# 

# בניית מודלים מתמטיים – מודל העושר (מבוסס סוכנים)

מנדי פישמן

המחלקה למתמטיקה שימושית

המכללה האקדאמית בראודה להנדסה

מנחה פרופ' כתריאל חגי

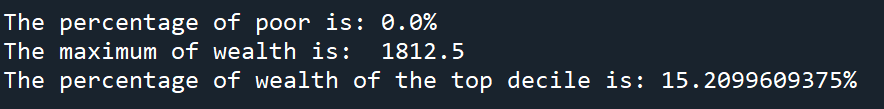
המחלקה למתמטיקה שימושית

המכללה האקדאמית בראודה להנדסה

חלק 1:

1. נבדוק את תוצאות המודל עבור מספר עסקאות שונה:

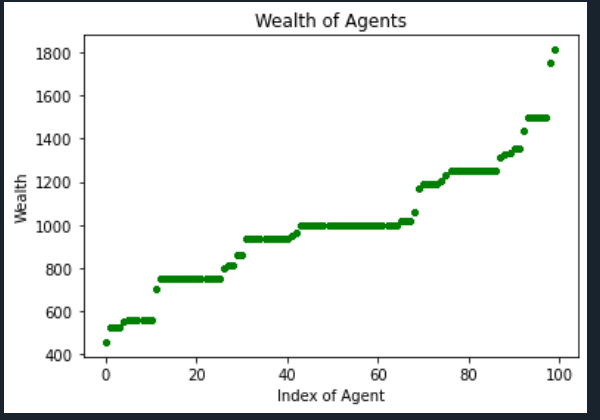
לאחר 1000 עסקאות קיבלנו את התוצאות הבאות:



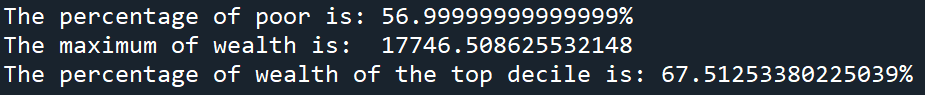
כלומר, אחוז העניים הוא 0%, האדם הכי עשיר הוא בעל 1812.5 ש"ח ואחוז העושר של העשירון העליון מכלל העושר הוא כ-15%.

\*עבור הרצות נוספות קיבלנו תוצאות שונות עקב הרנדומליות.

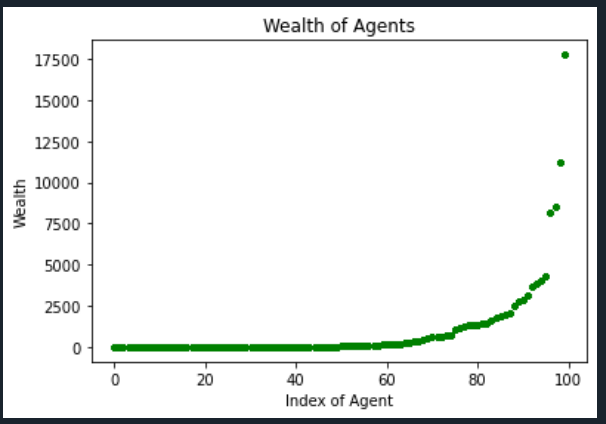
גרף של ווקטור העושר של האנשים:



לאחר 10000 עסקאות קיבלנו את התוצאות הבאות:



כלומר, אחוז העניים הוא כ- 57%, האדם הכי עשיר הוא בעל 17746.5 ש"ח ואחוז העושר של העשירון העליון מכלל העושר הוא כ-67.5%.

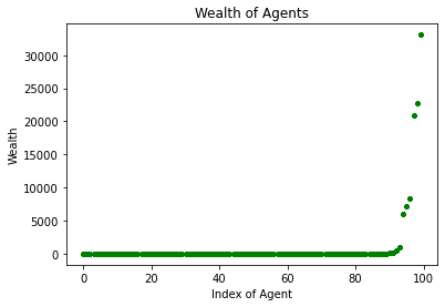


לאחר 100000 עסקאות קיבלנו את התוצאות הבאות:

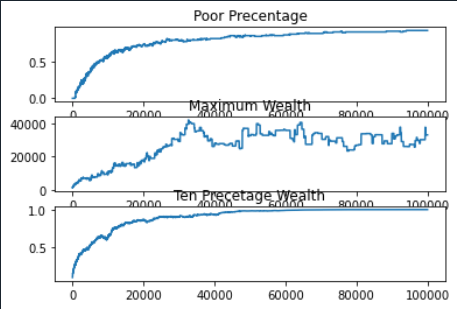
תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

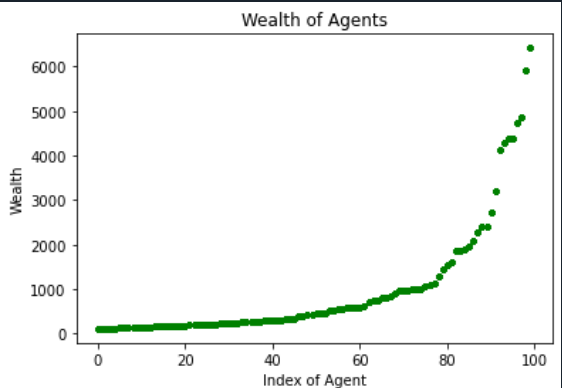
כלומר, אחוז העניים הוא כ- 92%, האדם הכי עשיר הוא בעל 33105 ש"ח ואחוז העושר של העשירון העליון מכלל העושר הוא כ-99.92%.



1. גרפים של שלושת המדדים במהלך 100,000 עסקאות:



1. המסקנות:
2. ניתן ללמוד מהגרפים כי רמת אי השוויון בחברה נמצאת במגמת עלייה באופן כללי.  
   ניתן לראות שככל שכמות העסקאות גדלות כך אחוז העניים בחברה גדל וכך גם אחוז העושר של העשירון העליו גדל.
3. נבחן מודל שבו קיים מיסוי (נבחן עבור 100000 עסקאות):



תמונה שמכילה טקסט, צילום מסך, קו, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

אחוז המס הוא 0.1%.

קיבלנו כי אחוז העניים הוא כ- 21%, האדם הכי עשיר הוא בעל 7024 ש"ח ואחוז העושר של העשירון העליון מכלל העושר הוא כ-49%.  
ניתן לראות כי מדדי אי השוויון ירדו באופן משמעותי למול המודל בו לא קיים מיסוי.

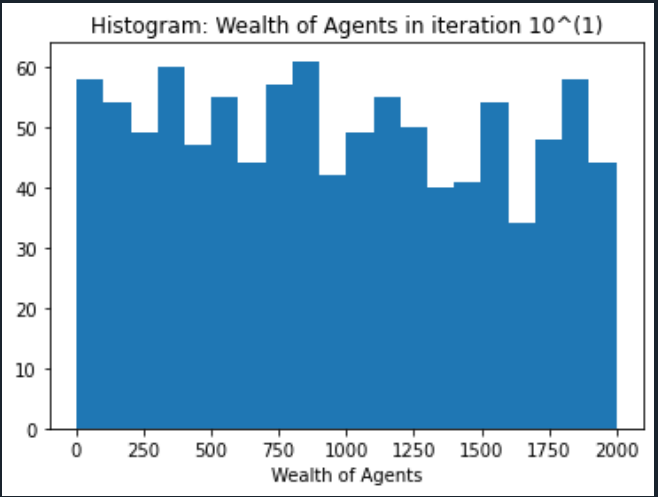
לסיכום, ניתן להסיק כי הטלת מס היא פעולה אשר יכולה לצמצם את אי השוויון.

חלק 2:

א. ההסיטוגרמות שקיבלנו:

\*כמות המחלקות בהיסטוגרמה הינו 20.

עבור k=1:



עבור k=2:

תמונה שמכילה טקסט, צילום מסך, עלילה, תרשים

התיאור נוצר באופן אוטומטי

עבור k=3:

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

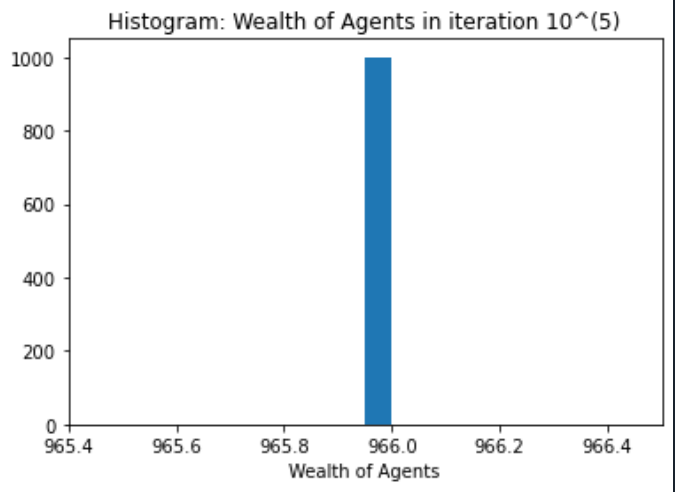
התיאור נוצר באופן אוטומטי

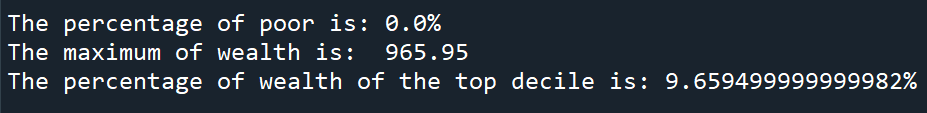
עבור k=4:

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

עבור k=5:



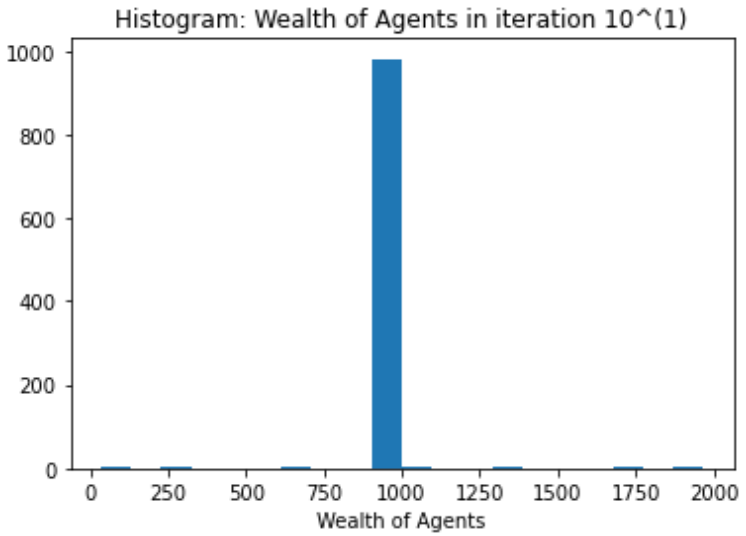


ב. רמת אי השוויון ירד, ואף ניתן לראות כי לבסוף כולם שווים. כמו גם, אפשר לראות שאכן כולם בעלי רמות עושר של 965.95 ש"ח. שכן זה מסתדר שהרי אנחנו מצפים שהגרלת הנתונים בהתחלה תהיה בעלת ממוצע 1000 שהרי זה הגרלת נקודות בהתפלגות אחידה.

\*בחלק זה אין

חלק 3:

עבור k=1:



עבור k=2:

תמונה שמכילה טקסט, צילום מסך, קו, עלילה

התיאור נוצר באופן אוטומטי

עבור k=3:

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

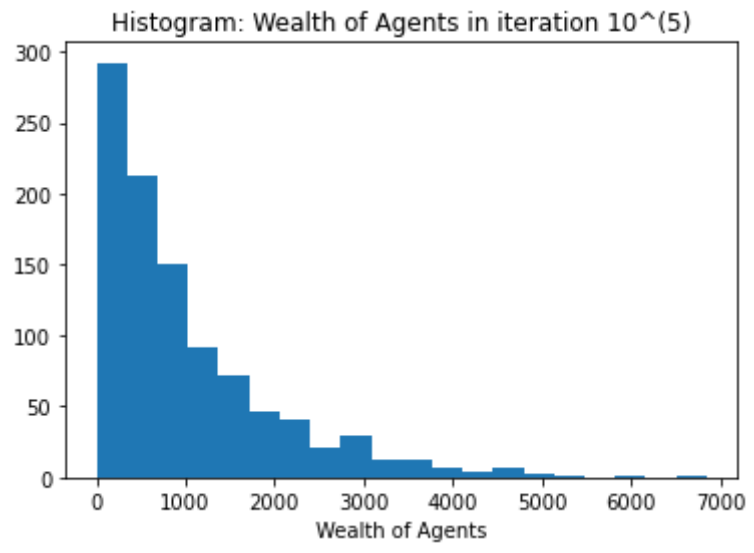
התיאור נוצר באופן אוטומטי

עבור k=4:

תמונה שמכילה טקסט, צילום מסך, עלילה, תרשים

התיאור נוצר באופן אוטומטי

עבור k=5:

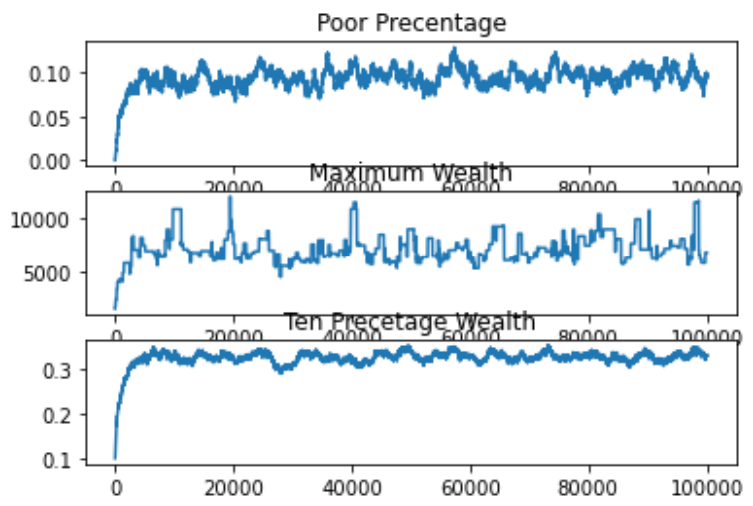


ב. ניתן לראות כי התפלגות שנוצרה דומה להתפלגות אקספוננציאלית, לעומת המודלים האחרים.

גרפים של מדדי העושר:

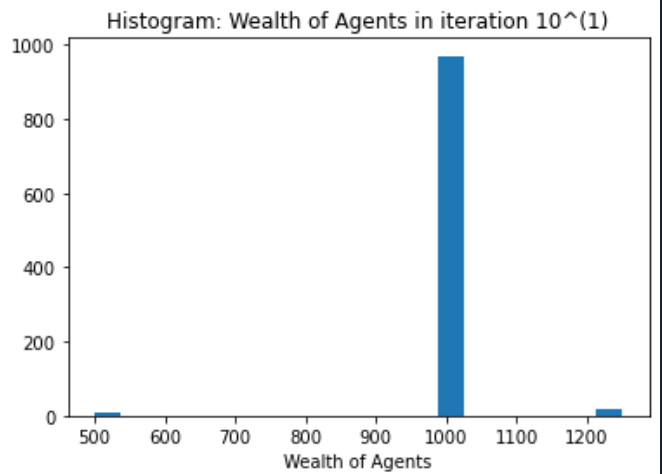
תמונה שמכילה טקסט, קו, עלילה, תרשים

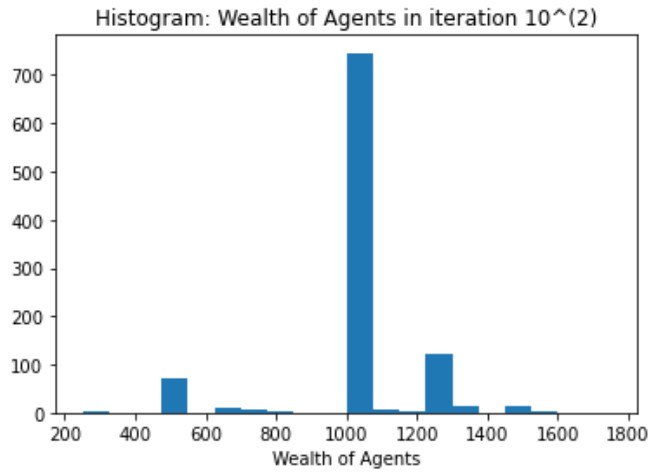
התיאור נוצר באופן אוטומטי



חלק 4:  
בחלק זה בחרנו לבנות מודל בו כל אינטראקציה בין סוכנים כלומר עסקה בין אנשים היא עסקה בין 3 אנשים כאשר מתוכם יש אדם אחד מפסיד ושניים מרוויחים.  
נעשה זאת באופן הבא – נניח שהמפסיד מפסיד מסכומו 50% אזי כל אחד משני המרוויחים (הסוכנים האחרים) מרוויח 25% מסכומו של המפסיד. (המודל הוא עבור 1000 סוכנים)

ההסיטוגרמות שקיבלנו:





תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

גרף מדד העושר של "הסוכנים":

תמונה שמכילה טקסט, קו, עלילה, תרשים

התיאור נוצר באופן אוטומטי

גרפים של המדדים:

תמונה שמכילה טקסט, צילום מסך, קו, גופן

התיאור נוצר באופן אוטומטי

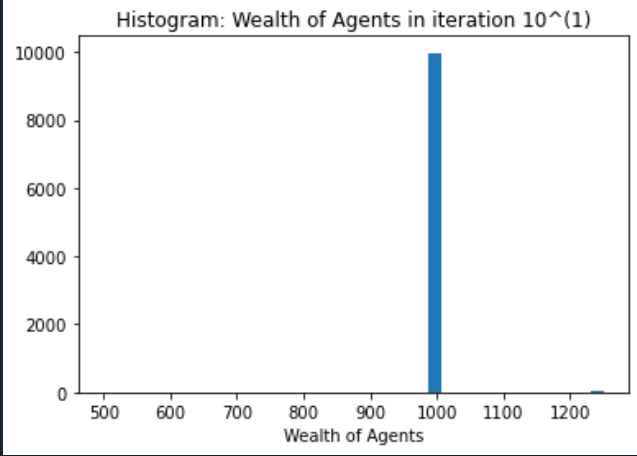
תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

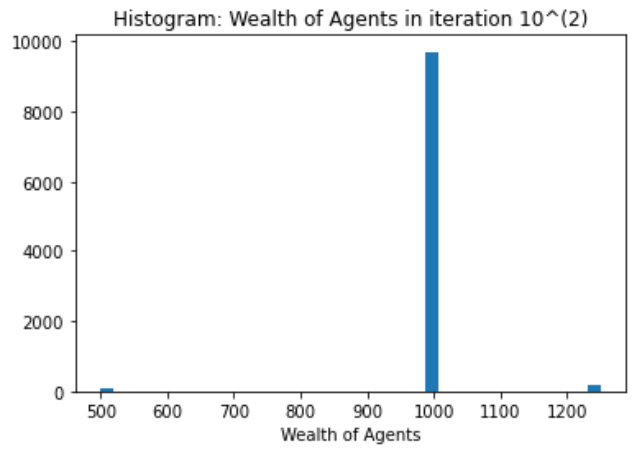
ניתוח התוצאות:  
1. קיבלנו כי אחוז העניים הוא מאוד נמוך, דבר שהיינו מצפים שיתקבל שהרי בכל עסקה יוצאים שני מרוויחים.   
2. קיבלנו כי העשירון העליון מחזיק כ-28% מכלל העושר וזה נוסע ככל הנראה מכך שבכל עסקה המרוויחים זוכים ל25% מסכומו של המפסיד דבר שמגדיל את עושרם באופן משמעותי.  
3. ניתן לראות כי הגרפים של המדדים אינם חלקים – אפשר להבין כי הפרמטרים של העסקה משנים את המדדים באופן קיצוני בין עסקה לעסקה.  
4. מהגרפים של ההסיטוגרמות ניתן לראות כי בהתחלה ישנה עלייה ולאחר מכן ירידה, לעומת הגרף שקיבלנו בחלק 3 שבו קיבלנו ירידה מההתחלה שדומה להתפלגות אקספוננציאלית.

חלק 4 עבור 10000 סוכנים ומיליון איטרציות: (ההתפלגות האחרונה מזכירה התפלגות ארלנג/גמא)

עבור k=1:



עבור k=2:



עבור k=3:

תמונה שמכילה טקסט, צילום מסך, קו, תרשים

התיאור נוצר באופן אוטומטי

עבור k=4:

תמונה שמכילה צילום מסך, טקסט, עלילה, תרשים

התיאור נוצר באופן אוטומטי

עבור k=5:

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

עבור k=6:

תמונה שמכילה טקסט, צילום מסך, תרשים, עלילה

התיאור נוצר באופן אוטומטי

הקוד:

import random  
import matplotlib.pyplot as plt  
  
def transaction(agent1,agent2,agents\_array,f):  
 x = random.randint(1,10)  
 **if** agents\_array[agent1]>=agents\_array[agent2]:  
 transaction\_size = f\*agents\_array[agent2]  
 **else**:  
 transaction\_size = f\*agents\_array[agent1]   
 **if** x>5:  
 agents\_array[agent1] = agents\_array[agent1] + transaction\_size  
 agents\_array[agent2] = agents\_array[agent2] - transaction\_size  
 **else**:  
 agents\_array[agent2] = agents\_array[agent2] + transaction\_size  
 agents\_array[agent1] = agents\_array[agent1] - transaction\_size  
 return agents\_array  
  
def transaction\_det\_col(agent1,agent2,agents\_array):  
 sum\_agents = agents\_array[agent1] + agents\_array[agent2];  
 avg\_agents = sum\_agents/2  
 agents\_array[agent1] = avg\_agents  
 agents\_array[agent2] = avg\_agents  
 return agents\_array  
  
def rand\_transaction(agent1,agent2,agents\_array):  
 r = random.random() #the ratio  
 sum\_agents = agents\_array[agent1] + agents\_array[agent2];  
 agents\_array[agent1] = r\*sum\_agents  
 agents\_array[agent2] = (1-r)\*sum\_agents  
 return agents\_array  
  
def agents(n,wealth,m,f,poor\_condition,tax\_bin,k,tax\_rate,transaction\_type,rng\_init\_wlth):  
 # n - number of agents,  
 # wealth - the firsy sum of each agent  
 # m - the number of iteration - transaction  
 # k - the num of iteration   
 t = 1 #for plotting the histogram each 10^t iteration  
   
 poor\_people\_precentage\_arr = m\*[0]  
 max\_wealth\_arr = m\*[0]  
 Precentage\_of\_largest\_ten\_precentage\_arr = m\*[0]  
   
 #define the init agents\_array according the type of transaction:  
 agents\_array\_initial = n\*[wealth]  
 **if** transaction\_type==2:  
 for i **in** range(0,n):  
 agents\_array\_initial[i] = random.randint(0,2000)   
 agents\_array = agents\_array\_initial  
  
 for i **in** range(0,m):  
 agent1 = 1  
 agent2 = 1  
 while agent1==agent2:  
 agent1 = random.randint(0,n-1)  
 agent2 = random.randint(0,n-1)  
 **if** transaction\_type == 1:  
 agents\_array = transaction(agent1,agent2,agents\_array,f)  
 elif transaction\_type == 2:  
 agents\_array = transaction\_det\_col(agent1,agent2,agents\_array)  
 **else**:  
 agents\_array = rand\_transaction(agent1,agent2,agents\_array)  
 poor\_people\_precentage,max\_wealth,Precentage\_of\_largest\_ten\_precentage = data(wealth,agents\_array,poor\_condition)   
 **if** tax\_bin == 1: #activate the tax function  
 **if** (i+1)/k == int((i+1)/k):  
 agents\_array = tax(agents\_array,tax\_rate)   
 #update the data arrays:  
 poor\_people\_precentage\_arr[i] = poor\_people\_precentage  
 max\_wealth\_arr[i] = max\_wealth  
 agents\_array.sort()  
 Precentage\_of\_largest\_ten\_precentage\_arr[i] = Precentage\_of\_largest\_ten\_precentage  
   
 #plot histogram:  
 **if** (i+1) == 10\*\*t:  
 plot\_histograms(agents\_array,t)  
 t += 1;  
   
 #print the consequences:  
 print("")  
 print("The agent array is: ",agents\_array)  
 print("")  
 print("The percentage of poor is: " + str(poor\_people\_precentage\*100) + "%")  
 print("The maximum of wealth is: ",max\_wealth)  
 print("The percentage of wealth of the top decile is: " + str(Precentage\_of\_largest\_ten\_precentage\*100) +"%")  
   
   
 return agents\_array,agents\_array\_initial, poor\_people\_precentage\_arr, max\_wealth\_arr, Precentage\_of\_largest\_ten\_precentage\_arr  
  
  
def plot\_agent\_arr(agents\_array):  
 index\_agent = list(range(len(agents\_array)))  
 for i **in** range(len(agents\_array)):  
 plt.scatter(index\_agent,agents\_array, color='g', marker = '.')  
 #if we want only points we need to write - plt.scatter  
 plt.xlabel("Index of Agent")  
 plt.ylabel("Wealth")  
 plt.title("Wealth of Agents")  
 plt.show()  
   
def data(init\_wealth,agents\_array,poor\_condition):  
 #check how many poor people:  
 n = len(agents\_array)  
 poor\_people\_amount = 0  
 ten\_precentage = int(n/10)  
 sum\_largest\_ten\_precentage = 0  
 for i **in** range(0,n):  
 **if** agents\_array[i] < poor\_condition:  
 poor\_people\_amount+=1  
 poor\_people\_precentage = poor\_people\_amount/n  
 for i **in** range(0,ten\_precentage):  
 sum\_largest\_ten\_precentage = sum\_largest\_ten\_precentage + agents\_array[n-i-1]  
 Precentage\_of\_largest\_ten\_precentage = sum\_largest\_ten\_precentage/(init\_wealth\*n)  
 max\_wealth = max(agents\_array)  
 return poor\_people\_precentage,max\_wealth,Precentage\_of\_largest\_ten\_precentage  
   
  
#plotting:   
   
def plot\_data\_array(arr\_poor,arr\_max, arr\_ten\_lrgst):  
 # Initialise the subplot function using number of rows and columns  
   
 Iteration\_vec = list(range(len(arr\_ten\_lrgst)))  
   
 figure, axis = plt.subplots(3)  
   
 # For Poor Precentage  
 axis[0].plot(Iteration\_vec, arr\_poor)  
 axis[0].set\_title("Poor Precentage")  
   
 # For Maximum Wealth  
 axis[1].plot(Iteration\_vec, arr\_max)  
 axis[1].set\_title("Maximum Wealth")  
   
 # For Ten Precetage Wealth  
 axis[2].plot(Iteration\_vec, arr\_ten\_lrgst)  
 axis[2].set\_title("Ten Precetage Wealth")  
   
 # Combine all the operations and display  
 plt.show()   
  
def plot\_histograms(agents\_array,t):  
 n = len(agents\_array)   
 n1 = int(n)  
 plt.hist(agents\_array, bins=n1)  
 plt.xlabel("Wealth of Agents")  
 plt.title("Histogram: Wealth of Agents in iteration 10^(" +str(t) +")")  
 plt.show()  
  
  
#Model with tax:  
def tax(agents\_array,tax\_rate):  
 total\_tax = 0  
 n = len(agents\_array)  
 for i **in** range(0,n):  
 total\_tax += tax\_rate\*agents\_array[i]  
 agents\_array[i] = agents\_array[i] - tax\_rate\*agents\_array[i]  
 per\_tax = total\_tax/n  
 for i **in** range(0,n):  
 agents\_array[i] += per\_tax  
 return agents\_array  
  
def main():  
   
 #Global input:  
 poor\_condition = 100 #the condition that defines the poor people  
 init\_wealth = 1000 #initial wealth of each agent   
 iteration\_num = 1000 #number of transactions  
 n = 100 #number of agents  
   
 #input according the transaction model:  
 #Model 1:  
 f = 0.25 #the size of transaction (Precentage)  
   
 #tax:  
 tax\_bin = 0 #with tax -> 1 else no tax  
 k = 1000 #the num of iteration to take a tax of the agents  
   
 #Model 2: sum of the agent divide equally  
 tax\_rate = 0.1  
 rng\_init\_wlth = 2000 #set the init range  
   
 #Model 3:  
 #just choose transaction\_type = 3  
   
 #choose the Model:  
 transaction\_type = 1 #1,2,3  
   
   
 #activate the program:  
 agents\_array, agents\_array\_initial,arr\_poor, arr\_max, arr\_ten\_lrgst = agents(n,init\_wealth,iteration\_num,f,poor\_condition,tax\_bin,k,tax\_rate,transaction\_type,rng\_init\_wlth)  
 agents\_array.sort()  
 plot\_agent\_arr(agents\_array)  
 data(init\_wealth,agents\_array,poor\_condition)  
 plot\_data\_array(arr\_poor,arr\_max, arr\_ten\_lrgst)  
  
  
   
main()