Demo 3 Cheat Sheet: Classical Uncertainty and Wavefunction Expectation values

Numpy : Numerical Vectors and arrays (np)

Usage	Purpose	Inputs

x=np.zeros(5)
x =np.zeros((5,5))
	Return a new array of given shape and type, filled with zeros.	**shape** = number or numbers

x = np.arange(-1.0,1.0, 0.1)	Return an array evenly spaced values within a given interval according to a step size.	**start**, **stop**, **step_size**
x = np.linspace(-1.0,1.0, 100)	Return an array of evenly spaced numbers over a specified interval.	**start**, **stop**, **num**
np.conjugate(x)	Return complex conjugate value of x.	**scalar or vector**
np.double(3)	Convert a number or integer in to a floating number (decimals).	**scalar**
np.random.normal(loc,scale)	Draw random samples from a normal (Gaussian) distribution.	**loc** = center, **scale** = standard deviation
np.mean(x)	Calculate average of a vector.	**vector**
np.exp(x)	Exponential function.	**scalar or vector**

Matplotlib : Plotting (plt)

Usage	Purpose	Inputs
plt.figure(figsize=(10,8))	Setup parameter for a graphic, in this case we will use it change size.	figsize= (inches width, inches height)
plt.plot(x,y)	Plot lines	**x, y = vectors**
plt.hist(x)	Plot a histogram.	**X =vector**
plt.xlabel("Axis x name")	Set the *x* axis label of the current plot.	**Name = string**
plt.xlim([xmin,xmax])	Set the *x* limits of the current axes.	**xmin,xmax = scalars**
plt.title("Plot name")	Set a title of the current plot.	**Name = string**
plt.show()	Display a figure.	

Demo Specific - Common variables:

psi_x == vector representing wavefunction
x == scalar o vector for position
x0 /xf = =initial /final position
v0 == initial velocity
p0 /pf == initial momentum
dt = time step for simulation
L== length of box

x_list == list of positions for multiple particles
p_list == list of momentum for multiple particle

Function	Purpose
verlet(x,v,dt,a)	Function to update positions and velocities on each timestep.
a_box(x)	Acceleration of a particle in a box
time, x_list, p_list = ode_integrate_box(x, v, a_box, stopTime = stopT)	Solve ode for motion using the verlet algorithm with boundaries for a particle in a box.
wavefunction(x, L)	given x, returns a valid wavefunction for the 1D particle in a box
probabilityDensity(psi_x)	get probability density function from psi.
expectation_value_generalized(x, f_x):	Return expectation value (<f(x)>) for the function operator f .
f_x_position(x)	Function to represent position
f_spread (x)	Function to represent spread of a function (x-L/2)^2