Double-click here to insert team members' names: William Taylor, Danny Garmendez

In [1]:
```
from sympy import *
from sympy.plotting import (plot, plot_parametric)
```

1a Equation of tangent line when t=pi/4

In [12]:
```
t = symbols('t')
xot = cos(t)
yot = sin(t) + 3

xp = diff(xot, t)
yp = diff(yot, t)
dydx = yp/xp

i = symbols('i')
tangent = dydx.subs(t, i) * (t - xot.subs(t, i)) + yot.subs(t, i)
print("The equation of the tangent line when t = pi/4 is y =", tangent.subs(i, pi/4))
```

```
The equation of the tangent line when t = pi/4 is y = -t + sqrt(2) + 3
```

1b points where tangent line is vertical

In [3]:
```
verticalTangent = solve(dydx, t)
print("The tangent line is vertical at t =", verticalTangent)
```

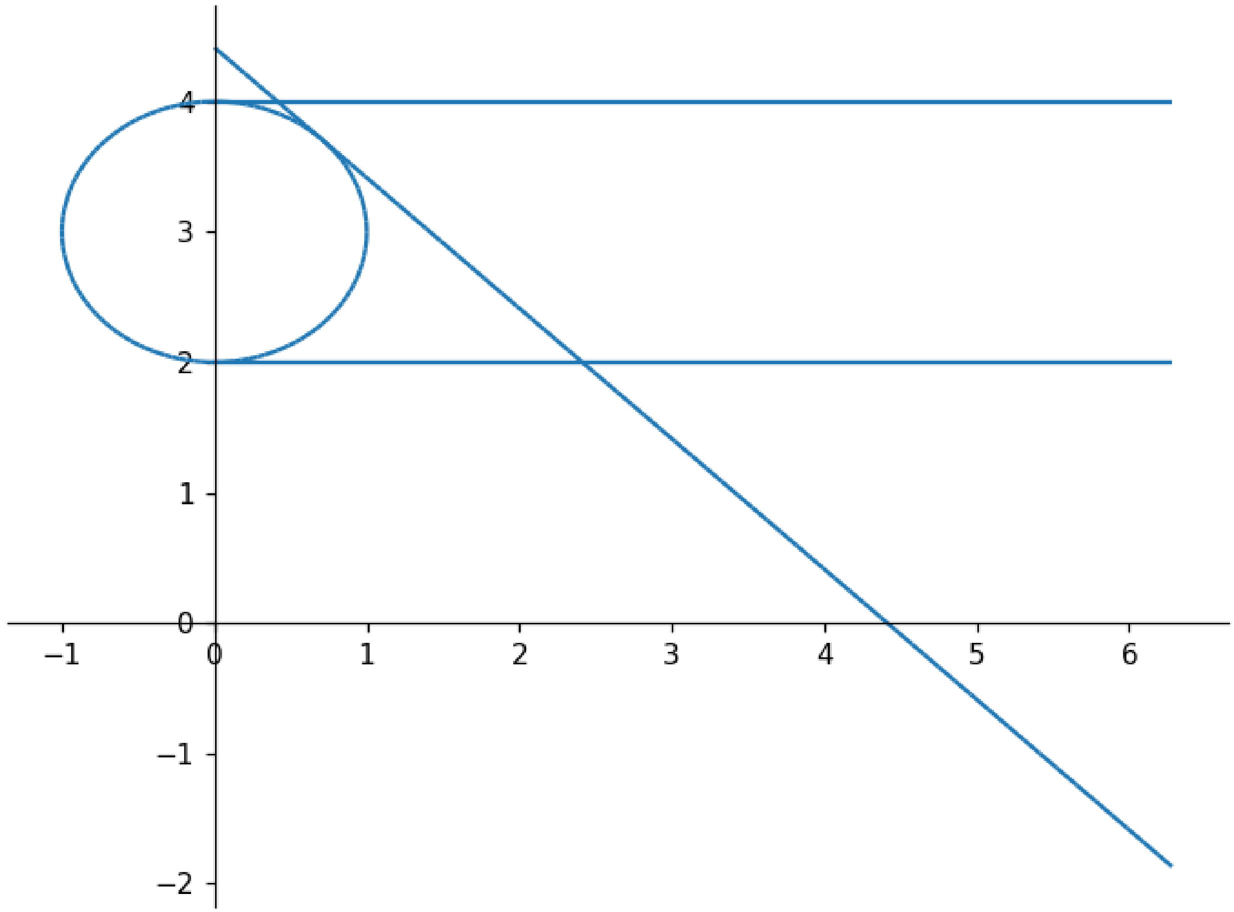
```
The tangent line is vertical at t = [pi/2, 3*pi/2]
```

1c Plot parametrized curve and all tangent lines

In [4]:
```
matplotlib notebook
```

In [15]:
```
graph = plot_parametric(xot, yot, (t, 0, 2*pi), show=False)
tangent1 = plot((tangent.subs(i, pi/4), (t, 0, 2*pi)),
                (tangent.subs(i, verticalTangent[0]), (t, 0, 2*pi)),
```

```
                (tangent.subs(i, verticalTangent[1]), (t, 0, 2*pi)), show=False)
graph.extend(tangent1)
graph.show()
```



2 Tangent lines at (3,0)

In [24]:

```
xot = t ** 2
yot = t ** 3 - 3 * t
xSolutions = solve(xot-3, t)
ySolutions = solve(yot, t)
```

```
dydx = diff(yot, t) / diff(xot, t)
tangent = dydx.subs(t, i) * (t - xot.subs(t, i)) + yot.subs(t, i)
print("The first intersection at t = {} is y = {}".format(xSolutions[0], tangent.subs(i, xSolutions[0])))
print("The second intersection at t = {} is y = {}".format(xSolutions[1], tangent.subs(i, xSolutions[1])))
```

```
The first intersection at t = -sqrt(3) is y = -sqrt(3)*(t - 3)
The second intersection at t = sqrt(3) is y = sqrt(3)*(t - 3)
```

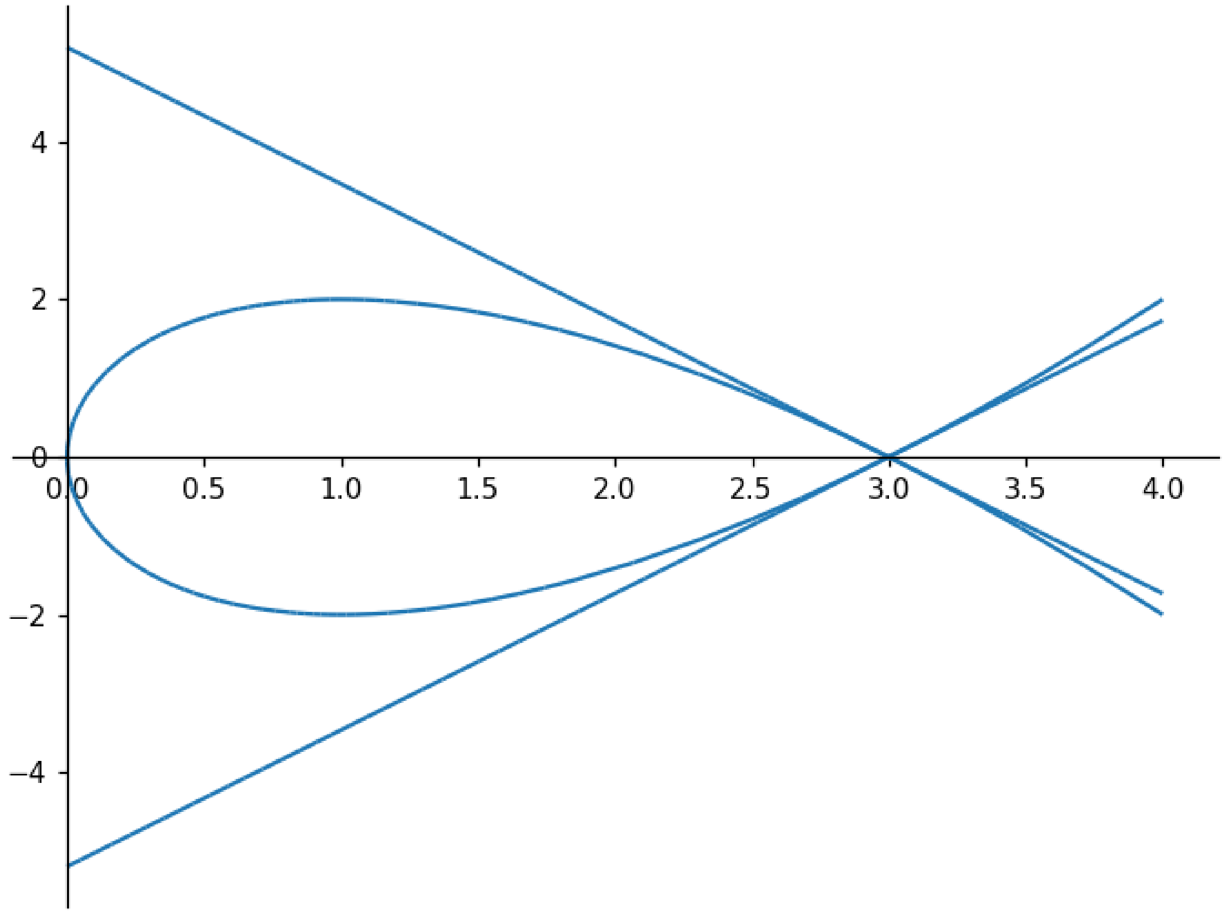
2b Plot of parametrized curve and tangent lines

In [ ]:

```
matplotlib notebook
```

In [27]:

```
graph = plot_parametric(xot, yot, (t, -2, 2), show=False)
tangents = plot((tangent.subs(i, xSolutions[0]), (t, 0, 4)), (tangent.subs(i, xSolutions[1]), (t, 0, 4)), show=False)
graph.extend(tangents)
graph.show()
```



3a find k and y0

In [97]:
```
time1 = Rational(1, 1)
time2 = Rational(1.5, 1)
count1 = 200
count2 = 360

y0 = symbols('y0', real=True)
k = symbols('k', real=True)
t = symbols('t', real=True)
y = symbols('y', real=True)
```

```
total = y0 * exp(k * t)

total1 = (total - y).subs({y:count1, t:time1})
total2 = (total - y).subs({y:count2, t:time2})

solutions = solve([total1, total2], [y0, k])
print("Given those values of t and y, y0 = {} and k = {}".format(solutions[0][0], solutions[0][1]))
print("Given those values of t and y, y0 = {:.4f} and k = {:.4f}".format(solutions[0][0].evalf(), solutions[0][1].evalf()
```

```
Given those values of t and y, y0 = 5000/81 and k = log(81/25)
Given those values of t and y, y0 = 61.7284 and k = 1.1756
```

3b When population = 2000

In [92]:

```
function = total.subs({y0:solutions[0][0], k:solutions[0][1]})
answer = solve(2000 - function, t)
print("At time t =", answer[0], " the total population is 2000, and")
print("at time t =", answer[0].evalf(), " the total population is 2000")
```

```
At time t = log((162/5)**(1/(2*log(9/5))))  the total population is 2000, and
at time t = 2.95869116338109  the total population is 2000
```

3c find k and population 1 hour before "initial"

In [107…

```
time1 = 0
time2 = .5

total = y0 * exp(k * t)


total1 = (total - y).subs({y:count1, t:time1})
total2 = (total - y).subs({y:count2, t:time2})

solutions = solve([total1, total2], [y0, k])
function = total.subs({y0:solutions[0][0], k:solutions[0][1]})

print("k =", solutions[0][1])
print("The total population 1 hour before the start time is", function.subs(t, -1))

print("The k values in both are the same")
print("Also, the initial population from part A is equal to the total population in part C\n\n")
```

```
print("This is because if you add one hour to the equation in part C, they would be the same function")
print("meaning if you calculate the time at one hour earlier, they should result in the same value.")
```

```
k = 1.17557332980424
The total population 1 hour before the start time is 61.7283950617284
The k values in both are the same
Also, the initial population from part A is equal to the total population in part C


This is because if you add one hour to the equation in part C, they would be the same function
meaning if you calculate the time at one hour earlier, they should result in the same value.
```

4a rate of change in f with respect to each variable

In [49]:

```
L = symbols('L')
T = symbols('T')
p = symbols('p')
f = symbols('f')
parameters = [L, T, p]
fotpl = sqrt(T / p) / (2 * L)

for i in parameters:
    print("The rate of change with respect to", i, " is", diff(fotpl, i))
```

```
The rate of change with respect to L  is -sqrt(T/p)/(2*L**2)
The rate of change with respect to T  is sqrt(T/p)/(4*L*T)
The rate of change with respect to p  is -sqrt(T/p)/(4*L*p)
```

4b interpret what happens to the pitch

In [34]:

```
print("If L decreases, the rate of change will be more negative, meaning the pitch will decrease but at an increasingly f
print("If T increases, the rate of change will be less positive, meaning the pitch will increase but at an increasingly s
print("If p increases, the rate of change will be less negative, meaning the pitch will decrease but at an increasingly s
```

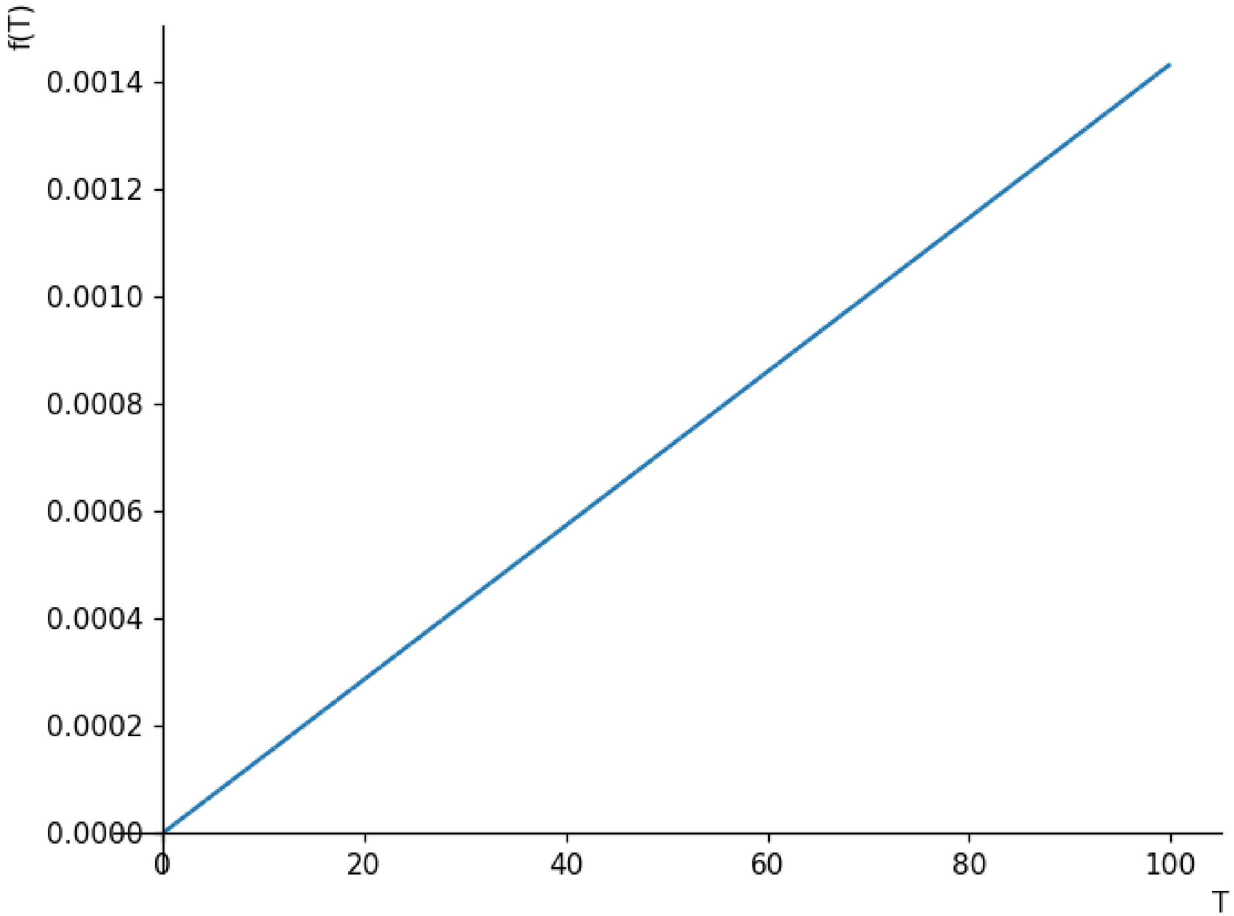
```
If L decreases, the rate of change will be more negative, meaning the pitch will decrease but at an increasingly faster r
ate
If T increases, the rate of change will be less positive, meaning the pitch will increase but at an increasingly slower r
ate
If p increases, the rate of change will be less negative, meaning the pitch will decrease but at an increasingly slower r
ate
```

4c Plot rho vs T

In [ ]:
```
matplotlib notebook
```

In [58]:
```
poT = solve(fotpl - f, p)[0].subs({L:.3, f:440}).evalf()
print("If L = .3 and f = 440 Hz, p =", poT)
plot(poT, (T, 0, 100))
```

```
If L = .3 and f = 440 Hz, p = 1.43480257116621e-5*T
```



Out[58]:
```
<sympy.plotting.plot.Plot at 0x2723b254970>
```

4d tuning tension when rho=.00078

In [60]:

```
print("If p is .00078, then T needs to be", solve(poT - .00078, T)[0])
```

```
If p is .00078, then T needs to be 54.3628800000000
```

In [ ]: