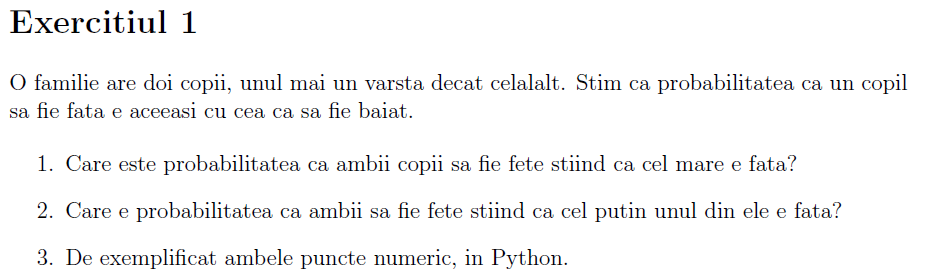
**Tema 1**



Ω = {FF, FB, BB, BF}

P(FF) = P(FB) = P(BB) = P(BF) =

1. FF = ambii copii sunt fete
   1. FB U FF = copilul cel mare sa fie fata
   2. P(FF | FB U FF) = = = =
2. FF U FB U BF = cel putin unul din copii sa fie fata
   1. P(FF U FB U BF) =
   2. P(FF | FF U FB U BF ) = = = =

import numpy as np  
import matplotlib.pyplot as plt  
  
def exemplifica1(N):  
 # 0 = baiat, 1 = fata  
 copil1 = np.random.randint(2, size=N)  
 copil2 = np.random.randint(2, size=N)  
  
 # probabilitatea ca ambii copii sa fie fete stiind ca cel mare e fata  
 primul\_fata = sum(copil1)  
 amadoi\_fete = 0  
 P\_FF = [0 for x in range(N)]  
 P\_F = [0 for x in range(N)]  
 for i in range(N):  
 if i != 0:  
 P\_FF[i] = P\_FF[i-1]  
 P\_F[i] = P\_F[i-1]  
  
 if copil1[i] == copil2[i] and copil1[i] == 1:  
 amadoi\_fete += 1  
 P\_FF[i] += 1  
  
 if copil1[i] == 1:  
 P\_F[i] += 1  
  
 # probabilitatea  
 p = amadoi\_fete / primul\_fata  
 print(f"Probabilitatea ca ambii copii sa fie fete stiind ca cel mare e fata este {p}")  
  
 # grafic  
 # Pentru a trata cazurile de impartire la 0 voi seta toate zerourile din vectorul la care se imparte la 1 si  
 # in vectorul din care se imparte voi seta la 0 ca sa dea probabilitatea 0  
 for i in range(N):  
 if P\_F[i] == 0:  
 P\_F[i] = 1  
 P\_FF[i] = 0  
 P = np.divide(P\_FF, P\_F)  
 plt.plot(P)  
 plt.show()

def exemplifica2(N):  
 # 0 = baiat, 1 = fata  
 copil1 = np.random.randint(2, size=N)  
 copil2 = np.random.randint(2, size=N)  
  
 # probabilitatea probabilitatea ca ambii sa fie fete stiind ca cel putin

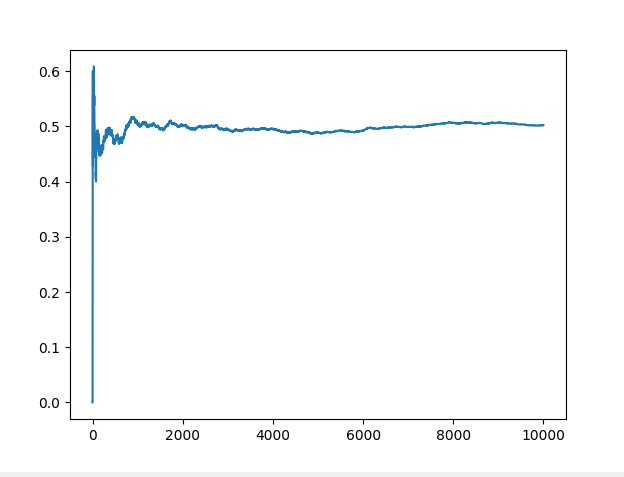
# unul din ele e fata  
 macar\_unul\_fata = 0  
 amadoi\_fete = 0  
 P\_FF = [0 for x in range(N)]  
 P\_F = [0 for x in range(N)]  
 for i in range(N):  
 if i != 0:  
 P\_FF[i] = P\_FF[i-1]  
 P\_F[i] = P\_F[i-1]  
  
 if copil1[i] == 1 or copil2[i] == 1:  
 macar\_unul\_fata += 1  
 P\_F[i] += 1  
 if copil1[i] == copil2[i] and copil1[i] == 1:  
 amadoi\_fete += 1  
 P\_FF[i] += 1  
  
 # probabilitatea  
 p = amadoi\_fete / macar\_unul\_fata  
 print(f"Probabilitatea ca ambii copii sa fie fete stiind stiind ca cel putin unul din ele e fata {p}")  
  
 # grafic  
 # Pentru a trata cazurile de impartire la 0 voi seta toate zerourile din

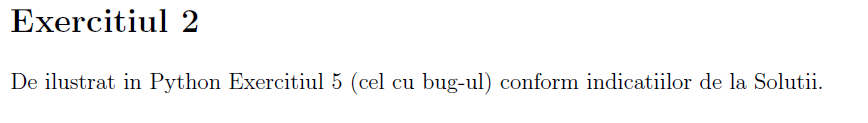
# vectorul la care se imparte la 1 si in vectorul din care se imparte voi

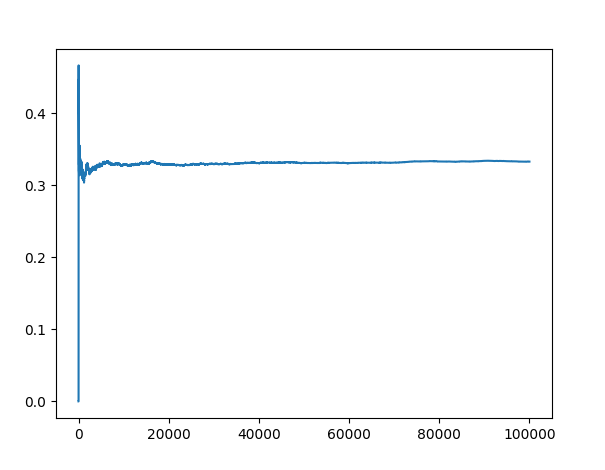
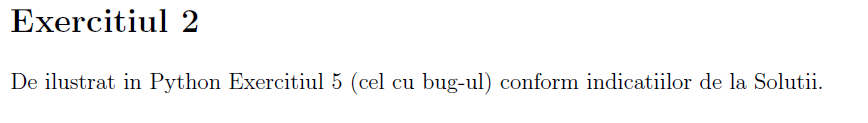
# seta la 0 ca sa dea probabilitatea 0  
 for i in range(N):  
 if P\_F[i] == 0:  
 P\_F[i] = 1  
 P\_FF[i] = 0  
 P = np.divide(P\_FF, P\_F)  
 plt.plot(P)  
 plt.show()

def main():  
 N = int(input("Introduceti numarul de experimente = "))  
 # exemplifica1(N)  
 exemplifica2(N)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

* grafic de la exemplificarea punctului 1:



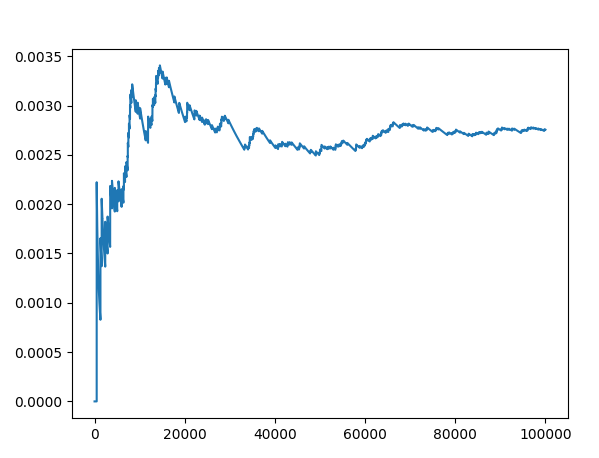
* grafic de la exemplificarea punctului 2:

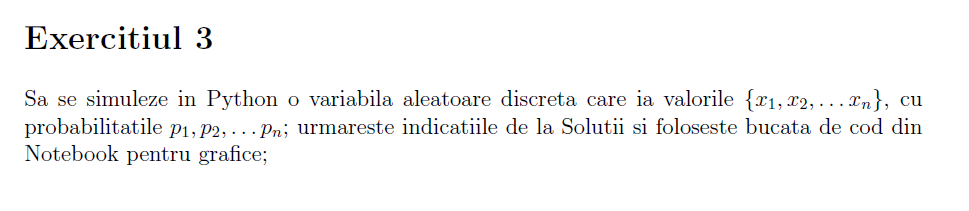


import numpy as np  
import matplotlib.pyplot as plt  
  
def generare(N):  
  
 v = [0 for x in range(N)]  
 for i in range(N):  
 v[i] = np.random.choice([0, 1], p=[0.95, 0.05])  
  
 perechi = 0  
 nr\_zerouri = 0  
 P\_P = [0 for x in range(N)]  
 P\_Z = [0 for x in range(N)]  
 for i in range(N):  
 if i != 0:  
 P\_P[i] = P\_P[i - 1]  
 P\_Z[i] = P\_Z[i - 1]  
  
 if v[i] == 1:  
 m = np.random.choice([0, 1], p=[0.05, 0.95])  
 else:  
 m = np.random.choice([0, 1], p=[0.98, 0.02])  
  
 if v[i] == 1 and m == 0:  
 perechi += 1  
 P\_P[i] += 1  
  
 if m == 0:  
 nr\_zerouri += 1  
 P\_Z[i] += 1  
  
 # probabilitatea  
 p = perechi / nr\_zerouri  
 print(f"Probabilitatea ca Mihai sa spuna ca nu are bug un cod care are este {p}")  
  
 # grafic  
 # Pentru a trata cazurile de impartire la 0 voi seta toate zerourile din

# vectorul la care se imparte la 1 si in vectorul din care se imparte voi

# seta la 0 ca sa dea probabilitatea 0  
 for i in range(N):  
 if P\_Z[i] == 0:  
 P\_Z[i] = 1  
 P\_P[i] = 0  
 P = np.divide(P\_P, P\_Z)  
 plt.plot(P)  
 plt.show()  
  
def main():  
 N = 100000  
 generare(N)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()





import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.cm as cm  
  
  
def discrete(N, values, probabilities):  
  
 v = np.random.uniform(0, 1, size=N)  
 X = [0 for x in range(N)]  
 for i in range(len(v)):  
 right = 0  
 for j in range(len(probabilities)):  
 right += probabilities[j]  
 if v[i] <= right:  
 X[i] = values[j]  
 break  
  
 fig, ax = plt.subplots()  
 cmap = cm.get\_cmap(name='rainbow')  
 for i in range(len(probabilities)):  
 nr\_fav = [list(X[:j]).count(values[i]) for j in range(1, N+1)]  
 P = np.divide(nr\_fav, range(1, N+1))  
  
 ax.annotate(' p' + str(i) + "= " + str(probabilities[i]),  
 xy=(10000, probabilities[i]),  
 xytext=(10000, probabilities[i]),  
 arrowprops=dict(facecolor='black', shrink=0.1, width=1))  
 ax.plot(range(N), P, color=cmap(i))  
  
 plt.show()  
  
  
def main():  
 N = 10000  
 values = [1, 2, 3, 4]  
 probabilities = [0.2, 0.3, 0.4, 0.1]  
 discrete(N, values, probabilities)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

