events and other periods wind Noting that to actually see the is the percieving of the photon	th long gaps. he flash of light, a certain th	hreshold number of photo	er in appearance. The randtom a exponential distribution on a time		
mu = lambd * T	od, T):  obability for n events  /(np.math.factorial(n))				
return poisson  # Let's plot the pdf f lam_rate = 40  # even T = 0.1  # 100 n_values = np.arange(0 pdf_vals = [poisson_pd	or n = 0 to 15 for a gr ts per second, say msec , 16)	iven rate.	s]		
<pre>plt.figure(figsize=(8, plt.bar(n_values, pdf_ plt.xlabel('Number of ) plt.ylabel('Probability plt.title('Poisson PDF plt.xticks(n_values) plt.show()</pre>	<pre>vals, color='skyblue') photons (n)') y')</pre>				
<pre>C:\Users\Ryan\AppData\Lo brary `math` module (Dep poisson = ((mu**n)/(np)  0.200 - 0.175 -</pre>	orecated Numpy 1.25). Fo.math.factorial(n))) *	Replace usages of `np	p.math` with `math`	o.math` is a deprec	ated alias for the
0.150 - Application of the control					
0.050 - 0.025 - 0.000 0 1		7 8 9 10 13 r of photons (n)	1 12 13 14 15		
<pre># And for fun lets jus lam_rate = 80  # even T = 0.1  # 100 n_values = np.arange(0 pdf_vals = [poisson_pd  plt.figure(figsize=(8,</pre>	ts per second, say msec , 16) f(n, lambd=lam_rate, T=			as much.	
<pre>plt.bar(n_values, pdf_) plt.xlabel('Number of ) plt.ylabel('Probability plt.title('Poisson PDF plt.xticks(n_values) plt.show()  C:\Users\Ryan\AppData\Lob brary `math` module (Deg</pre>	<pre>photons (n)') y') for λ=40, T=0.1 sec')  pcal\Temp\ipykernel_560 precated Numpy 1.25). F</pre>	Replace usages of `np		o.math` is a deprec	ated alias for the
<pre>poisson = ((mu**n)/(np  0.14 -  0.12 -  0.10 -</pre>		* np.exp(-mu) or λ=40, T=0.1 sec			
Probability - 80.0 - 90.0 - 90.0					
0.02 - 0.00 0 1 2  As you can observe, the me	Number	7 8 9 10 11 of photons (n)	12 13 14 15 ch photons.		
<pre>def detectionprob(K, 1)     """     Probability of det     """     mu = lambd * T</pre>					
<pre>mu = lambd * T     return 1-poisson_p  # Let's plot the detec K_values = np.arange(0 det_prob_values = [detection] plt.figure(figsize=(8, plt.bar(K_values, det_plt.xlabel('Detection))</pre>	tion probability as a section probability as a section prob (K, lambd=40,4)) prob_values, color='sas	<pre>, T=0.1) for K in K</pre>			
<pre>plt.ylabel('Detection p plt.title('Detection p plt.xticks(K_values) plt.show()  C:\Users\Ryan\AppData\Lo brary `math` module (Dep poisson = ((mu**n)/(np)</pre>	probability') robability vs threshold  pcal\Temp\ipykernel_560  precated Numpy 1.25). Formath.factorial(n))) *	d (λ=40, T=0.1 sec)') 00\1394053765.py:7: I Replace usages of `np * np.exp(-mu)	DeprecationWarning: `npp.math` with `math`	o.math` is a deprec	ated alias for the
0.8 -	ction probability vs t	threshold (λ=40, T=	=0.1 sec)		
Detection probability 7.0 0.0 - + 9.0					
3a	3 4 5 6 7 Detection thres	8 9 10 11 shold K (photons)	12 13 14 15		
	t1=0.8, t2=2.2):  process with rate `lar  te (events per unit tir	me)	rval [t1, t2].		
t1, t2 - Start and	end time of the proces				
<pre>t1, t2 - Start and """  T = t2 - t1 # Dur  # Sample the numbe</pre>	ation of process  r of events from a Pois son(lambd * T) # Numbe event times				
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(event)  def plotflash(photon_t """ Plot photon arriva	r of events from a Poisson(lambd * T) # Numberevent times times(N, t1, t2) ent times nt_times) imes, ax):  l times as vertical limes	er of events			
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(event  def plotflash(photon_t """  Plot photon arriva  photon_times: array ax: matplotlib axi """ ax.eventplot(photon ax.set_ylabel("Photon def simulate_flash(): """	r of events from a Poisson(lambd * T) # Numberson(lambd * T) # Numbe	nes. horizontal', colors=	'black', linewidths=0.	3)	
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(event  def plotflash(photon_t """  Plot photon arrival  photon_times: arran ax: matplotlib axi """  ax.eventplot(photon ax.set_ylabel("Photon ax.set_ylabel("Photon times = rand ax: matplotlib axi """  ax.eventplot(photon ax.set_ylabel("Photon ax.set_ylabel("Photo	r of events from a Poisson(lambd * T) # Numberson(lambd * T) # Numbe	nes.  horizontal', colors=  ges: t is open from s1 to	s2.	3)	
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(even  def plotflash(photon_t """  Plot photon arriva  photon_times: arranax: matplotlib axi """  ax.eventplot(photonax.set_ylabel("Phonomy ax.set_ylabel("Phonomy a	r of events from a Poisson(lambd * T) # Numberson(lambd * T) # Son(lambd * T) # Son	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with process  ons per ms erval in ms pen from 1 to 2 ms sorbed/detected by r	s2. obability $\alpha$ ).		
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(even  def plotflash(photon_t """  Plot photon arriva  photon_times: arranax: matplotlib axi """  ax.eventplot(photonax.set_ylabel("Photonax.set_ylab	r of events from a Poisson(lambd * T) # Numberson(lambd * T) # Numbe	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with properties on the properties of the properties o	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabili stream)) < alpha]		
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(even  def plotflash(photon_t """  Plot photon arriva  photon_times: arrandax: matplotlib axi """  ax.eventplot(photonax.set_ylabel("Photonax.set_yla	r of events from a Poisson (lambd * T) # Number son (lambd * T) # Number sevent times times (N, t1, t2) ent times nt_times)  imes, ax):  I times as vertical limes sobject  n_times, orientation='Next ton Events")  If light with three stages am. through a shutter that do by rods (each photon done in milliseconds.  photon rate: 100 photon 0.8, 2.2 # flash into # shutter op # fraction abs  oton stream tflash(lam, t1=full_t1, through shutter: those call_stream[(full_stream: each photon from the shutter_stream[np.random with aligned x-axes (in the photon of the shutter_stream (in the photon of the photon of the shutter_stream (in the photon of the photon of the shutter_stream (in the photon of the photon of the photon of the shutter_stream (in the photon of the pho	nes.  horizontal', colors=  ges:  t is open from s1 to  is detected with pr  ons per ms  erval in ms  pen from 1 to 2 ms  sorbed/detected by r  , t2=full_t2)  events between s1 a  m >= s1) & (full_str  shutter stream is d  om.rand(len(shutter_  n ms)  ue, figsize=(10, 6))  =100 photons/ms)")	s2. obability α).  rods  nd s2. eam <= s2)]  retected with probabili stream)) < alpha]		
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(event  def plotflash(photon_t """ Plot photon arriva  photon_times: arranax: matplotlib axi """ ax.eventplot(photonax.set_ylabel("Photonax.set_ylabe	r of events from a Poisson(lambd * T) # Number son(lambd * T) # Number son(lam	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with properties on the properties of the properties o	s2. obability α).  rods  nd s2. eam <= s2)]  retected with probabili stream)) < alpha]		
t1, t2 - Start and """  T = t2 - t1  # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(event  def plotflash(photon_t """  Plot photon arriva  photon_times: arrandax: matplotlib axin """ ax.eventplot(photonax.set_ylabel("Photonax.set_yl	r of events from a Poisson(lambd * T) # Numberson(lambd * T) # Numbe	nes.  horizontal', colors=  ges:  t is open from s1 to   is detected with pr  ons per ms   erval in ms   pen from 1 to 2 ms   sorbed/detected by r  , t2=full_t2)  events between s1 a m >= s1) & (full_str   shutter stream is d   om.rand(len(shutter_   n ms)  ue, figsize=(10, 6))  =100 photons/ms)")  rom %.1f to %.1f ms)  %.2f)" % alpha)	s2. obability α).  rods  nd s2. eam <= s2)]  retected with probabili stream)) < alpha]		
T = t2 - t1 # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(eve.  def plotflash(photon_t """	ation of process  r of events from a Poisson(lambd * T) # Number  son(lambd * T) # Number  event times  times(N, t1, t2)  ent times  nt_times, ax):  I times as vertical limes  sobject  n_times, orientation='Number  ton Events")  f light with three stage  eam.  through a shutter that and by rods (each photon of the shutter of the shutter of the shutter of the shutter; those constream  tflash(lam, t1=full_t1,  through shutter: those constream[(full_stream[(f	nes.  horizontal', colors=  ges:  t is open from s1 to     is detected with pr  ons per ms     erval in ms     pen from 1 to 2 ms     sorbed/detected by r  , t2=full_t2)  events between s1 a  m >= s1) & (full_str  shutter stream is d     om.rand(len(shutter_ n ms)  ue, figsize=(10, 6))  =100 photons/ms)")  rom %.1f to %.1f ms)  %.2f)" % alpha)	s2. obability α).  ods  nd s2. eam <= s2)]  etected with probabili stream)) < alpha]		
# Plot full stream plotflash (shutter_stream = fill fill stream plotflash (shutter_axes[0].set_title(axes[0].set_title(axes[2].set].set_title(axes[2].set_title(axes[2].set].set_title(axes[2].set].set_title(axes[2].set_title(axes[2].set].set_title(axes[	ation of process  r of events from a Poisson(lambd * T) # Number  son(lambd * T) # Number  event times  times(N, t1, t2)  ent times  nt_times, ax):  I times as vertical limes  sobject  n_times, orientation='Number  ton Events")  f light with three stage  eam.  through a shutter that and by rods (each photon of the shutter of the shutter of the shutter of the shutter; those constream the shutter those constream through shutter: those construction through shutter through shutter: through shutter: through shutter: through shutter: through shutter: through shutter: through shutter through shutte	nes.  horizontal', colors=  ges:  t is open from sl to  is detected with pr  ons per ms     erval in ms     pen from 1 to 2 ms     sorbed/detected by r  , t2=full_t2)  events between sl a     m >= sl) & (full_str     shutter stream is d     om.rand(len(shutter_     n ms)  ue, figsize=(10, 6))  =100 photons/ms)")  rom %.lf to %.lf ms)  %.2f)" % alpha)	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabili stream)) < alpha]  " % (s1, s2))  (λ=100 photons/ms)		
t1, t2 - Start and """  T = t2 - t1 # Dur  # Sample the numbe N = np.random.pois  # Generate random event_times = rand  # Return sorted ev return np.sort(eve.  def plotflash(photon_t """  Plot photon arriva  photon_times: arra; ax: matplotlib axi """ ax.eventplot(photo. ax.set_ylabel("Pho  def simulate_flash(): """ Simulate a flash o 1. Full photon str. 2. Photons passing 3. Photons detecte.  The simulation is event full photon str. 2. Photons passing 3. Photons detected.  The simulation is event full photons detected.  # Parameters lam = 100 # Generate full photons passing shutter_stream = ligh: # Photons passing shutter_stream = fi # Detected photons detected_stream = # Create subplots fig, axes = plt.sui  # Plot full stream plotflash(full_str. axes[0].set_title( axes[0].set_title( axes[0].set_title( axes[2].set_title( axes[2].set_title( axes[2].set_title( axes[2].set_title()  plt.tight_layout()  simulate_flash()  2.0  2.0  2.0  4.5  1.0  0.5  0.0  2.0  4.7  2.0  2.0  2.0  4.7  4.7  4.7  4.7  4.7  4.7  4.7  4	r of events from a Poisson(lambd * T) # Number son (lambd * T) # Number	nes.  horizontal', colors=  ges:  t is open from s1 to  is detected with pr  ons per ms  erval in ms  pen from 1 to 2 ms  sorbed/detected by r  , t2=full_t2)  events between s1 a  m >= s1) & (full_str  shutter stream is d  om.rand(len(shutter_  n ms)  ue, figsize=(10, 6))  =100 photons/ms)")  rom %.1f to %.1f ms)  %.2f)" % alpha)  Tull Photon Stream  After Shutter (open  Detected by r	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream)) < alpha]  "% (s1, s2))  from 1.0 to 2.0 ms)		2.0 2
# Photons passing shutter_stream = full_stream = ligh full_stream = li	r of events from a Poisson(lambd * T) # Number son (lambd * T) # Number	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with properties detected by respect to the properties of the properties	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream)) < alpha]  " % (s1, s2))  rods (α = 0.06)		2.0 2
t1, t2 - Start and """  T = t2 - t1 # Dur  # Sample the numbe N = np.random.pois # Generate random event_times = rand # Return sorted ev return np.sort(eve)  def plotflash(photon_t """  ax.eventplot(photo. ax.set_ylabel("Photon arriva photon_times: arra ax: matplotlib axi """  ax.eventplot(photo. ax.set_ylabel("Photon structure) Simulate a flash o 1. Full photon structure 2. Photons passing 3. Photons detected  The simulation is '"""  # Parameters lam = 100 # full_t1, full_t2 = s1, s2 = 1.0, 2.0 alpha = 0.06  # Generate full ph full_stream = ligh # Photons passing shutter_stream = f # Detected photons detected_stream =  # Create subplots fig, axes = plt.sui # Plot full stream plotflash(full_str. axes[0].set_xlitle( axes[0].set_xlitle( axes[0].set_xlitle( axes[0].set_xlitle( axes[0].set_xlitle( axes[1].set_xlitle( axes[2].set_title( axes[2].set_title( axes[2].set_title( axes[2].set_xlabel plt.tight_layout()  simulate_flash()  2.0  2.0  3. Plot the percent def plotdetectioncurve "Int the detection It she total num The "axis is on a "" "Lyalues = np.logs; "Values = np.logs; "Yalues	r of events from a Poisson (lambd * T) # Number son (lambd * T) # Numbe	nes.  horizontal', colors=  ges:  t is open from sl to  is detected with pr  ons per ms     erval in ms     pen from 1 to 2 ms     sorbed/detected by r  , t2=full_t2)  events between sl a     m >= sl) & (full_str     shutter stream is d     om.rand(len(shutter_     n ms)     ue, figsize=(10, 6))  =100 photons/ms)")  Tom %.lf to %.lf ms)  %.2f)" % alpha)  Photons  s a function of light     in tensity I.     g at the cornea.     ranges from 0.01 to	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream)) < alpha]  " % (s1, s2))  rods (α = 0.06)  t intensity I.  100.		2.0 2
T = t2 - t1 # Dur  # Sample the numbe N = np.random.pois # Generate random event_times = rand # Return sorted ev return np.sort(eve.  def plotflash(photon_t """	r of events from a Poisson(lambd * T) # Number son (lambd * T) # Number	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with pr  ons per ms erval in ms pen from 1 to 2 ms sorbed/detected by r  , t2=full_t2)  events between s1 a m >= s1) & (full_str shutter stream is d om.rand(len(shutter_n ms))  ue, figsize=(10, 6))  =100 photons/ms)")  Tom %.1f to %.1f ms)  %.2f)" % alpha)  Detected by r  Detected by r  After Shutter (open  """  Detected by r  After Shutter (open ""  photons  s a function of light aranges from 0.01 to  (N=K) for I in I_va	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream)) < alpha]  " % (s1, s2))  rods (α = 0.06)  t intensity I.  100.		2.0 2
t1, t2 - Start and """  T = t2 - t1 # Dur  # Sample the numbe N = np.random.pois # Generate random event_times = rand # Return sorted ev return np.sort(eve.  def plotflash (photon_t """  Plot photon arriva  ax: eventplot (photo. ax.set_ylabel("Photo. by Parameters lam = 100 # Farameters lam = 100 # full_t1, full_t2 = sl., s2 = 1.0, 2.0 alpha = 0.06 # Generate full ph full_stream = ligh # Photons passing shutter_stream = f # Detected photons detected_stream = # Create subplots fig, axes = plt.sui # Plot full stream plotflash(full_stream) plotflash(shutter_ axes[0].set_xlim(f) # Plot after shutt plotflash(shutter_ axes[2].set_xlabel plt.tight_layout()  simulate_flash()  2.0  2.0  2.0  3.5  4 Plot detected ph plotflash(shutter_ axes[2].set_xlabel plt.tight_layout()  simulate_flash()  2.0  2.0  3.6  4 Plot detected ph plotflash(shutter_ axes[2].set_xlabel plt.tight_layout()  simulate_flash()  2.0  3.7  4 Plot detected ph plotflash(shutter_ axes[2].set_xlabel plt.tight_layout()  simulate_flash()  2.0  3.7  4 Plot detected ph plotflash(shutter_ axes[2].set_xlabel plt.tight_layout()  simulate_flash()  2.0  4 Plot the detection axes[2].set_xlabel axes[2]	ation of process  ation of process  r of events from a Pois son(lambd * T) # Number son(lambd * T)  ent times nt_times nt_times nt_times, orientation='!  ton Events")  f light with three stag sam. through a shutter that diby rods (each photon done in milliseconds.  photon rate: 100 photo 0.8, 2.2 # flash int	nes.  horizontal', colors=  ges:  t is open from s1 to  is detected with pr  ons per ms     erval in ms     pen from 1 to 2 ms     sorbed/detected by r  , t2=full_t2)  events between s1 a     m >= s1) & (full_str     shutter stream is d     om.rand(len(shutter_     n ms)     ue, figsize=(10, 6))  =100 photons/ms)")  rom %.1f to %.1f ms)  %.2f)" % alpha)  Tull Photon Stream  After Shutter (open  ### After Shutter (open	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream) < alpha]  " % (s1, s2))  from 1.0 to 2.0 ms)  1.6 e (ms)  1.6 e (ms)		2.0 2
T = t2 - t1 # Dur  # Sample the numbe N = np.random.pois # Generate random event_times = rand # Return sorted ev return np.sort(eve.  def plotflash(photon_t	ation of process  r of events from a Poisson (lambd * T) # Number of Poisson (lambd * T) # Num	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with pr  ons per ms erval in ms pen from 1 to 2 ms sorbed/detected by r  , t2=full_t2)  events between s1 a m >= s1) & (full_str shutter stream is d om.rand(len(shutter_ n ms) ue, figsize=(10, 6))  =100 photons/ms)")  rom %.lf to %.lf ms)  %.2f)" % alpha)  Teull Photon Stream  After Shutter (open at the cornea. ranges from 0.01 to  yet (open)  After Shutter (open)  %.2f)" % alpha)  tion of intensity I. gat the cornea. ranges from 0.01 to  yet (open)  at the cornea. ranges from 0.01 to  yet (open)  at the cornea. ranges from 0.01 to  yet (open)  at the cornea. ranges from 0.01 to  cts of varying and abel="K=4, a=0.06") abel="K=6, a=0.06")	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream) < alpha]  " % (s1, s2))  from 1.0 to 2.0 ms)  1.6 e (ms)  1.6 e (ms)		2.0 2
Test to the number of photons.  Is the number of photons are as the simulation is simulated photons are as the simulation and detected photons as the simulation and detected photons are as the simulation are as the simul	r of events from a Poison (lambd * T) # Number son * I * I * I * I * I * I * I * I * I *	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with property on the property on the property of the proper	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream) < alpha]  " % (s1, s2))  from 1.0 to 2.0 ms)  1.6 e (ms)  1.6 e (ms)		2.0 2
T t2 - Start and  """  T t2 - t1 f pur  # Sample the numbe N = np.random.pois # Generate random event_times = rand # Return sorted ev return np.sort(eve  def plotflash(photon_t Plot photon arriva photon_times: arra ax: matplotlib axi """ ax.eventplot(photo ax.set_ylabel("Pho def simulate flash(): """ Simulate a flash o 1. Full photon str 2. Photons passing 3. Photons detecte The simulation is ' """ # Parameters lam = 100 # full_t1, full_t2 = s1, s2 = 1.0, 2.0 alpha = 0.06 # Generate full ph full_stream = ligh # Photons passing shutter_stream = f # Detected photons detected_stream = # Create subplots fig, axes = plt.su # Plot full stream plotflash(shutter_ axes[0].set_xlim(f # Plot after shutt plotflash(shutter_ axes[1].set_title( axes[2].set_title( axes[2].set_ti	r of events from a Poison (lambd * T) # Number son * I * I * I * I * I * I * I * I * I *	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with property on the property on the property of the proper	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilistream) < alpha]  " % (s1, s2))  from 1.0 to 2.0 ms)  1.6 e (ms)  1.6 e (ms)		2.0 2
Teach and T = t2 - t1  # Dur # Sample the numbe N = np.random.pois # Generate random event_times = rand # Return np.str(eve  def plotflash (photon_t	r of events from a Pois son(lambd * T) # Numbol son(la	nes.  horizontal', colors=  ges:  t is open from sl to  is detected with pr  ons per ms erval in ms pen from 1 to 2 ms sorbed/detected by r  , t2=full_t2)  events between sl a m >= sl) & (full_str shutter stream is d om.rand(len(shutter_ n ms) ue, figsize=(10, 6))  =100 photons/ms)")  rom %.lf to %.lf ms)  %.2f)" % alpha)  full Photon Stream  After Shutter (open  Detected by r  Detected by r  1.4  Time  T	s2. obability \( \alpha \).  ods  nd s2. eam <= s2) ] etected with probabilistream) \( < \alpha \) alpha]  (\( \lambda = 100 \) photons/ms)  ifrom 1.0 to 2.0 ms)  rods (\( \alpha = 0.06 \))  t intensity I.  1.6  e (ms)  R.		2.0 2
Test the number of photons.  Is the number of photons.  Is the simulate a flash of sing share a	photons are period. Photons are period photon stream and aligned x-axes [1] and aligned x-axes [2] and aligned x-a	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with property on the property on the property of the proper	s2. obability \( \alpha \).  ods  and s2. eam <= s2)   etected with probability stream) \( \cdot \) alpha]  from 1.0 to 2.0 ms)  from 1.0 to 2.0 ms)  and the intensity I.  1.6  and the intensity I.  1.7  1.8  1.9  1.9  1.9  1.9  1.1  1.9  1.1  1.0  1.1  1.0  1.0		2.0 2
### To be produced by the prod	ation of process  r of events from a Pois son(lambd * T) # Number son(lambd * T) son(la	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with pr  ons per ms even in ms even in ms even in to 2 ms sorbed/detected by r  t2=full_t2)  events between s1 a m >= s1) & (full_str shutter stream is d om.rand(len(shutter_ n ms) ue, figsize=(10, 6))  =100 photons/ms)")  rom %.lf to %.lf ms)  %.2f)" % alpha)  Detected by r  Detected by r  Detected by r  tig at the cornea, ranges from 0.01 to , K=K) for I in I_va ) 1.4  Time  Tim	s2. obability \( \alpha \).  ods  and s2. eam <= s2)   etected with probability stream) \( \cdot \) alpha]  from 1.0 to 2.0 ms)  from 1.0 to 2.0 ms)  and the intensity I.  1.6  and the intensity I.  1.7  1.8  1.9  1.9  1.9  1.9  1.1  1.9  1.1  1.0  1.1  1.0  1.0		2.0 2
### To be produced by the prod	ation of process  r of events from a Pois son(lambd * T) # Number son(lambd * T) son(la	nes.  horizontal', colors=  ges:  t is open from s1 to is detected with pr  ons per ms even in ms even in ms even in to 2 ms sorbed/detected by r  t2=full_t2)  events between s1 a m >= s1) & (full_str shutter stream is d om.rand(len(shutter_ n ms) ue, figsize=(10, 6))  =100 photons/ms)")  rom %.lf to %.lf ms)  %.2f)" % alpha)  Detected by r  Detected by r  Detected by r  tig at the cornea, ranges from 0.01 to , K=K) for I in I_va ) 1.4  Time  Tim	s2. obability \( \alpha \).  ods  and s2. eam <= s2)   etected with probability stream) \( \cdot \) alpha]  from 1.0 to 2.0 ms)  from 1.0 to 2.0 ms)  and the intensity I.  1.6  and the intensity I.  1.7  1.8  1.9  1.9  1.9  1.9  1.1  1.9  1.1  1.0  1.1  1.0  1.0		2.0 2
### Test	ation of process  r of events from a Pois son(lambd * T) # Number son(lambd * T) son(la	nes.  horizontal', colors=  ges:  tis open from sl to is detected with pr  ons per ms erval in ms pen from 1 to 2 ms sorbed/detected by r  , t2=full_t2)  events between sl am events between sl am shutter stream is d om.rand(len(shutter_ n ms) ue, figsize=(10, 6))  =100 photons/ms)")  rom %.lf to %.lf ms)  %.2f)" % alpha)  Full Photon Stream  After Shutter (open  Detected by r  Detected by r  cts of varying a and abel="K=0, a=0.06") abel="K=0,	s2. obability α).  ods  nd s2. eam <= s2)] etected with probabilif stream)) < alpha]  " % (s1, s2))  (λ=100 photons/ms)  infrom 1.0 to 2.0 ms)  t intensity I.  100.  lues]		2.0 2

In [1]: from A2b\_rh172 import \* import numpy as np

In [2]: def randtimes(N, t1, t2):

1a

import matplotlib.pyplot as plt

Simulate a Poisson process

N - number of random times

[ t1 , t2 ] - time interval

return random\_times

plt.xlabel('Time')

plt.ylabel('Event Height')

def plotflash(times):

plt.show()

plotflash(t)

1.0 -

0.8

Event Height .0 .9 .9

0.2 -

0.0 -

1b

1.0

0.8

1.0

In [4]: def randintervals(N, lambd, t1):

return event\_times

In [5]: intervals = randintervals(10,10,1)

plotflash(intervals)

In [3]: t = randtimes(10, t1=1, t2=2)

ппп

random\_times = [random.uniform(t1,t2) for \_ in range(N)]

Plot the given times as a stem plot with unit heights.

Stem Plot of Random Times

1.4

Generate N random time intervals from  ${\tt t1}$  with lambda rate.

intervals = np.random.exponential(scale=1.0/lambd, size = N)

Stem Plot of Random Times

Time

1.6

1.8

is a deprecated alias for the standard li

is a deprecated alias for the standard li

is a deprecated alias for the standard li

plt.stem(times, [1] \* len(times), markerfmt='')

plt.title('Stem Plot of Random Times')

1.2

event\_times = t1 + np.cumsum(intervals)

