1. Overview

Data Science Playlist on YouTube

Introduction

Welcome to the course on data science with Python. This course steps through basic data science and machine learning skills to analyze data and create actionable information. It address major steps of the Cross-Industry Standard Process for Data Mining Cycle (CRISP-DM) that has 6 phases.

1. Business Understanding - purpose of the data science project
2. Data Understanding - assess available data to address the business case
3. Data Preparation - exploratory data analysis, feature engineering, and transformations
4. Modeling - select and train regressors or classifiers
5. Evaluation - test models with various criteria
6. Deployment - production phase to deploy models in data pipelines

The 12 exercises are designed to be completed in 2-3 hours (15-20 minutes each) but sections can be skipped if you already have the background knowledge.

1. Overview (this lesson)
2. Data Import and Export
3. Data Analysis
4. Visualize Data
5. Prepare (Cleanse, Scale, Divide) Data
6. Regression
7. Features
8. Classification
9. Interpolation
10. Solve Equations
11. Differential Equations
12. Time Series

It is best to follow the lessons in these steps because the later lessons build upon the information from the prior lessons.

Final Project

You are designing a next-generation cell phone and the battery and processor on the cell phone generate a lot of heat. You want to make sure that the material between them will prevent over-heating of the battery by the processor.

The final project will help you answer questions about material properties for predicting the temperature of the battery and processor. It uses data from the temperature control lab to determine thermal conductivity. See the Final Project after going through the 12 lessons and when you are ready for the challenge.

What Runs Python Commands?

The Python executable runs the commands. A Jupyter notebook starts a Python session called a kernel. Restarting the kernel removes any of the prior cell results and stops any running code.

In [1]:

```
import sys
print(sys.executable)
```

```
/usr/bin/python3
```

Where is the Current Working Directory?

The current working directory is where the IPython notebook is located. It is also where files will be stored and retrieved. It is possible to change the current working directory with `os.chdir(path)`

In [2]:

```
import os
print(os.getcwd())
```

```
/home/curtis/classes/dynamic_optmization/data_science-master
```

There are seperate Jupyter notebook files to help with TCLab installation. There are also Frequently Asked Questions for setup and troubleshooting. More information on installation of TCLab package is in the IPython notebook **TCLab Help**.

Python Package Management

Package management in data science and machine learning is an important topic because of the pace of new releases. It may be important to have the latest or a specific version of a package.

To install packages, use the `pip` (or `conda`) package manager. An alternative is to start an Anaconda prompt from the start menu, start a DOS prompt with Windows-key+r `cmd`, or terminal on MacOS/Linux and type:

```
python -m pip install gekko
```

If you would like to run this same command in a Jupyter notebook then use an exclamation mark in front.

```
!python -m pip install gekko
```

If you want to be sure that you are using the distribution of the current Python kernel use `sys.executable`.

```
import sys
!{sys.executable} -m pip install gekko
```

Packages used in this course include `gekko`, `keras`, `numpy`, `pandas`, `pyserial`, `sklearn`, `tclab`, `tensorflow`, and others. If there is an error that you are missing a package then `!pip install gekko` is typically sufficient to retrieve and install it. The `!` tells Jupyter to run from the system command line, not as a Python command although the `!` is now optional for `pip` in the latest version of Jupyter. Add `--user` (not your specific user name) if you do not have administrative privilege on the computer. The package manager `pip` retrieves the lastest version of gekko from `pypi` and installs it in the Python distribution.

Install Package

In [3]:

```
!python -m pip install --user gekko
```

```
/usr/bin/python: No module named pip
```

List Available Packages and Versions

In [4]:

```
!python -m pip list
```

```
/usr/bin/python: No module named pip
```

Uninstall Package

If you need to `uninstall` a package then use the command line because it will ask for a `yes` / `no` response or else add `--yes` to automatically answer the confirmation message. Don't forget to reinstall the package if you need it.

In [ ]:

```
!python -m pip uninstall gekko --yes
```

Install Specific Version

You may need to get a specific package version by specifying a version such as `0.2.0`.

In [ ]:

```
!python -m pip install gekko==0.2.0
```

Upgrade Package

If you have an older version and want to upgrade to the latest stable release use `--upgrade`

In [ ]:

```
!python -m pip install gekko --upgrade
```

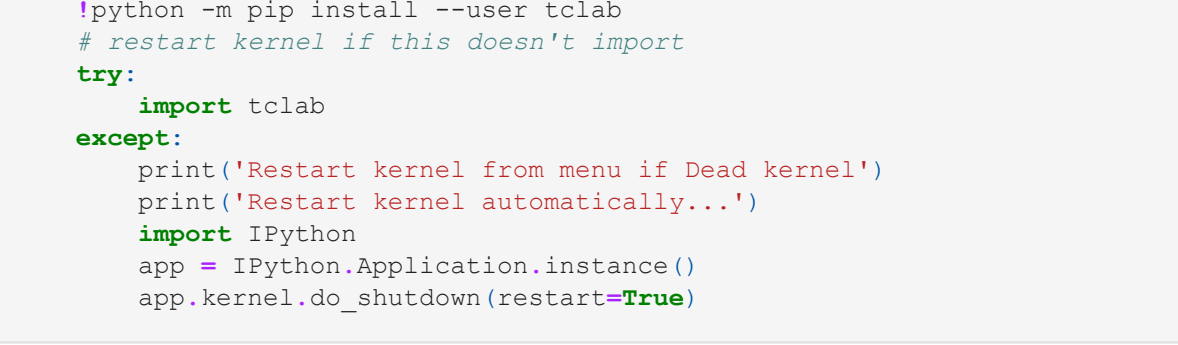
`pip` Summary

The `pip` tool facilitates package management in Python using `install`, `list`, and `uninstall` commands.

```
!python -m pip {command} {package name}
```

Install TCLab Module

You need the TCLab to do the exercises but there are sample data files if you do not have a device. The TCLab is available online with Amazon or with a few other options.



As a first step, plug in the TCLab (USB blue cable only) and install the package with `python -m pip install tclab` or by running the cell below (`Ctrl+Enter`). Restart the Python kernel with `Kernel...Restart & Run All` from the menu if there is an error importing `tclab` after the installation.

In [1]:

```
# install tclab
try:
    import tclab
except:
    # Needed to communicate through usb port
    !python -m pip install --user pyserial
    # The --user is put in for accounts without admin privileges
    !python -m pip install --user tclab
    # restart kernel if this doesn't import
    try:
        import tclab
    except:
        print('Restart kernel from menu if Dead kernel')
        print('Restart kernel automatically...')
        import IPython
        app = IPython.Application.instance()
        app.kernel.do_shutdown(restart=True)
```

TCLab LED Test

The following code show how to turn on the LED for 5 seconds. You only need to connect the blue USB cable to the computer. The white power cable is needed for heater power in the next example.

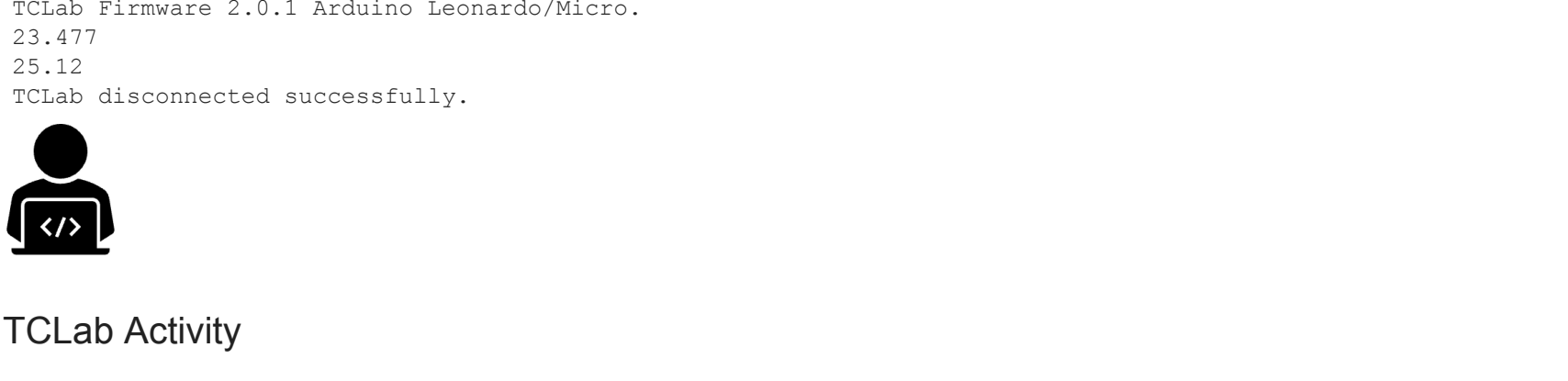
In [2]:

```
import tclab
import time

with tclab.TCLab() as lab:
    lab.LED(100)
    time.sleep(5.0) # wait 5.0 seconds
    lab.LED(0)
```

```
TCLab version 0.4.9
Arduino Leonardo connected on port /dev/ttyACM1 at 115200 baud.
TCLab Firmware 2.0.1 Arduino Leonardo/Micro.
TCLab disconnected successfully.
```

TCLab Heater Test



The following code show how to turn on the heater `Q1`. It reads temperature `T1` before and after a 10 second pause. You need to connect the white power cable and plug in the power supply. The white power cable is needed to supply power to the heaters.

In [3]:

```
with tclab.TCLab() as lab:
    print(lab.T1)  # print temperature 1
    lab.Q1(100)    # turn on Q1 to 100%
    time.sleep(15) # sleep for 15 seconds
    print(lab.T1)  # print temperature 1
```

```
TCLab version 0.4.9
Arduino Leonardo connected on port /dev/ttyACM1 at 115200 baud.
TCLab Firmware 2.0.1 Arduino Leonardo/Micro.
23.477
25.12
TCLab disconnected successfully.
```

TCLab Activity

TCLab functions allow Python to interact with the TCLab through the USB serial connection.

TCLab Function	Example	Description
`TCLab()`	`tclab.TCLab()`	Create new lab object and connect
`LED`	`lab.LED(45)`	Turn on the LED to 45%. Valid range is 0-100%
`Q1`	`lab.Q1(63)`	Turn on heater 1 (`Q1`) to 63%. Valid range is 0-100%
`Q2`	`lab.Q2(28)`	Turn on heater 2 (`Q2`) to 28%. Valid range is 0-100%
`T1`	`print(lab.T1)`	Read temperature 1 (`T1`) in °C. valid range is -40 to 150°C (TMP36 sensor)
`T2`	`print(lab.T2)`	Read temperature 2 (`T2`) in °C. with +/- 1°C accuracy (TMP36 sensor)
`close()`	`lab.close()`	Close serial USB connection to TCLab - not needed if using `with` to open

Complete the following exercises.

Blink LED

Blink the LED 5 times for 1 second each.

In [12]:

```
with tclab.TCLab() as lab:
    OFF = 0
    ON = 100

    lab.LED(OFF)
    for i in range(5):
        lab.LED(ON)
        time.sleep(1)
        lab.LED(OFF)
        time.sleep(1)
```

```
TCLab version 0.4.9
Arduino Leonardo connected on port /dev/ttyACM0 at 115200 baud.
TCLab Firmware 2.0.1 Arduino Leonardo/Micro.
TCLab disconnected successfully.
```

Heat to 50°C

Turn on heater 1 (`Q1`) to 80% until `T1` reaches 50°C. Update the LED blink time (`t`).

$$t = \frac{50-T_1}{10}$$

This displays a visual indication of the temperature (`T1`) with the formula for alternating time off and time on. As the temperature approaches 50°C, the LED blinks more rapidly until it turns off.

Use a `while lab.T1<50:` loop to continuously check that the temperature is less than 50°C.

In [4]:

```
import tclab
import time

def blink(lab, t):
    lab.LED(100)
    time.sleep(t)
    lab.LED(0)
    time.sleep(t)

with tclab.TCLab() as lab:
    while lab.T1 < 50:
        lab.Q1(80)
        t = (50 - lab.T1)/10
        blink(lab, t)
        print(lab.T1)
    lab.close()
```

```
TCLab version 0.4.9
Arduino Leonardo connected on port /dev/ttyACM1 at 115200 baud.
TCLab Firmware 2.0.1 Arduino Leonardo/Micro.
28.633
29.277
30.212
31.146
32.113
33.08
34.079
34.917
35.723
36.367
37.108
37.882
38.591
39.268
39.912
40.524
41.169
41.813
42.168
42.813
43.135
43.715
44.102
44.424
44.746
45.068
45.391
45.713
46.035
46.325
46.615
46.712
47.002
47.292
47.324
47.646
47.904
47.969
48.259
48.291
48.549
48.613
48.903
48.936
49.258
49.258
49.451
49.548
49.58
49.87
49.87
49.902
50.225
TCLab disconnected successfully.
```

In [ ]: