In [2]:
```
from sympy import * # You need to start by importing this symbolic package
```

Example 1 (part a): Verify y = Ce^(at) is a solution to the ODE dy/dt = a * y. **NOTE: The second part of this solution will be important for all future computer assignments!!!**

In [3]:
```
t=symbols('t')  # Defines t as a symbolic variable
C,a=symbols('C a')  # Defines C and a as symbolic variables at the same time
y=C*exp(a*t)
LHS=diff(y,t)
RHS=a*y
print('LHS =',LHS,'and RHS =',RHS)  # the print command produces output and explanatory text
```

```
LHS = C*a*exp(a*t) and RHS = C*a*exp(a*t)
```

In [4]:
```
# ************Let's actually solve the ODE using the dsolve command ************
y=Function('y')  # Defines y as a symbolic function
deq=diff(y(t),t)-a*y(t) # PERSONAL PREFERENCE: I like to move everything to one side to solve EXPRESSION = 0
ysoln=dsolve(deq,y(t))
print('The solution to the ODE is',ysoln)
print('Eq stands for "Equation". Arguments are LHS and RHS.')
```

```
The solution to the ODE is Eq(y(t), C1*exp(a*t))
Eq stands for "Equation". Arguments are LHS and RHS.
```

Example 1 (part b): If y(0)=10, we can use the **ics** option to solve an IVP

In [5]:
```
ysoln=dsolve(deq,y(t),ics={y(0):10})
print('The solution to the IVP is',ysoln)
```

```
The solution to the IVP is Eq(y(t), 10*exp(a*t))
```

Example 3: Find the values of r for which y = t^r is a solution to the ODE t^2y''' - 4ty'' + 4y' = 0

In [6]:
```
r,t=symbols('r t')
y=t**r
LHS=t**2*diff(y,t,3)-4*t*diff(y,t,2)+4*diff(y,t)
print('Equation becomes',LHS,'=0, or ',LHS.factor(),'=0')
# Solution seems obvious, but will solve in Python to confirm
r_soln=solve(LHS,r)
print('Values of r are',r_soln)
# Confirming with dsolve
y=Function('y')
LHS=t**2*diff(y(t),t,3)-4*t*diff(y(t),t,2)+4*diff(y(t),t)
dsolve(LHS,y(t))
```

```
Equation becomes -4*r*t**r*(r - 1)/t + r*t**r*(r**2 - 3*r + 2)/t + 4*r*t**
r/t =0, or  r*t**r*(r - 5)*(r - 2)/t =0
Values of r are [0, 2, 5]
```

Out[6]: $y(t) = C_1 + C_2t^2 + C_3t^5$

Example 4: Object dropped from rest. Forces are the weight of the object (m *g) and air resistance (c* v). By Newton's Second Law, F = ma = mg - cv, or mv' = mg - cv and the initial condition is v(0)=0 ("from rest")

In [10]:
```
t,m,g,c=symbols('t m g c')
v=Function('v')
deq=m*diff(v(t),t)-m*g-c*v(t) # moved everything to one side
vsoln=dsolve(deq,v(t),ics={v(0):0})  # assumed  deq = 0
print('The solution to the ODE is',vsoln.expand())
# General solution without the initial condition:
gensoln=dsolve(deq,v(t))
print('Without the initial condition, the solution is',gensoln.expand())
print('Note that exp(C1*c/m) is also an arbitrary constant and can be repla
ced with C1')
```

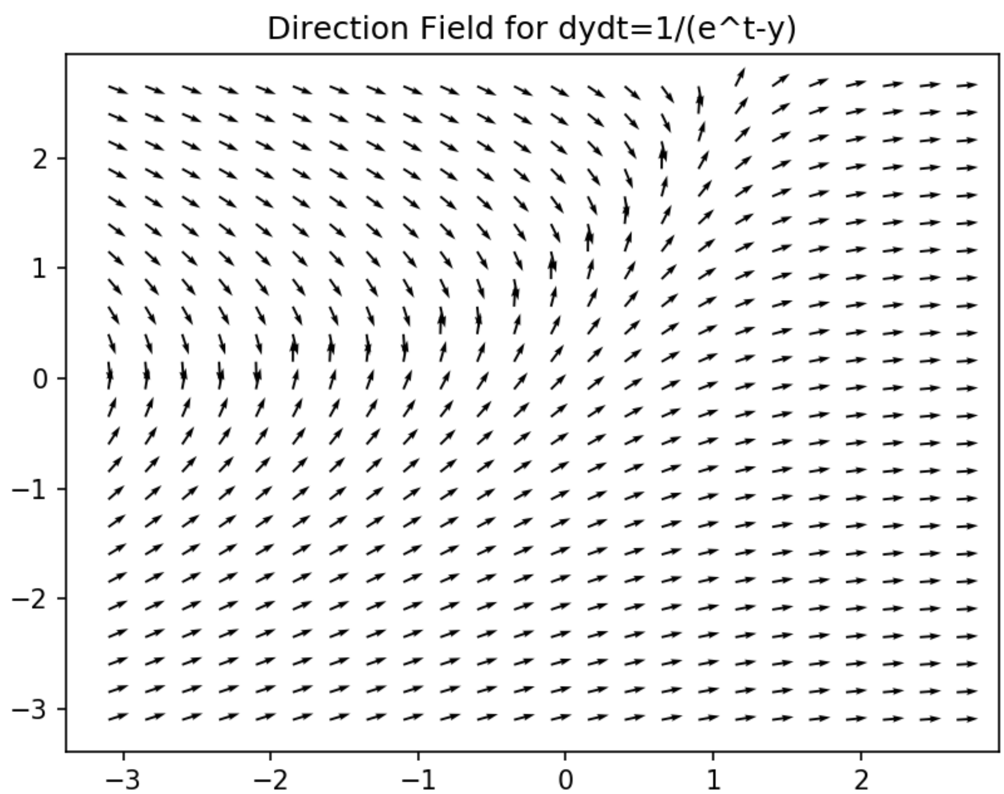
```
The solution to the ODE is Eq(v(t), g*m*exp(c*t/m)/c - g*m/c)
Without the initial condition, the solution is Eq(v(t), -g*m/c + exp(C1*c/
m)*exp(c*t/m)/c)
Note that exp(C1*c/m) is also an arbitrary constant and can be replaced wit
h C1
```

Example 5: Direction field. To do this, we need to use the NUMPY package instead of SYMPY.

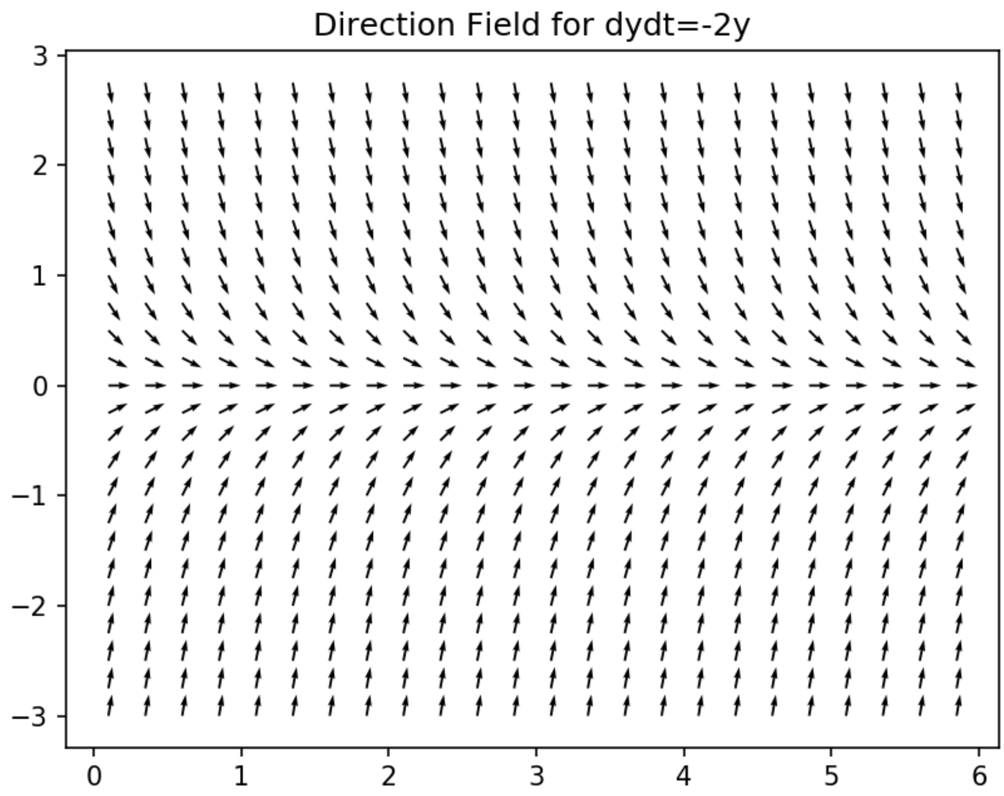
In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [12]:
```
matplotlib notebook
```

In [13]:
```
T, Y = np.meshgrid(np.arange(-3.1, 2.9, .25), np.arange(-3.1, 2.9, .25)) # adjust domain and range and spacing as needed
dYdT = 1/(np.exp(T)-Y) # put f(t,y) here to find slope
U = 1/(1+dYdT**2)**0.5*np.ones(T.shape) # Normalizes the arrows to see near-zero slopes.
V = 1/(1+dYdT**2)**0.5*dYdT
plt.figure()
plt.title('Direction Field for dydt=1/(e^t-y)')
Q = plt.quiver(T, Y, U, V) # draws the arrows at (X,Y) with slope dYdX
```



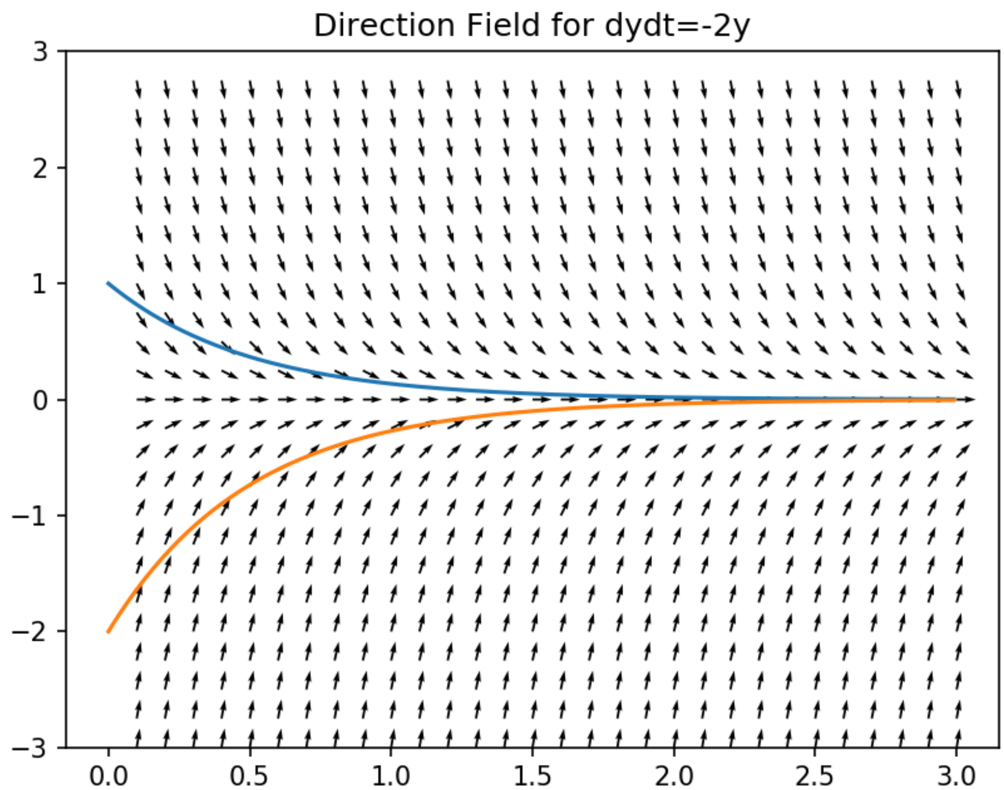
In [14]:
```
T, Y = np.meshgrid(np.arange(0.1, 6.1, .25), np.arange(-3, 3, .25)) # adjus
ting to avoid zero denominator
dYdT = -2*Y # put f(t,y) here to find slope
U = 1/(1+dYdT**2)**0.5*np.ones(T.shape) # Normalizes the arrows to see near
-zero slopes.
V = 1/(1+dYdT**2)**0.5*dYdT
plt.figure()
plt.title('Direction Field for dydt=-2y')
Q = plt.quiver(T, Y, U, V) # draws the arrows at (X,Y) with slope dYdX
```



In [15]:
```
matplotlib notebook
```

We know the solution to y' = -2y is y = y0 * e^(-2t) based on the first example. Let's plot a couple of solution curves with the direction field

In [21]:
```
T, Y = np.meshgrid(np.arange(0.1, 3.1, .1), np.arange(-3, 3, .25)) # adjusting to avoid zero denominator
dYdT = -2*Y # put f(t,y) here to find slope
U = 1/(1+dYdT**2)**0.5*np.ones(T.shape) # Normalizes the arrows to see near-zero slopes.
V = 1/(1+dYdT**2)**0.5*dYdT
plt.figure()
plt.title('Direction Field for dydt=-2y')
Q = plt.quiver(T, Y, U, V) # draws the arrows at (X,Y) with slope dYdX
tplot=np.arange(0,3,0.01)  # a range of t-values from 0 to 6 with stepsize .01
# Let y(0)=1 so solution is y=e^(-2t)
yplot=np.exp(-2*tplot)
plt.plot(tplot,yplot)
#NOW let y(0)=-2 so solution is y=-2e^(-2t)
yplot2=-2*np.exp(-2*tplot)
plt.plot(tplot,yplot2)
plt.ylim(-3,3)  # changes the y-range to match the direction field
```



Out[21]: (-3, 3)

In [ ]: