Programming for Cybersecurity
Lab Topic 08-plotting

Introduction.

This lab is just an introduction to plotting, there is a lot of messing you can do yourself on this.
I would suggest that you create another folder in labs called **Topic08-plotting**.

Lab: Hey! numpy exists. (handy for matrix manipulation)

1. Write a program that makes a list, called *salaries*, that has (say 10) random numbers (20000 – 80000).

```
import numpy as np
# it is a good idea to have your absolute values set into variables at the
beginning of your program
minSalary= 20000
maxSalary = 80000
numberOfEntries = 10

salaries = np.random.randint(minSalary, maxSalary,numberOfEntries)
print (salaries)
```

2. Modify the program so that the "random" salaries are the same each time the program is run, to make the program easier to test, ie "seed" the random number generator.

```
np.random.seed(1) # this is so that the "random" numbers are the same each time t
o make it easier to debug.
```

3. Modify the program, to make another array of numbers that has the salaries plus 5000 (numpy is great for matrix operations)

```
salariesPlus = salaries + 5000 # this will add 5000 to each value
print(salariesPlus)
```

4. Modify the program so that it increases all the salaries by 5% (store in another array).

```
# you can also multiply
salariesMult = salaries * 1.05 # add 5% by multiplying by 1.05
print(salariesMult)
# This is a float array, here I convert it to an int array (it does a floor)
newSalaries = salariesMult.astype(int)
print(newSalaries)
```

Plotting

5. Write a program that plots the function $y = x^2$.

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

xpoints = np.array(range(1,101))
ypoints = xpoints * xpoints #multiply each entry by itself

plt.plot(xpoints, ypoints)
plt.show()
```

6. Write a program that plots a histogram of the salaries (from Question 1)

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

minSalary = 20000
maxSalary = 80000
numberOfEntries = 100

# this is so that the "random" numbers are the same each time to make it easier t
o debug.
np.random.seed(1)
salaries = np.random.randint(minSalary, maxSalary, numberOfEntries)

plt.hist(salaries)
plt.show()
```

7. Write a program that makes a list called ages that has, the same number random values as salaries, (range:21 to 65) . Make scatter plot of this data.

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

minSalary = 20000
maxSalary = 80000
numberOfEntries = 100

# this is so that the "random" numbers are the same each time to make it easier t
o debug.
np.random.seed(1)
salaries = np.random.randint(minSalary, maxSalary, numberOfEntries)
ages = np.random.randint(low=21, high = 65, size = numberOfEntries) # I don't
like this, I prefer having absolute values set at the top

plt.scatter(ages, salaries ) # this will be random
plt.show()
```

8. Add a line to this plot that shows $y= x^2$ in a different colour.

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

minSalary = 20000
maxSalary = 80000
numberOfEntries = 100

# this is so that the "random" numbers are the same each time
np.random.seed(1)
salaries = np.random.randint(minSalary, maxSalary, numberOfEntries)
# I prefer putting the abolute values into variables that are set at the top
ages = np.random.randint(low=21, high=65, size=numberOfEntries)

plt.scatter(ages, salaries)
#plt.show() # if you do this the proram will halt here until the plot is closed

# add x squared
xpoints = np.array(range(1, 101))
ypoints = xpoints * xpoints  # multiply each entry by itself

plt.plot(xpoints, ypoints, color= 'r')
plt.show() # see how the axis have changed
```

9. Add a legend, title and axis labels to this plot.

```
#same code as above except we need to add label to the plot and set the titles
and labels on the axis


plt.scatter(ages, salaries, label="salaries")
#plt.show() # if you do this the proram will halt here until the plot is closed

# add x squared
xpoints = np.array(range(1, 101))
ypoints = xpoints * xpoints  # multiply each entry by itself

plt.plot(xpoints, ypoints, color='r', label = "x squared")

plt.title("random plot")
plt.xlabel("Salaries")
plt.ylabel("age")
plt.legend()
plt.show()  # see how the axis have changed
```

10. Save this to a file called “prettier-plot.py”

```
#add this line to the end and you can comment out the show()
#plt.show()  # see how the axis have changed

plt.savefig('prettier-plot.png')
```

11. Write program that has a list of counties, make it a long list (pick from five counties). Make some counties appear more than others. Make a pie chart of the counties.

```
# Demonstrate making a pie plot of the unique occurences of values in an array .
# I am demonstrating more than just how to do pie plots in this.
# Author: Andrew Beatty

import numpy as np
import matplotlib.pyplot as plt

# make the array of occurences
possibleCounties = ['mayo', 'Galway', 'Rosommon', 'DirtyDublin','Donegal']
# pick 100 randomly from possible counties with the frequence set in p
counties = np.random.choice(
       possibleCounties ,
       p=[0.1, 0.3, 0.2, 0.12, 0.28 ],
       size=(100)
)

# right now we need the number of occurences of each county
# this returns a tuple of the unique values and how many times they appear
unique, counts = np.unique(counties, return_counts=True)
# we can now put this into a pie plot
plt.pie(counts, labels= unique)

plt.show()
```

12. Modify the program to make bar chart of the counties.

```
# same above except
plt.bar(unique, counts)

plt.show()
```

Extra:

13. Mess around with real data.
14. Mess around with normalized random data.
15. Add a line of normal distribution to the plot in 9 above. (use seaborn)

Discussion

- This lab only scrapes the surface of what you can do with plots, you will want to do a lot more exploring of these packages yourself.
- Picking the scales of your plots is not always straightforward and can take a bit of trial and error