Outcomes

By the end of this notebook, you will be able to...

- Read Python code in a Jupyter notebook.
- Run Python code in a Jupyter notebook.
- Read a provided data set into a Jupyter notebook.
- Use the `print()` function in Python to display data.
- Create a graph in Python to visualize and interpret data.
- Change basic elements of a graph in Python.

Importing the libraries we need

One important feature of Python is its ability to import **libraries** of functions that we might want to use. These libraries are created by Python developers and are available from the internet, meaning if you've opened this notebook in a web browser, you have access to over a hundred thousand libraries that already perform much of what you want to do.

To import a library, we use the `import` command, usually formatted as `from library import abbreviation`, where `library` is the name of the library you want and `abbreviation` is a shorthand we use for that library name. While you can name `abbreviation` anything you want, most libraries have a preferred `abbreviation` to help make it easier for programmers to read each other's code.

For this notebook we need the following libraries:

- `pandas` enables us to **read data from a spreadsheet**.
- `matplotlib.pyplot` gives us a wealth of **graphing utilities**.

Below is a **code cell** that imports each of these libraries. Click inside the cell and press Ctrl+Enter, or press the triangle play button next to the cell.

In [ ]:
```
import pandas as pd
print('imported pandas')
import matplotlib.pyplot as plt
print('imported matplotlib.pyplot')
```

This code cell includes a `print()` function, which allows us to easily retrieve information from Python. Since Python reads through code one line at a time from top to bottom, the code produces these two messages in order just below the code cell.

Let's add one more line to the code cell above to confirm we're finished with importing everything. If you don't see line numbers on the left-hand side of the code cell, click on

Tools (in the menu above) → Settings → Editor and check the box next to Show Line Numbers.

Copy `print('finished importing, NAME')` and paste it into Line 6 in the code cell above. (Ctrl+C is a handy shorcut for copying and Ctrl+V is a handy shortcut for pasting.) Then, in the code cell, change `NAME` to be your name. Run the code cell again, and check that your new message appears.

Now that we've imported these libraries, we will have access to them throughout this notebook, because **anything you do in one code cell carries forward to future code cells**. We'll use this continuity to our advantage throughout these notebooks.

Error Messages

If you copy and paste something incorrectly in a code cell, Python will tell you. Just do your best in trying to understand the error messages, and email me if you get stuck! The most common error in this context is probably not closing a parentheses, just like when you're writing a sentence!

Reading data from a Google Sheet

Our big operation for this notebook is to **read data into Python** from a spreadsheet. The easiest way to do this is to upload the data into a Google Sheet, turning on sharing for anyone to read, and then use the method `pd.read_excel()` to read in the data. Notice that the function begins with `pd`, which means we're telling Python that `read_excel()` is a **function** that lives inside the `pandas` **module** (which we abbreviated `pd`).

The code cell below reads data from a Google Sheet (which we've already set up for you) from the link https://docs.google.com/spreadsheets/d/1Z4pTRxwg1ZCAcaN5DNRuL4oLCR1MsD29gHv8toSZ usp=sharing, where we have some position, velocity, and acceleration data for a 500-gram cart going up and down a ramp inclined by 6°, measured using a sonic motion sensor like you might be familiar with. Click this link to open this sheet in a new tab so you can compare the spreadsheet's contents with what we see in Python.

We need to **store** this data somewhere, so we're going to create a **database** named `database` by adding `database =` in front of the `pd.read_excel` command. This means that Python will read all the data in our spreadsheet, then store those numbers under the name `database` so we can use them later.

Run the code cell below to read in the data.

In [ ]:
```
database = pd.read_excel("https://docs.google.com/spreadsheets/d/1Z4pTRxwg1ZCAc
```

```
print('finished reading')
```

Printing information from a database

It's always a good idea to check whether the data imported correctly. After all, we might have pasted in the wrong URL, or misformatted our data. Pandas gives us lots of options to display the information in our `database`. The code cell below prints the beginning and end of the *entire* database.

Checkpoint

Run the code cell and describe the output. Double-click on this **text cell** to edit it and type your answers in the square brackets below. Specifically, discuss whether the values printed match up with those in the spreadsheet.

[Add your description here.]

In [ ]:
```
print(database['time'][1:5])
```

Now, replace the print function with each of the following (one at a time) and run the code cell. For each of these print functions, describe what output you see and explain how Python knows which information from the spreadsheet to display.

1. `print( database.head(4) )`
2. `print( database.head(5) )`
3. `print( database.tail(4) )`
4. `print( database.tail(3) )`
5. `print( database['time'] )` This single column of data is called an **array** in Python.
6. `print( database['ay'] )`
7. `print( database['time'][0] )` This one and the next few might seem weird at first. Check out Why Programmers Start Counting At Zero.
8. `print( database['time'][1] )`
9. `print( database['time'][2] )`
10. `print( database['time'][1:5] )`
11. `print( database['ay'][7:10] )`

[Add your answers here.]

Outline a few ways in which these commands might be useful in a lab experiement.

[Add your answers here.]

Creating a graph

Printing numerical values is great, but we need **visualizations** to help us understand what's going on in the data. Fortunately, `pandas` knows how to call `matplotlib.pyplot` for us to create a graph using the `.plot` command, if we tell it which columns in `database` we want to graph.

Run the code cell below and describe the graph.

Then, swap the order of `'time'` and `'position'`. How does the graph change?

In [ ]:
```
database.plot('time','position')

plt.show() # This displays the graph cleanly
```

Manipulating basic graph elements

The above graph is great, and we were able to generate it with a total of 4 lines of code (not counting the `print()` commands). We'll finish out this notebook by learning how to make a few simple changes to the graph. Follow these steps in the code cell above.

1. First, change the graph back to acceleraiton versus time. Run the code cell.
2. Next, change the graph to a scatter plot by adding `.scatter` between `plot` and the open parenthesis. Run the code cell. When would you prefer a line plot, and when would you prefer a scatter plot?
3. Now let's add some additional information to the graph. In line 2 of the code cell above, copy-paste `plt.xlabel('time (seconds)')`. Run the code cell. What changes in the graph?
4. Now add code to line 3 to create a similar label on the vertical axis. What would you need to change in the command you copy-paste? Do this and run the code cell.
5. Let's zoom in on the second peak in the data. Copy-paste `plt.xlim(18200,MAXTIME)` into line 4. The first value, `18200`, will set the minimum time shown in the graph. Change `MAXTIME` to a numerical value for the maximum time you'd like the graph to show. Pick one that's to the right of the second peak. Run the code cell. You can adjust your `xlim` values as needed to show the peak to your liking.
6. Now change the vertical range that the graph shows. What do you think you'll need to change in the command you just used in Step 5? You can adjust your `xlim` and `ylim` values as needed to show the peak to your liking.
7. Finally, save the figure using `plt.savefig("Figure_name",dpi=800)` and fill in the name of the file. Be sure to keep the dpi to a high number, this will save an image with high resolution.

You try

Use the code cell below to create graphs of velocity versus time and acceleration versus time using the values in `database`. How can you use the previous code cell as an example? What do you need to change?

In [ ]:

References

This notebook is based on Brian Lane's LetsCodePhysics tutorials and Adam LaMee's CODINGinK12 tutorials