```
# === exponentialChart.ipynb ===

#%%
# `%matplotlib notebook` not only makes a plot interactive,
# but it also allows it to be updated after the initial plot
%matplotlib notebook
# the line below is the expansion of %matplotlib notebook
# when run/edit in visual code studio
#get_ipython().run_line_magic('matplotlib', 'notebook')


#%%
# Import Numpy for calculations and matplotlib for charting
# ---------------------------------------------------------------------------
# The NumPy library is oftentimes used alongside PyPlot. This package contains
# plenty of built-in methods which allow for simple scientific computing
# ---------------------------------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt


#%%
# ---------------------------------------------------------------------------
# `np.arange(start, end, step)` creates a list of numbers from `start` to
# `end-before` where each number in the list is `step/increment`
# ---------------------------------------------------------------------------
# Creates a list from 0 to 5 with each step being 0.1 higher than the last
x_axis = np.arange(0, 5, 0.1)
x_axis


#%%
# ---------------------------------------------------------------------------
# Creates an exponential series of values which we can then chart
# The `e_x` list is being created using a "list comprehension"
# In this example: takes values from the `x_axis` list one at a time,
# finds the exponential of them, and stores the response within a list
# ---------------------------------------------------------------------------
e_x = [np.exp(x) for x in x_axis]
e_x

#%%
# ---------------------------------------------------------------------------
# Create a graph based upon the two lists we have created
# Matplotlib allows users to generate plots by setting one list as the x-axis
# and another as the y-axis and passed in to .plot() as arguments.
# ---------------------------------------------------------------------------
plt.plot(x_axis, e_x)
# Show the graph that we have created (not needed if use %matplotlib notebook)
#plt.show()

#%%
# Give our graph axis labels by calling functions .xlabel and .ylabel
plt.xlabel("Time With MatPlotLib")
plt.ylabel("How Cool MatPlotLib Seems")
```

```
56
57 # Have to plot our chart once again as it doesn't stick after being shown
58 plt.plot(x_axis, e_x)
59
60 # Show the graph that we have created (not needed if use %matplotlib notebook)
61 # plt.show()
62
63 #%%
64
65
66
67
```