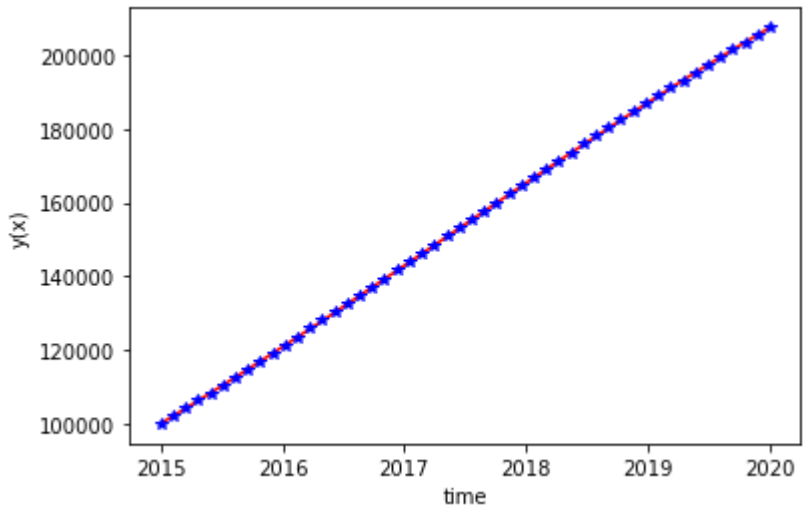
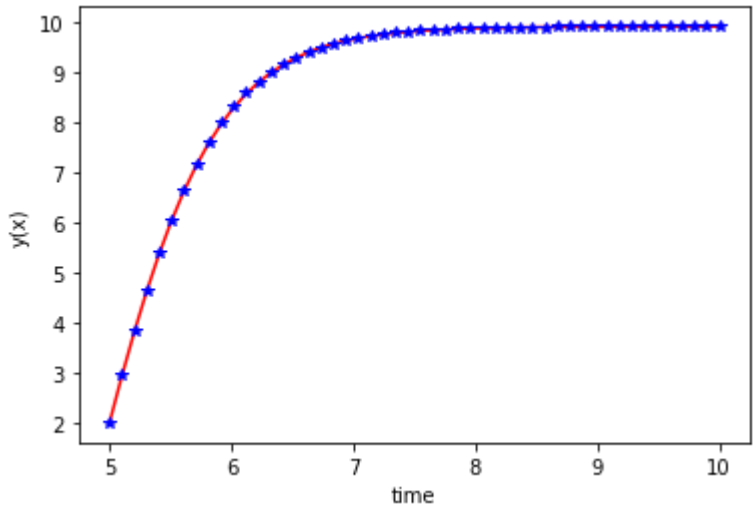
In [5]:

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


def ForwardEuler(f, U0, a, b, n):
    t = np.zeros(n+1)
    y = np.zeros(n+1)
    y[0] = U0
    t[0] = a
    dt = (b-a)/float(n)
    for k in range(n):
        t[k+1] = t[k] + dt
        y[k+1] = y[k] + dt*f(y[k],t[k])
    return y, t

def f(y,t):
    alpha=0.3
    R=300000
    return alpha*y*(1-y/R)

y,t=ForwardEuler(f, 100000, 2015, 2020, 50)
#conditie initiala

#time points
x = np.linspace(2015,2020)

#solve ODE
yo=odeint(f,100000,x)

#plot results
plt.plot(t,y, "r-", x, yo, "b*")
plt.xlabel('time')
plt.ylabel('y(x)')
plt.show()
```



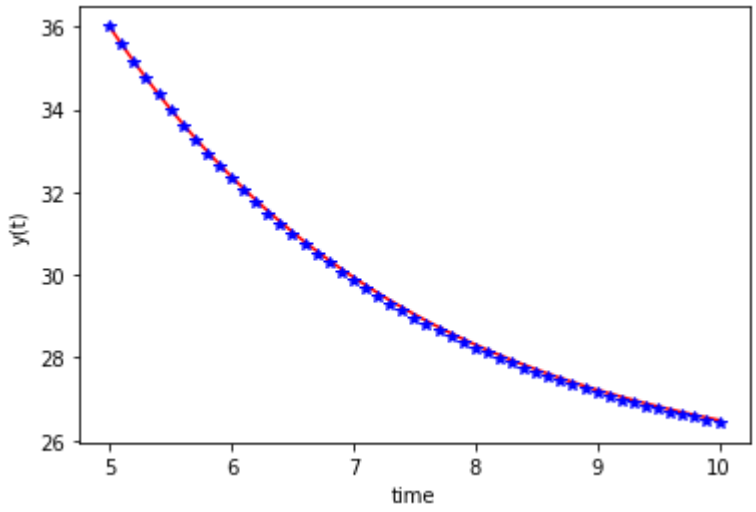
In [6]:

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


def Heun(f, U0, a, b, n):
    t = np.zeros(n+1)
    y = np.zeros(n+1)
    y[0] = U0
    t[0] = a
    dt = (b-a)/float(n)
    for k in range(n):
        t[k+1] = t[k] + dt
        ys=y[k]+dt*f(y[k],t[k])
        y[k+1] = y[k] + 1/2*dt*f(y[k],t[k]) + 1/2*dt*f(ys,t[k+1])
    return y, t

def f(y,t):
    b=0.1
    g=9.8
    return -b*abs(y)*y+g

y,t=Heun(f, 2, 5, 10, 50)
#conditie initiala

#time points
x = np.linspace(5,10)

#solve ODE
yo=odeint(f,2,x)

#plot results
plt.plot(t,y, "r-", x, yo, "b*")
plt.xlabel('time')
plt.ylabel('y(x)')
plt.show()
```



In [7]:

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

def ForwardEuler(f, U0, a, b, n):
    # Solve y'=f(y,t), y(0)=U0, with n steps until t=T.
    t = np.zeros(n+1)
    y = np.zeros(n+1)  # y[k] is the solution at time t[k]
    y[0] = U0
    t[0] = a
    dt = (b-a)/float(n)
    for k in range(n):
        t[k+1] = t[k] + dt
        y[k+1] = y[k] + dt*f(y[k], t[k])
    return y, t

def Heun(f, U0, a, b, n):
    # Solve y'=f(y,t), y(0)=U0, with n steps until t=T.
    t = np.zeros(n+1)
    y = np.zeros(n+1)  # y[k] is the solution at time t[k]
    y[0] = U0
    t[0] = a
    dt = (b-a)/float(n)
    for k in range(n):
        t[k+1] = t[k] + dt
        ys = y[k] + dt * f(y[k],t[k])
        y[k+1] = y[k] + 1/2*dt*f(y[k], t[k]) + 1/2 * dt * f(ys,t[k+1])
    return y, t

def f(y,t):
    h = 0.4
    S = 25
    dydt = -h *(y - S)
    return dydt

y, t = Heun(f, U0=36, a=5, b=10, n=50)

y1, t1 = ForwardEuler(f, U0=36, a=5, b=10, n=50)

plt.plot(t, y, "r-", t1, y1, "b*")
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
```



In [ ]: