1. Solve $\frac{dy(t)}{dt} = -ky(t)$ with parameter k = 0.5 and the initial condition $y_0 = 10$

In [17]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.integrate import odeint
import numpy as np
import sympy as sy
import sympy, math
```

In [2]:

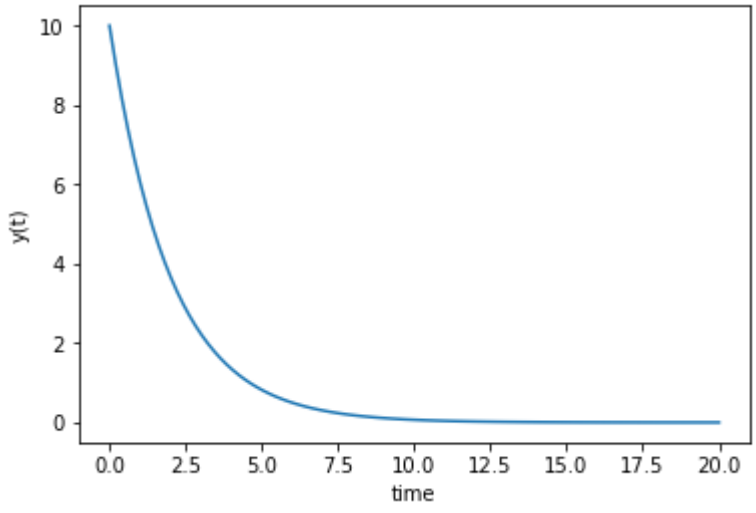
```
#function that returns dy/dt
def model(y,t):
    k=0.5
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



2. Solve the above problem for k = 0.1; 0.4; 0.6; 0.7 and 0.9

In [3]:

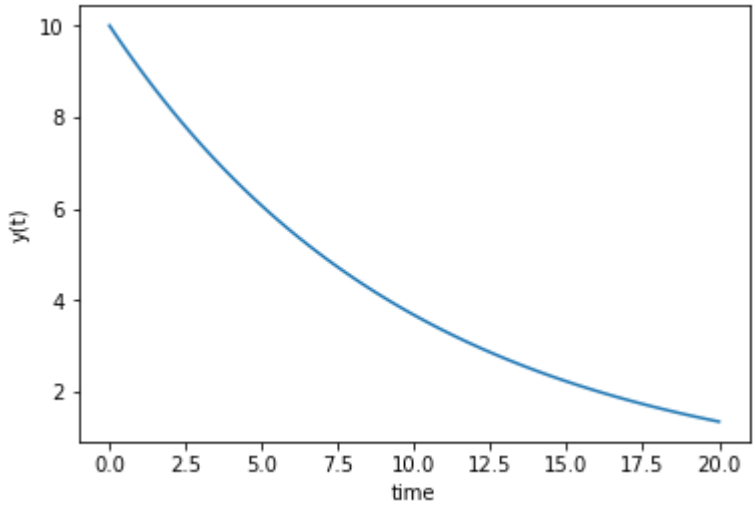
```
#function that returns dy/dt
def model(y,t):
    k=0.1
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



In [4]:

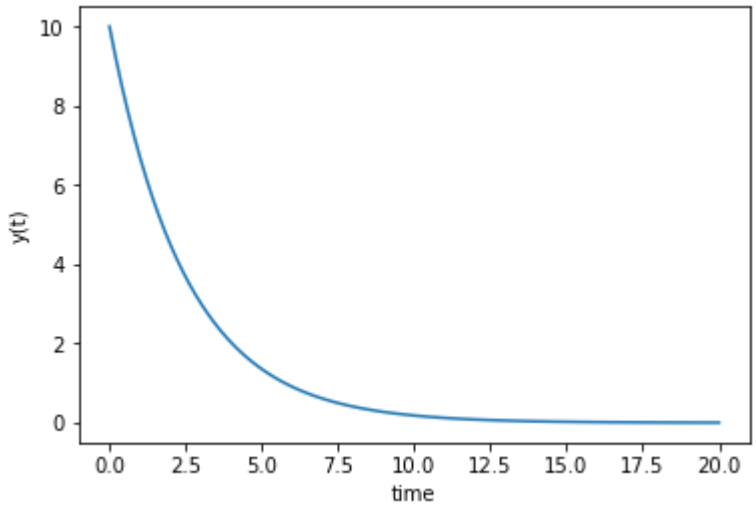
```
#function that returns dy/dt
def model(y,t):
    k=0.4
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



In [5]:

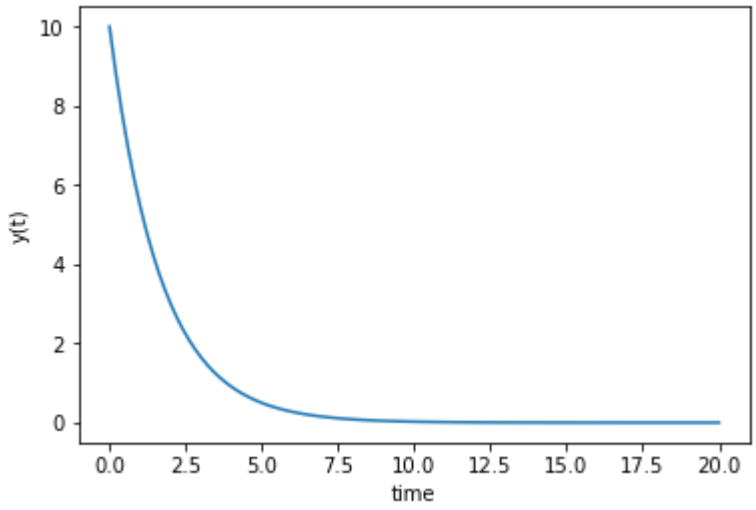
```
#function that returns dy/dt
def model(y,t):
    k=0.6
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



In [6]:

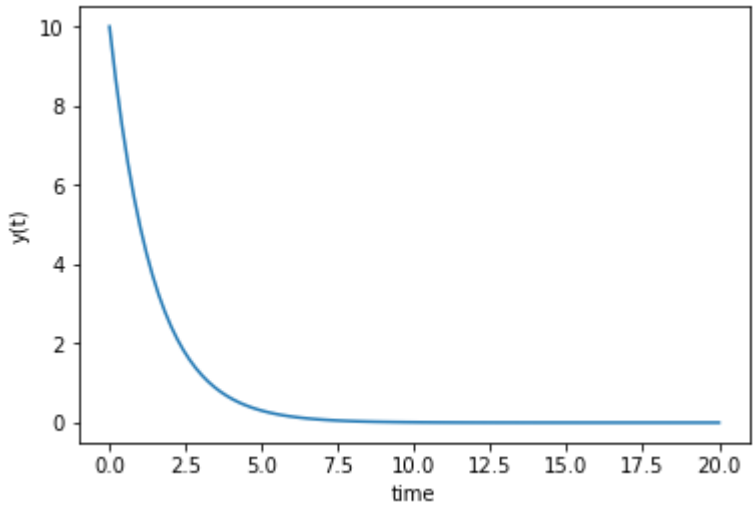
```
#function that returns dy/dt
def model(y,t):
    k=0.7
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



In [7]:

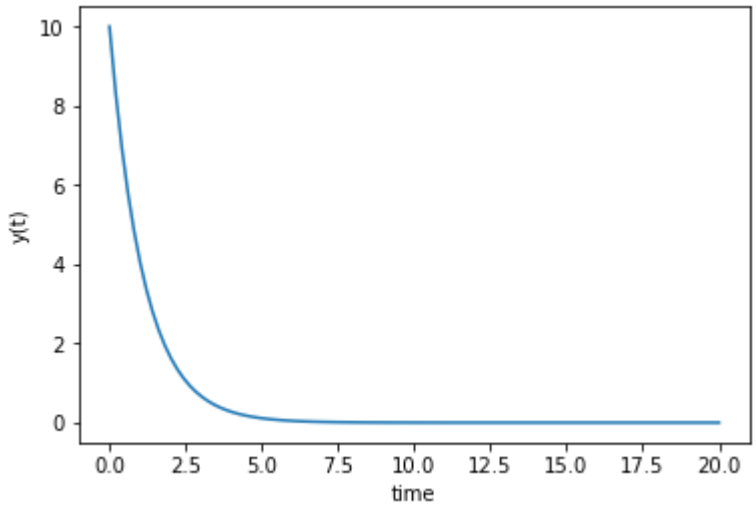
```
#function that returns dy/dt
def model(y,t):
    k=0.9
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



In [8]:

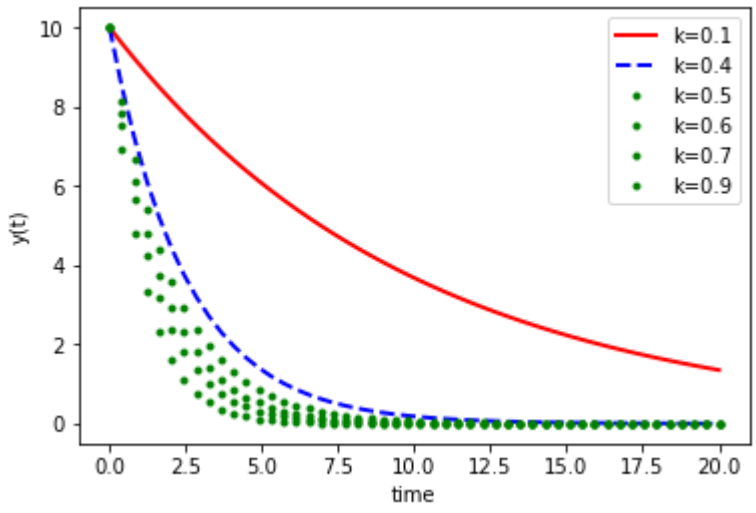
```
#function that returns dy/dt
def model(y,t,k):
    dydt = -k*y
    return dydt

# initial condition
y0=10

# time points
t = np.linspace(0,20)

#solve ODE
k = 0.1
y1 = odeint(model,y0,t,args=(k,))
k = 0.4
y2 = odeint(model,y0,t,args=(k,))
k = 0.5
y3 = odeint(model,y0,t,args=(k,))
k = 0.6
y4 = odeint(model,y0,t,args=(k,))
k = 0.7
y5 = odeint(model,y0,t,args=(k,))
k = 0.9
y6 = odeint(model,y0,t,args=(k,))


#plot results
plt.plot(t,y1,'r-',linewidth=2,label='k=0.1')
plt.plot(t,y2,'b--',linewidth=2,label='k=0.4')
plt.plot(t,y3,'g.',linewidth=2,label='k=0.5')
plt.plot(t,y4,'g.',linewidth=2,label='k=0.6')
plt.plot(t,y5,'g.',linewidth=2,label='k=0.7')
plt.plot(t,y6,'g.',linewidth=2,label='k=0.9')
plt.xlabel('time')
plt.ylabel('y(t)')
plt.legend()
plt.show()
```



3. Solve $\frac{7dy(t)}{dt} = -y(t) + u(t), y(0)$ **= 2**u **steps from 0 to 2 at** t **=**

12

In [16]:

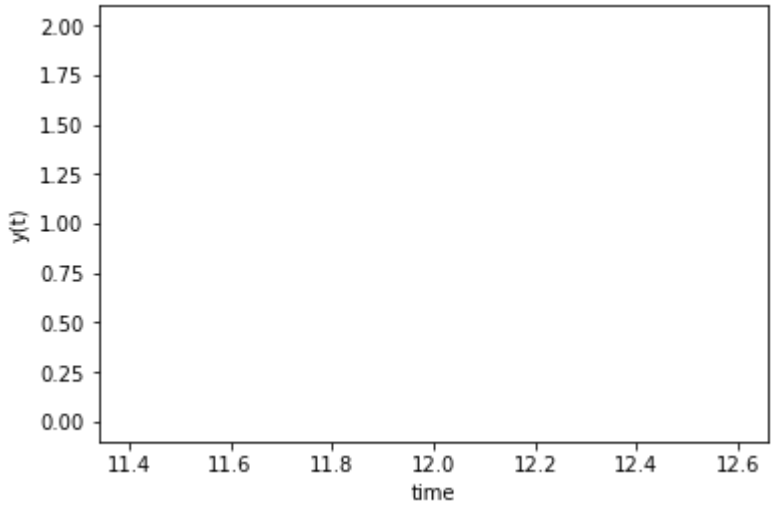
```
#function that returns dy/dt
def model(y,u,t):
    dydt = (-y(t) + u(t))/7
    return dydt

# initial condition
u=np.linspace(0,1,2)
y0=2*u

# time points
t = 12

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```

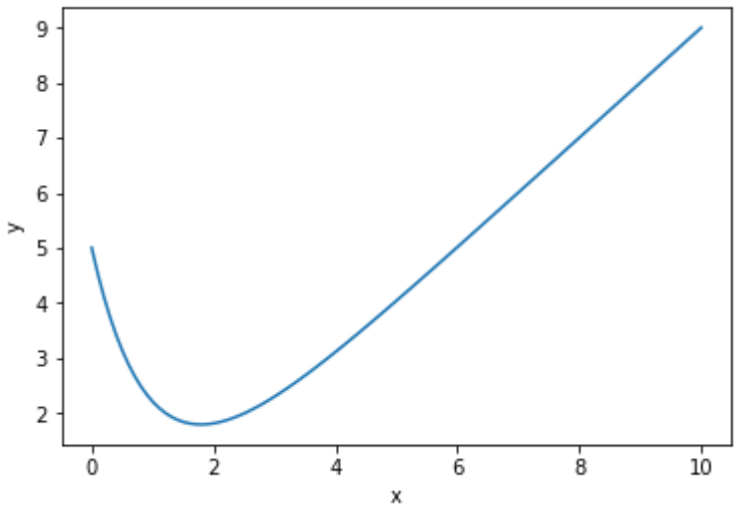



```
[[ 0.  2.]]
```

4. Solve $\frac{dy}{dx} - 2y = 0$ with y(0) = 5

In [20]:

```
#Define a function that calculates the derivative
def dy_dx(y,x):
    return x-y
xs = np.linspace(0,10,100)
y0 = 5.0 #the initial condition
ys = odeint(dy_dx,y0,xs)
#plot results
plt.plot(xs,ys)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
print(ys)
```



In [ ]: