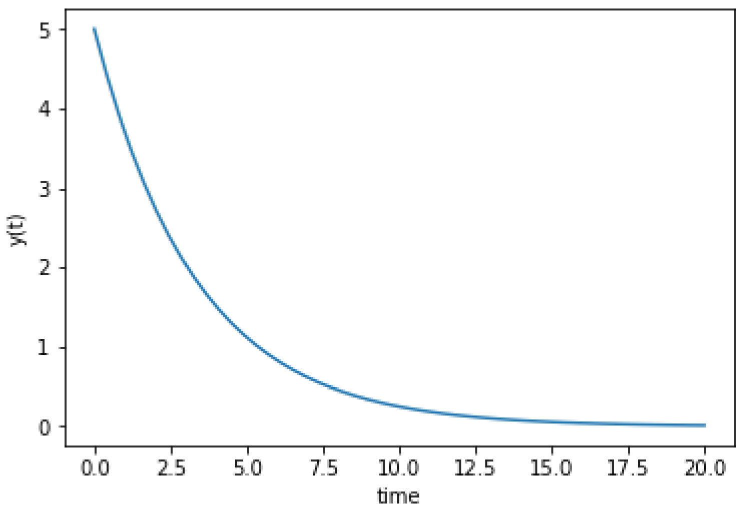
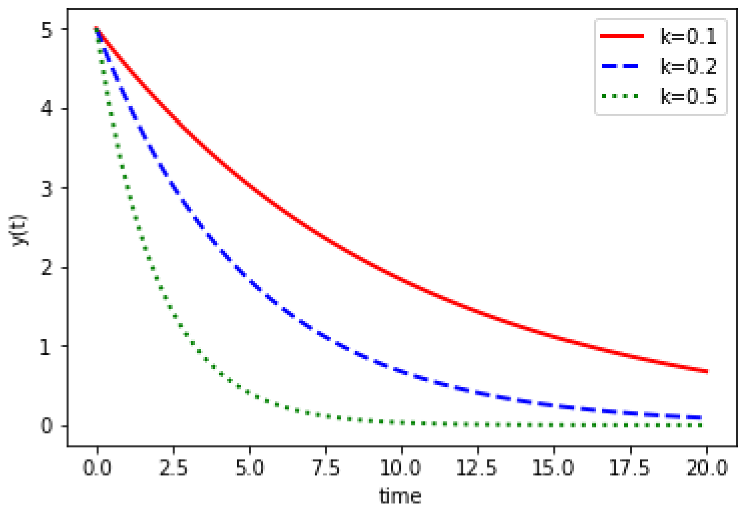
In [2]:

```python
# fundamental package for scientific computing with Python
import numpy as np
# integrate/ solve a system of ordinary differential equations
# dy/dt = func(y, t) where y can be a vector
from scipy.integrate import odeint
# provides a MATLAB-like plotting framework
import matplotlib.pyplot as plt

# we want to solve the equation dy(t)/dt = -ky(t)

# function that returns dy/dt
def f(y,t):
    k = 0.3
    dydt = -k * y
    return dydt

# initial condition
y0 = 5

# time points
t = np.linspace(0,20)
# numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)
# returns num evenly spaced numbers over the interval [start, stop] (de tip ndarray)
# num - number of samples to generate; default is 50
# endpoint - if true, stop is the last sample; otherwise, it is not included
# retstep - if true, return (samples, step), where step is the spacing between samples
# dtype - the type of the output array

# solve ODE
# the model, initial conditions and time points are defined as inputs to ODEINT to numerica
y = odeint(f,y0,t)

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



In [28]:

```
# an optional fourth input is args that allows additional information to be passed into the
# the args input is a tuple sequence of values
# the argument k is now an input to the model function by including an additional argument

import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

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

# initial condition
y0 = 5

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

# solve ODEs
# args - extra arguments to pass to function
k = 0.1
y1 = odeint(f,y0,t,args=(k,))
k = 0.2
y2 = odeint(f,y0,t,args=(k,))
k = 0.5
y3 = odeint(f,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=0.2')
plt.plot(t,y3,'g:',linewidth=2,label='k=0.5')
plt.xlabel('time')
plt.ylabel('y(t)')
plt.legend()
plt.show()
```

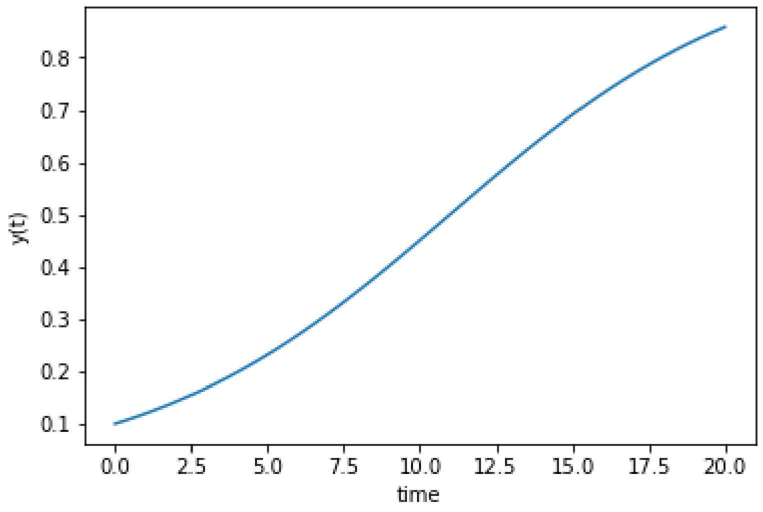


In [ ]:

```
```

In [30]:

```
# fundamental package for scientific computing with Python
import numpy as np
# integrate/ solve a system of ordinary differential equations
# dy/dt = func(y, t) where y can be a vector
from scipy.integrate import odeint
# provides a MATLAB-like plotting framework
import matplotlib.pyplot as plt

# we want to solve the equation dy(t)/dt = 0.2*y(1-y)

# function that returns dy/dt
def f(y,t):
    return 0.2 * y * (1 - y)

# initial condition
y0 = 0.1

# time points
t = np.linspace(0,20)
# numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)
# returns num evenly spaced numbers over the interval [start, stop] (de tip ndarray)
# num - number of samples to generate; default is 50
# endpoint - if true, stop is the last sample; otherwise, it is not included
# retstep - if true, return (samples, step), where step is the spacing between samples
# dtype - the type of the output array

# solve ODE
# the model, initial conditions and time points are defined as inputs to ODEINT to numerica
y = odeint(f,y0,t)

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



In [ ]:

```
```