COMP 364: Computer Tools for Life Sciences
Data visualization with MatPlotLib

Mathieu Blanchette, based on material from Christopher J.F. Cameron and Carlos G. Oliver
Visualization is an important way for humans to understand data. Python programs can generate plots about the data they are handling. This is commonly done using the Matplotlib module. https://matplotlib.org/devdocs/api/pyplot_summary.html

To use Matplotlib, you first need to import the module within your program.

```python
import matplotlib.pyplot as plt
```

This imports the functions of the matplotlib module, and gives the module a shorter name: plt.
A first example

```python
import matplotlib.pyplot as plt

my_numbers = [1, 5, 2, 4, 1]
plt.plot(my_numbers)
plt.ylabel("My numbers")
plt.show()  # displays figure
```
import matplotlib.pyplot as plt
import math

my_x = range(0, 30)

# calculate the value of sin(x) for all x in my_x
# this uses the approach of list comprehension
my_sin = [math.sin(x) for x in my_x]

# here plot takes two arguments: the list of x coordinates
# and the list of y coordinates
plt.plot(my_x, my_sin)
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.show() # displays figure
A second example
Saving a figure

With Wing101, you can save a figure when it pops-up during the program’s execution.
But you can also save the figure programmatically (without the need for user intervention):

Use the `plt.savefig()` function: https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.savefig.html

```python
import matplotlib.pyplot as plt
import math

my_x = range(0, 30)
my_sin = [math.sin(x) for x in my_x]

plt.plot(my_x, my_sin)
plt.xlabel("x")

# this won’t show the figure, but will save it
# in a file named my_sin.png
plt.savefig("my_sin.png")
```
Colors and markers

We can select the color of the plots, the style/size of markers, etc. Useful when multiple data are being plotted!

See plot() documentation for details.
```python
import matplotlib.pyplot as plt
import math

my_x = range(0, 30)
my_sin = [math.sin(x) for x in my_x]
my_cos = [math.cos(x) for x in my_x]

# plots my_sin with a blue line
plt.plot(my_x, my_sin, "b")

# plots my_cos with a red line and marker *
plt.plot(my_x, my_cos, "r*--")
plt.xlabel("x")

plt.show()
```
More about colors

Matplotlib functions can handle many different colour codes:

1. character:
   - 'b': blue
   - 'r': red
   - 'k': black
   - ...

2. RGB (Red-Green-Blue)
   - (0,1,0) = green
   - (1,0,1) = purple
   - (0,0,0) = black
   - (0.5, 0.5, 0.5) = gray

See https://matplotlib.org/api/colors_api.html for more information.
A more advanced example

Suppose you have measured the expression of 5 genes in a set of healthy patients and a set of cancer patients:

```python
genes = ["ACTB", "P53", "RPL3", "POLR2A", "RB"]

# expression data in 4 healthy individuals
normals = [[0.4, 6.4, 3.2, 3.5, 4.1],
            [0.6, 5.4, 3.6, 4.2, 4.9],
            [0.7, 5.7, 3.7, 4.1, 4.2],
            [0.4, 5.2, 3.6, 3.3, 4.8]]

# expression data in 5 cancer patients
cancer = [[0.5, 9.2, 3.4, 3.6, 0.9],
          [0.7, 8.7, 3.5, 4.6, 0.7],
          [0.4, 8.2, 2.9, 4.2, 1.2],
          [0.6, 9.7, 3.8, 3.9, 1.3],
          [0.6, 9.6, 3.1, 3.8, 1.0]]
```

Goal: Visualize this data to learn which genes may be dysregulated in cancer.
A more advanced example

Idea: generate plot with x-axis = gene, y-axis = expression
Use dots of different colors for normals and cancer patients

```python
import matplotlib.pyplot as plt

for n in normals:
    plt.plot(genes, n, "ro")
for c in cancer:
    plt.plot(genes, c, "bx")

plt.ylabel("Expression")
plt.savefig("cancer1.png")
```
Now add legend, and show different individuals in different tones of red and blue

```python
for index, n in enumerate(normals):
    plt.plot(genes, n, "o", color=(1-0.1*index, 0, 0),
             label="Normal"+str(index))

for index, c in enumerate(cancer):
    plt.plot(genes, c, "o", color=(0,0,1-0.1*index),
             label="Cancer"+str(index))

plt.ylabel("Expression")
plt.legend(loc="best")  # displays legend
plt.savefig("cancer2.png")
```
Bar graph

Generate a bar graph of expression for P53.
# extract data for P53
p53_exp_normals = [n[1] for n in normals]
p53_exp_cancer = [c[1] for c in cancer]

# generate identifiers for samples
normals_names = ["Norm" + str(i) for i in range(0, len(normals))]
cancer_names = ["Cancer" + str(i) for i in range(0, len(cancer))]

plt.bar(normals_names, p53_exp_normals)
plt.bar(cancer_names, p53_exp_cancer)
plt.ylabel("P53 expression")
plt.show()
plt.savefig("cancer3.png")
For more information

Tutorial:
https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py

Documentation:
https://matplotlib.org/devdocs/api

Important: You don’t need to know everything in Matplotlib! You just need to know how to read the document to figure out how to do what you want to do.