PRÁCTICA 01 - Modelación mediante E.D.O.

In [1]:
```
# importar librerias necesarias
import numpy # manejo de vectores y matrices
import matplotlib.pyplot as pl # graficar
import sympy # matematicas simbolica

sympy.init_printing(use_latex='mathjax')
```

1. Considere el modelo de población $\frac{dP}{dt} = 0.4P(1 - \frac{P}{230})$, donde $P_{(t)}$ es la población en el tiempo t.

a. Para qué valores de P está en equilibrio la población?

Punto de equilibrio: no ha cambio en la pobación, es decir:

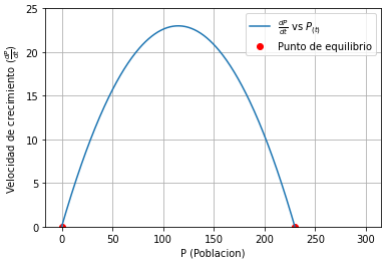
- $\frac{dP}{dt} = 0$
- $0.4P(1 - \frac{P}{230}) = 0$
- $0.4P = 0$ entonces $P = 0$
- $1 - \frac{P}{230} = 0$ entonces $P = 230$

In [2]:
```
# resolución con python
P = sympy.Symbol('P')
sympy.solve(0.4 * P * (1 - P/230),P)
```

Out[2]: $[0.0,\ 230.0]$

b. Graficar

In [3]:
```
# eje de la poblacion, tomemos desde -1 hasta 300, tomando 400 puntos
p = numpy.linspace(-1, 300, 400)
# eje de la velocidad de cre3cimiento, dpdt, la ecuación diferencial
dpdt = 0.4 * p * (1. - p / 230.)

# graficamos p vs dpdt
pl.plot(p, dpdt, label=r"$\frac{dP}{dt}$ vs $P_{(t)}$")

# graficamos los puntos de equilibrio
pl.scatter(0, 0, color='red', label="Punto de equilibrio")
pl.scatter(230, 0, color='red')

# mejoramos visibilidad
pl.grid()  # cuadricula
pl.ylim(0,25) # limitar eje y para mejorar la visibilidad
pl.xlabel('P (Poblacion)') # etiqueta al eje x
pl.ylabel(r'Velocidad de crecimiento ($\frac{dP}{dt}$) ')  # etiqueta al eje y
pl.legend()  # mostramos la leyenda para identificar grafica y puntos
pl.show() # mostramos la grafica
```



c. Para qué valores de "P" está en crecimiento la población?

Para el intervalo $0 < P < 230$

d. Para qué valores de P está decrecimiento la población?

Para $P < 0$ y para $P > 230$

2. Considere el modelo de población $\frac{dP}{dt} = 0.3P(1 - \frac{P}{200})(\frac{P}{50} - 1)$, donde $P_{(t)}$ es la población en el tiempo t.

a. Para qué valores de P está en equilibrio la población?

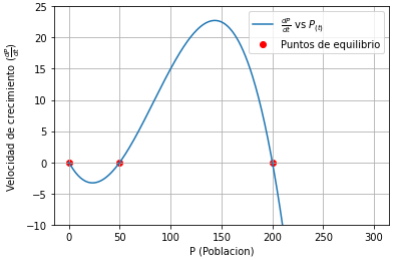
- $\frac{dP}{dt} = 0$
- $0.3P(1 - \frac{P}{200})(\frac{P}{50} - 1) = 0$
- $0.3P = 0$ entonces $P = 0$
- $(1 - \frac{P}{200}) = 0$ entonces $P = 200$
- $(\frac{P}{50} - 1) = 0$ entonces $P = 50$

In [4]:
```
# resolucion con python
P = sympy.Symbol('P')
sympy.solve(0.3 * P * (1 - P/200) * (P / 50 - 1),P)
```

Out[4]: [0.0, 50.0, 200.0]

b. Graficar

In [5]:
```
p = numpy.linspace(0, 300, 400)
dpdt = 0.3 * p * (1. - p / 200.) * (p / 50. - 1)

pl.plot(p, dpdt, label=r"$\frac{dP}{dt}$ vs $P_{(t)}$")

pl.scatter(0, 0, color='red', label="Puntos de equilibrio")
pl.scatter(50, 0, color='red')
pl.scatter(200, 0, color='red')

pl.grid()
pl.ylim(-10,25)
pl.xlabel('P (Poblacion)')
pl.ylabel(r'Velocidad de crecimiento ($\frac{dP}{dt}$) ')
pl.legend()
pl.show()
```



c. Para qué valores de "P" está en crecimiento la población?

En el rango $50 < x < 200$

d. Para qué valores de P está decrecimiento la población?

Para $0 < P < 50$ y para $P > 200$

3. Considere la ecuación diferencia $\frac{dP}{dt} = P^3 - P^2 - 12P$

a. Para qué valores de P está en equilibrio P?

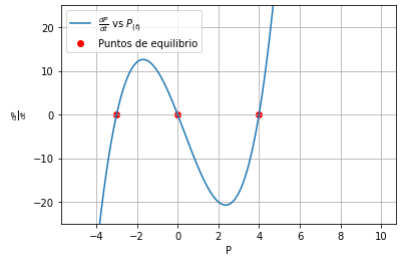
- $\frac{dP}{dt} = 0$
- $P^3 - P^2 - 12P = 0$
- $P(P^2 - P - 12) = 0$
- $P(P - 4)(P + 3) = 0$
- $P = 0$
- $P = 4$
- $P = -3$

In [6]:
```
# resolución con python
P = sympy.Symbol('P')
sympy.solve(P**3 - P**2 - 12*P,P)
```

Out[6]: $[-3, 0, 4]$

b.Graficar

In [7]:
```
p = numpy.linspace(-5, 10, 400)
dpdt = p **3 - p ** 2 - 12*p

pl.plot(p, dpdt, label=r"$\frac{dP}{dt}$ vs $P_{(t)}$")
pl.scatter(-3, 0, color='red', label="Puntos de equilibrio")
pl.scatter(0, 0, color='red')
pl.scatter(4, 0, color='red')

pl.grid()
pl.ylim(-25,25)
pl.xlabel('P')
pl.ylabel(r'$\frac{dP}{dt}$')
pl.legend()
pl.show()
```



c. Para qué valores de "P" está en crecimiento la población?

Para $-3 < P < 0$ y para $P > 4$

d. Para qué valores de P está decrecimiento?

Para $0 < P < 4$ y para $P < -3$

4. El archivo de texto adjunto ejer04.csv proporciona una tabla con información del área de terreno en Australia colonizada por el sapo marino americano (Bufo marinis) cada cinco años desde 1939 hasta 1974. La primera columna corresponde al Año, y la segunda al Área ocupada acumulativa (km^2) . Modele la migración de este sapo bajo el supuesto que la velocidad de crecimiento es proporcional a la población de sapos. Haga predicciones acerca de la superficie de terreno ocupada en los años 2010, 2050 y 2100:

a. Formular la ecuación diferencial

- $\frac{dP}{dt} \propto P$
- $\frac{dP}{dt} = kP$

In [8]:
```
# Formular la Ecuación con python

t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
P = sympy.Function('P') # variable dependiente, poblacion de sapos

# expresamos la ecuacion
dpdt = k*P(t)
modelo = sympy.Eq(P(t).diff(t), dpdt)
modelo
```

Out[8]: $\frac{d}{dt}P(t) = kP(t)$

b. Resolver la Ecuación planteada

- $\frac{dP}{dt} = kP$

- $\frac{1}{P}\frac{dP}{dt} = k$
- $\int \frac{1}{P}\frac{dP}{dt}dt = \int kdt$
- $\int \frac{1}{P}dP = \int kdt$
- $\ln(P) = kt + c$
- $P = e^{kt+c}$
- $P = e^{kt}e^{c}$
- $P = ce^{kt}$

In [9]:
```
#Solucion general utilizando python
solucion = sympy.dsolve(modelo)
solucion
```

Out[9]: $P(t) = C_1 e^{kt}$

c. Determina la constante k

In [10]:
```
# cargar datos
datos = numpy.loadtxt("ejer04.csv", delimiter=',')
datos
```

Out[10]:
```
array([[  1939.,  32800.],
       [  1944.,  55000.],
       [  1949.,  73600.],
       [  1954., 138000.],
       [  1959., 202000.],
       [  1964., 257000.],
       [  1969., 301000.],
       [  1974., 584000.]])
```

en t=0 la Población de Sapos P=32800

- $P = ce^{kt}$
- $32800 = Ce^{0*k}$
- $C = 32800$
- $P = 32800e^{kt}$

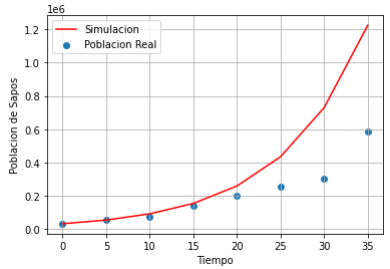
en t=5 la Población de Sapos P=55000

- $55000 = 32800e^{5k}$
- $k = \frac{1}{5}\ln(\frac{55000}{32800})$
- k= 0,1034
- $P = 32800e^{0.1034t}$

d. Graficar población real y población del modelo

In [11]:
```
# creamos una funcion implementando el modelo resuelto
def poblacion(p0, k, t):
    p = p0 * numpy.exp(k * t)
    return p
```

In [12]:
```
tiempo = datos[:,0]  # capturamos todas las filas (:) de la primera columna ([:,0]) de la data
poblacion_real = datos[:, 1]  # capturamos todas las filas (:) de la segunda columna ([:,1]) de la data

tiempo = tiempo - tiempo[0]  # restamos a todo el vector para tener t=0 en t=1939

p0 = datos[0, 1] # capturamos poblacion inicial (32800) es el primer elemento de la segunda columna de la data
k = 0.1034          # coeficiente de crecimiento computada previamente

poblacion_modelo = poblacion(p0, k, tiempo) # poblacion que predice nuestro modelo

pl.scatter(tiempo, poblacion_real, label="Poblacion Real") # poblacion real

pl.plot(tiempo, poblacion_modelo, color='red', label="Simulacion") # nuestra solucion

pl.ylabel("Poblacion de Sapos")
pl.xlabel("Tiempo")
pl.grid()
pl.legend()
pl.show()
```



e.Compara la solución con los datos reales. Crees en tus predicciones?

Hasta 15 años posteriores al inicio es bastante factible, posterir a ello el modelo tiene una mayor velocidad de crecimiento

f. Haga predicciones acerca de la superficie de terreno ocupada en los años 2010, 2050 y 2100.

In [13]:
```
# Para los años solicitados tenemos las siguientes predicciones:
# 2010
anio_inicial = datos[0, 0] # capturamos el año inicial
anio_buscado = 2010 - anio_inicial
p = poblacion(p0, 0.1034, anio_buscado)
print("Predicción de la población de sapos al año 2010: ", round(p))

anio_buscado = 2050 - anio_inicial
p = poblacion(p0, 0.1034, anio_buscado)
print("Predicción de la población de sapos al año 2050: ", round(p))

anio_buscado = 2100 - anio_inicial
p = poblacion(p0, 0.1034, anio_buscado)
print("Predicción de la población de sapos al año 2100: ", round(p))
```

```
Predicción de la población de sapos al año 2010:  50606156
Predicción de la población de sapos al año 2050:  3165521957
Predicción de la población de sapos al año 2100:  556862280675
```

5. Considere un modelo elemental del proceso de aprendizaje: si bien el aprendizaje humano es un proceso extremadamente complicado, es posible construir modelos de ciertos tipos simples de memorización. Por ejemplo, considere una persona a quien se le da una lista para estudiar, y posteriormente se le hacen pruebas periódicas para determinar exactamente qué tanto de la lista ha memorizado. (Por lo general las listas consisten en sílabas sin sentido, números de tres dígitos generados al azar o entradas de tablas de integrales. Si $L_{(t)}$ es la fracción de la lista aprendida en el tiempo t, donde $L = 0$ corresponde a no saber nada del listado y $L = 1$ corresponde a saberlo por completo, podemos entonces formar un simple modelo de este tipo de aprendizaje con base en las hipótesis: La rapidez de aprendizaje es proporcional a la fracción que queda por aprender.

a. Formular la Ecuación diferencial.

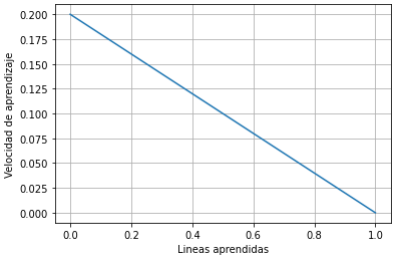
- $\frac{dL}{dt} \propto 1 - L$
- $\frac{dL}{dt} = k(1 - L)$

In [14]:
```
# Formular la Ecuación con python
t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
L = sympy.Function('L') # variable dependiente, lineas aprendidas

# expresamos la ecuacion
dpdt = k*(1 - L(t))
modelo = sympy.Eq(L(t).diff(t), dpdt)
modelo
```

Out[14]: $\frac{d}{dt}L(t) = k\,(1 - L(t))$

b. Graficar la la tasa de cambio con respecto del tiempo. (utilizar $k = 0.2$)

In [15]:
```
# vector de 100 posiciones, desde no tener nada aprendido (0) hasta totalmente aprendido (1)
L = numpy.linspace(0, 1, 100)
k = 0.2
dLdt = k * (1 - L)

pl.plot(L, dLdt)
pl.grid()
pl.xlabel("Lineas aprendidas")
pl.ylabel("Velocidad de aprendizaje")
pl.show()
```



c. Generar el modelo de memorización resolviendo la ecuación planteada.

- $\frac{dL}{dt} = k(1 - L)$
- $\frac{1}{1-L}\frac{dL}{dt} = k$
- $\int \frac{1}{1-L}\frac{dL}{dt}dt = \int k dt$
- $-\ln(1 - L) = kt + c$
- $\ln(1 - L) = -kt - c$
- $1 - L = ce^{-kt}$
- $L = 1 - ce^{-kt}$

In [16]:
```
# resolucion con python
t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
L = sympy.Function('L') # variable dependiente, lineas aprendidas

# expresamos la ecuacion
dpdt = k*(1 - L(t))
modelo = sympy.Eq(L(t).diff(t), dpdt)
sympy.dsolve(modelo)
```

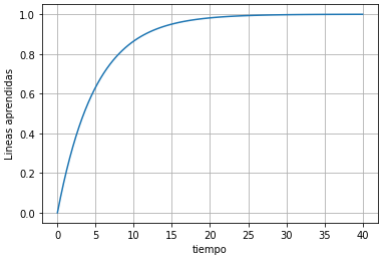
Out[16]: $L(t) = C_1 e^{-kt} + 1$

Para $t = 0$ $L = L_0$

- $L_0 = 1 - c$
- $c = 1 - L_0$
- $L = 1 - (1 - L_0)e^{-kt}$

In [17]:
```
def get_lineas(L0, k, t):
    L = 1 - (1 - L0) * numpy.exp(-k * t)
    return L
```

In [18]:
```
# generamos eje x con tiempo desde 0 hasta 10
t = numpy.linspace(0, 40, 100)
Lineas = get_lineas(0, 0.2, t)
pl.plot(t, Lineas)
pl.grid()
pl.xlabel("tiempo")
pl.ylabel("Lineas aprendidas")
pl.show()
```



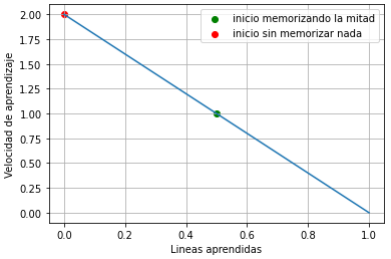
d. Para que valores de L con $0 < L < 1$ ocurre mas rápido el aprendizaje?

Con $L = 0$ se produce la mayor velocidad de aprendizaje, ver grafica del inciso b) $\frac{dL}{dt}$ vs L

6. Suponga que dos estudiantes memorizan listas de acuerdo con el mismo modelo: $\frac{dL}{dt} = 2(1 - L)$

a. Si uno de los estudiantes aprende la mitad de la lista en el tiempo $t = 0$ y el otro no memoriza nada de ella, qué estudiante está aprendiendo más rápidamente en este instante? Graficar para sustentar su análisis.

In [19]:
```
# vector de 100 posiciones, desde no tener nada aprendido (0) hasta totalmente aprendido (1)
L = numpy.linspace(0, 1, 100)
k = 2
dLdt = k * (1 - L)

pl.plot(L, dLdt)
pl.scatter(0.5, 2*(1-0.5), color='green', label="inicio memorizando la mitad")
pl.scatter(0, 2*(1-0), color='red', label="inicio sin memorizar nada")
pl.grid()
pl.xlabel("Lineas aprendidas")
pl.ylabel("Velocidad de aprendizaje")
pl.legend()
pl.show()
```

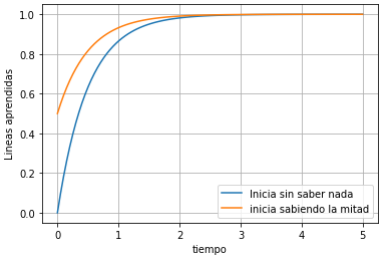


En el instante $t = 0$ el estudiante que no sabía ninguna linea (velocidad = $2[\frac{lineas}{min}]$) aprende mas rápido que el estudiante inicialmente se sabía la mitad (velocidad = $1[\frac{lineas}{min}]$) de las lineas

b. Alcanzará el estudiante que comienza sin saber nada de la lista al estudiante que empieza sabiendo la mitad de la lista? (Graficar para sustentar respuesta)

In [20]:
```
def get_lineas(L0, k, t):
    L = 1 - (1 - L0) * numpy.exp(-k * t)
    return L
```

In [21]:
```
# generamos eje x con tiempo desde 0 hasta 10
t = numpy.linspace(0, 5, 100)
Lineas_1 = get_lineas(0, 2, t)
Lineas_2 = get_lineas(0.5, 2, t)
pl.plot(t, Lineas_1, label = "Inicia sin saber nada")
pl.plot(t, Lineas_2, label="inicia sabiendo la mitad")
pl.grid()
pl.xlabel("tiempo")
pl.ylabel("Lineas aprendidas")
pl.legend()
pl.show()
```



7. Considere las dos siguientes ecuaciones diferenciales que modelan las tasas de memorización de un poema por dos estudiantes. La tasa de Juan es proporcional a la cantidad por aprender, con una constante de proporcionalidad de $k = 2$. La tasa de Ana es proporcional al cuadrado de la cantidad por aprender y cuya constante de proporcionalidad es de $k = 3$.

a. Formular las ecuaciones

Apredizaje de Juan:

- $\frac{dL}{dt} \propto (1 - L)$
- $\frac{dL}{dt} = k(1 - L)$
- $\frac{dL}{dt} = 2(1 - L)$

Aprendizaje de Ana:

- $\frac{dL}{dt} \propto (1 - L)^2$
- $\frac{dL}{dt} = k(1 - L)^2$
- $\frac{dL}{dt} = 3(1 - L)^2$

b. Resolver las ecuaciones (puede utilizar sympy)

Resolución para Juan:

- $\frac{dL}{dt} = k(1 - L)$
- $\frac{1}{1-L}\frac{dL}{dt} = k$
- $\int \frac{1}{1-L}\frac{dL}{dt}dt = \int k dt$
- $-\ln(1 - L) = kt + c$
- $\ln(1 - L) = -kt - c$
- $1 - L = ce^{-kt}$
- $L = 1 - ce^{-kt}$

Resolución para Ana:

- $\frac{dL}{dt} = k(1 - L)^2$
- $\frac{1}{(1-L)^2}\frac{dL}{dt} = k$
- $\int \frac{1}{(1-L)^2}\frac{dL}{dt}dt = \int k dt$
- $\int \frac{1}{(1-L)^2}dL = \int k dt$

Resolvemos la integral con un cambio de variable: Sea $u = 1 - L$ entonces $du = -dL$

- $\int \frac{1}{(1-L)^2}dL$
- $-\int \frac{1}{u^2}du = \frac{1}{u}$ sustituyendo $\frac{1}{1-L}$

Volvemos a la ecuación diferencial:

- $\int \frac{1}{(1-L)^2} dL = \int k dt$
- $\frac{1}{1-L} = kt + c$
- $1 - L = \frac{1}{kt+c}$
- $L = 1 - \frac{1}{kt+c}$
- $L = \frac{kt+c}{kt+c} - \frac{1}{kt+c}$
- $L = \frac{c+kt-1}{c+kt}$

In [22]:
```
# resolucion utilizando python


t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
L = sympy.Function('L') # variable dependiente, lineas aprendidas

# # para Juan
dpdt = k*(1 - L(t))
modelo = sympy.Eq(L(t).diff(t), dpdt)
sympy.dsolve(modelo)
```

Out[22]: $L(t) = C_1 e^{-kt} + 1$

In [23]:
```
# resolucion utilizando python


t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
L = sympy.Function('L') # variable dependiente, lineas aprendidas

# # para Juan
dpdt = k*(1 - L(t))**2
modelo = sympy.Eq(L(t).diff(t), dpdt)
sympy.dsolve(modelo)
```

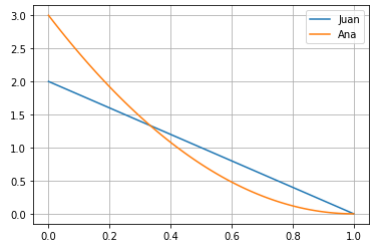
Out[23]: $L(t) = \frac{C_1 + kt - 1}{C_1 + kt}$

c. Graficar ambos modelos de velocidad

In [24]:
```
def dLdt_juan(k, L):
    dldt = k * (1 - L)
    return dldt
```

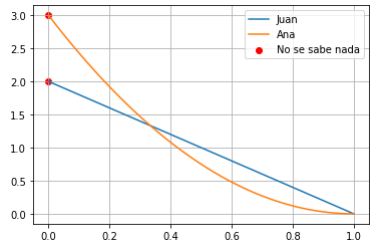
In [25]:
```
def dLdt_ana(k, L):
    dldt = k * (1 - L) ** 2
    return dldt
```

In [26]:
```
lineas = numpy.linspace(0, 1, 100)
juan = dLdt_juan(2, lineas)
ana = dLdt_ana(3, lineas)

pl.plot(lineas, juan, label='Juan')
pl.plot(lineas, ana, label='Ana')

pl.grid()
pl.legend()
pl.show()
```



d. Qué estudiante tiene una tasa más rápida de aprendizaje en $t = 0$, si ambos empiezan la memorización juntos y nunca antes han visto el poema? Encuentre la solución particular de ambos y graficar.

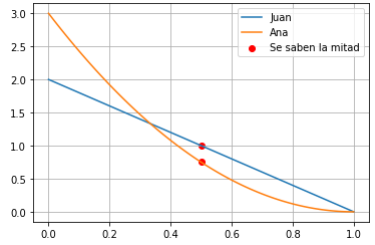
In [27]:
```
lineas = numpy.linspace(0, 1, 100)
juan = dLdt_juan(2, lineas)
ana = dLdt_ana(3, lineas)

pl.plot(lineas, juan, label='Juan')
pl.plot(lineas, ana, label='Ana')
pl.scatter(0, dLdt_juan(2, 0), color='red', label="No se sabe nada" )
pl.scatter(0, dLdt_ana(3, 0), color='red' )
pl.grid()
pl.legend()
pl.show()
```



Ambos tienen la misma tasa de aprendizaje.

e. Qué estudiante tiene una tasa más rápida de aprendizaje en $t = 0$, si ambos empiezan a memorizar juntos habiendo aprendido la mitad del poema? Encuentre la solución particular de ambos y graficar.

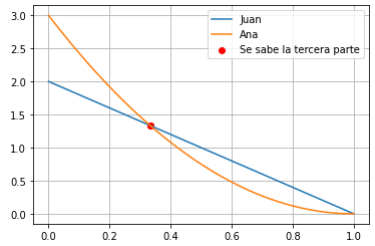
In [28]:
```
lineas = numpy.linspace(0, 1, 100)
juan = dLdt_juan(2, lineas)
ana = dLdt_ana(3, lineas)

pl.plot(lineas, juan, label='Juan')
pl.plot(lineas, ana, label='Ana')
pl.scatter(0.5, dLdt_juan(2, 0.5), color='red', label="Se saben la mitad" )
pl.scatter(0.5, dLdt_ana(3, 0.5), color='red' )
pl.grid()
pl.legend()
pl.show()
```



Juan tiene mayor velocidad cuando ambos saben la mitad de las lineas.

f. Qué estudiante tiene una tasa más rápida de aprendizaje en $t = 0$, si ambos empiezan la memorización juntos y habiendo aprendido un tercio del poema? Encuentre la solución particular de ambos y graficar.

In [29]:
```
lineas = numpy.linspace(0, 1, 100)
juan = dLdt_juan(2, lineas)
ana = dLdt_ana(3, lineas)

pl.plot(lineas, juan, label='Juan')
pl.plot(lineas, ana, label='Ana')
pl.scatter(1/3, dLdt_juan(2, 1/3), color='red', label="Se sabe la tercera parte" )
pl.scatter(1/3, dLdt_ana(3, 1/3), color='red' )
pl.grid()
pl.legend()
pl.show()
```



Juan y Ana tienen la misma velocidad cuando ambos saben la tercera parte de las lineas.

8.En los siguientes ejercicios, consideramos el fenómeno de la desintegración radiactiva que, por experimentación, sabemos que se comporta de acuerdo con la ley siguiente: La tasa a la que una cantidad de un isótopo radiactivo se desintegra es proporcional a la cantidad del isótopo presente. La constante de proporcionalidad depende sólo de la partícula radiactiva considerada.

a) Modele la desintegración radiactiva en base a E.D.O. usando la notación: t tiempo, $r_{(t)}$ cantidad de isótopo radiactivo particular en el tiempo t, λ tasa de desintegración

- $\frac{dr}{dt} \propto r$
- $\frac{dr}{dt} = -\lambda M$

b) Usando esta notación, escriba un modelo para la desintegración de un isótopo radiactivo particular

- $\frac{dr}{dt} = -\lambda r$
- $\frac{1}{r}\frac{dr}{dt} = -\lambda$
- $\int \frac{1}{r}\frac{dr}{dt}dt = \int -\lambda dt$
- $\int \frac{1}{r}dr = \int -\lambda dt$
- $\ln(r) = -\lambda t + c$
- $r = e^{-\lambda t + c}$
- $r = e^{-\lambda t}e^{c}$
- $r = ce^{-\lambda t}$

c) Si la cantidad del isótopo presente en t = 0 es r 0 , establezca el problema de valor inicial correspondiente para el modelo resuelto.

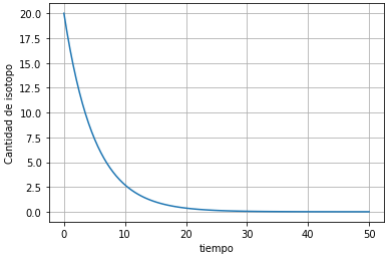
- $r = ce^{-\lambda t}$
- $r_0 = ce^{-\lambda(0)}$
- $r_0 = c$
- $r = r_0 e^{-\lambda t}$

d) Implementar una función en Python para implementar el modelo generado.

In [30]:
```
def get_desintegracion(r0, t, lam):
    r = r0 * numpy.exp(-lam * t)
    return r
```

e) Graficar el modelo cantidad de isotopo vs tiempo.

In [31]:
```
# Sea una cantidad de isotopo de 20kg y una constante de desintegración de 0.2
t = numpy.linspace(0, 50, 100)
r = get_desintegracion(20, t, 0.2)

pl.plot(t, r)
pl.grid()
pl.xlabel("tiempo")
pl.ylabel("Cantidad de isotopo")
pl.show()
```



9. La vida media de un isótopo radiactivo es la cantidad de tiempo que toma a una cantidad de material radiactivo desintegrarse a la mitad de su cantidad original.

a) La vida media del carbono 14 (C-14) es de 5230 años. Determine el parámetro λ de tasa de desintegración del C-14.

- $r = r_0 e^{-\lambda t}$
- $\frac{1}{2} r_0 = r_0 e^{-\lambda (5230)}$
- $\frac{1}{2} = e^{-\lambda (5230)}$
- $\lambda = -\frac{1}{5230} \ln(\frac{1}{2})$
- $\lambda = 0,000132533$

b) La vida media del yodo 131 (1-131) es de 8 días. Calcule el parámetro de tasa de desinte- gración del 1-131.

- $r = r_0 e^{-\lambda t}$
- $\frac{1}{2} r_0 = r_0 e^{-\lambda (5230)}$
- $\frac{1}{2} = e^{-\lambda (8)}$
- $\lambda = -\frac{1}{8} \ln(\frac{1}{2})$
- $\lambda = 0,086643398$

(c) Cuáles son las unidades de los parámetros de tasa de desintegración en las partes (a) y (b)?

En el caso del Carbono en el inciso a) las unidades son $\frac{1}{años}$

En el caso del lodo en el inciso b) las unidades son $\frac{1}{dias}$

d) Para estimar la vida media de un isótopo, podríamos comenzar con 1000 átomos del isótopo y medir la cantidad de tiempo que le toma a 500 de ellos desintegrarse o podríamos comenzar con 10 000 átomos del isótopo y medir la cantidad de tiempo que le toma desintegrarse a 5 000 de ellos. Obtendremos la misma respuesta? Explícalo.

Si, se obtendrá la misma cantidad, dado que con la relación de la cantidad inicial, y la desintegración del mismo a la mitad, básicamente se simplifican las cantidades, por lo que son iguales

10. El fechado por carbono es un método para determinar el tiempo transcurrido desde la muerte del material orgánico. Las hipótesis implícitas en el fechado por carbono son que:

- El carbono 14 (C-14) constituye una proporción constante del carbono que la materia viva ingiere según una base regular.
- Una vez que la materia muere, el C-14 presente se desintegra, pero ningún átomo nuevo es agregado a la materia.

Entonces, al medir la cantidad de C-14 que aún permanece en la materia orgánica y al compararla con la cantidad de C-14 encontrada en la materia viva, puede calcularse el "tiempo desde la muerte". Usando el parámetro de la tasa de desintegración que usted estimó en el anterior ejercicio, determine el tiempo desde la muerte si:

a) 88% del C-14 original aún está presente en el material.

- $r = r_0 e^{-0,000132533t}$
- $\frac{88}{100} r_0 = r_0 e^{-0,000132533t}$
- $\frac{88}{100} = e^{-0,000132533t}$
- $t = \frac{1}{-0,000132533} \ln(\frac{88}{100})$
- $t = 964,5$ años

b) 12% del C-14 original aún está presente en el material.

- $r = r_0 e^{-0,000132533t}$
- $\frac{12}{100} r_0 = r_0 e^{-0,000132533t}$
- $\frac{12}{100} = e^{-0,000132533t}$
- $t = \frac{1}{-0,000132533} \ln(\frac{12}{100})$
- $t = 15998$ años

c) 2% del C-14 original aún está presente en el material.

- $r = r_0 e^{-0,000132533t}$
- $\frac{2}{100} r_0 = r_0 e^{-0,000132533t}$
- $\frac{2}{100} = e^{-0,000132533t}$
- $t = \frac{1}{-0,000132533} \ln(\frac{2}{100})$
- $t = 29517$ años

d) 98% del C-14 original aún está presente en el material.

- $r = r_0 e^{-0,000132533t}$
- $\frac{98}{100} r_0 = r_0 e^{-0,000132533t}$
- $\frac{98}{100} = e^{-0,000132533t}$
- $t = \frac{1}{-0,000132533} \ln(\frac{98}{100})$
- $t = 152,43$ años

10. El isótopo radiactivo I-131 se usa en el tratamiento de la hiper tiroides. El I-131 administrado a un paciente se acumula en forma natural en la glándula tiroides, donde se desintegra y acaba con parte de la glándula.

(a) Suponga que se requieren 72 horas para enviar el I-131 del productor al hospital. Qué porcentaje de la cantidad originalmente enviada llega al hospital?

- $r = r_0 e^{-0,086643398t}$
- $pr_0 = r_0 e^{-0,086643398(72/24)}$
- $p = e^{(-0,086643398)(3)}$
- $p = 77\%$

(b) Si el I-131 es almacenado en el hospital 48 horas adicionales antes de ser usado, ¿qué tanto queda de la cantidad original enviada por el productor cuando el material radiactivo se utilice?

- $r = r_0 e^{-0,086643398t}$
- $pr_0 = r_0 e^{-0,086643398(48/24+3)}$
- $p = e^{(-0,086643398)(5)}$
- $p = 64.8\%$

(c) Qué tiempo le tomará al I-131 desintegrarse completamente de manera que el hospital pueda deshacerse de los residuos sin precauciones especiales?

Nunca se desintreará por completo ya que $e^{-\lambda t}$ nunca es cero.

11.Suponga que una especie de pez en un lago específico tiene una población que sigue el modelo logístico de población con razón k de crecimiento, capacidad N de soporte y tiempo t medido en años. Ajuste el modelo para tomar en cuenta cada una de las situaciones siguientes.

(a) 100 peces son cultivados cada año.

$\frac{dP}{dt} = k(1 - \frac{P}{N})P - 100$

(b) Un tercio de la población de peces es cultivada anualmente.

$$\frac{dP}{dt} = k(1 - \frac{P}{N})P - \frac{1}{3}P$$

(c) El número de peces cultivados cada año es proporcional a la raíz cuadrada del número de peces en el lago.

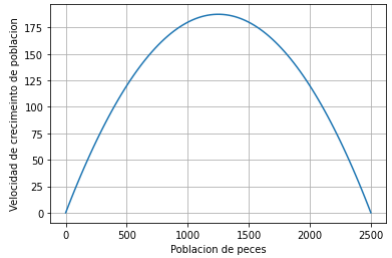
$$\frac{dP}{dt} = k(1 - \frac{P}{N})P - r(\sqrt{P})$$

12.Suponga el parámetro k = 0.3 de razón de crecimiento y la capacidad N = 2500 de soporte en el modelo logístico de población del ejercicio anterior. Y también que P (0) = 2500.

a) Graficar la tasa de crecimiento.

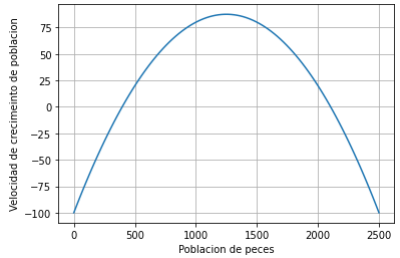
In [32]:
```
def get_dpdt(P, k, N):
    dpdt = k * (1 - P / N) * P
    return dpdt
```

In [33]:
```
P = numpy.linspace(0, 2500, 100)
dpdt = get_dpdt(P, 0.3, 2500)
pl.plot(P, dpdt)
pl.grid()
pl.xlabel("Poblacion de peces")
pl.ylabel("Velocidad de crecimeinto de poblacion")
pl.show()
```



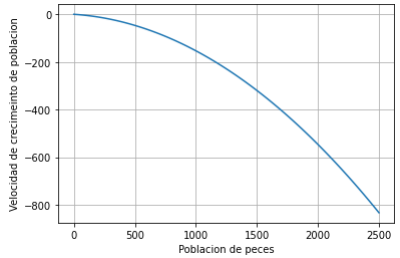
In [34]:
```
def get_dpdt(P, k, N):
    dpdt = k * (1 - P / N) * P - 100
    return dpdt
```

In [35]:
```
P = numpy.linspace(0, 2500, 100)
dpdt = get_dpdt(P, 0.3, 2500)
pl.plot(P, dpdt)
pl.grid()
pl.xlabel("Poblacion de peces")
pl.ylabel("Velocidad de crecimeinto de poblacion")
pl.show()
```



In [36]:
```
def get_dpdt(P, k, N):
    dpdt = k * (1 - P / N) * P - P/3
    return dpdt
```

In [37]:
```
P = numpy.linspace(0, 2500, 100)
dpdt = get_dpdt(P, 0.3, 2500)
pl.plot(P, dpdt)
pl.grid()
pl.xlabel("Poblacion de peces")
pl.ylabel("Velocidad de crecimeinto de poblacion")
pl.show()
```



13. El rinoceronte es actualmente muy raro. Suponga que se aparta suficiente terreno para su preservación y que hay entonces suficiente espacio para muchos más territorios de rinocerontes que rinocerontes. En consecuencia, no habrá peligro de una sobre población. Sin embargo, si la población es muy pequeña, los adultos fértiles tendrán dificultad en encontrarse cuando sea el tiempo de apareamiento.

(a) Escriba una ecuación diferencial que modele la población de rinocerontes con base en esas hipótesis. (Note que hay más de un modelo razonable que se ajusta a esas suposiciones.)

Son posibles varios modelos diferentes. Sea R la población de rinocerontes en el tiempo t. El supuesto básico es que existe un umbral mínimo que la población debe superar para sobrevivir. En términos de la ecuación diferencial, esta suposición significa que $\frac{dR}{dt}$ debe ser negativo si R está cerca de cero. Por ejemplo: Si k es un parámetro de tasa de crecimiento y b es un parámetro que determina el nivel en el que la población comenzará a disminuir ($R < \frac{b}{k}$), entonces: $\frac{dR}{dt} = kR - b$

In [38]:
```
# solucion con python
# resolucion utilizando python


t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
b = sympy.Symbol('b') # parametro
R = sympy.Function('R') # variable dependiente, lineas aprendidas

drdt = k*R(t)-b
modelo = sympy.Eq(R(t).diff(t), drdt)
modelo
```

Out[38]:
$$\frac{d}{dt}R(t) = -b + kR(t)$$

b) Encuentre la solución general.

- $\frac{dR}{dt} = kR - b$
- $\frac{1}{kR-b}\frac{dR}{dt} = 1$
- $\int \frac{1}{kR-b}\frac{dR}{dt}dt = \int dt$
- $\int \frac{1}{kR-b}dR = \int dt$
- $\frac{1}{k}\ln(kR - b) = t + c$
- $\ln(kR - b) = k(t + c)$
- $kR - b = e^{k(t+c)}$
- $kR = e^{k(t+c)} + b$
- $R = \frac{1}{k}(e^{k(t+c)} + b)$
- $R = \frac{1}{k}(Ce^{kt} + b)$

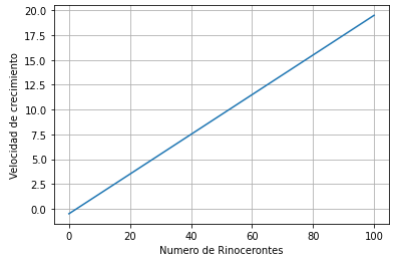
In [39]:
```
sympy.dsolve(modelo)
```

Out[39]:
$$R(t) = \frac{b + e^{k(C_1+t)}}{k}$$

c) Graficar, velocidad - tiempo, y población tiempo. Tiene sentido el modelo generado?

In [40]:
```
def velocidad(R, k, b):
    v = k * R - b
    return v
```

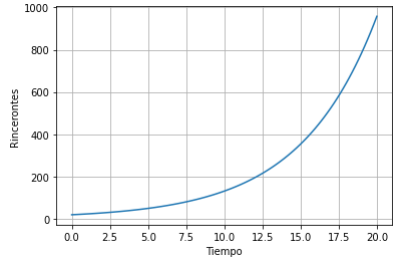
In [41]:
```
R = numpy.linspace(0, 100, 100)
v = velocidad(R, 0.2, 0.5)
pl.plot(R, v)
pl.xlabel("Numero de Rinocerontes")
pl.ylabel("Velocidad de crecimiento")
pl.grid()
```



- $R = \frac{1}{k}(Ce^{kt} + b)$
- $R_0 = \frac{1}{k}(C + b)$
- $kR_0 = C + b$
- $C = kR_0 - b$
- $R = \frac{1}{k}((kR_0 - b)e^{kt} + b)$

In [42]:
```
def poblacion(r0, k, b):
    r = 1/k * ((k * r0 - b)*numpy.exp(k*t)+b)
    return r
```

In [43]:
```
t = numpy.linspace(0,20, 100)
p = poblacion(20, 0.2, 0.5)
pl.plot(t, p)
pl.xlabel("Tiempo")
pl.ylabel("Rincerontes")
pl.grid()
pl.show()
```



14. Considere las siguientes hipótesis respecto a la fracción de una pieza de pan cubierta por moho:

- Las esporas de moho caen sobre el pan a una razón constante.
- Cuando la proporción cubierta es pequeña, la fracción del pan cubierto por el moho se incrementa a una razón proporcional a la cantidad de pan cubierto.
- Cuando la fracción de pan cubierto por el moho es grande, la razón de crecimiento disminuye.
- Para sobrevivir, el moho debe estar en contacto con el pan.

(a) Usando estas hipótesis, escriba una ecuación diferencial que modele la proporción de una pieza de pan cubierta por moho. (Observe que hay más de un modelo razonable que se ajuste a esas hipótesis.) (b) Encuentre la solución general. (c) Graficar, velocidad - tiempo, y población tiempo. Tiene sentido el modelo generado?

- $M_{(t)}$: Área cubierda por el moho $[mm^2]$, variable dependiente
- t: tiempo $[s]$, variable independiente
- $\frac{dM}{dt}$: velocidad de crecimeinto poblacional $[\frac{mm^2}{s}]$
- A: Área del pan $[mm^2]$, parámetro.
- k: Coeficiente de crecimeinto $[s^{-1}]$, parámetro.

"Cuando la proporción cubierta es pequeña, la fracción del pan cubierto por el moho se incrementa a una razón proporcional a la cantidad de pan cubierto.Cuando la fracción de pan cubierto por el moho es grande, la razón de crecimiento disminuye".

- $\frac{dM}{dt} = k(1 - \frac{M}{A})M$

Supongamos que el toro de pan tiene un área de $A = 150[mm^2]$

- $\frac{dM}{dt} = k(1 - \frac{M}{150})M$

In [102]:
```
# Con el conocimiento hasta este momento, no podemos resolver la ecuación sin técncias mas avanzadas
# Utilicemos python para la solución por el momento

# representación del modelo en python

t = sympy.Symbol('t') # variable independiente, el tiempo
k = sympy.Symbol('k') # parametro
M = sympy.Function('M') # variable dependiente, area del moho

dmdt = k*(1 - M(t) / 150)*M(t)
modelo_moho = sympy.Eq(M(t).diff(t), dmdt)
modelo_moho
```

Out[102]:
$$\frac{d}{dt}M(t) = k\left(1 - \frac{M(t)}{150}\right)M(t)$$

(b) Encuentre la solución general.

In [103]:
```
# Con el conocimiento hasta este momento, no podemos resolver la ecuación sin técncias mas avanzadas
# Utilicemos python para la solución por el momento

solucion_moho = sympy.dsolve(modelo_moho)
solucion_moho
```

Out[103]:
$$M(t) = \frac{150}{C_1 e^{-kt} + 1}$$

Supongamos que en un $t = 0$, detectamos area inicial poblada por el moho $10[mm^2]$

$M_{(t=0)} = 10$

In [106]:
```
# Condición inicial en t=0 tenemos M=10
ci = {M(0): 10}

C_eq = sympy.Eq(solucion_moho.lhs.subs(t, 0).subs(ci), solucion_moho.rhs.subs(t, 0))
C_eq
```

Out[106]:
$$10 = \frac{150}{C_1 + 1}$$

In [107]:
```
# Despejamos C1

sympy.solve(C_eq)
```

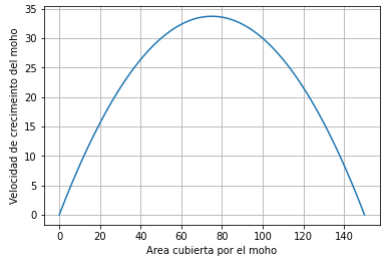
Out[107]: [14]

La solución particular para la condición inicial: $M_{(t=0)} = 1$ es: $M = \frac{150}{14e^{-kt}+1}$

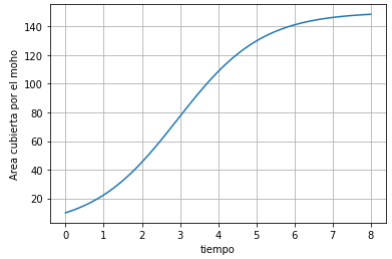
(c) Graficar, velocidad - tiempo, y población tiempo. Tiene sentido el modelo generado?

In [110]:
```
# poblacion vs velocidad
M = numpy.linspace(0, 150, 100)
k = 0.9
A = 150
dmdt = k * (1 - M / A) * M

pl.plot(M, dmdt)
pl.grid()
pl.xlabel("Area cubierta por el moho")
pl.ylabel("Velocidad de crecimeinto del moho")
```

Out[110]: Text(0, 0.5, 'Velocidad de crecimeinto del moho')



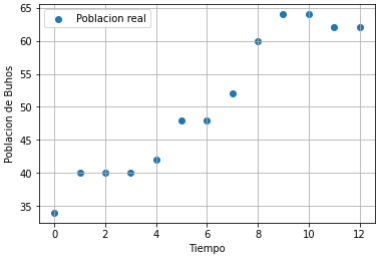
In [109]:
```
# Sea A= 150[mm^2] el área del pan de nuestro caso
# Sea C=10
# Sea el coeficiente de crecimiento k=0.05.
t= numpy.linspace(0, 8, 100)
k = 0.9


M = 150 / (14* numpy.exp(-k*t)+1)
pl.plot(t,M)
pl.grid()
pl.xlabel("tiempo")
pl.ylabel("Area cubierta por el moho")
pl.show()
```



15.El archivo ejer11.csv contiene datos sobre la población de búhos amarillos en Inglaterra. La primera columna corresponde a los años en que se tomo el control de población, la segunda respectivamente pertenece al tamaño de la población de Búhos. :

(a) Grafique utilizando matplotlib.

In [54]:
```
data = numpy.loadtxt("ejer11.csv", delimiter=",")
buhos = data[:, 1]
tiempo = data[:, 0] - data[0,0]
pl.scatter(tiempo, buhos, label="Poblacion real")
pl.grid()
pl.xlabel("Tiempo")
pl.ylabel("Poblacion de Buhos")
pl.legend()
pl.show()
```



(b) Qué modelo de población usaría usted para modelar esta población?

$P(t) = ce^{kt}$

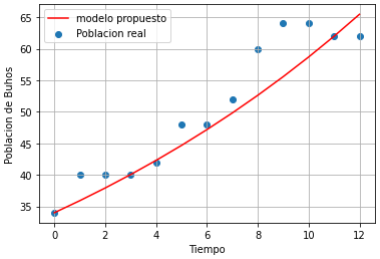
Población inicial:

- $P(t = 0) = 34$
- $P(t) = ce^{kt}$

Población en t=11

- $P(t = 11) = 34$
- $62 = 34e^{11k}$
- $k = \frac{\ln(\frac{62}{34})}{11}$
- $P(t) = 34e^{0.056t}$

In [57]:
```
data = numpy.loadtxt("ejer11.csv", delimiter=",")
buhos = data[:, 1]
tiempo = data[:, 0] - data[0,0]
pl.scatter(tiempo, buhos, label="Poblacion real")
pl.plot(tiempo, 34*numpy.exp(0.0546*tiempo), color='red', label="modelo propuesto")
pl.grid()
pl.xlabel("Tiempo")
pl.ylabel("Poblacion de Buhos")
pl.legend()
pl.show()
```



(c) Puede usted calcular (o por lo menos hacer estimaciones razonables) los valores del parámetro?

- $P(t = 11) = 34$
- $62 = 34e^{11k}$
- $k = \frac{\ln(\frac{62}{34})}{11}$
- $P(t) = 34e^{0.056t}$

(d) Qué predice su modelo para la población actual?

In [62]:
```
print("t \t real \t propuesto")
for i in range(tiempo.size):
    print(int(tiempo[i]), "\t", int(buhos[i]), "\t", int(34*numpy.exp(0.0546*tiempo[i])))
```

```
t        real    propuesto
0        34      34
1        40      35
2        40      37
3        40      40
4        42      42
5        48      44
6        48      47
7        52      49
8        60      52
9        64      55
10       64      58
11       62      61
12       62      65
```