ()): 233, ((42,), ()): 433494437, ((20,), ()): 10946,
((50,), ()): 20365011074, ((35,), ()): 14930352, ((17,), ()): 2584
((5,), ()): 8, ((0,), ()): 1, ((43,), ()): 701408733, ((30,), ()):
1346269, ((25,), ()): 121393, ((31,), ()): 2178309, ((13,), ()):
377, ((40,), ()): 165580141, ((38,), ()): 63245986, ((39,), ()):
102334155, ((10,), ()): 89, ((48,), ()): 7778742049, ((18,), ()):
4181, ((3,), ()): 3, ((28,), ()): 514229, ((11,), ()): 144, ((36,)
()): 24157817, ((8,), ()): 34, ((6,), ()): 13, ((33,), ()): 570288
((7,), ()): 21, ((21,), ()): 17711, ((16,), ()): 1597, ((46,), ())
2971215073, ((41,), ()): 267914296, ((47,), ()): 4807526976, ((29,
()): 832040, ((26,), ()): 196418, ((4,), ()): 5, ((34,), ()):
```

```
9227465, ((19,), ()): 6765, ((1,), ()): 1, ((44,), ()): 1134903170
((27,), ()): 317811, ((14,), ()): 610, ((9,), ()): 55, ((15,), ())
987, ((24,), ()): 75025, ((22,), ()): 28657, ((49,), ()):
12586269025, ((23,), ()): 46368, ((37,), ()): 39088169, ((32,), ()
3524578, ((2,), ()): 2, ((45,), ()): 1836311903}
```

```
fib.clear_cache()
```

```
fib.get_cache()
```

```
{}
```

```
fib(500)
```

```
WARNING: Output truncated!
full_output.txt
```

```
Traceback (click to the left of this block for traceback)
...
RuntimeError: maximum recursion depth exceeded

full_output.txt
```

```
fib(250)
```

```
127765235729247325860370338946550318986595564473522 49
```

```
fib(500)
```

```
225591516161936330872512695036072072046011324913758190588638866418
46277386868834050159870527969684986 26
```

```
l=range(10)
l
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
l[2:5]
```

```
[2, 3, 4]
```

```
l[2:10:3]
```

```
[2, 5, 8]
```

```
l2=range(20,30)
l2
```

```
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29]
```

```
l +l2
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29]
```

```
l1=[1,2,3]
l1
```

```
[1, 2, 3]
```

```
l2=l1
l2
```

```
[1, 2, 3]
```

```
l1==l2
```

```
True
```

```
l1.append(4)
l1
```

```
[1, 2, 3, 4]
```

```
l2
```

```
[1, 2, 3, 4]
```

```
l3=copy(l1)
l3
```

```
[1, 2, 3, 4]
```

```
l1.append(5)
l1
```

```
[1, 2, 3, 4, 5, 5]
```

```
l3
```

```
[1, 2, 3, 4]
```

```
l=range(1,15)
l
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
map(factor,l)
```

```
[1, 2, 3, 2^2, 5, 2 * 3, 7, 2^3, 3^2, 2 * 5, 11, 2^2 * 3, 13, 2 *
```

```
l
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
filter(is_prime,l)
```

```
[2, 3, 5, 7, 11, 13]
```

```
15 in l
```

```
False
```

```
l1.count(5)
```

```
2
```

```
def quadrat(x):
    return x^2

map(quadrat,l)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]
```

```
f=lambda x: x^2
f(3)
```

```
9
```

```
map(lambda x: x^3, l)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197,
2744]
```

```
add = lambda x,y: x+y
add(3,4)
```

```
7
```

```
add42 = lambda y: add(y,42)
```

```
add42(10)
```

```
52
```

```
l=[1,2,3,4]
l
```

```
[1, 2, 3, 4]
```

```
reduce(operator.mul,l)
```

```
24
```

```
l2=range(1,13)
l2
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

```
map(is_prime,l2)
```

```
[False, True, True, False, True, False, True, False, False, False,
True, False]
```

```
[is_prime(k) for k in l2]
```

```
[False, True, True, False, True, False, True, False, False, False,
True, False]
```

```
[k^2 for k in l2 if is_prime(k)]
```

```
[4, 9, 25, 49, 121]
```

```
rat5=[[m/n for m in [1..n]] for n in [1..5]]
rat5
```

```
[[1], [1/2, 1], [1/3, 2/3, 1], [1/4, 1/2, 3/4, 1], [1/5, 2/5, 3/5,
4/5, 1]]
```

```
rat5a=reduce(operator.add,rat5)
rat5a
```

```
[1, 1/2, 1, 1/3, 2/3, 1, 1/4, 1/2, 3/4, 1, 1/5, 2/5, 3/5, 4/5, 1]
```

```
l6=list(set(rat5a))
```

```
l6
```

```
[1, 1/2, 1/3, 2/3, 1/4, 3/4, 1/5, 2/5, 3/5, 4/5]
```

```
l6.sort()
```

```
l6
```

```
[1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 1]
```

```
t=1,-2,3
for i in t:
    print i
```

```
1
-2
3
```

```
t.append(5)
```

```
Traceback (click to the left of this block for traceback)
...
AttributeError: 'tuple' object has no attribute 'append'
```

```
x=1
y=2
x,y
```

```
(1, 2)
```

```
x,y=y,x
x,y
```

```
(2, 1)
```

```
n=17
m=5
q,r = n.quo_rem(m)
```

```
q,r
```

```
(3, 2)
```

```
q
```

```
3
```

```
l[1]
```

```
2
```

```
l
```

```
[1, 2, 3, 4]
```

```
l1=['a','b','c']
l2=[1,2,3]
l12=zip(l1,l2)
l12
```

```
[('a', 1), ('b', 2), ('c', 3)]
```

```
l12[0]
```

```
('a', 1)
```

```
d={'a':1,5:3,'c':ZZ}
d
```

```
    {'a': 1, 'c': Integer Ring, 5: 3}
```

```
d[5]
```

```
    3
```

```
d['y']
```

```
    Traceback (click to the left of this block for traceback)
    ...
    KeyError: 'y'
```

```
d['y']=5
```

```
d['y']
```

```
    5
```

```
d
```

```
    {'a': 1, 'c': Integer Ring, 'y': 5, 5: 3}
```

```
d.keys()
```

```
    ['a', 'y', 'c', 5]
```

```
d.values()
```

```
    [1, 5, Integer Ring, 3]
```

```
d.items()
```

```
    [('a', 1), ('y', 5), ('c', Integer Ring), (5, 3)]
```

```
var('x,y')
sol=solve([x+y==1,x-3*y==5],[x,y])
sol
```

```
    [[x == 2, y == -1]]
```

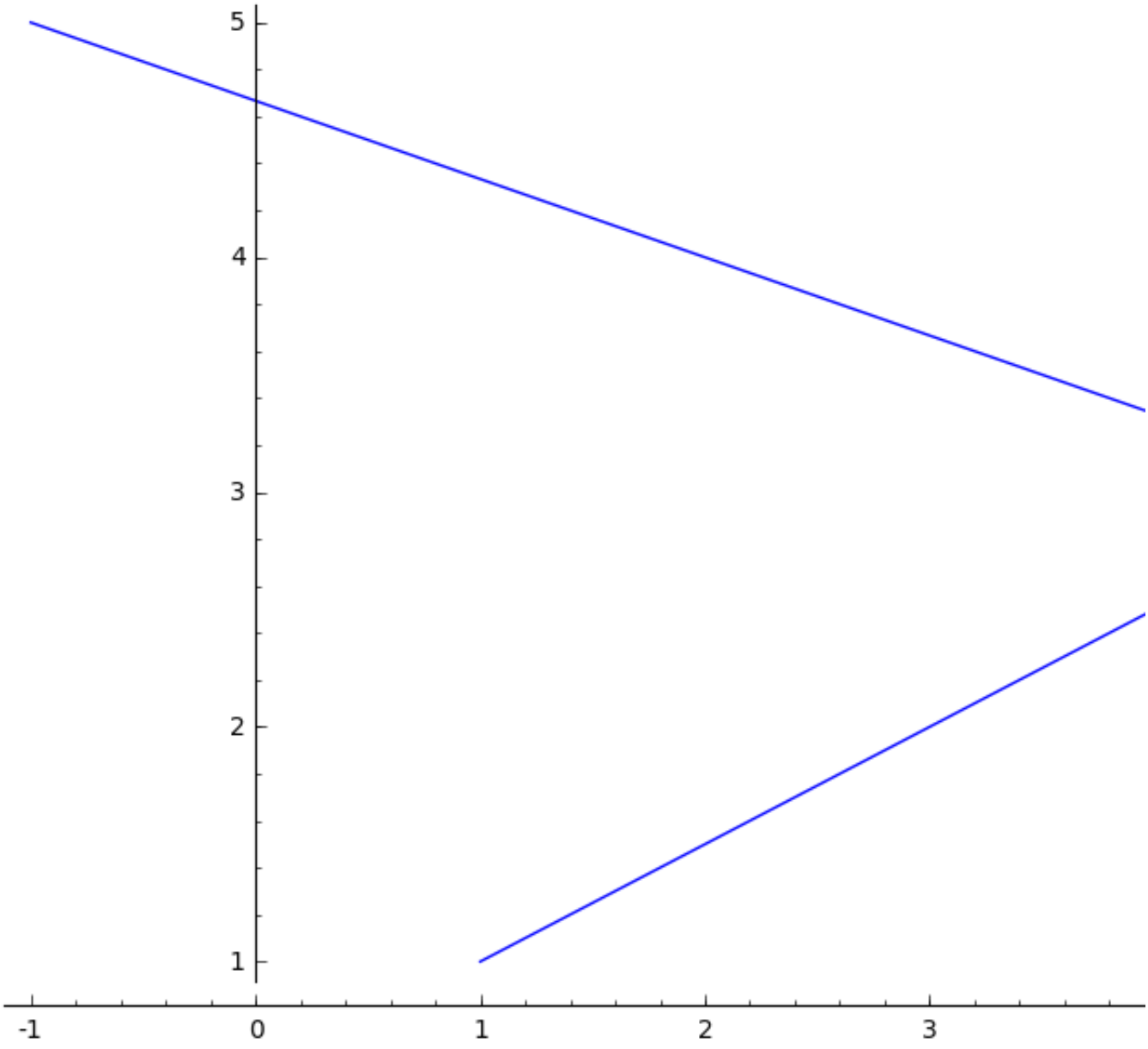
```
sol=solve([x+y==1,x-3*y==5],[x,y],solution_dict=True)
sol
```

```
    [{y: -1, x: 2}]
```

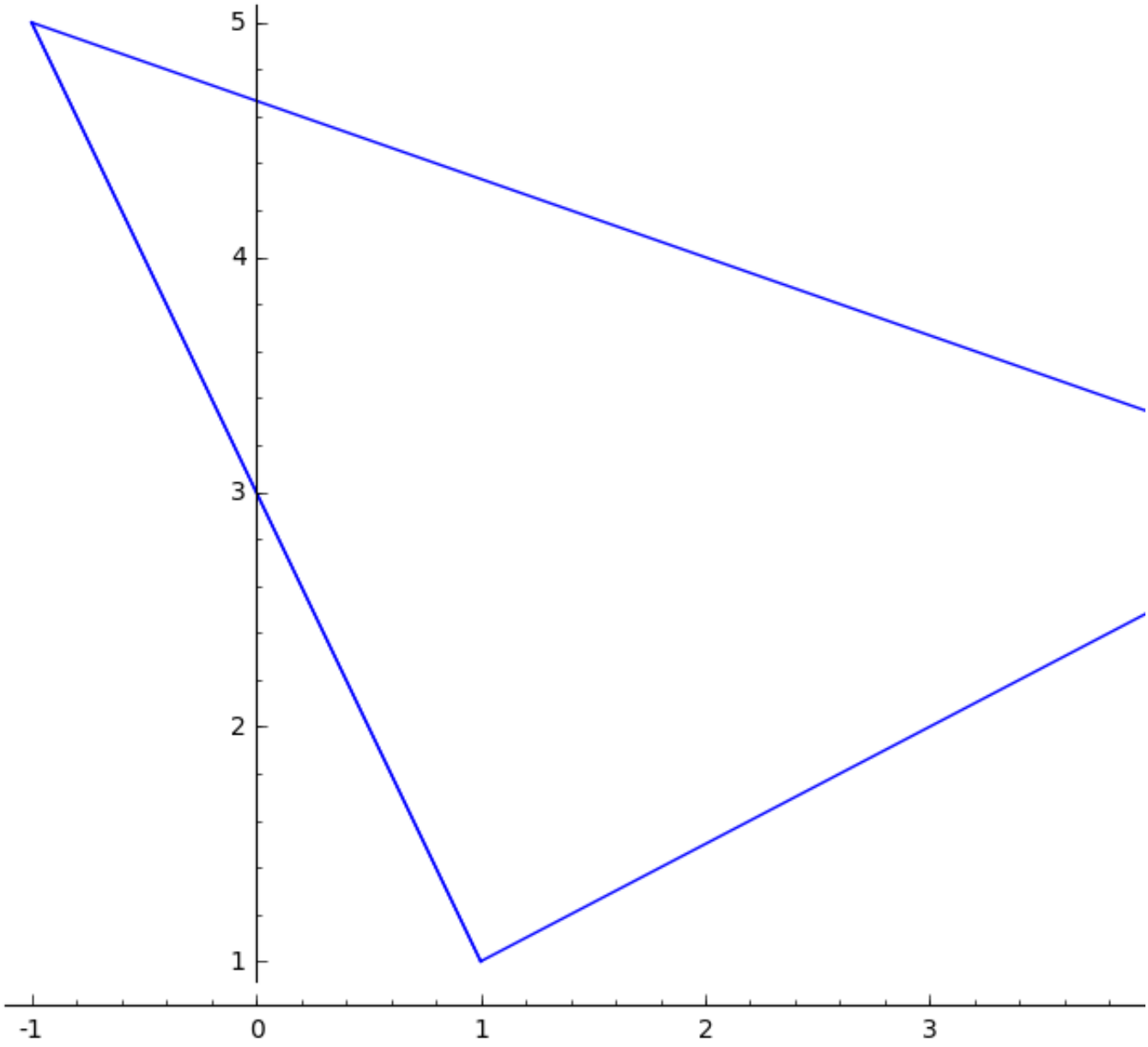
```
sol[0][x]
```

```
    2
```

```
A=[1,1]
B=[5,3]
C=[-1,5]
g=line([A,B,C])
g
```




```
g=g+ line([A,C])
g
```




```
(A+B)/2
```

```
Traceback (click to the left of this block for traceback)
...
TypeError: unsupported operand parent(s) for /: '<type
'list'>' and 'Integer Ring'
```

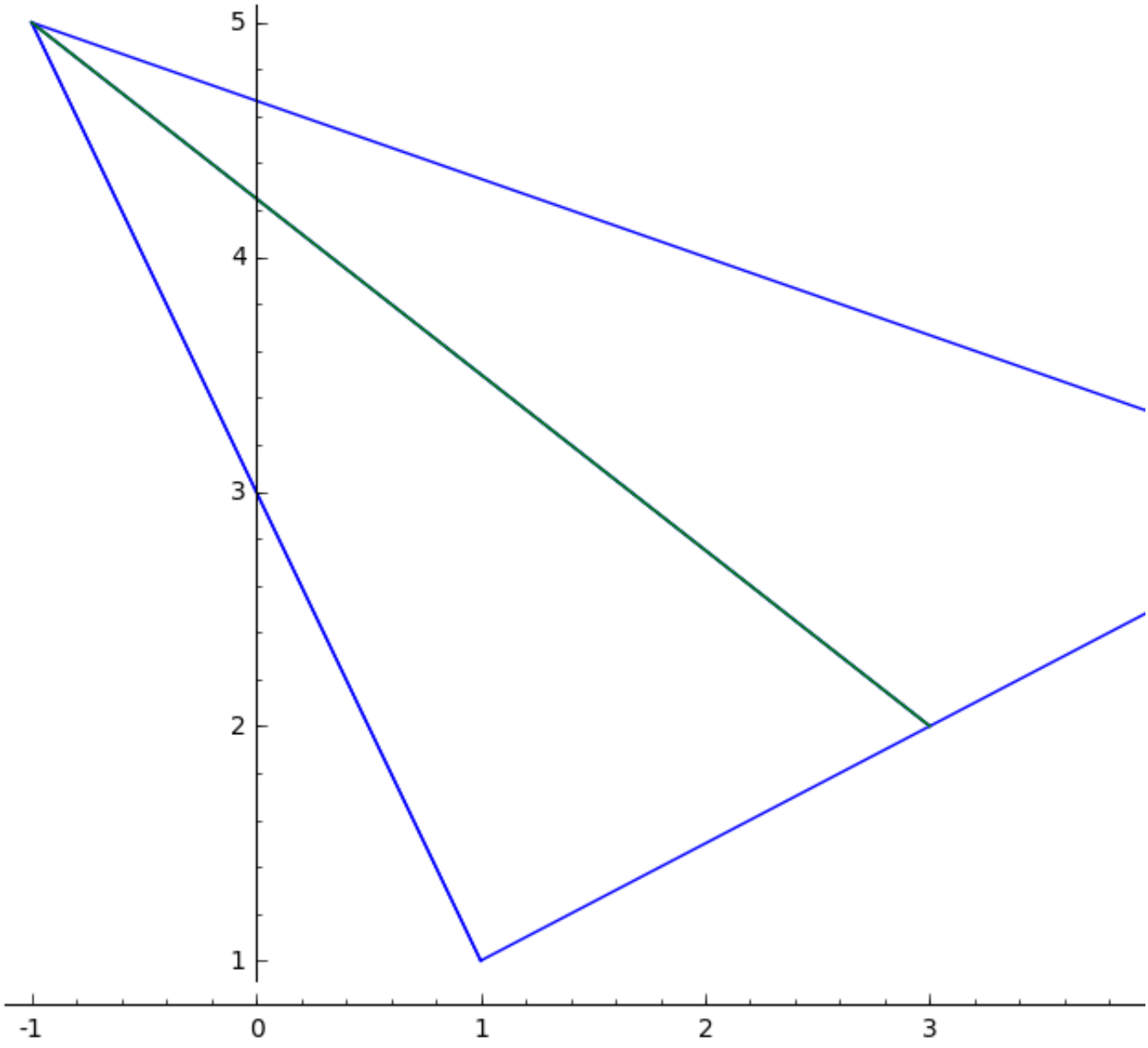
```
A=vector(A)
B=vector(B)
C=vector(C)
A
```

```
(1, 1)
```

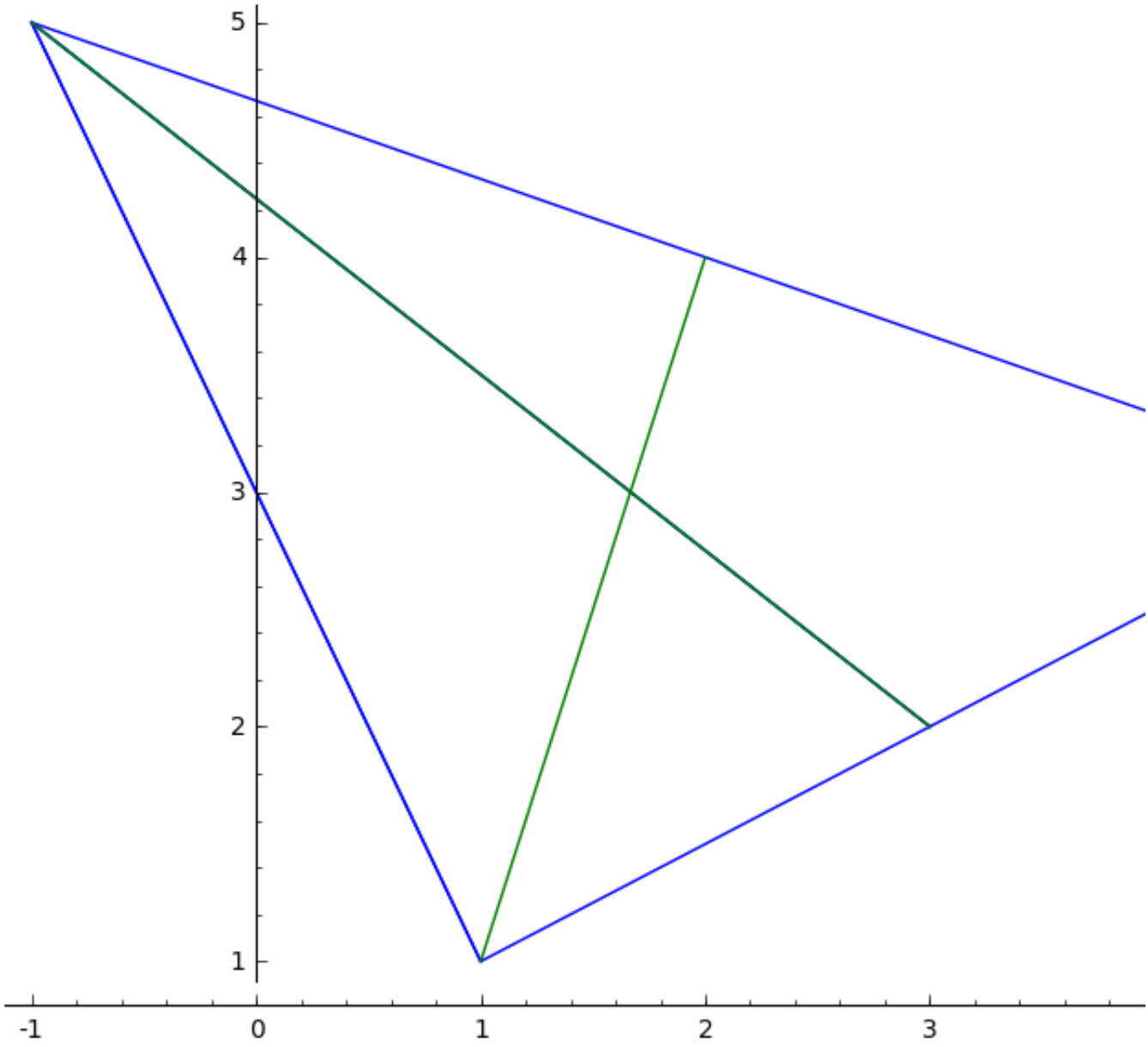
```
(A+B)/2
```

```
(3, 2)
```

```
g=g+ line([(A+B)/2,C],color='green')
g
```




```
g=g+ line([(B+C)/2,A],color='green')
g
```




```
var('s,t')
```

```
(s, t)
```

```
sA=A+s*((B+C)/2-A)
sA
```

```
(s + 1, 3*s + 1)
```

```
sC=C+t*((A+B)/2-C)
sC
```

```
(4*t - 1, -3*t + 5)
```

```
solve(sA-sC,[s,t])
```

```
Traceback (click to the left of this block for traceback)
...
TypeError: The first argument must be a symbolic expression or a
list of symbolic expressions.
```

```
st=solve(list(sA-sC),[s,t])
```

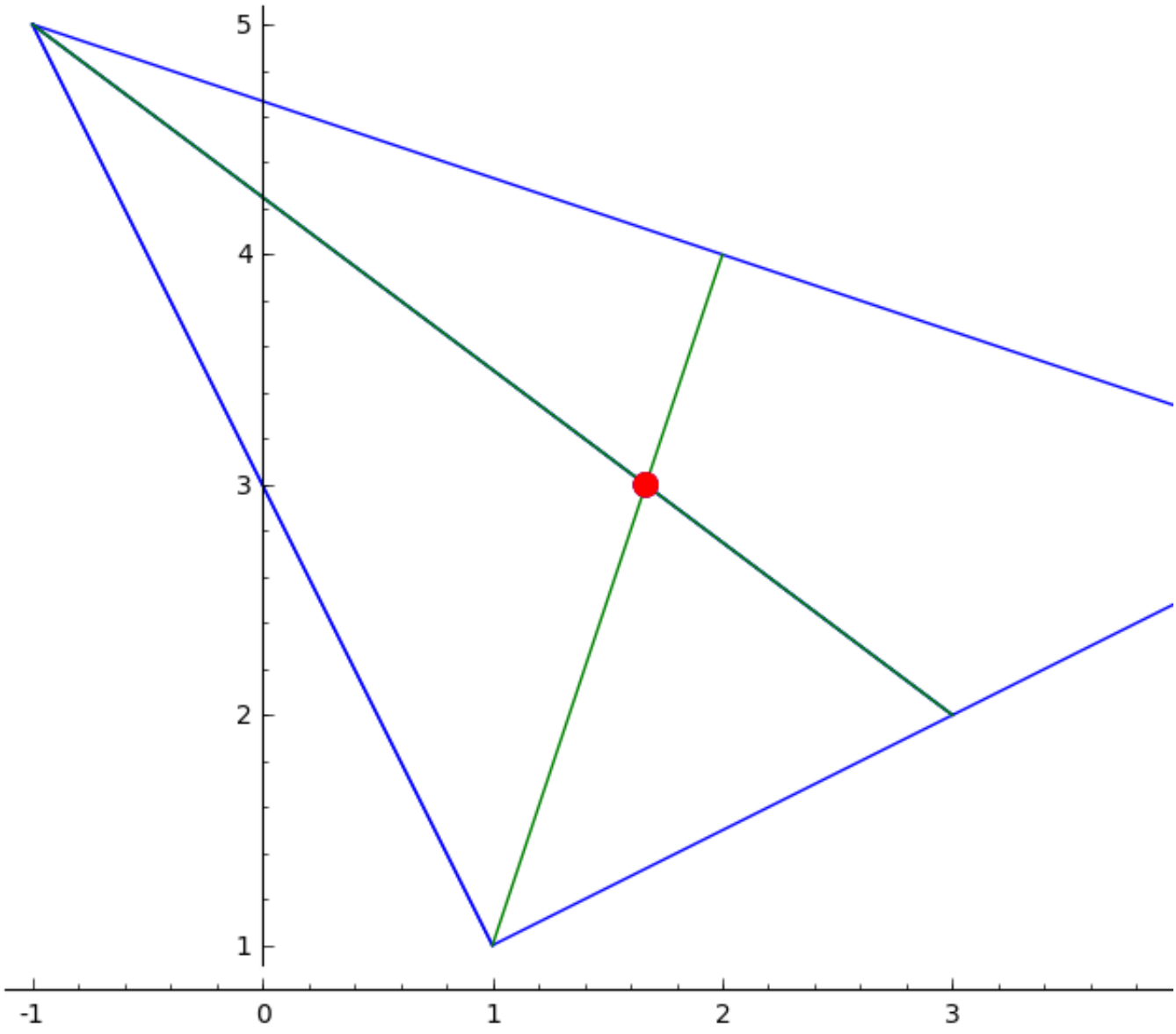
```
st
```

```
[[s == (2/3), t == (2/3)]]
```

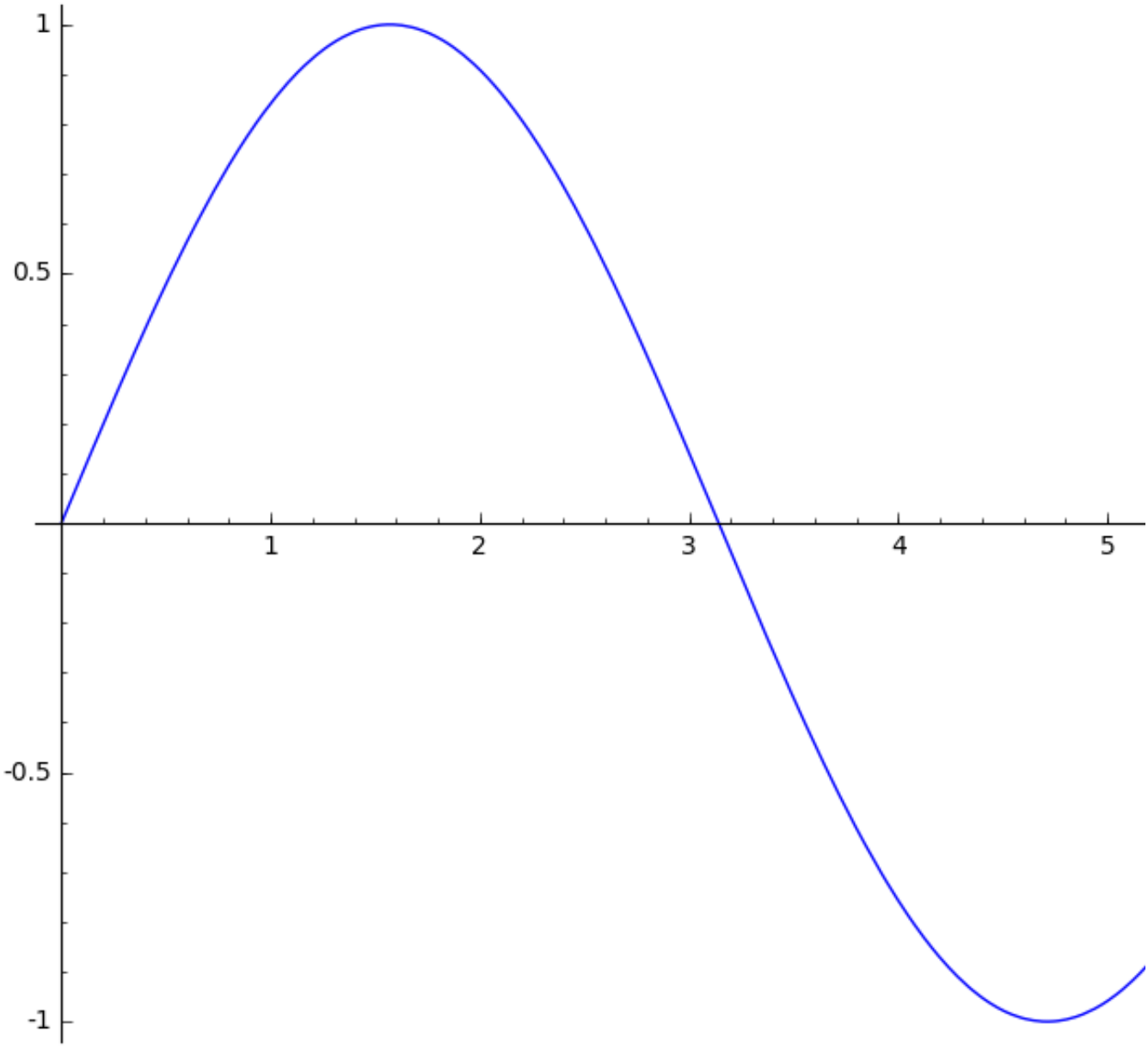
```
S=sA.subs(st[0])
S
```

```
(5/3, 3)
```

```
g += circle(S,0.05,color='red',fill='red')
g
```



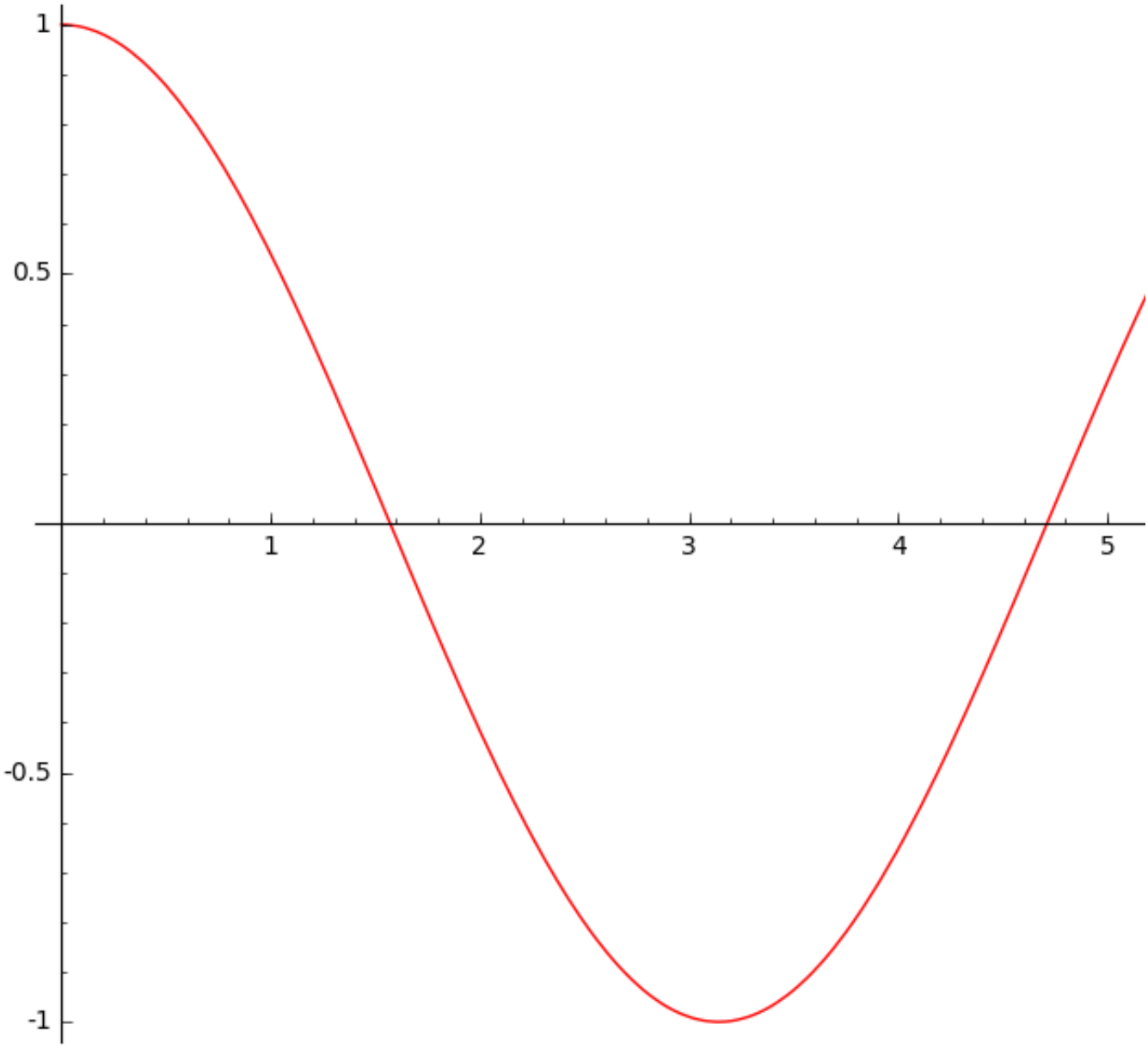

```
g1=plot(sin(x),x,0,2*pi)
g1
```



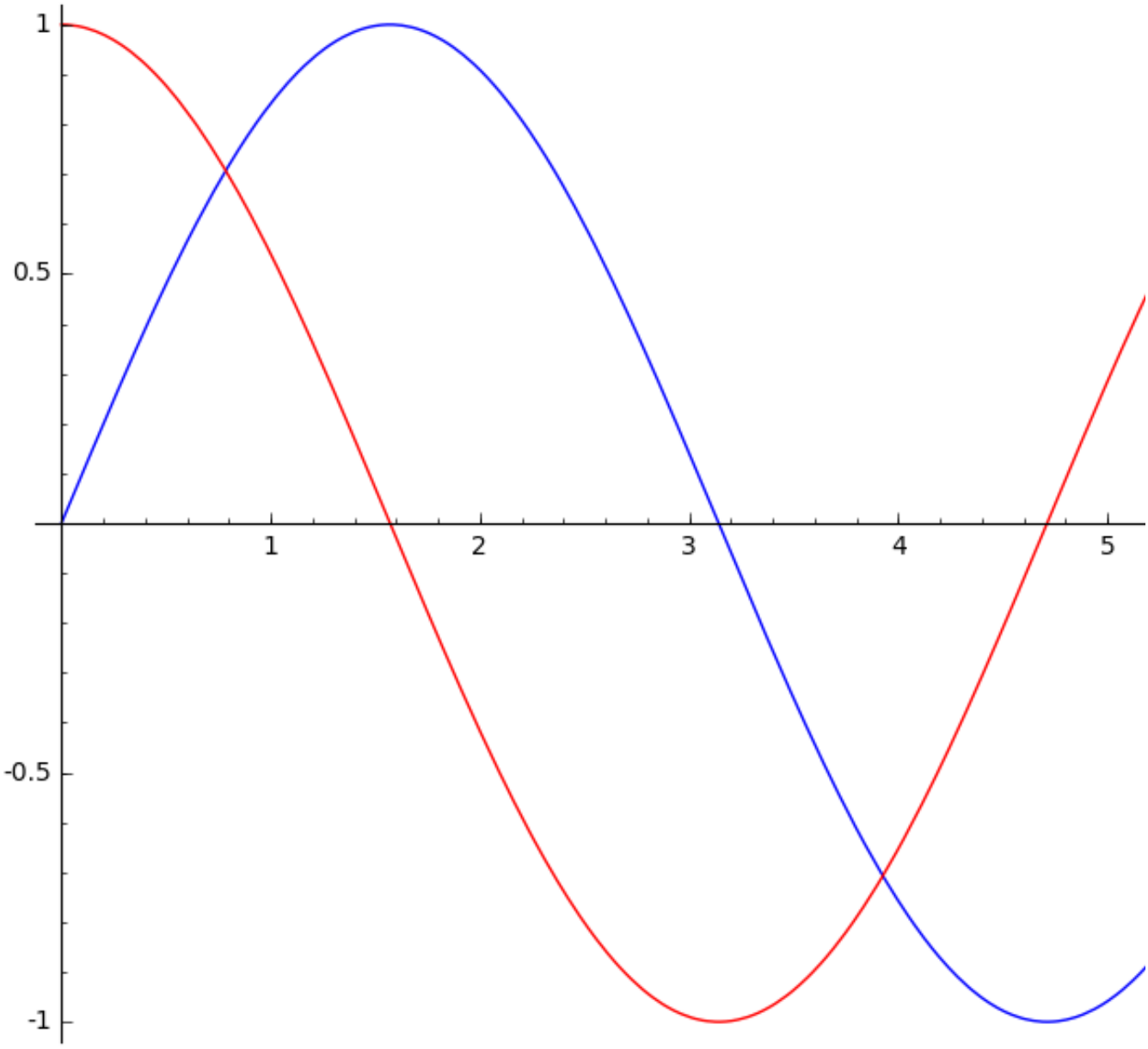

```
g1.save('plot.pdf')
```

plot.pdf

```
g2=plot(cos(x),x,0,2*pi,color='red')
g2
```

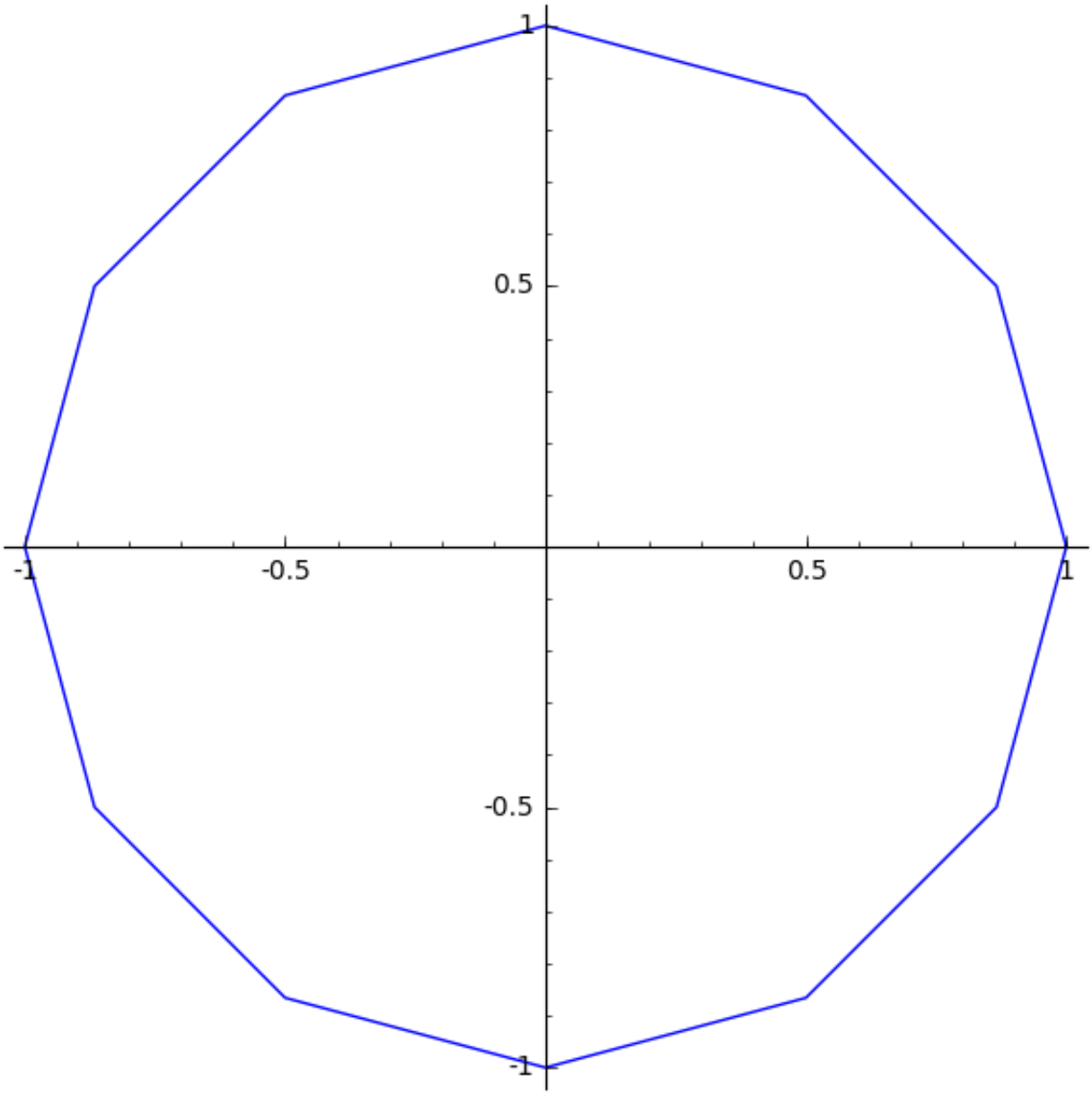



```
g3=g1 + g2
g3.show()
```

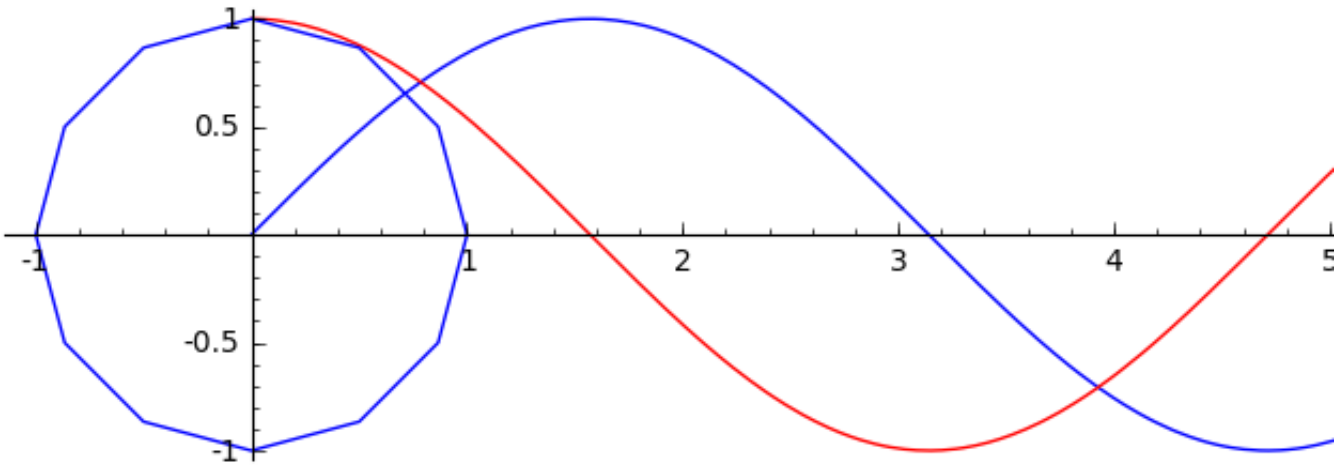



```
def ngon(n):
    return [[cos(2*pi*k/n),sin(2*pi*k/n)] for k in range(0,n+1)]
```

```
g12 = list_plot(ngon(12),aspect_ratio=1,plotjoined=True)
g12
```



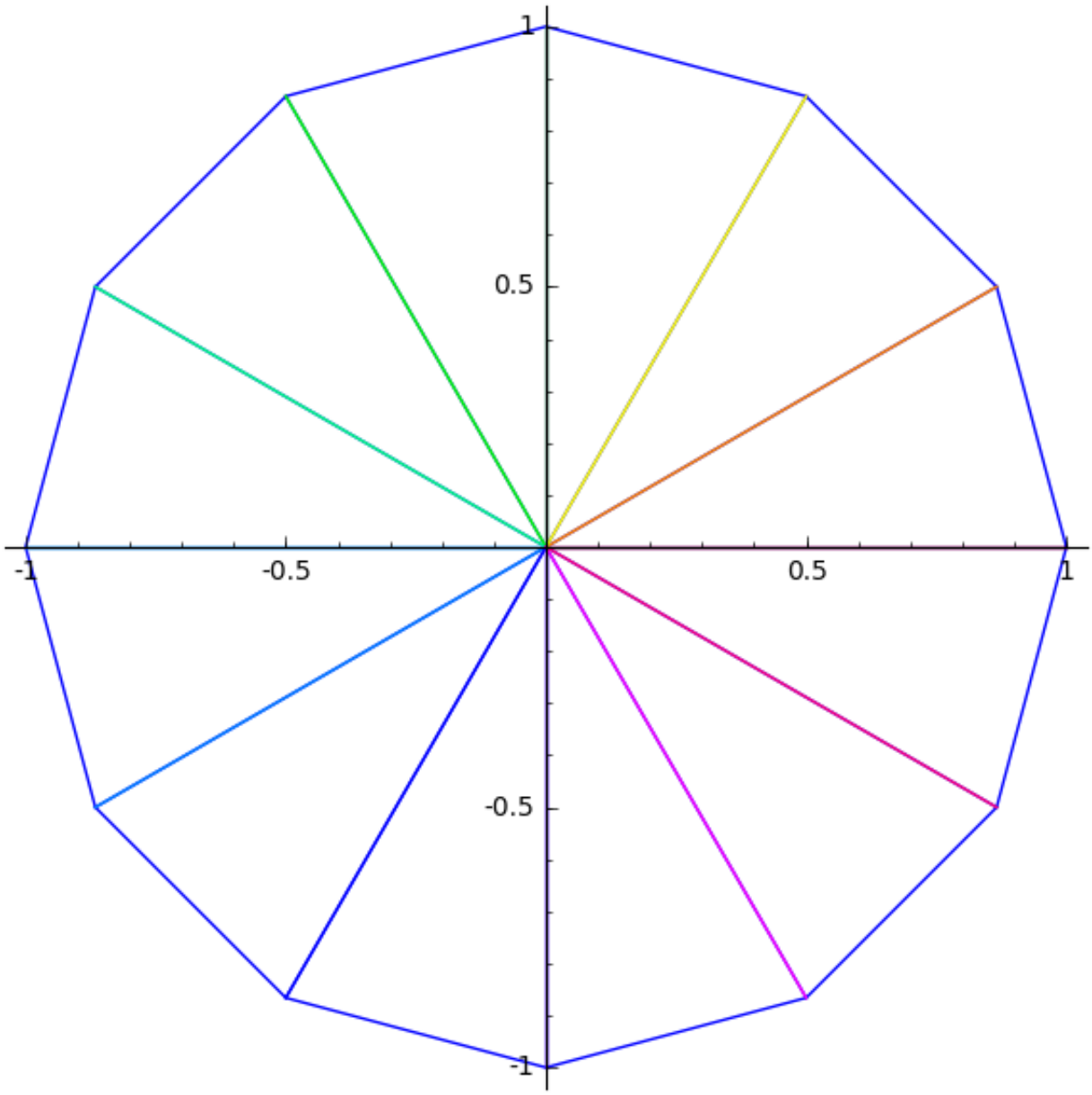

```
h = g3+g12
h
```



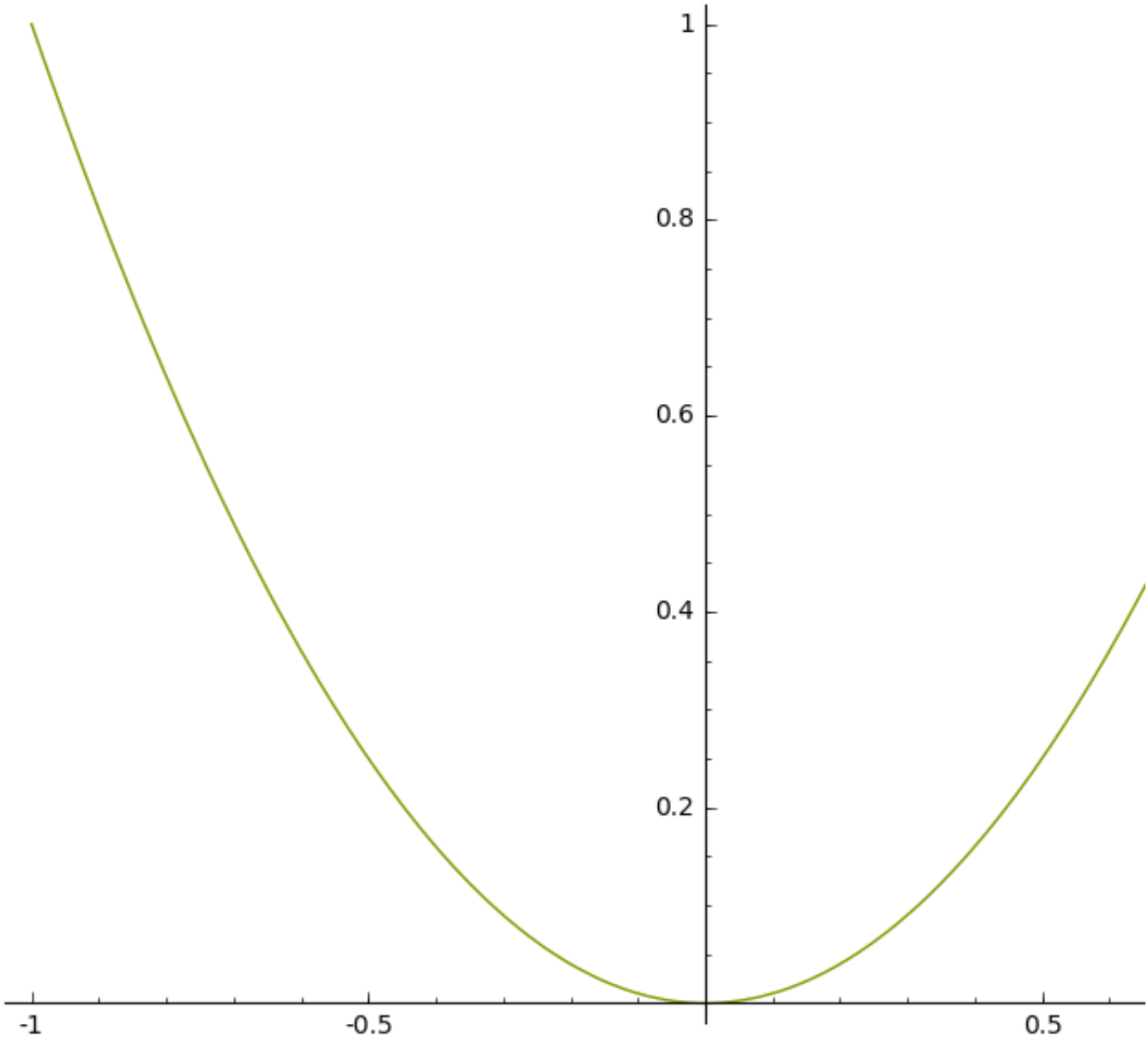

```
for k in range(12):
```

```
    g12 += line([[0,0],[cos(2*pi*k/12),sin(2*pi*k/12)]],hue=k/12)
```

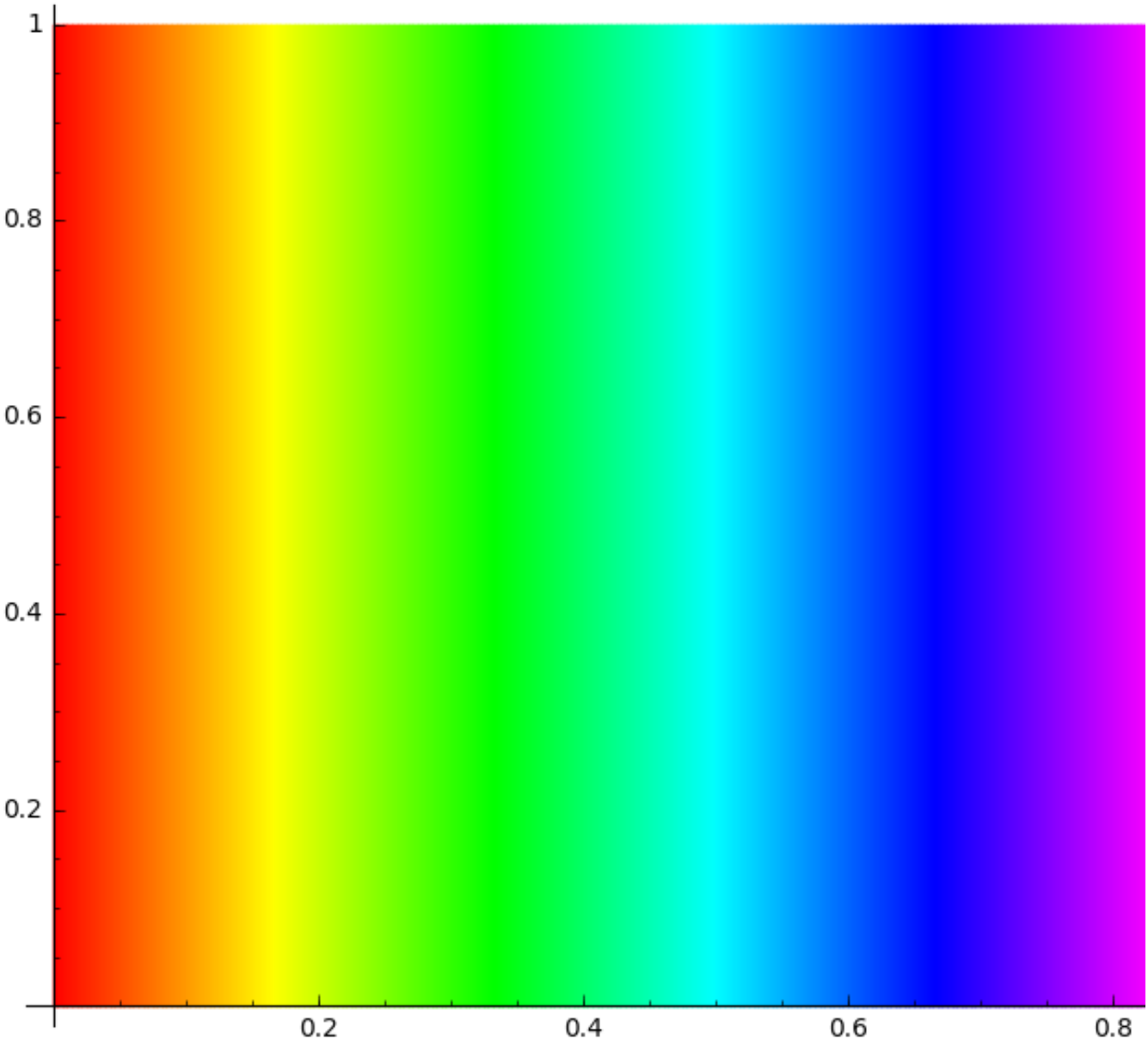
```
g12
```



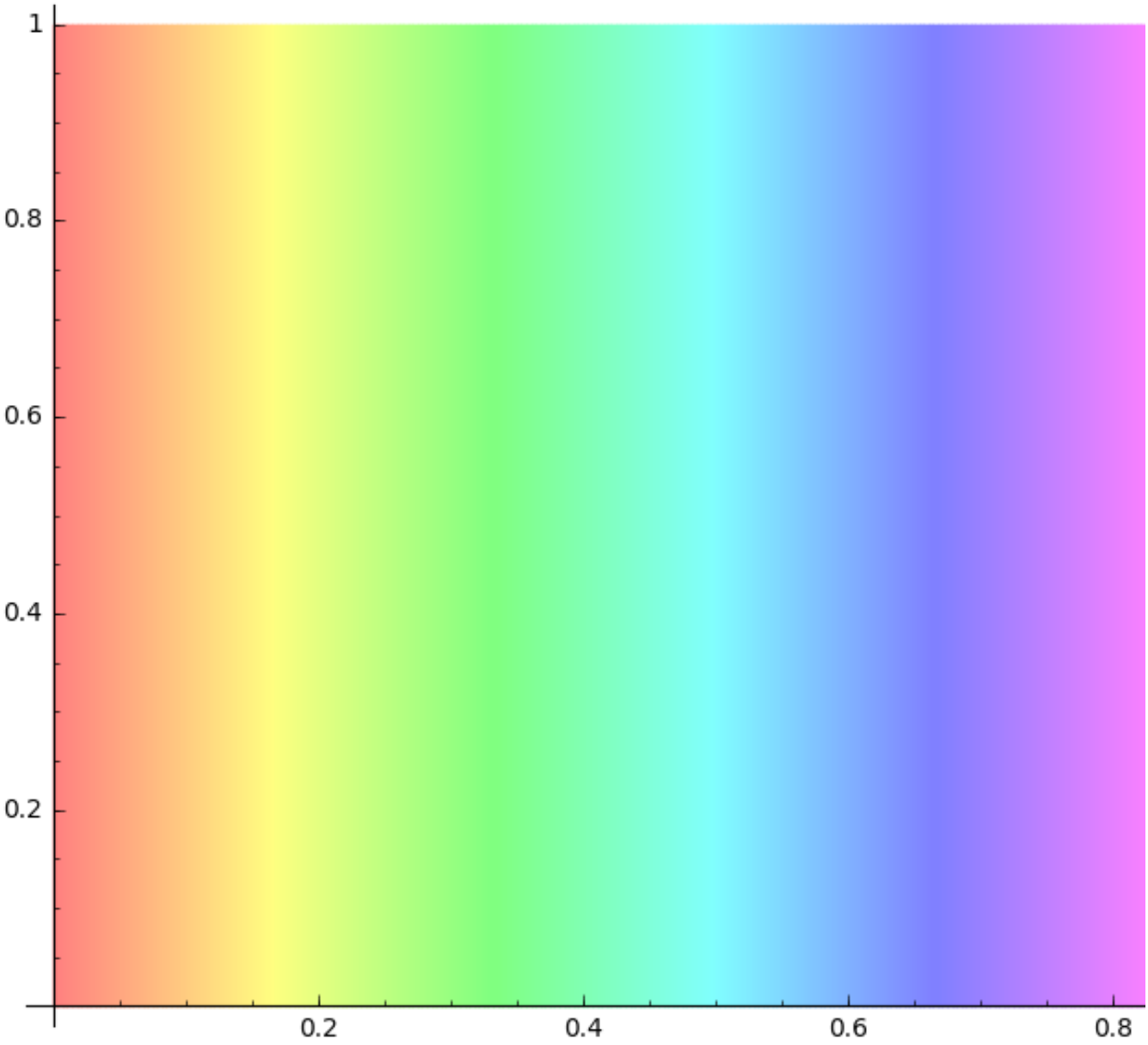

```
plot(x^2,x,-1,1,rgbcolor=[0.5,0.6,0])
```



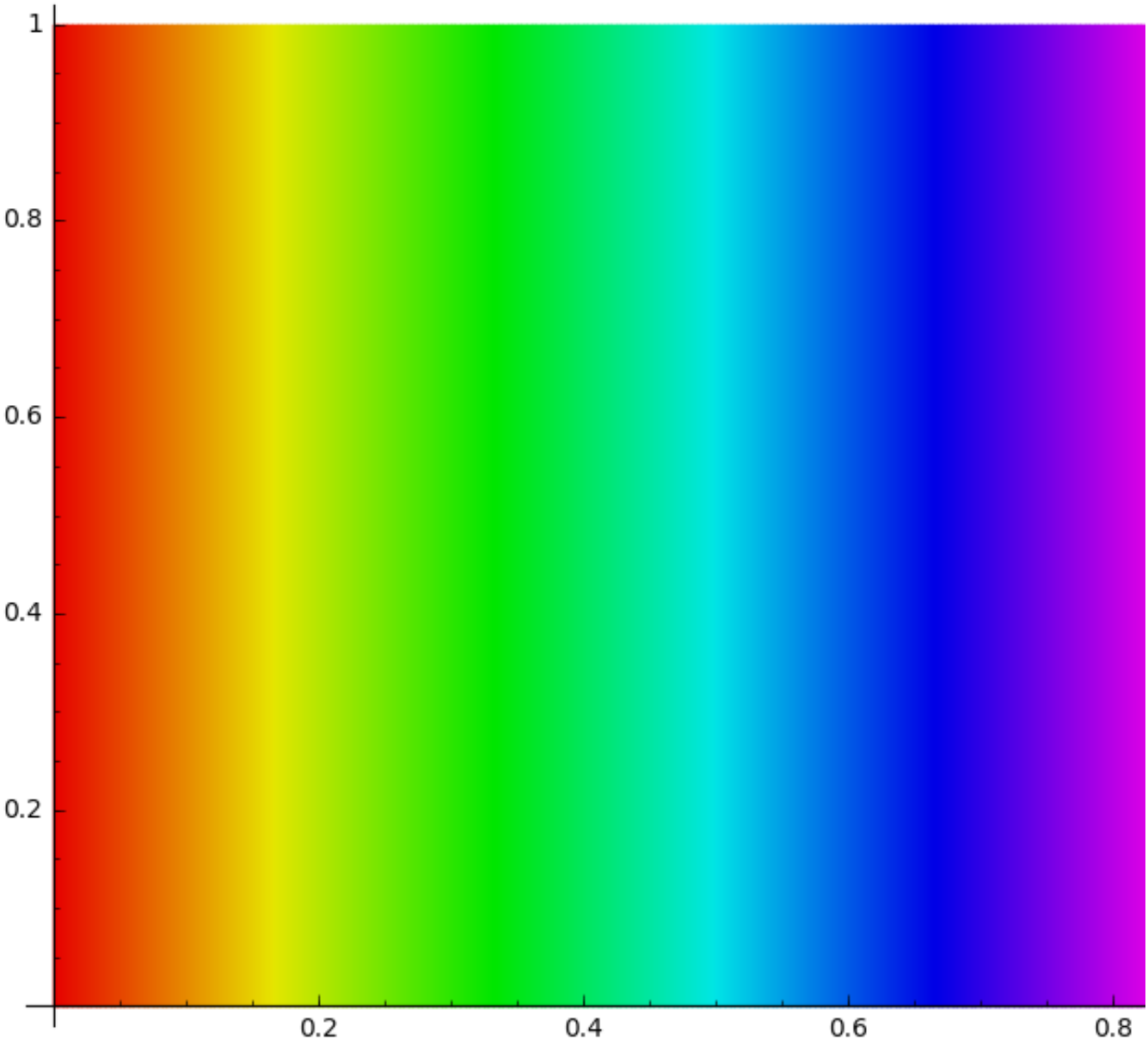

```
p=Graphics()
for x in srange(0,1,0.001):
    p=p+ line([[x,0],[x,1]],hue=x)
p
```




```
p=Graphics()
for x in srange(0,1,0.001):
    p=p+ line([[x,0],[x,1]],rgbcolor=hue(x,s=0.5))
p
```




```
p=Graphics()
for x in srange(0,1,0.001):
    p=p+ line([[x,0],[x,1]],rgbcolor=hue(x,v=0.9))
p
```


1
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8