MP307 Practical 3 Discrete Population Models

Dara Corr ID: 18483836

In [3]:
```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```

```

Q.1(*) Geometric Population Model

Consider the geometric population model $P_{n+1} = (1+r)P_n$ for $n = 0, 1, \ldots$ with solution $P_n = P_0 e^{sn}$ for $s = \log(1+r)$ as a model of

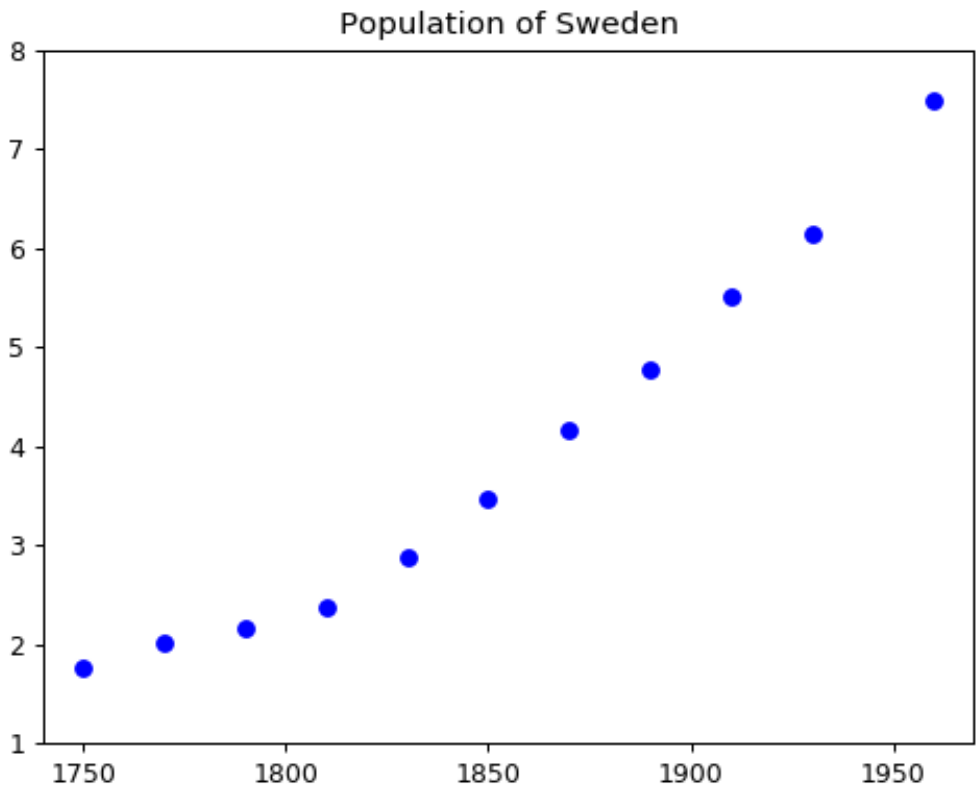
1. Sweden's population with 11 samples from 1750 to 1960
2. The US population with 19 samples from 1790 to 1990.

Population of Sweden

The data (in millions) are given in the lists `Year` and `Pop` below.

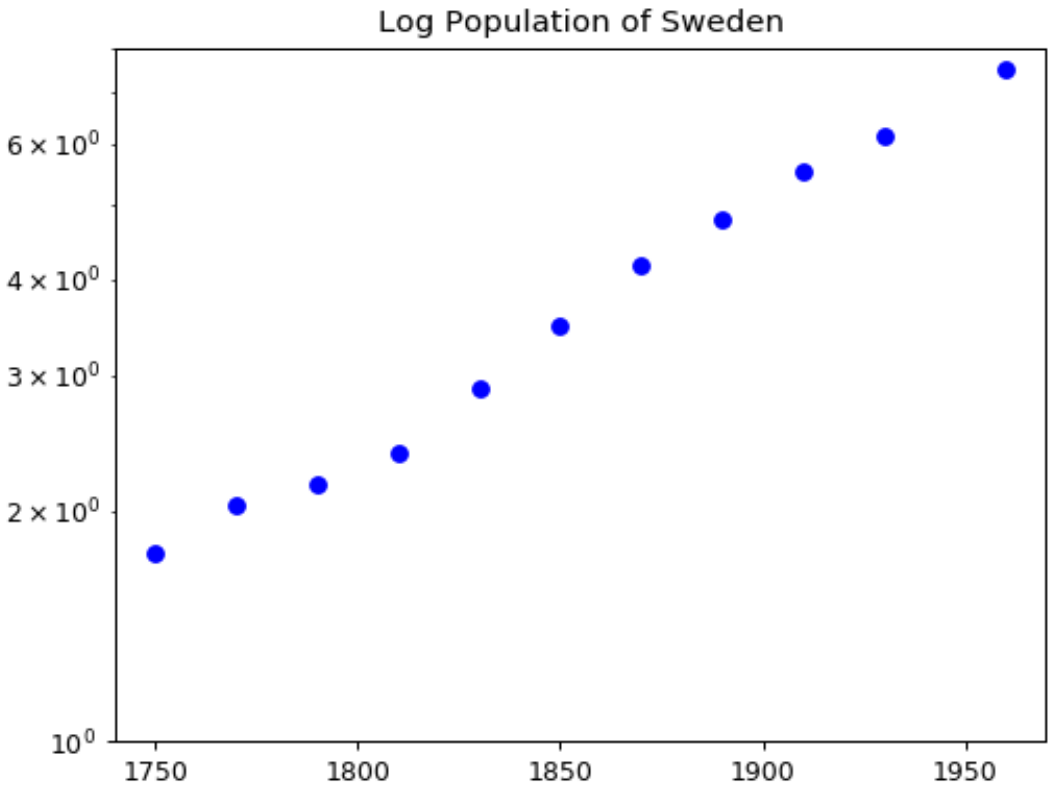
In [1]:
```
Year = [1750, 1770, 1790, 1810, 1830, 1850, 1870, 1890, 1910, 1930, 1960]
Pop= [1.76, 2.03, 2.158, 2.378, 2.888, 3.48, 4.17, 4.785, 5.522, 6.14, 7.49]
```

In [4]:
```
plt.figure(1)
plt.plot(Year,Pop,'bo')
plt.axis([1740,1970,1,8])
plt.title('Population of Sweden')
plt.show()
```



Look at t vs log(P):

In [5]:
```
plt.figure(2) # new figure
plt.plot(Year,Pop,'bo')
plt.yscale("log")
plt.axis([1740,1970,1,8])
plt.title('Log Population of Sweden')
plt.show()
```



The log plot suggest a linear relationship to between $\log P$ and t of the form $\log P(t) = st + b$ for some s, b. Thus $P(t) = P_0 e^{st}$ with $P_0 = \exp(b)$.

In [6]:
```
log_Pop=np.log(Pop) # take log of each entry in S_Pop
```

In [7]:
```
print(log_Pop)
```

```
[0.56531381 0.70803579 0.76918187 0.8662598  1.06056422 1.24703229
 1.42791604 1.56548602 1.70874011 1.81482474 2.0135688 ]
```

Use a least squares approximation to find a line as explained in the notes for matching data x_i to y_i.

In [8]:
```
N=len(Year) # no of elements in S_Year
sumx=0; sumy=0; sumxx=0; sumxy=0;
for i in range(N):
    sumx=sumx+Year[i]
    sumy=sumy+log_Pop[i]
    sumxx=sumxx+Year[i]**2
    sumxy=sumxy+log_Pop[i]*Year[i]
Xbar=sumx/N
Ybar=sumy/N
XXbar=sumxx/N
XYbar=sumxy/N
```

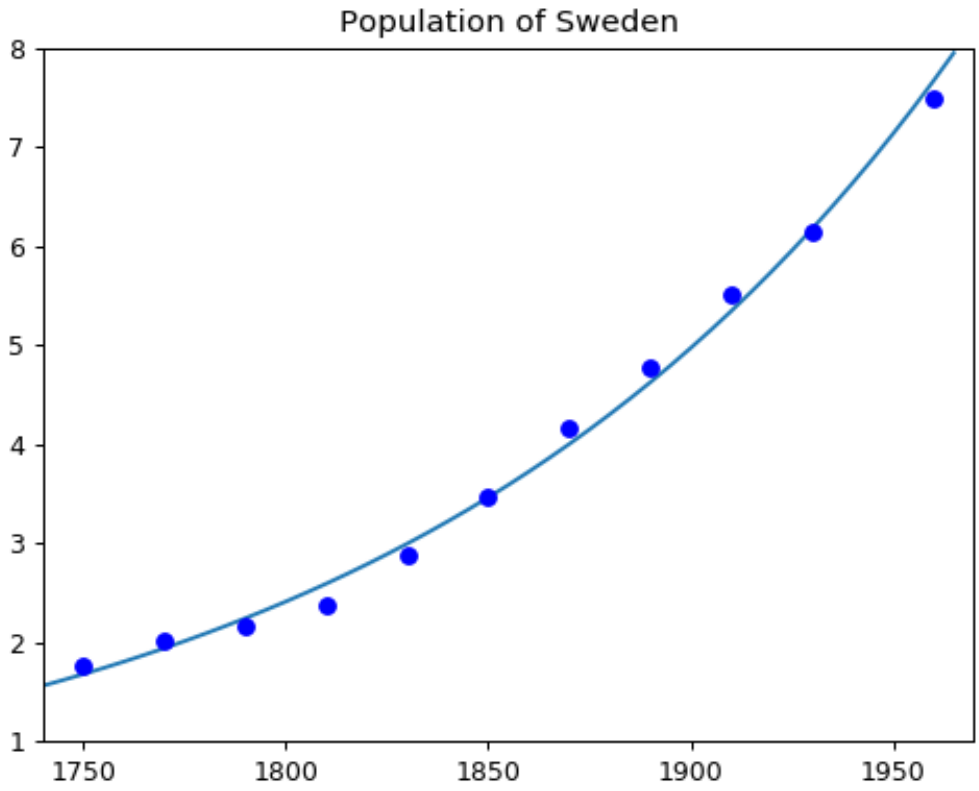
In [9]:
```
s=(XYbar-Xbar*Ybar)/(XXbar-Xbar**2)
b=(XXbar*Ybar-Xbar*XYbar)/(XXbar-Xbar**2)

P_0=np.exp(b)
print(s,P_0)
```

```
0.007218271157956407 5.500654960458621e-06
```

Plot population $P(t) = P_0 e^{st}$ vs t.

In [10]:
```
t=np.arange(1740.0,1970.,5.)
P_LS=P_0*np.exp(s*t) # least squares approximation
```

In [11]:
```
plt.figure(3)
plt.plot(t,P_LS)
plt.plot(Year,Pop,'bo')
plt.axis([1740,1970,1,8])
plt.title('Population of Sweden')
plt.show()
```

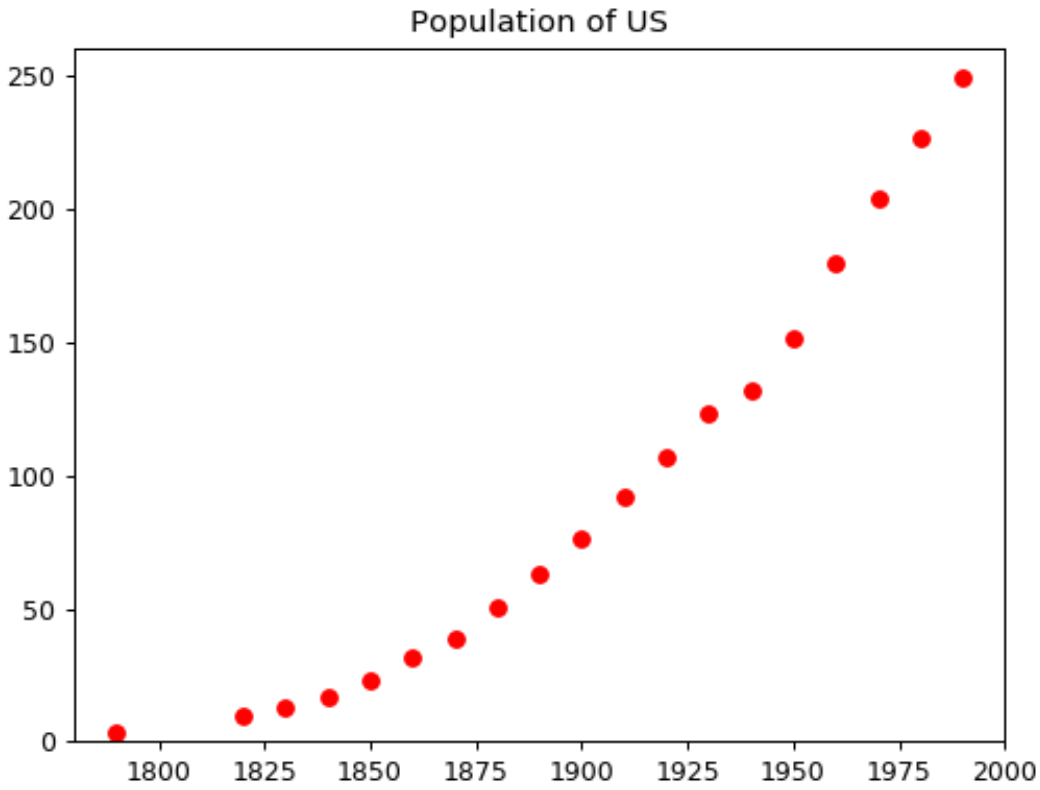


From Figure 1 we see the growth of a population P that appears to be growing exponentially. By finding Log(P) and plotting it against time t, we get a plot which resembles a straight line plot with some variance which shows that the data could be modelled using a geometric population model. Finding parameters s and b from the straight line plot $Log(P) = st + b$ we can model exponential growth by plotting $P(t) = P_0 e^{st}$ vs t. This model appears to fit the data reasonably well. It does not account for a slight decline in population growth in the 1800s but otherwise it is a reasonably good model for the Population of Sweden from 1750 to 1960.

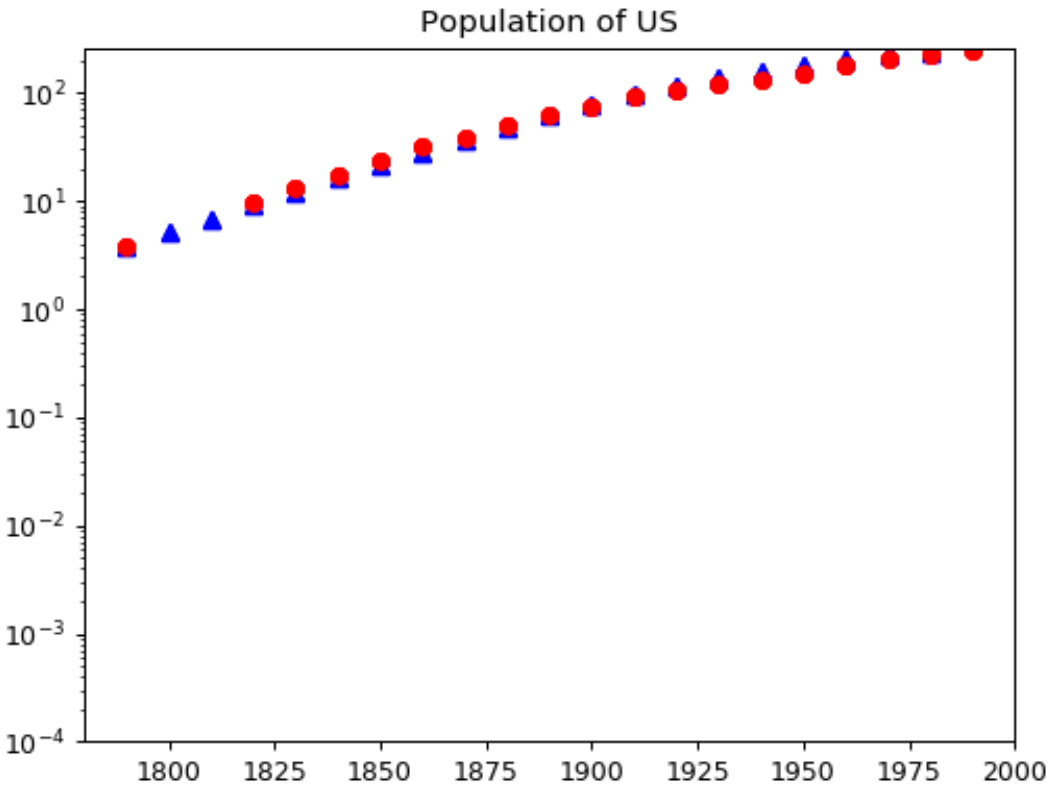
Repeat this analysis for the US population as follows:

In [24]:
```
Year = [1790, 1820, 1830, 1840, 1850, 1860, 1870, 1880, 1890, 1900, 1910, 1920
, 1930, 1940, 1950, 1960, 1970, 1980, 1990]
Pop= [3.79, 9.6, 12.9, 17.1, 23.2, 31.4, 38.6, 50.2, 62.9, 76.0, 92.0, 106.5,
123.2, 132.0, 151.3, 179.3, 203.3, 226.54, 248.7]
```

In [25]:
```
plt.figure(4)
plt.plot(Year,Pop,'ro')
plt.axis([1780,2000,0,260])
plt.title('Population of US')
plt.show()
```



In [28]:
```
plt.figure(5) # new figure
plt.plot(Year,Pop,'ro')
plt.yscale("log")
plt.axis([1780,2000,0,260])
plt.title('Log Population of the USA')
plt.show()
```

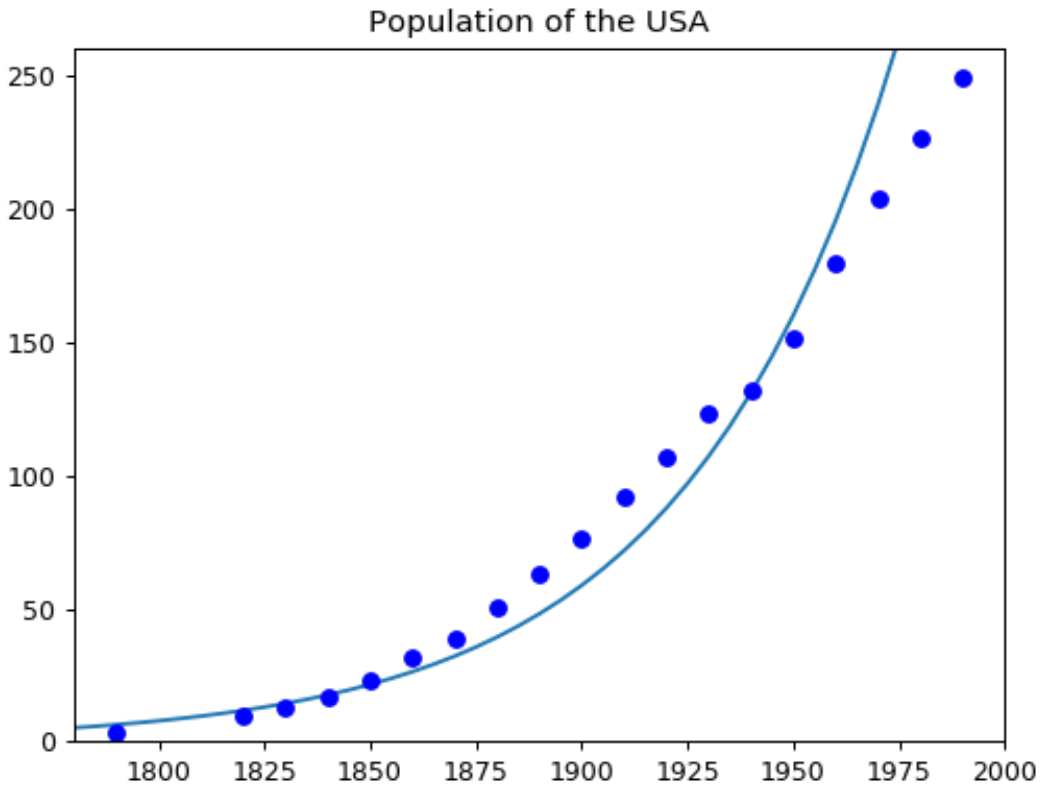



```
C:\Users\Dara\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarnin
g: Attempted to set non-positive bottom ylim on a log-scaled axis.
Invalid limit will be ignored.
  after removing the cwd from sys.path.
```

In [29]:
```
#attempt to model using exponential growth model

log_Pop=np.log(Pop) # take log of each entry in S_Pop
print(log_Pop)


#least squares approximation to find constants s and b
N=len(Year) # no of elements in S_Year
sumx=0; sumy=0; sumxx=0; sumxy=0;
for i in range(N):
    sumx=sumx+Year[i]
    sumy=sumy+log_Pop[i]
    sumxx=sumxx+Year[i]**2
    sumxy=sumxy+log_Pop[i]*Year[i]
Xbar=sumx/N
Ybar=sumy/N
XXbar=sumxx/N
XYbar=sumxy/N

s=(XYbar-Xbar*Ybar)/(XXbar-Xbar**2)
b=(XXbar*Ybar-Xbar*XYbar)/(XXbar-Xbar**2)

P_0=np.exp(b)
print(s,P_0)
```

```
[1.33236602 2.2617631  2.55722731 2.83907846 3.14415228 3.44680789
 3.65325228 3.91601503 4.14154616 4.33073334 4.52178858 4.66814499
 4.81380905 4.88280192 5.01926462 5.18906038 5.31468272 5.42292153
 5.51624735]
0.020051106626461583 1.6718326017274779e-15
```

In [30]:
```
t=np.arange(1780,2000,5.)
P_LS=P_0*np.exp(s*t) # least squares approximation

plt.figure(6)
plt.plot(t,P_LS)
plt.plot(Year,Pop,'bo')
plt.axis([1780,2000,0,260])
plt.title('Population of the USA')
plt.show()
```



We try to apply the same Geometric Model we used on the Swedish Population from 1750-1960 to the Population of the USA from 1790-1990. In figure 5, we see that the log of Population P (log(P)) resembles a curve more than a straight line which suggests that a good linear fit of the plot Log(P) vs t cannot be achieved. In Figure 6, we see that the Geometric Popuation Growth Model does not produce a good fit against the real world data for the Population in the USA from 1790-1990. The plot gives a reasonable fit from 1800 until about 1850 and then the Model Curve underestimates the population of the USA by a large margin until about 1940 when it begins to drastically overestimate the population. The Geometric Model we use here does not work in this case with the supplied data for US population.

Q.2 (*) Discrete Verhulst Model

The general discrete Verhulst model of population growth is as follows

$$P_{n+1} = \frac{1+r}{1+\frac{rP_n}{K}}P_n$$

where $P_n \to K$ as $n \to \infty$.

For the choice of parameters $P_0 = 3.79, r = 0.35$ and $K = 300$ compare the Verhulst model to the US population from 1790 to 1990. Explain what you see.

In [41]:
```
print(Year)
```

```
[1790, 1820, 1830, 1840, 1850, 1860, 1870, 1880, 1890, 1900, 1910, 1920, 193
0, 1940, 1950, 1960, 1970, 1980, 1990]
```

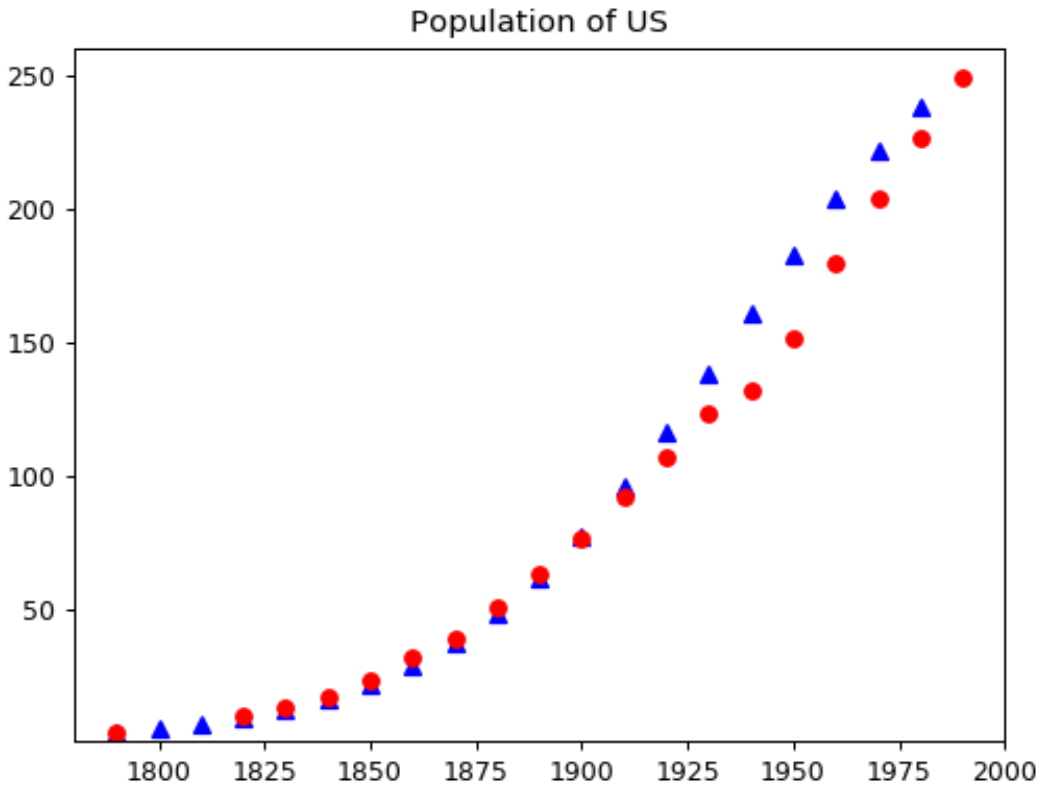
In [42]:
```
P_0=3.79
r=0.35
K=300.0
```

In [43]:
```
# plot Verhulst pop PV from 1790 to 1990 with 10 year intervals
PV=[0]*20 # list for Verhulst P initialized with 20 zeros
YV=[0]*20 # list for years initialized with 20 zeros

PV[0]=P_0
YV[0]=1790
# choose 10 year intervals
for n in range(1,20):
    YV[n]=1790+10*n
    PV[n]=(1+r)*PV[n-1]/(1+r*PV[n-1]/K)
```

In [44]:
```
print(PV)
```

```
[3.79, 5.093976135520772, 6.836240197791433, 9.15590034404303, 12.22982786510
7413, 16.278010802773355, 21.56575898309768, 28.39924806493404, 37.1094574197
0735, 48.0188229724253, 61.386423256309854, 77.33325731948169, 95.76020718123
476, 116.28490230208658, 138.23136107104716, 160.69678033606021, 182.69000801
767106, 203.3004004123143, 221.83893811890553, 237.9088727981478]
```

In [45]:
```
plt.figure(7)
plt.plot(YV,PV,'b^')
plt.plot(Year,Pop,'ro')
plt.axis([1780,2000,0.0001,260])
plt.title('Population of US')
plt.show()
```



The Discrete Verhulst Model fits the data for the US population very well, it fits the data from 1790 to 1910 quite closely. After the year 1910, the Verhulst Model overpredicts the population of the US from 1920 to 1980. So we see that it works very well for the first half of the data and then after a certain point it is not as effective at modelling the population of the US with the data provided here. A possible cause for the discrepency between the model and the real data here is the events of World War 2 which caused a slight decrease in population growth that the model did not account for.

Q.3 (*) Cohort Model

Consider the human population divided into three age groups $0-14$, $15-39$ and ≥ 40 with population size P_i , $i=1,2,3$ and yearly birth rates of 0, 0.06 and 0 and death rates of 0.005, 0.01 and 0.015, respectively.

(a). Show that $\mathbf{P}(t+1)=\mathbf{A}\mathbf{P}(t)$ where

$$\mathbf{P}=\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}, \quad \mathbf{A}=\begin{bmatrix} 0.92867 & 0.06 & 0 \\ 0.0663 & 0.9504 & 0 \\ 0 & 0.0396 & 0.985 \end{bmatrix}.$$

(b). Find the largest eigenvalue of $\mathbf{A}$ and describe the long-term behaviour of this system.

(c). Consider the growth of the population with the following initial populations. How long does it take for the total population to double in each case?

(i) $P_1=200$, $P_2=P_3=400$.

(ii) $P_1=400$, $P_2=P_3=300$.

(iii) $P_1=200$, $P_2=500$ and $P_3=300$.

Explain what you see.

In [1]:
```
b=[0, 0.06 ,0]
d=[0.005, 0.01, 0.015]
```

In [25]:
```
A=np.mat([[b[0] + 14/15 - 14/15 * d[0], b[1], b[2]],
          [1/15*(1-d[0]), 24/25*(1-d[1]), 0],
          [0, 1/25*(1-d[1]), 1-d[2]]])
# recall Python indexing begins at 0
```

In [5]:
```
print(A)
```

```
[[0.92866667 0.06       0.        ]
 [0.06633333 0.9504     0.        ]
 [0.         0.0396     0.985     ]]
```

Find eigenvalues and eigenvectors

In [6]:
```
X, V=np.linalg.eig(A)
print(X) # eigen values
print(V) # columns are right eigenvectors
```

```
[0.985      1.00354962 0.87551705]
[[ 0.          0.32180453  0.72790385]
 [ 0.          0.40162787 -0.64479684]
 [ 1.          0.85740124  0.23322312]]
```

In [18]:
```
P = [[100],
     [100],
     [200]]
print(P)

p1 = 100
p2 = 100
p3 = 200
```

```
[[100], [100], [200]]
```

In [34]:
```
#P1(t+1) = b0*P1(t) + b1*P2(t) + b2*P3(t)
#P2(t+1) = (1-d0)P1(t)
#P3(t+1) = (1 - d1)P2(t)

#python indexing starts at 0 so P1 = P[0]... etc

P_t_plus_1 = [ [(b[0]+ 14/15 - 14/15 * d[0] )*p1 + b[1]*p2 + b[2]*p3 ],
       [ 1/15*(1 - d[0]) * p1 + 24/25 * (1 - d[1]) * p2 ],
       [ 1/25*(1-d[1])* p2 + (1-d[2]) * p3 ] ]




print(P_t_plus_1)
```

```
[[98.86666666666666], [101.67333333333332], [200.96]]
```

In [10]:
```
print(A*P) #this is equal to P(t+1)
```

```
[[ 98.86666667]
 [101.67333333]
 [200.96      ]]
```

(b) Can extract first, second, third eigenvalues/vectors as follows:

In [29]:
```
lambda0=X[0]
u0=V[:,0] # This is a 3 by 1 matrix
print(lambda0,u0)
```

```
0.985 [[0.]
 [0.]
 [1.]]
```

In [30]:
```
np.matmul(A,u0)/lambda0 # A u = lambda u
```

Out[30]:
```
matrix([[0.],
        [0.],
        [1.]])
```

In [31]:
```
lambda1=X[1]
u1=V[:,1]
print(lambda1,u1)
```

```
1.0035496159842685 [[0.32180453]
 [0.40162787]
 [0.85740124]]
```

In [55]:
```
np.matmul(A,u1)/lambda1 # A u = lambda u
```

Out[55]:
```
matrix([[0.32180453],
        [0.40162787],
        [0.85740124]])
```

In [32]:
```
lambda2=X[2]
u2=V[:,2]
print(lambda2,u2)
```

```
0.8755170506823979 [[ 0.72790385]
 [-0.64479684]
 [ 0.23322312]]
```

In [57]:
```
np.matmul(A,u2)/lambda2 # A u = lambda u
```

Out[57]:
```
matrix([[ 0.72790385],
        [-0.64479684],
        [ 0.23322312]])
```

The largest Eigenvalue in this system has a value of 1.003545.. .Since this value is greater than 1 this means the population exponentially increases over time since the largest eigenvalue is the one which domintates over large time steps.

(c) Initial poplulation $P_1 = P_2 = P_3 = 100$.

In [58]:
```
P0=[100,100,100]
```

In [59]:
```
np.matmul(A,P0) # population after one year
```

Out[59]:
```
matrix([[ 98.86666667, 101.67333333, 102.46      ]])
```

In [60]:
```
n=20 # 20 years
An=np.linalg.matrix_power(A,n) # matrix A^n
print(An)
```

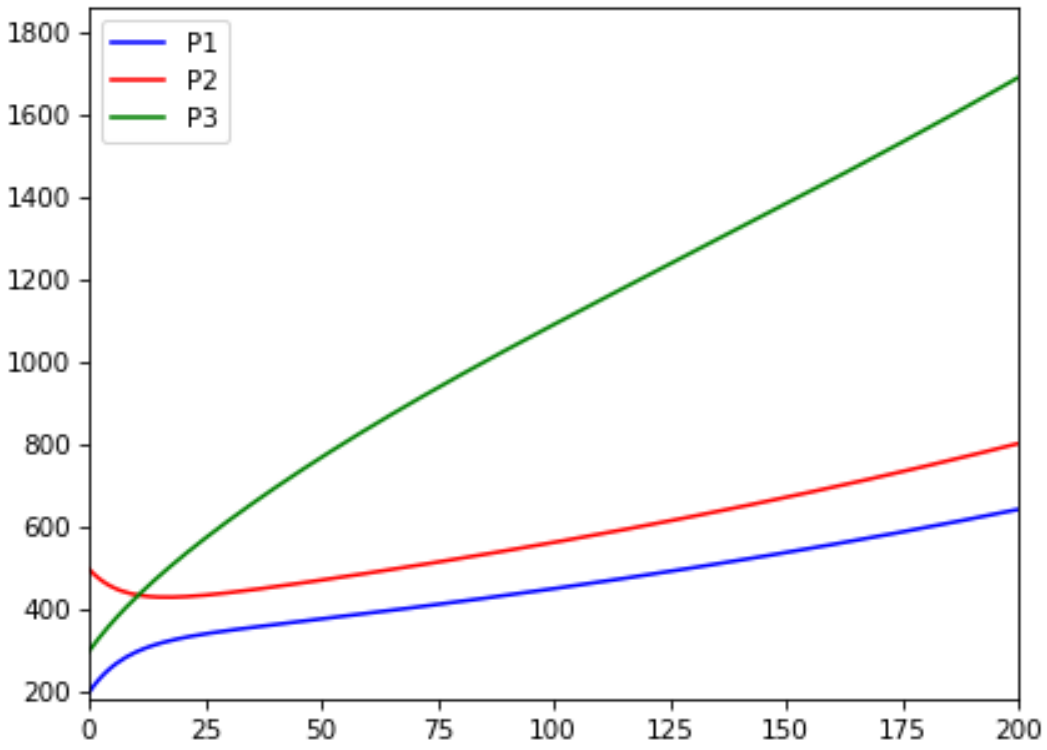
```
[[0.48657128 0.47022728 0.        ]
 [0.51986238 0.65689805 0.        ]
 [0.24436408 0.51787528 0.73913643]]
```

In [61]:
```
Pn=np.matmul(An,P0) # population after n years
print(Pn)
```

```
[[ 95.67985614 117.67604324 150.13757869]]
```

Define lists P1, P2, P3 giving population at time 0,1,2,3, etc

In [64]:
```
nit= 200 # nit is no of time steps

#initialise list to zero

P1=[0]*(nit+1)
P2=[0]*(nit+1)
P3=[0]*(nit+1)

#Given initial population

P1[0],P2[0],P3[0]=200,500,300

# compute P1,P2,P3
for n in range(1,nit+1):
    AP=np.matmul(A,[P1[n-1],P2[n-1],P3[n-1]]) # AP[n-1]
    P1[n],P2[n],P3[n]=AP[0,0],AP[0,1],AP[0,2]

# plotting


t=np.arange(0,nit+1,1) #  time steps
plt.figure()
P1plot,=plt.plot(t,P1,'b-')
P2plot,=plt.plot(t,P2,'r-')
P3plot,=plt.plot(t,P3,'g-')
plt.legend((P1plot,P2plot,P3plot),('P1','P2','P3'))
Pmax=max(max(P1),max(P2),max(P3))
Pmin=min(min(P1),min(P1),min(P1))
plt.axis([0,nit,Pmin/1.1,Pmax*1.1])
plt.show()
```



Compare populations to eigenvector for largest eigenvalue

In [63]:
```
print(P2[50]/P1[50],u1[1,0]/u1[0,0]) # compare first and second cohorts to eig
envector entries
```

```
1.247744063179402 1.248049155293366
```

In [64]:
```
print(P3[50]/P1[50],u1[2,0]/u1[0,0]) # compare first and third cohorts to eige
nvector entries
```

```
2.0444670328229053 2.664354161915348
```

(i) P1 doubles after about 160 years, P2 doubles after about 215 years and P3 doubles after about 55 years. P1 and P2 gradually increase and P3 grows rapidly. P2 declines slightly for the first 10 years until it starts to increase.

(ii) P1 doubles after about 266 years, P2 doubles after about 125 years, P3 doubles after about 34 years. The Values of P2 and P3 are near zero at the start of the plot. P1 starts to decline for about 20 years and then it starts to gradually increase. P2 and P3 both start increasing rapidly until about 8 years where P3 continues to grow at a rapid rate and P2 increases at a more gradual rate as before.

(iii) P1 doubles after about 60 years, P2 doubles after about 260 years, P3 doubles after about 28 years. P1 rapidly increases for about 15 years and then continues to grow at a steady rate. P2 declines for the first 15 years and then starts to grow gradually afterwards. P3 grows at a rapid rate as in case (i) and (ii)

Q.4 (*) Chaotic Models

Consider the Ricker model of Salmon population size as follows

$$P_{n+1} = re^{-\frac{P_n}{K}} P_n$$

where r is the rate and K is another parameter. For simplicity choose $K = 500$ and consider the behaviour of this system for

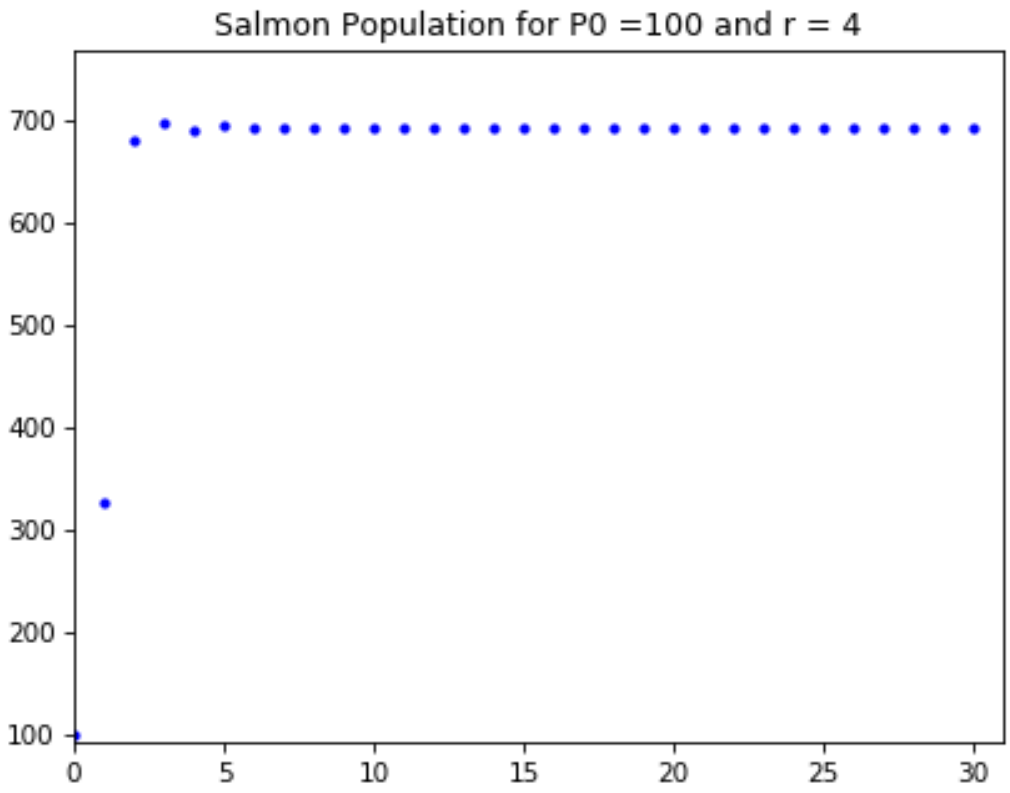
(a). $r < 1$, with initial population of 1000.

(b). $1 < r < e^2$, for initial population 100, 500 and 1000.

(c). $r > e^2$, for initial population 100, 500 and 1000.

Explain what you see.

In [132]:
```
# initial values
P_0=100
K=500.
r=4

nit=30 # nit is no of time steps

t=np.arange(0,nit+1,1) #  time steps

P=[0]*(nit+1) # list for P initialized with nit zeros

P[0]=P_0 # given initial population
for n in range(1,nit+1):
    P[n]=r*P[n-1]*np.exp(-P[n-1]/K)

# plotting
plt.figure()
plt.plot(t,P,'b.')
Pmax=max(P)
Pmin=min(P)
plt.axis([0,nit+1,Pmin/1.1,Pmax*1.1])
plt.title('Salmon Population for P0 =' +str(P_0)+ ' and r = ' +str(r))
plt.show()
```



Population dies out for $r < 1$.

Explore the other values of r where one can find convergence to non-zero population for $1 < r < e^2$ and period doubling phenomena and chaos for $r > e^2$.

In [129]:
```
e2=np.exp(2)
print(e2)
```

```
7.38905609893065
```

(a)Population decreases exponentially and dies down to 0 after about 10 time steps.

(b) The population increases rapidly and converges to a fixed point $P* = K \cdot ln(R_0)$. The population will usually overshoot $P*$ by a small margin before converging to $P*$ in the next couple of time steps. when P_0 = 1000, the population decreased to 544 and then started to increase again in order to converge to $P*$. When r approaches e^2, points oscillate above and below $P*$ for some time but eventually converge to $P*$

(c) Period Doubling starts to occur where the series appears to oscillate between two fixed values of P with points alternating between these P values. This does not converge. The plot becomes more and more chaotic when r is increased beyond values of 12 and we observe random values of P for r $\geq$ 20.

In [ ]:
```
```