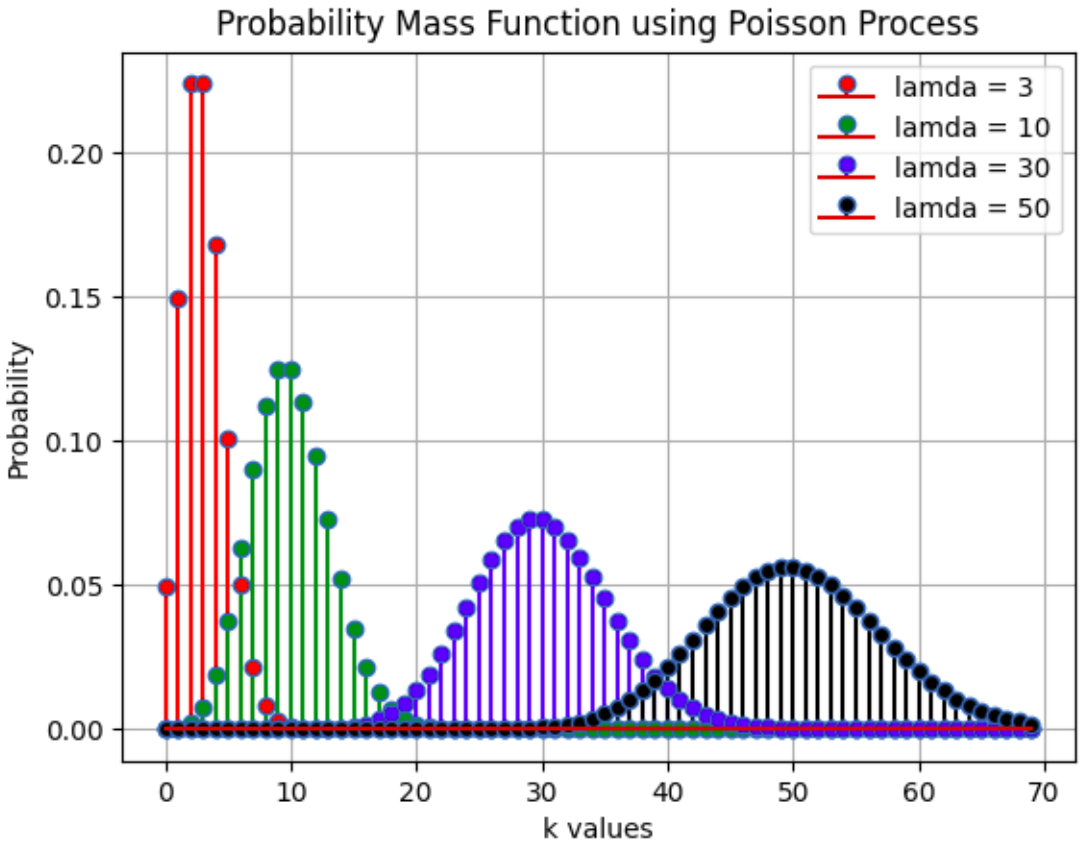
Συστήματα Αναμονής (Queuing Systems)

1ο Εργαστήριο
Θοδωρής Παπαρρηγόπουλος (el18040)

Γενικά για την εργασία αυτή χρησιμοποίησα python3.6. Επίσης, για να τρέξουν οι κώδικες πρέπει να είναι κατεβασμένα τα πακέτα της matplotlib, του scipy και του numpy.

Κατανομή Poisson

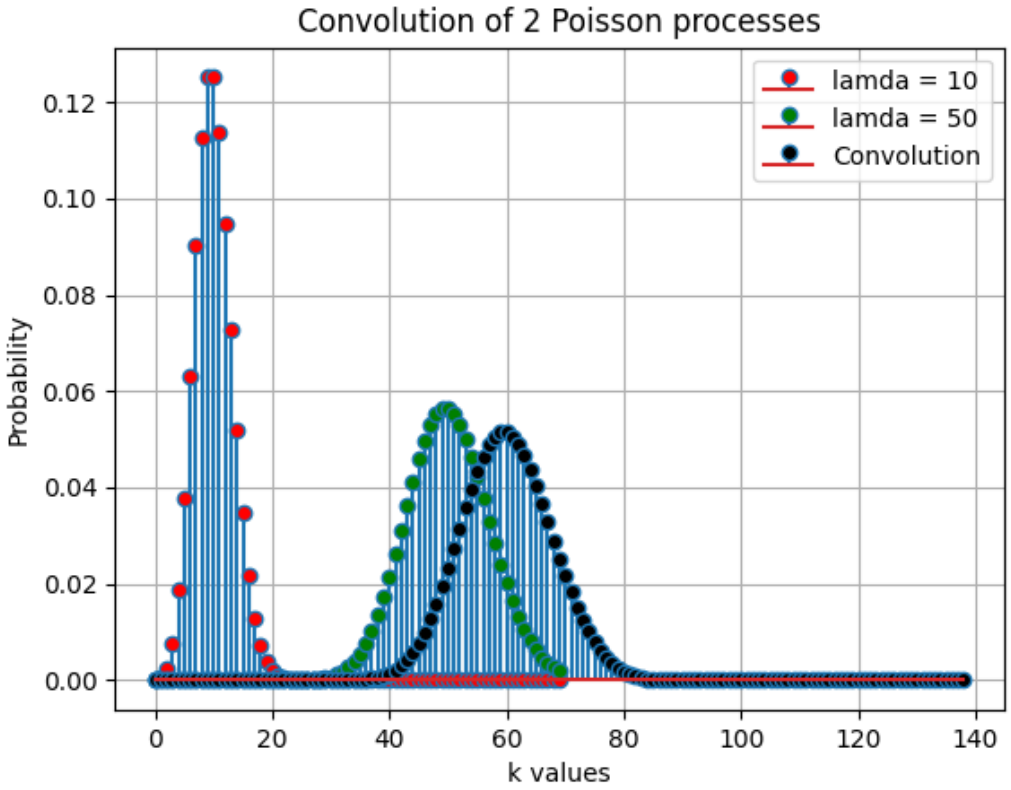
Α) Στο παρακάτω διάγραμμα παρατηρούμε πως όσο μεγαλώνει το λ τόσο μεγαλώνει ο μέση όρος και το variance. Για μια ακολουθία a_n *τότε*, $Mean = \frac{\sum_{k=1}^{N} a_k}{N}$ όπως και $\sigma^2 = \frac{\sum_{k=1}^{N} (a_k - Mean)^2}{N}$.



Β) Παρατηρούμε παρακάτω τα means και τα variance για τα διάφορα λ = 3, 10, 30, 50:

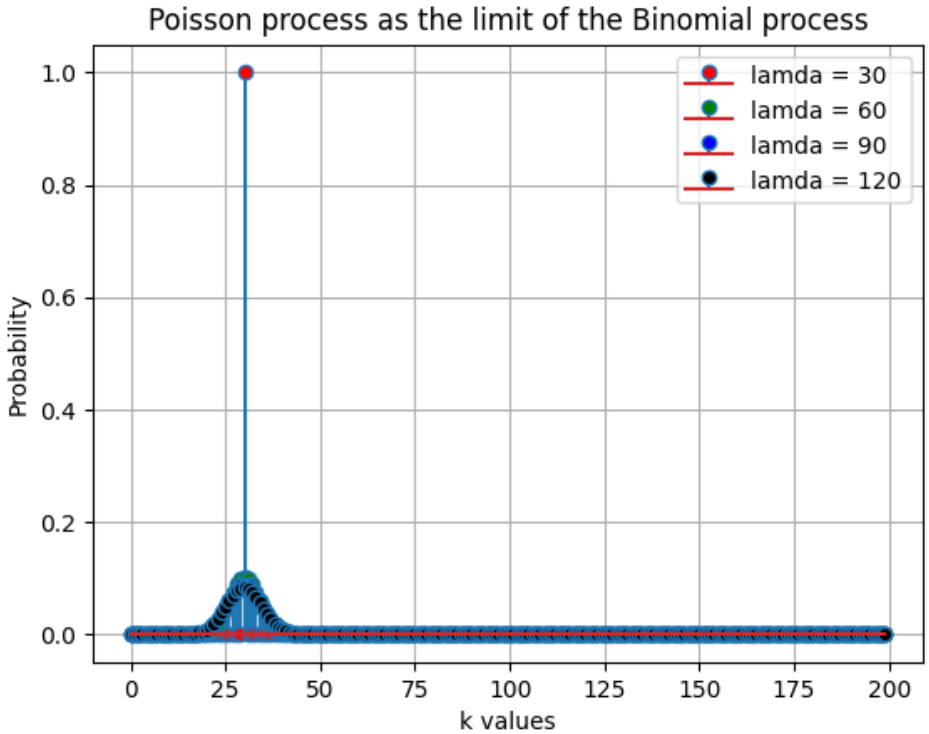
For lamda = 3 , we have mean = 3.0 and variance = 3.0
For lamda = 10 , we have mean = 10.0 and variance = 10.0
For lamda = 30 , we have mean = 30.0 and variance = 30.0
For lamda = 50 , we have mean = 50.0 and variance = 50.0

Γ) Παρακάτω παρατηρούμε πως η συνέλευση 2 poisson distributions είναι επίσης μια poisson distribution, το οποίο επαληθεύει την θεωρία μας. Επιπλέον, γνωρίζουμε πως $\lambda_{convolution} = 10+50 = 60$. Η απαραίτητη προϋπόθεση είναι να είναι και οι άλλες 2 άλλα κατανομές poisson και ανεξάρτητες.



Δ) Σε θεωρητικό υπόβαθρό ισχύει πως $Poisson = P[X \le k] = \frac{(\lambda t)^k}{k!} e^{-\lambda t} = \lim_{n \to \infty} \binom{n}{k} p^k (1-p)^k$.

Φαίνεται από το παρακάτω διάγραμμα πως η Poisson είναι απλά μια ειδική περίπτωση της διωνυμικής όπου παίρνουμε μεγάλο αριθμό δοκιμών.



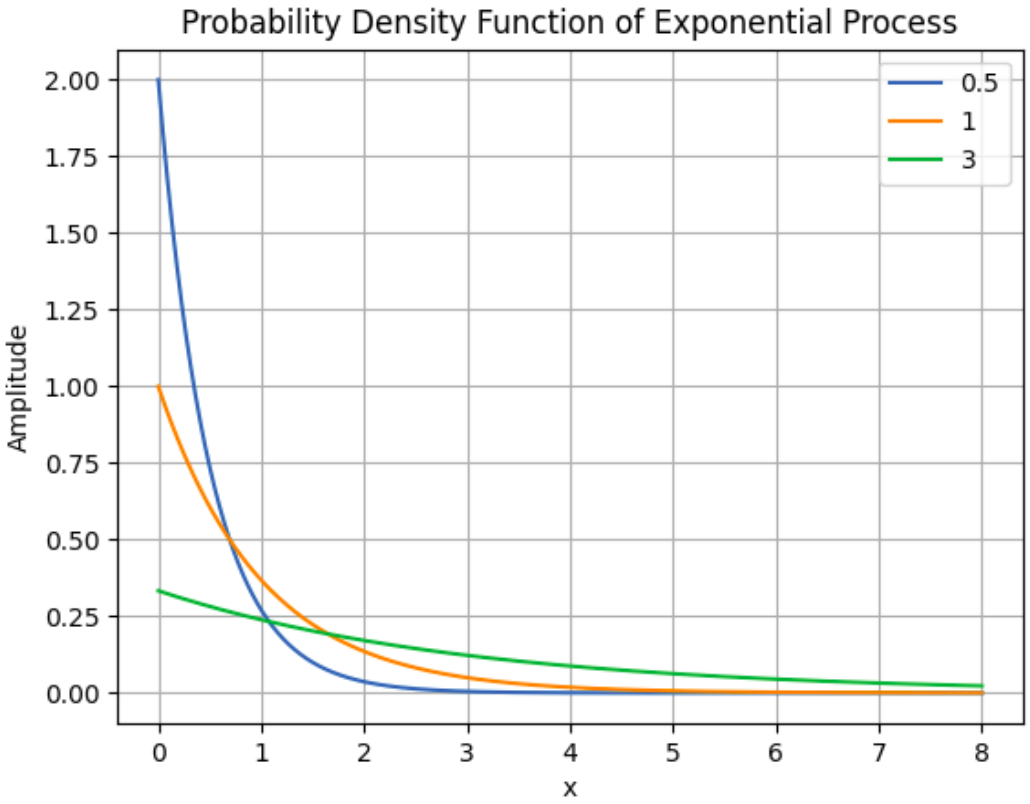
```
import numpy as np
from scipy.stats import binom
from scipy.stats import poisson
import matplotlib.pyplot as plt

figure_counter = -1

figure_titles = [
	"Probability Mass Function using Poisson Process",
	"Convolution of 2 Poisson processes",
	"Poisson process as the limit of the Binomial process",
]

labelTuple = [
	("k values","Probability"),
	("k values","Probability"),
	("k values","Probability"),
]



################################
# A                            #
################################

lamdas = [3,10,30,50]
axis = np.arange(70)

figure_counter += 1
plt.figure(figure_counter)
plt.title(figure_titles[figure_counter])
colors = ['red','green','blue','black']
for i, lamda in enumerate(lamdas):
	markerline, stemlines, baseline = plt.stem(axis, poisson.pmf(axis, lamda), colors[i] ,
use_line_collection=True,label="lamda = " + str(lamda) )
	markerline.set_markerfacecolor(colors[i])


xlabel, ylabel = labelTuple[figure_counter]
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid()
plt.legend()
plt.savefig("../figures/PoissonDistribution/" + str(figure_counter) + ".png")

################################
# B                            #
################################

for lamda in lamdas:
	mean,var ,skew, kurt = poisson.stats(lamda,moments='mvsk')
	print ("For lamda = ", lamda, ", we have mean = ", mean, " and variance = ", var)
```

```
################################
# C                              #
################################

lamdas = [10,50]

conv = np.convolve(poisson.pmf(axis, lamdas[0]),poisson.pmf(axis, lamdas[1]))

figure_counter += 1
plt.figure(figure_counter)
plt.title(figure_titles[figure_counter])

for i, lamda in enumerate(lamdas):
        markerline, stemlines, baseline = plt.stem(axis, poisson.pmf(axis, lamda), label="lamda = "
+ str(lamda) , use_line_collection=True)
        markerline.set_markerfacecolor(colors[i])


        xlabel, ylabel = labelTuple[figure_counter]
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)


markerline, stemlines, baseline = plt.stem(np.arange(len(conv)),conv, label="Convolution" ,
use_line_collection=True)
markerline.set_markerfacecolor(colors[3])
xlabel, ylabel = labelTuple[figure_counter]
plt.xlabel(xlabel)
plt.ylabel(ylabel)

plt.grid()
plt.legend()
plt.savefig("../figures/PoissonDistribution/" + str(figure_counter) + ".png")


################################
# D                              #
################################

axis = np.arange(200)
lamdas = [30,60,90,120]
P = [30/lamda for lamda in lamdas]

figure_counter += 1
plt.figure(figure_counter)
plt.title(figure_titles[figure_counter])

for i, p in enumerate(P):
```

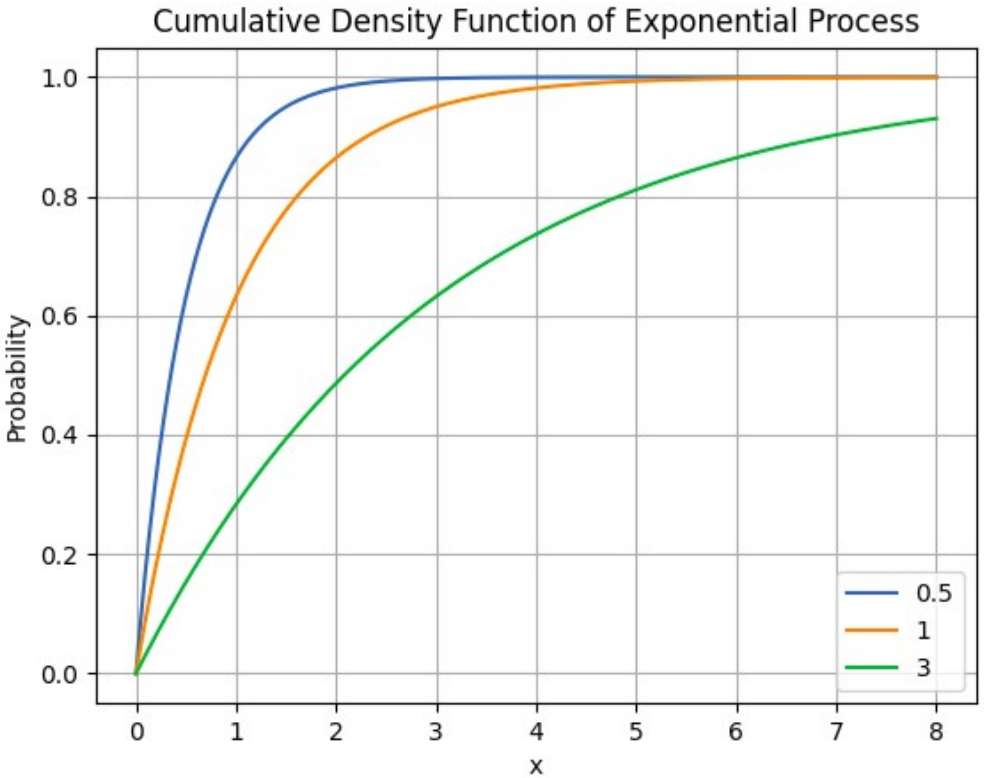
```
        markerline, stemlines, baseline = plt.stem(axis, binom.pmf(axis,lamdas[i],p),label=('lamda =
'+ str(lamdas[i])),use_line_collection=True)
        markerline.set_markerfacecolor(colors[i])

        xlabel, ylabel = labelTuple[figure_counter]
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)

plt.grid()
plt.legend()
plt.savefig("../figures/PoissonDistribution/" + str(figure_counter) + ".png")




plt.show()
```

Εκθετική Κατανομή

A) Για $\frac{1}{\lambda} = \{ 0.5,1,3 \}$



B) Για $\frac{1}{\lambda} = \{ 0.5,1,3 \}$



Γ)

$$P(X>t+s|X>s)=\frac{P((X>t+s)\wedge(X>s))}{P(X>s)}=\frac{P(X>t+s)}{P(X>s)}=P(X>t)$$

$$P(X>50000|X>20000)=P(X>30000+20000|X>20000).$$

Συνεπώς $P(X>50000|X>20000)=P(X>30000)$.

$$P(X>a)=1-P(X\le a)=1-CDF(a)\Rightarrow P(X>30000)=1-CDF(30000)$$

Προκύπτει :

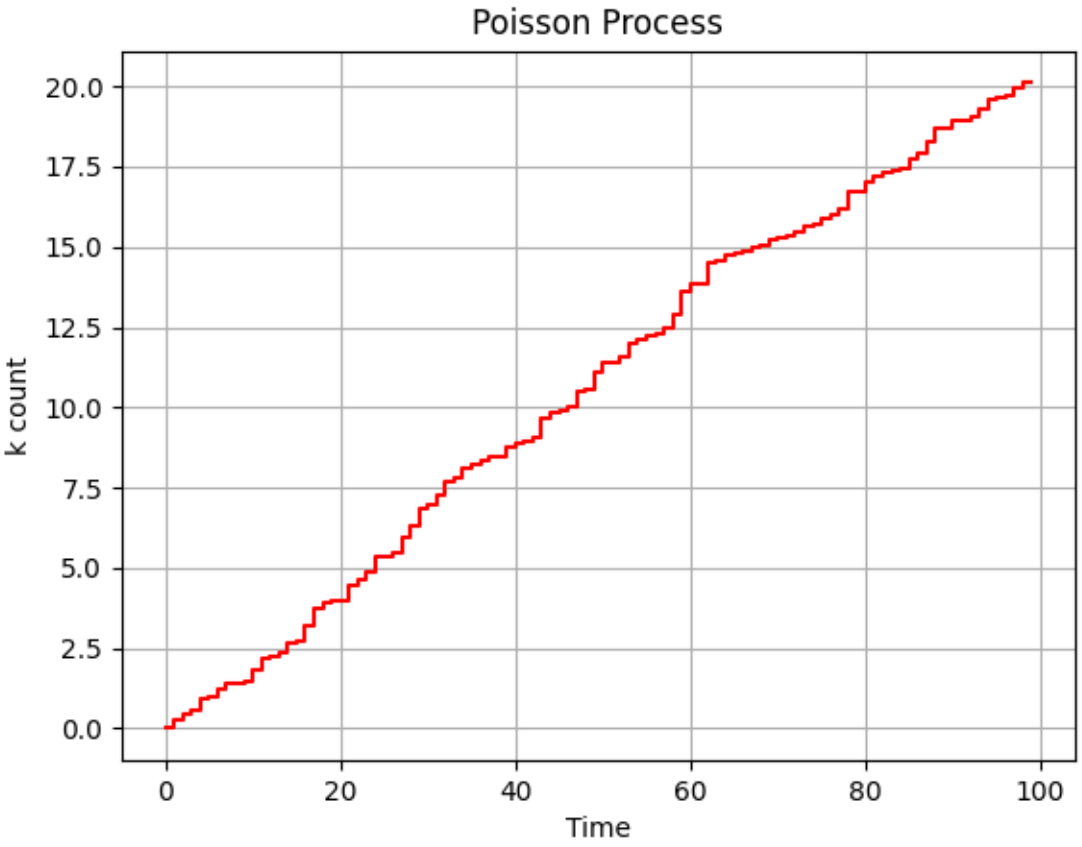
- Pr(X>30.000) = 0.8869204367171575
- Pr(X>50.000 | X>20.000) = 0.8869204367171575
  Pr(X>50.000) = 0.8187307530779818
  Pr(X>20.000) = 0.9231163463866358

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon


figure_counter = -1

figure_titles = [
        "Probability Density Function of Exponential Process",
        "Cumulative Density Function of Exponential Process",
]

labelTuple = [
        ("k values","Probability"),
        ("k values","Probability"),
]

################################
# A                            #
################################

colors = ['red','green','blue','black']
inverse_lamdas = [0.5,1,3]
axis = np.arange(0,8,0.00001)

figure_counter += 1
plt.figure(figure_counter)
plt.title(figure_titles[figure_counter])
for i, lamda in enumerate(inverse_lamdas):
        plt.plot(axis, expon.pdf(axis, 0,lamda), label=str(lamda))


xlabel, ylabel = labelTuple[figure_counter]
plt.xlabel(xlabel)
```

```
plt.ylabel(ylabel)
plt.grid()
plt.legend()
plt.savefig("../figures/ExponentialDistribution/" + str(figure_counter) + ".png")

################################
# B                            #
################################

figure_counter += 1
plt.figure(figure_counter)
plt.title(figure_titles[figure_counter])
for i, lamda in enumerate(inverse_lamdas):
        plt.plot(axis, expon.cdf(axis, 0,lamda), label=str(lamda))

xlabel, ylabel = labelTuple[figure_counter]
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid()
plt.legend()
plt.savefig("../figures/ExponentialDistribution/" + str(figure_counter) + ".png")


################################
# C                            #
################################

print ("Pr(X>30.000) = ", 1 - expon.cdf(axis[30000],0,2.5))
print ("Pr(X>50.000) = ", 1 - expon.cdf(axis[50000],0,2.5))
print ("Pr(X>20.000) = ", 1 - expon.cdf(axis[20000],0,2.5))
print ("Pr(X>50.000 | X>20.000) = ",(1 - expon.cdf(axis[50000],0,2.5))/(1-
expon.cdf(axis[20000],0,2.5)))


plt.show()
```

Διαδικασία Poisson

Α) Μεταξύ 2 διαδοχικών γεγονότων ακολυθείτε επίσης κατανομή Poisson



Β) Σε παράθυρο $dT = t_1 - t_2$ ο αριθμός των γεγονότων ακολουθεί κατανομή Poisson με μέση τιμή λt. Προκύπτει:

For Lambda = 200, mean is 4.725
For Lambda = 300, mean is 4.763333333333334
For Lambda = 500, mean is 5.06
For Lambda = 1000, mean is 4.918
For Lambda = 10000, mean is 4.9839

```
import numpy as np
import matplotlib.pyplot as plt


figure_counter = -1

figure_titles = [
        "Poisson Process"
]

labelTuple = [
```

```
        ("Time","k count")
]



################################
# A                            #
################################

lamda = 5

grid = np.random.exponential(1/lamda,100)
#print ("This is MEAN",np.mean(grid))
grid = [sum(grid[0:i]) for i in range(100)]


figure_counter += 1
plt.figure(figure_counter)
plt.title(figure_titles[figure_counter])


plt.step(range(100), grid,label = "Poisson Process Counting", color = "red")


xlabel, ylabel = labelTuple[figure_counter]
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid()
plt.savefig("../figures/PoissonProcesses/" + str(figure_counter) + ".png")
plt.show()


################################
# B                            #
################################

for i in [2,3,5,10,100]:
    grid = np.random.poisson(lamda,i*100)
    print ("For Lambda = {}, mean is {}".format(i*100,np.mean(grid)))
```