# CSE231 - Lab 14

A Soft Intro to Pylab/Matplotlib

## Congratulations! 🎉

You officially know (pretty much) everything about Python! There's still a few things you might want to look into yourself (either because they were recent additions to Python, or because they're things you could easily live without), like

1. Walrus operator, `:=`
2. Decorators, `@staticmethod`, `@property`, `@classmethod`, ...
3. `yield`, `with`, `await`, `async`, `assert`, `lambda`
4. Bitwise operators, `>>`, `<<`, `~`, `|`, ...
5. `zip()`, `map()`, `filter()`
6. Advanced data structures, linked lists, hash tables, trees (CSE331 stuff)

## Modules

For right now, though, we’re going to talk about a widely used module in data science, `matplotlib`, since Python is the big language in that field.

Something that we didn’t address with modules earlier, is that there are literally thousands of them. There is only a small fraction of modules built by the Python developers, the rest of them are all built by random programmers across the world. GitHub is a great place to look into some of the modules that people make.

Surprisingly, `matplotlib` wasn’t made by the Python developers despite it being such a well constructed module. You now have the knowledge required to create modules of your own and release them to the public, “module” is basically just a fancy word for “Python file”.

## Now you might be thinking...

Braedyn, why are you talking about “`matplotlib`” when we were taught `pylab`?

`pylab` is a module composed of *two other* modules. `matplotlib` is *inside* `pylab`, alongside another module, `numpy`, which you might become familiar with later if you get deeper into data science stuff.

So, you can either import `matplotlib.pyplot` (the specific part of the `matplotlib` module we’ll be using), or you can import `pylab`. `pylab` simply calls upon `matplotlib` functions but is renamed.

## Creating a Figure and Axes Object

```
import pylab

figure, axes = pylab.subplots( figsize=(10, 7) )
```

To get started, you’ll want to create a figure and an axes object. This is the popular and most concise way of doing it.

The figure object acts as a “canvas” to the axes object. You can create multiple axes objects and plot them onto the same figure, but we won’t be going into that today.

## Making a Bar Graph

```
import pylab

figure, axes = pylab.subplots( figsize=(10, 7) )

axes.bar( x_data_names, y_data )
```

In order to then create some sort of graph, you call the desired graph *method function* on the axes object. `.bar()`, for instance, takes two required parameters. A list of things you want on the x-axis, and a list of things you want on the y-axis.

The two lists must correspond to each other’s data points in-parallel.

## Labels / Title / Seeing The Figure

```
import pylab

figure, axes = pylab.subplots( figsize=(10, 7) )

axes.bar( x_data_names, y_data )

axes.set_xlabel(“My Cool X Data”)

axes.set_ylabel(“My Cool Y Data”)

axes.set_title(“My Extremely Cool Graph”)

pylab.show()
```

## Setting X Labels

```
import pylab

figure, axes = pylab.subplots( figsize=(10, 7) )

axes.bar( x_data_names, y_data )

axes.set_xticklabels( x_data_names, rotation=90)

axes.set_xlabel(“My Cool X Data”)

axes.set_ylabel(“My Cool Y Data”)

axes.set_title(“My Extremely Cool Graph”)

pylab.show()
```

## Alternatively...

What I just showed you is the *object-oriented* way of doing things. It’s the preferred way of using the module nowadays.

But, you can also call the module directly without needing to create an axes object. The function names change, for whatever reason. This is the most confusing part of `matplotlib`. The parameters all stay the same, the names are just slightly different.

Pretty commonly, you’ll see people use:

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

You can replace all of the instances we’ve called `pylab`, and instead replace it with `plt` in that case.

## Pylab Without OOP

```
import pylab

pylab.figure(figsize=(10, 7))

pylab.bar( x_data_names, y_data )    # axes.bar()

pylab.xticks( x_data_names, rotation=90)    # axes.set_xticklabels()

pylab.xlabel(“My Cool X Data”)    # axes.set_xlabel()

pylab.ylabel(“My Cool Y Data”)    # axes.set_ylabel()

pylab.title(“My Extremely Cool Graph”)    # axes.set_title()

pylab.show()
```

## Using Matplotlib Directly

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 7))

plt.bar( x_data_names, y_data )

plt.xticks( x_data_names, rotation=90)

plt.xlabel(“My Cool X Data”)

plt.ylabel(“My Cool Y Data”)

plt.title(“My Extremely Cool Graph”)

plt.show()
```

## Using Matplotlib Directly (cont.)

```
import matplotlib.pyplot as plt

figure, axes = plt.subplots( figsize=(10, 7) )

axes.bar( x_data_names, y_data )

axes.set_xticklabels( x_data_names, rotation=90)

axes.set_xlabel(“My Cool X Data”)

axes.set_ylabel(“My Cool Y Data”)

axes.set_title(“My Extremely Cool Graph”)

plt.show()
```

## Project 11

Alright, that’s all I got for today.

For project 11, *please* start early. It’s worth a whopping 60 points (6% of the course grade), and it’s due on Friday, April 24th.

I’ll try to host more of my own help-rooms this upcoming week, but I can’t make any guarantees -- I got my homework of my own unfortunately.