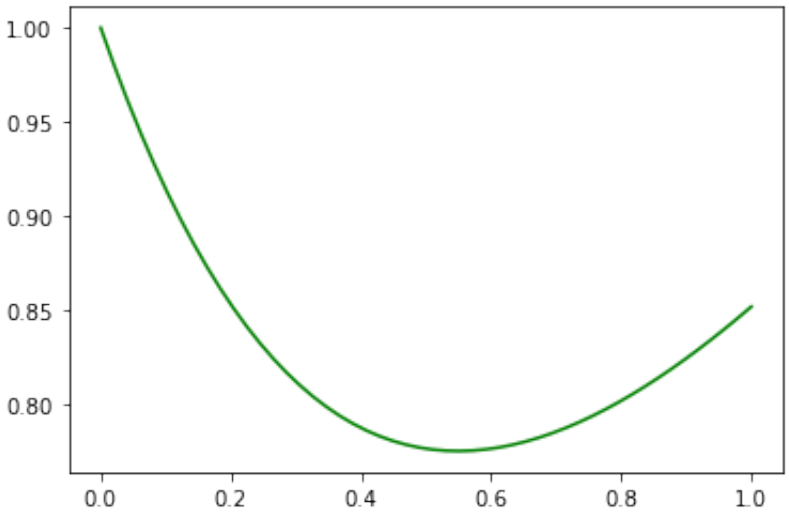
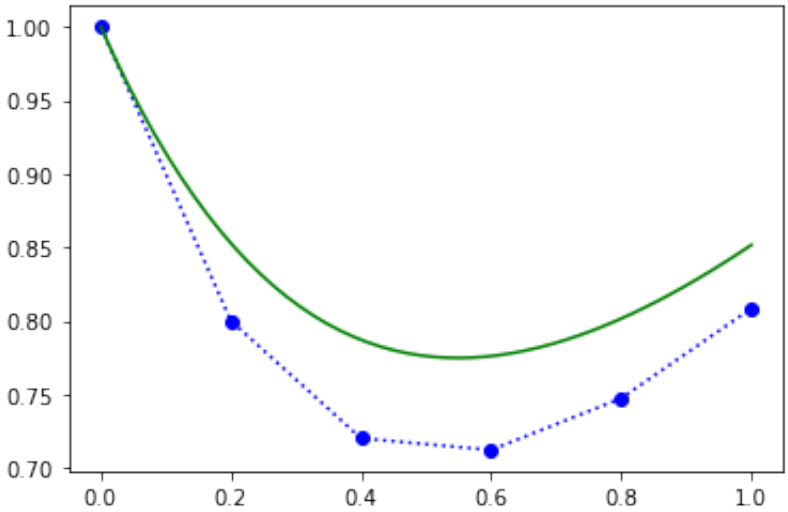
Métodos numéricos de integración de EDOs

In [1]:
```
from pylab import *

# Ecuación diferencial
def f(t,x):
    return 1 + t - 2*x

# Solucion exacta
t = linspace(0, 1);
x = (1 + 2*t + 3*exp(-2*t)) / 4.0
plot(t,x,'g');
```



Método de Euler (progresivo)

In [2]:
```
h=0.2

t0=0
x0=1

x1=x0 + h*f(t0,x0)
t1=t0+h


x2=x1 + h*f(t1,x1)
t2=t1+h

x3=x2 + h*f(t2,x2)
t3=t2+h

x4=x3 + h*f(t3,x3)
t4=t3+h

x5=x4 + h*f(t4,x4)
t5=t4+h

tiempos = [t0,t1,t2,t3,t4,t5]
X=[x0,x1,x2,x3,x4,x5]

plot(tiempos,X,'bo:',t,x,'g');
```



Definamos una función que implemente el método de Euler. Queremos usarla en la forma

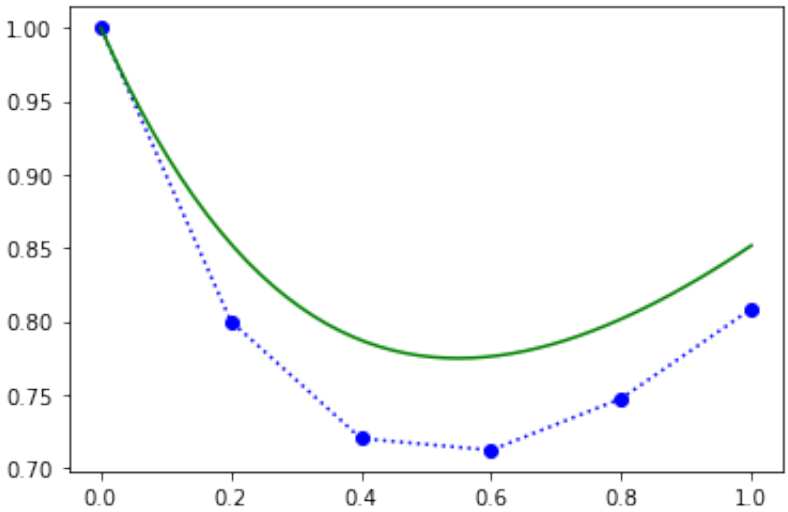
```
Euler_p(f, t_range=[0,1], x0=1, N=5)  ---> tiempos,X
```

In [3]:

```
# Método Euler progresivo
def Euler_p(f, t_range, x0, N):
    (a, b) = t_range
    x = asarray(x0)
    tiempos, h = linspace(a, b, N+1, retstep=True)
    X = [x]
    for t in tiempos[0:-1]:
        x = x + h*f(t,x)
        X.append(x)
    return (tiempos, asarray(X))
```

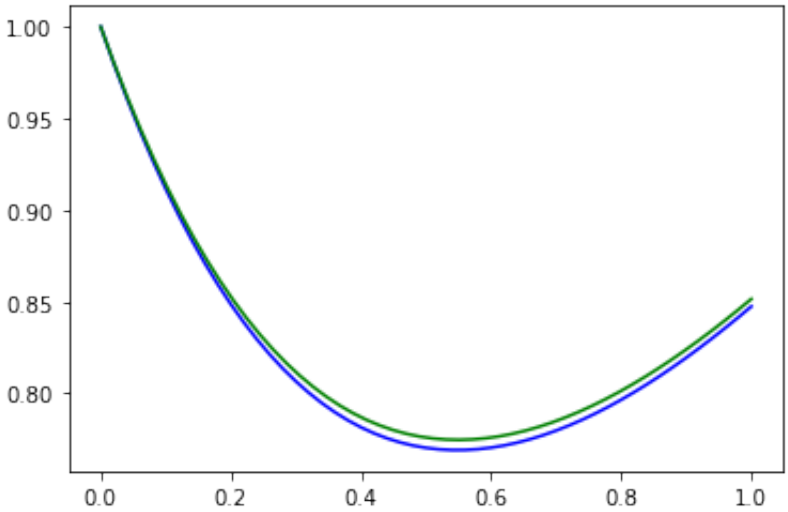
In [4]:

```
r,X=Euler_p(f, t_range=[0,1], x0=1, N=5)
plot(r, X, 'bo:',  t, x, 'g');
```



In [5]:

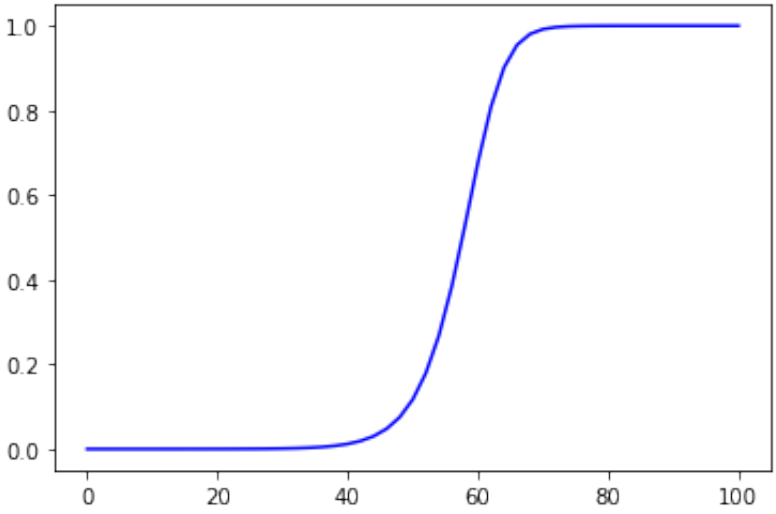
```
r,X=Euler_p(f, t_range=[0,1], x0=1, N=50)
plot(r, X, 'b',  t, x, 'g');
```



Otros ejemplos

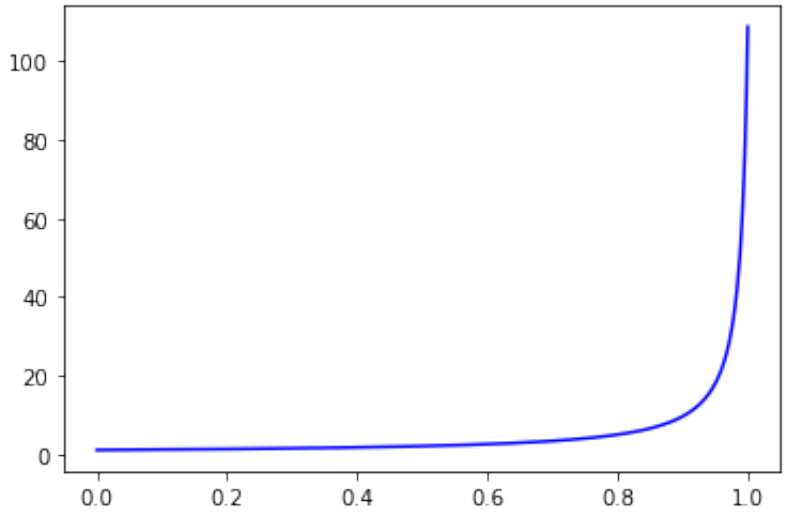
La ecuación logística $\dot{x} = rx(1-x)$.

In [6]:
```
def logistica(t,x):
    r=0.3
    return r*x*(1-x)

r,X=Euler_p(f=logistica, t_range=[0,100], x0=0.000001, N=50)
plot(r, X, 'b');
```



La ecuación $\dot{x} = x^2$

In [7]:
```
def xdos(t,x):
        return x**2
r, X = Euler_p(xdos, t_range=[0,1], x0=1, N=500)
plot(r, X, 'b');
```



Sistemas de ecuaciones

El programa anterior vale tanto para ecucaiones escalares como para sistemas. Veamos como resolver el pvi

$$\begin{aligned} \dot{x} &= -3x + 2y \\ \dot{y} &= -2x - 3y \\ x(0) &= 0 \\ y(0) &= 1 \end{aligned}$$

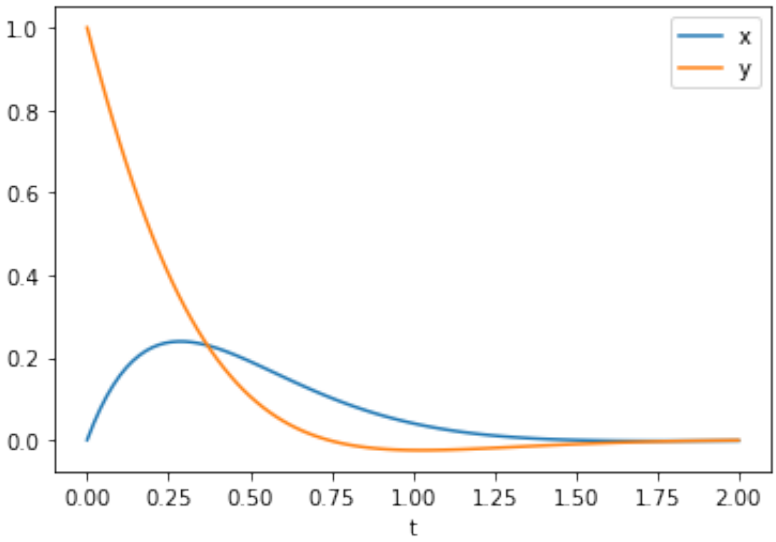
para valores de t en el intervalo $[0, 2]$

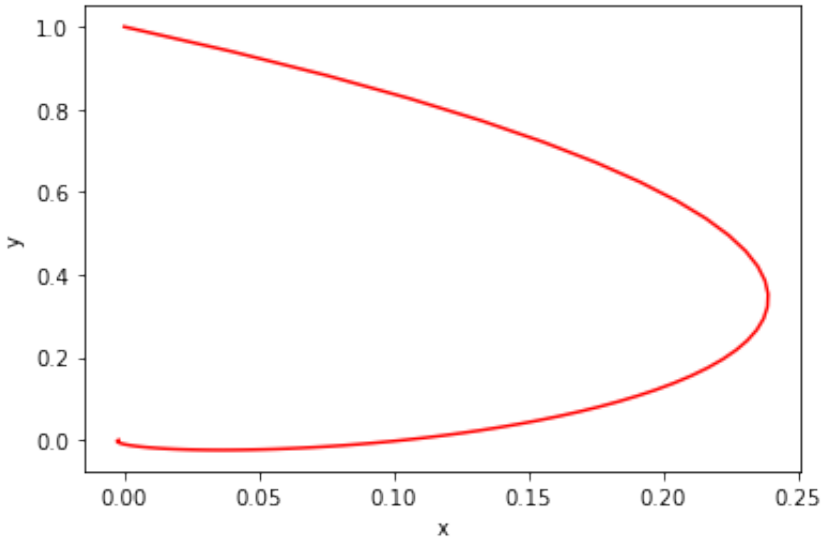
In [8]:
```
def f(t,u):
    x,y=u
    return array([-3*x+2*y,-2*x-3*y])

t, U = Euler_p(f,[0,2],[0,1],8)
# La fila i-ésima de U es [x_i,y_i]
U
```

Out[8]:
```
array([[ 0.        ,  1.        ],
       [ 0.5       ,  0.25      ],
       [ 0.25      , -0.1875    ],
       [-0.03125   , -0.171875  ],
       [-0.09375   , -0.02734375],
       [-0.03710938,  0.04003906],
       [ 0.01074219,  0.02856445],
       [ 0.01696777,  0.00177002],
       [ 0.00512695, -0.00804138]])
```

In [9]:

```
t, U = Euler_p(f,[0,2],[0,1],100)
x,y = U.T

figure()
xlabel('t')
plot(t,x, label='x');
plot(t,y, label='y');
legend()

figure()
xlabel('x')
ylabel('y')
plot(x,y,'r');
```





Usando las librerías de Python

Python tiene ya programados multitud de métodos eficientes para la integración numérica de EDOs. Uno de los más cómodos es `odeint`. Veamos como usarlo.

In [10]:

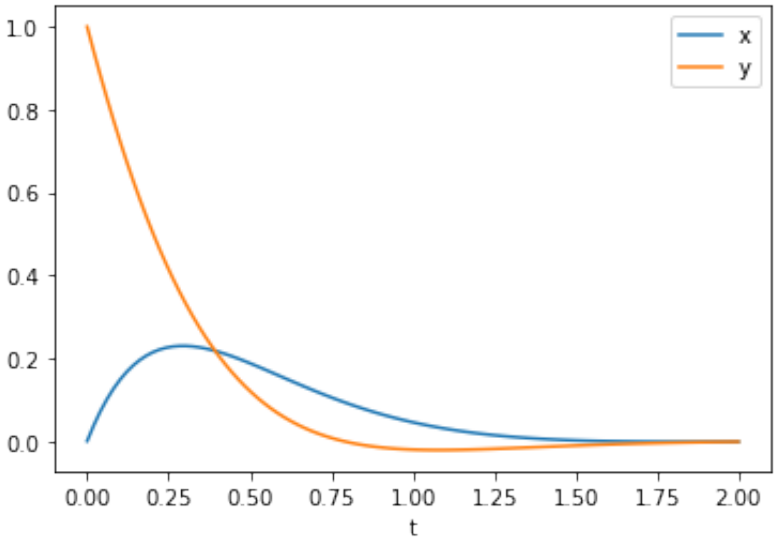
```
from scipy.integrate import odeint

# para odeint el orden de las variables debe ser x,t
def f(u,t):
    x,y=u
    return array([-3*x+2*y,-2*x-3*y])

t=linspace(0,2,8)
U=odeint(f, [0,1], t)

U
```

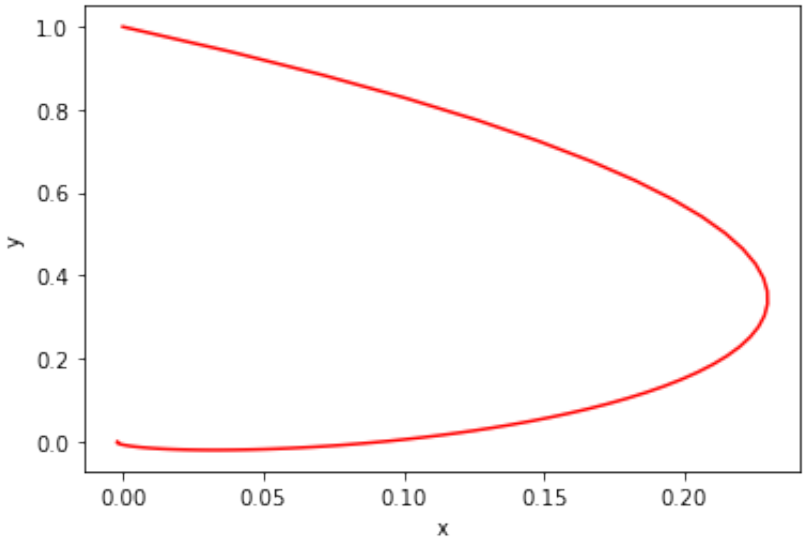
Out[10]:

```
array([[ 0.        ,  1.        ],
       [ 0.22951534,  0.35695241],
       [ 0.16385211,  0.07473774],
       [ 0.07564086, -0.01092876],
       [ 0.02449187, -0.02126179],
       [ 0.00386252, -0.01321071],
       [-0.00165332, -0.0056021 ],
       [-0.00187593, -0.00162022]])
```

In [11]:

```
t=linspace(0,2,100)
U=odeint(f, [0,1], t)
x,y = U.T

figure()
xlabel('t')
plot(t,x, label='x');
plot(t,y, label='y');
legend()

figure()
xlabel('x')
ylabel('y')
plot(x,y,'r');
```



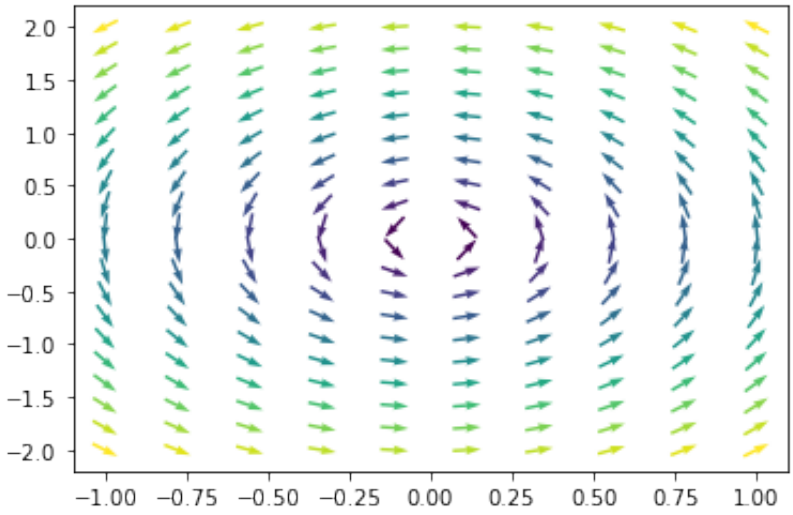


Métodos gráficos

Dibujo del campo vectorial y diagrama de fases

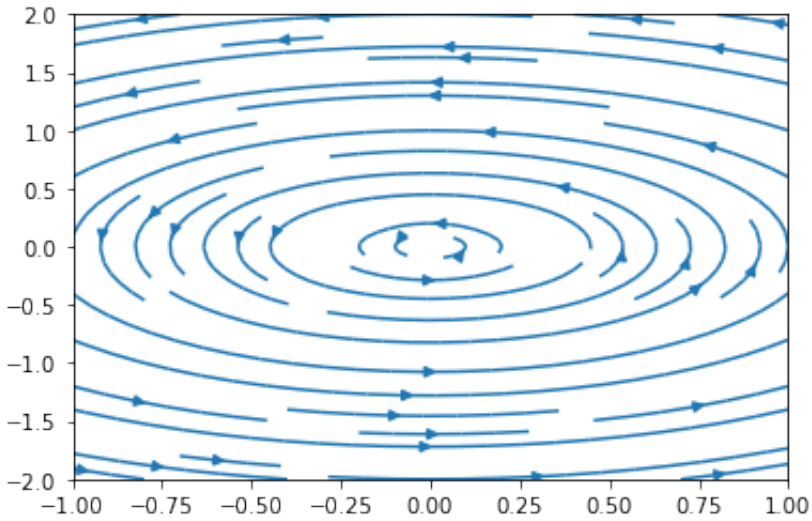
In [12]:

```
from pylab import *
X,Y=meshgrid(linspace(-1,1,10),linspace(-2,2,20))
def N(x,y):
    return sqrt(x**2+y**2)

def F(x,y):
    return array([-y,x])/N(x,y)

U,V=F(X,Y)
color = N(X,Y)
quiver(X,Y,U,V,color,pivot='mid');
```



In [13]:

```
streamplot(X,Y,U,V,density=0.7);
```

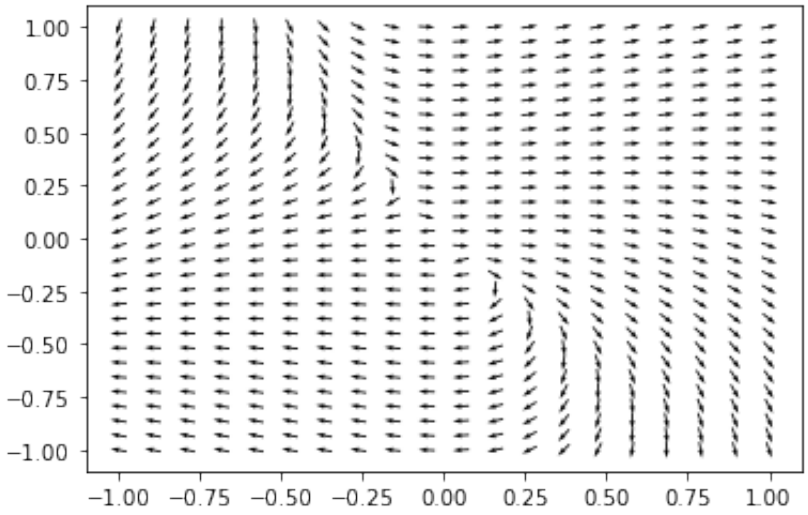


In [14]:

```
def plot_vector_field(f,x_range,y_range, N=[30,30]):
    a,b = x_range
    c,d = y_range
    nx,ny = N
    X,Y=meshgrid(linspace(a,b,nx),linspace(c,d,ny))
    U,V=f([X,Y])
    return quiver(X,Y,U,V,pivot='mid')
```

In [15]:

```
def F(u):
    x,y=u
    dx=3*x+2*y
    dy=(2*y-x)*x
    return array([dx,dy])/sqrt(dx**2+dy**2)

plot_vector_field(F,[-1,1],[-1,1],[20,30]);
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

In [ ]: