Modelo Lotka Volterra

Graficacion computacional

El modelo de Lotka-Volterra, también conocido como modelo de presa-depredador, se ocupa de la interacción entre dos especies, donde una de ellas (presa) tiene abundante comida, y la segunda especie (depredador) tiene suministro de alimentos exclusivamente a la población de las presas.

Flores Hidalgo Yair Uriel

Importacion de bibliootecas

In [17]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys
```

Se muestra los parámetros del modelo (a, b, c y e) los cuales muestran las tasas de crecimiento y decrecimiento

• ‘a’: Es la tasa del aumento natural de las presas

• ‘b’: Es la tasa de destrucción por depredadores

• ‘c’: Es la tasa de muerte en ausencia de presas

• ‘e’: Es la tasa de aumento causado por alimentación disponible

También se muestra una simulación del tiempo, donde se inicializan variables (t, x y y) donde:

• ‘t’: Es el tiempo inicial

• ‘x’: La población inicial de presas

• ‘y’: La población inicial de los depredadores

Y por ultimo se muestran unas listas donde se almacenan los valores de tiempo, población tanto de presas como de depredadores

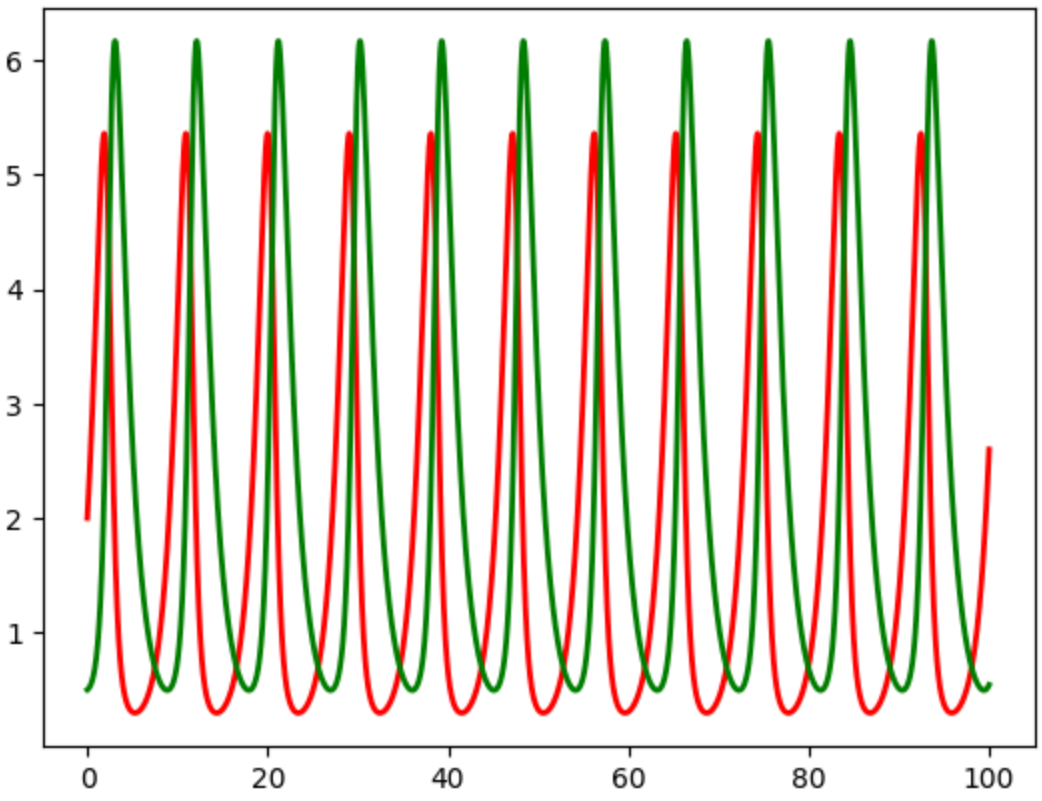
Parametros

In [18]:
```
a = 0.9; b = 0.4; c = 0.7;  e = 0.4
dt = 0.001; max_time = 100
```

```
t = 0; x = 2.0; y = 0.5
```

Creacion de listas

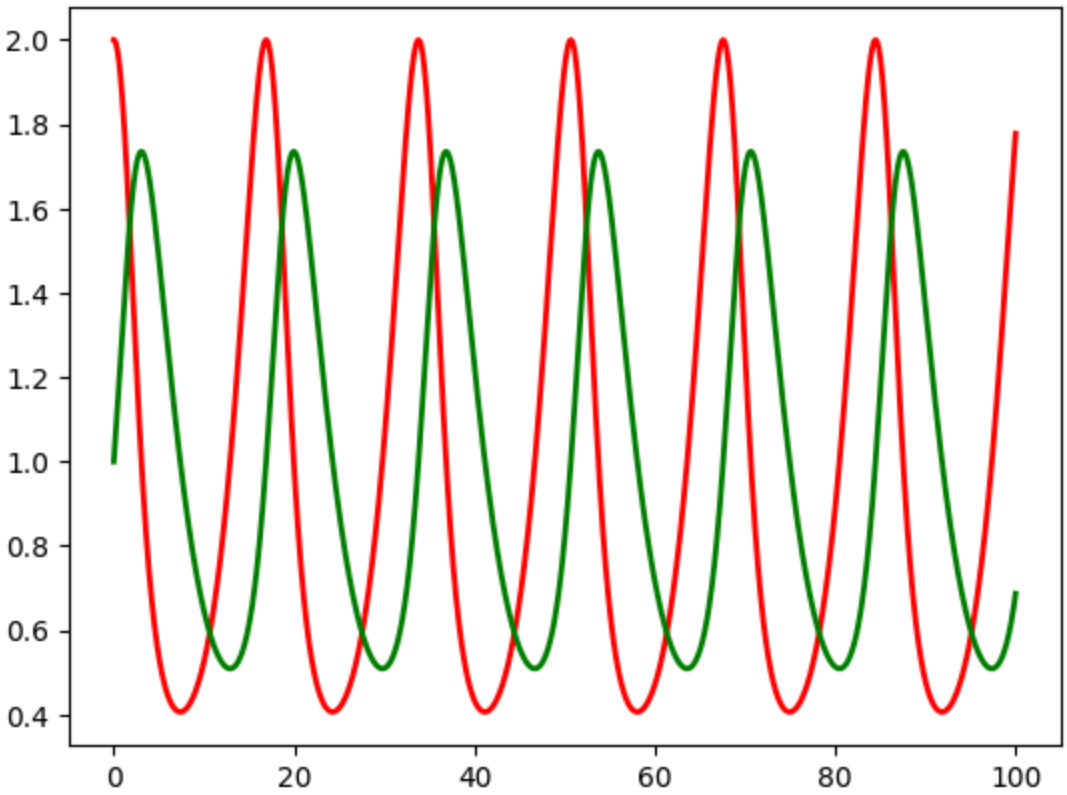
In [20]:
```
t_list = []; x_list = []; y_list = []

t_list.append(t); x_list.append(x); y_list.append(y)
while t < max_time:
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)

plt.show()
```



Caso 1

1. En este primer modelo, podemos observar que al inicio como la población de las presas es mayor (línea roja) ya que la población es mayor a la de los depredadores, posteriormente dado a la disminución de presas casi de automático baja el número de depredadores y al bajar en número de este, las presas subieran y así scesivamente.

In [21]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys


# model parameters
a = 0.5; b = 0.5; c = 0.3;  e = 0.3
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 2.0; y = 1.0

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)

plt.show()
```

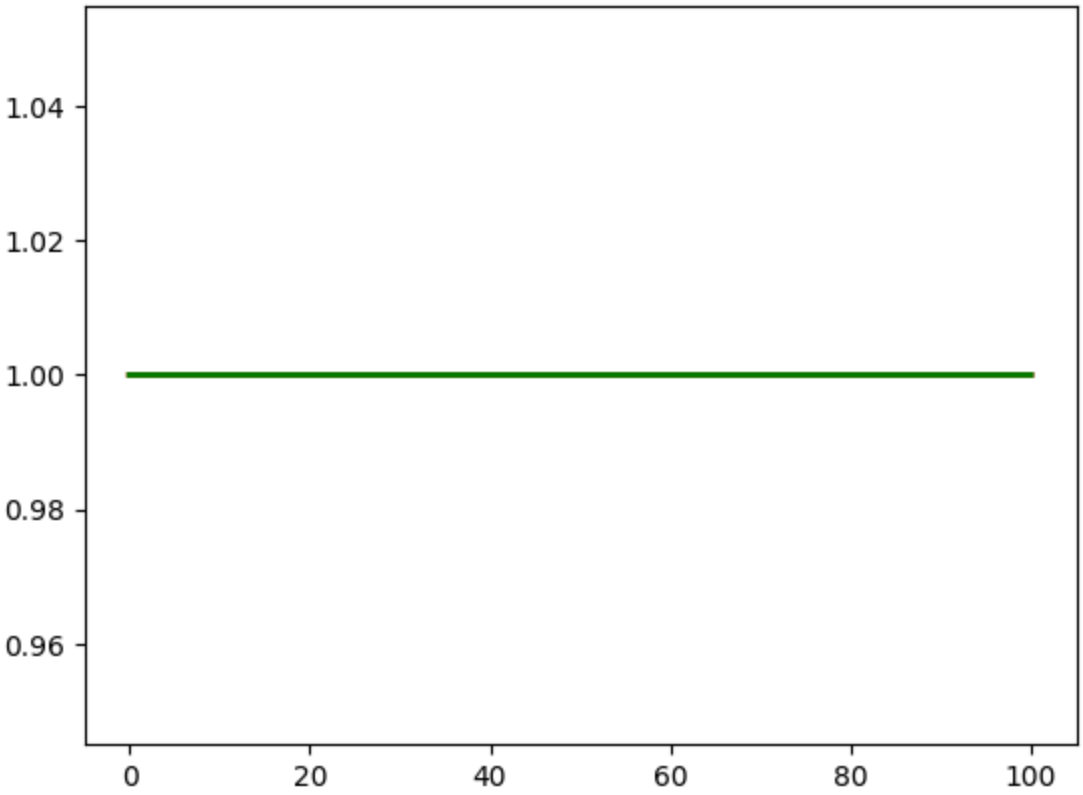


Caso 2

En el ejemplo dos, pusimos los datos iguales para ver si encontrábamos un punto de equilibrio, dando las mismas tasas tanto para el crecimiento como decrecimiento y a esto se le agrega que ambos tienen la misma cantidad de población inicial. Al final observamos una línea verde, la cual puede indicar que ambas líneas se encuentran al mismo nivel.

In [23]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys


# model parameters
a = 0.7; b = 0.7; c = 0.3;  e = 0.3
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 1.0

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
```

```
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)

plt.show()
```

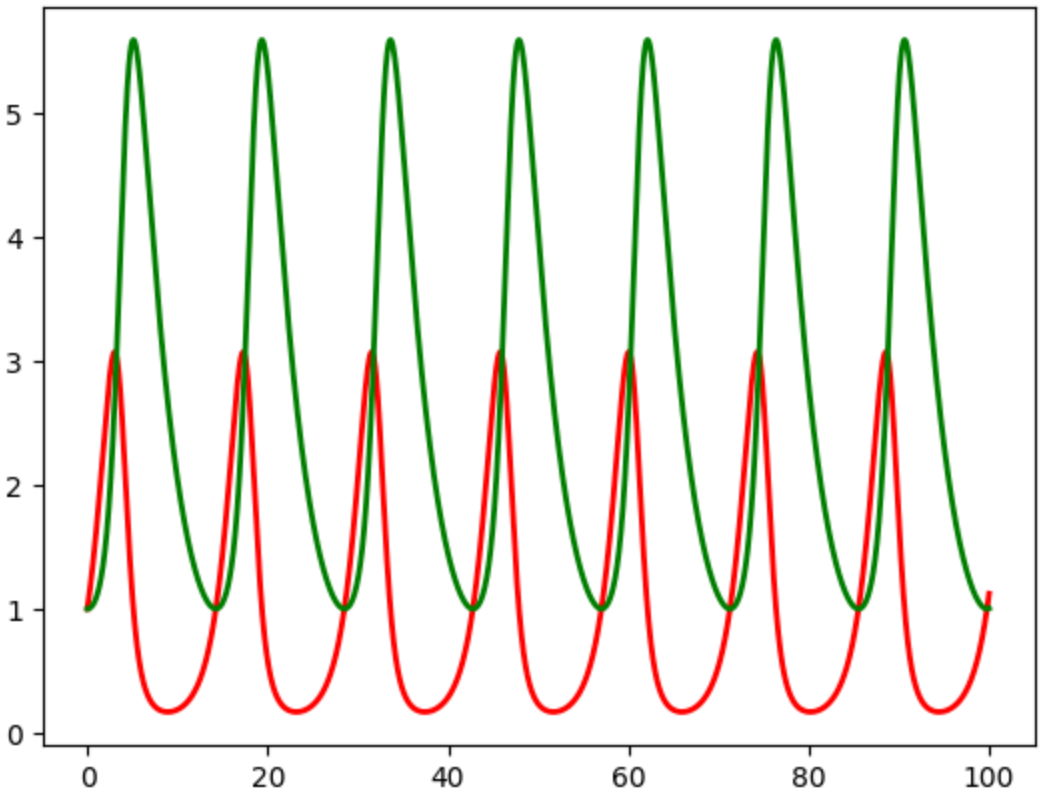


Caso 3

3. Para este caso, se modifico con la finalidad de hacer a las presas mas “fuertes” para ello, se aumento la tasa en la que se reproducen las presas y se bajo la tasa en la que forma que atacan los depredadores. Podemos observar que efectivamente los depredadores son mas “fuertes” ya que aun en su nivel mas bajo, no bajo mas de su población inicial a comparación de las presas

In [24]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys
```

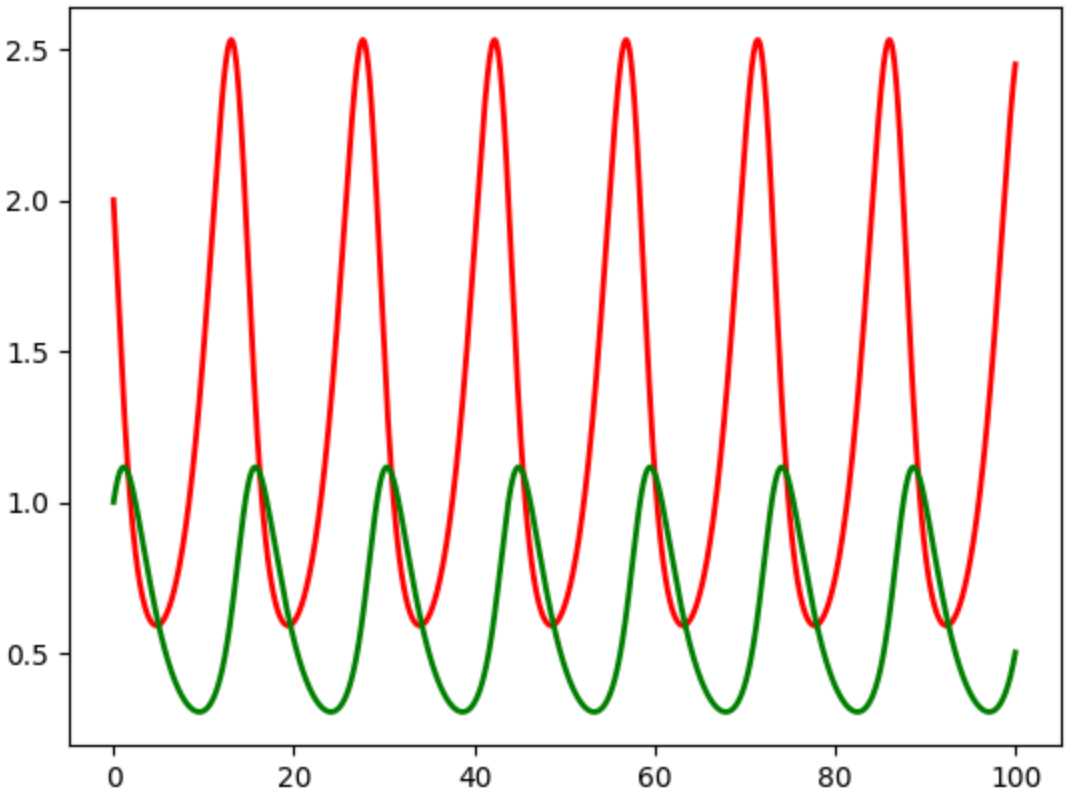
```
# model parameters
a = 0.8; b = 0.3; c = 0.3;  e = 0.3
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 1.0; y = 1.0

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)

plt.show()
```



Caso 4

Para este ejemplo, decidimos hacer lo contrario del ejemplo anterior, en este caso aumento la manera de crecer de las presas, además de agregar el doble de la población inicial de las presas

In [25]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys


# model parameters
a = 0.5; b = 0.8; c = 0.4;  e = 0.3
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 2.0; y = 1.0

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)

plt.show()
```

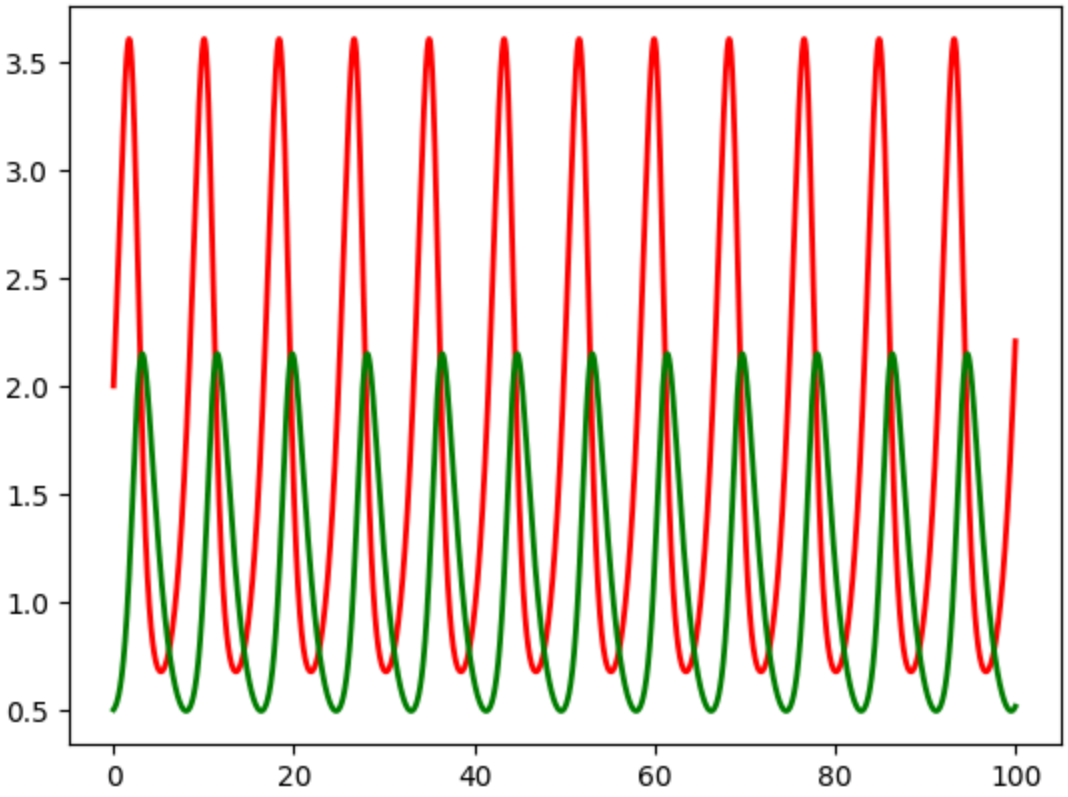


Caso 5

Para este último ejemplo, pusimos números aleatorios para interpretar el resultado, y el diagnostico es que la población inicial de presas es mayor a la de los depredadores, y de nueva cuenta podemos ver cómo es la relación entre presas y depredadores a que si una crece la otra hará lo mismo y viceversa

In [26]:
```
import matplotlib.pyplot as plt
from random import *
from numpy import *
import sys


# model parameters
a = 0.9; b = 0.8; c = 0.7;  e = 0.4
dt = 0.001; max_time = 100

# initial time and populations
t = 0; x = 2.0; y = 0.5

# empty lists in which to store time and populations
t_list = []; x_list = []; y_list = []

# initialize lists
t_list.append(t); x_list.append(x); y_list.append(y)

while t < max_time:
    # calc new values for t, x, y
```

```
    t = t + dt
    x = x + (a*x - b*x*y)*dt
    y = y + (-c*y + e*x*y)*dt

    # store new values in lists
    t_list.append(t)
    x_list.append(x)
    y_list.append(y)

# Plot the results
p = plt.plot(t_list, x_list, 'r', t_list, y_list, 'g', linewidth = 2)

plt.show()
```



In [ ]: