Advanced Python modules and functionalities

Lesson 8 – 9/30/16

Today's schedule

1. Advanced Modules
    a) `matplotlib` – graphing and figure generation
    b) `optparse` – flagged inputs
2. Raspberry Pi as a Learning Tool

1. matplotlib

matplotlib

Purpose: Graphing and visualization of data:

1. Plots
2. Histograms
3. Barplots, etc

Say goodbye to figure generation using Excel

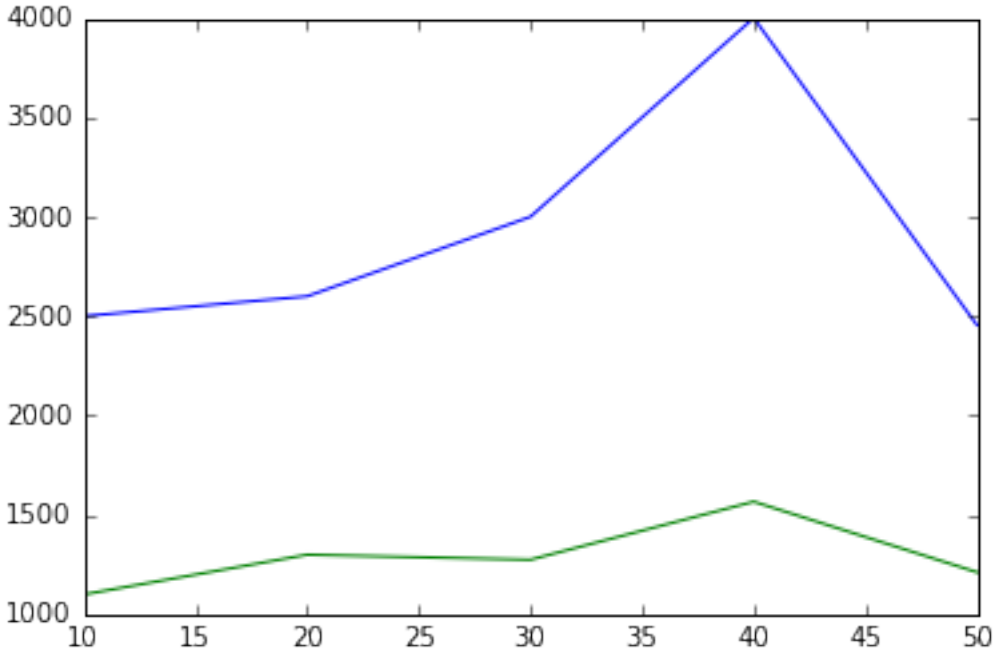
Example matplotlib

- **Problem:**
  - You want to plot the expression of *YFG* over five different time points in your dataset. You want to also to compare this time course to that of *YFHKG*.

Example: Plotting Lines

```
%matplotlib inline
import matplotlib.pyplot as plt

days = [10, 20, 30, 40, 50]
YFG = [2500, 2600, 3000, 4000, 2455]
YFHKG = [1100, 1300, 1275, 1567, 1212]

plt.plot(days, YFG)
plt.plot(days, YFHKG)
plt.show()
```

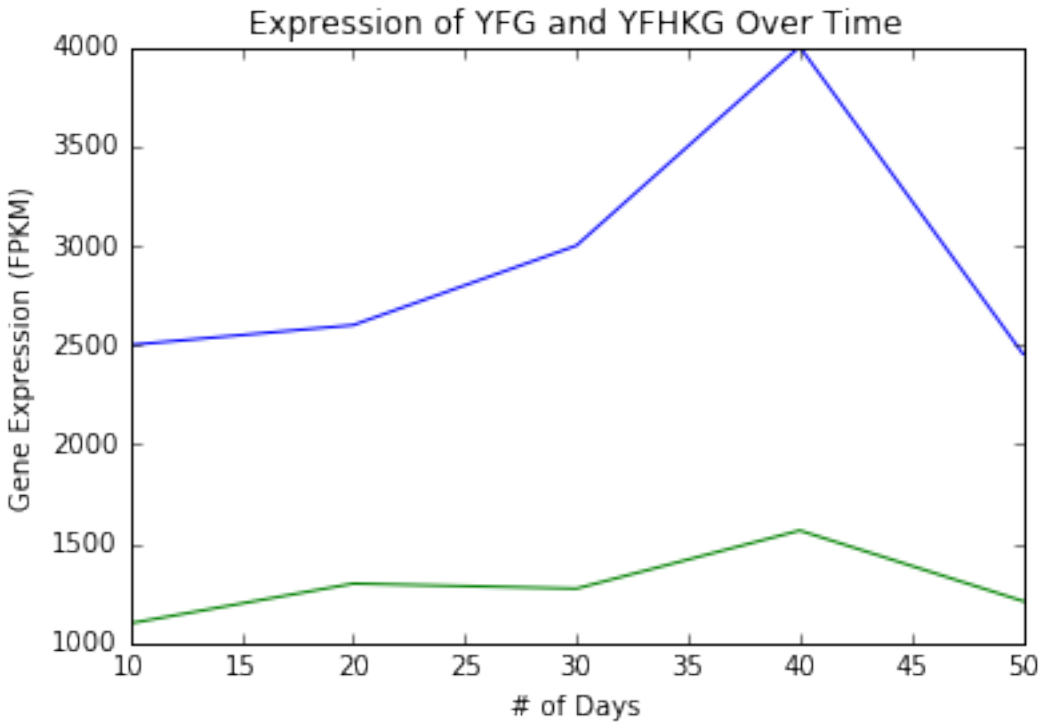
Example Result



Example: Labeling Axes

```
%matplotlib inline
import matplotlib.pyplot as plt

days = [10, 20, 30, 40, 50]
YFG = [2500, 2600, 3000, 4000, 2455]
YFHKG = [1100, 1300, 1275, 1567, 1212]

plt.plot(days, YFG)
plt.plot(days, YFHKG)
plt.xlabel("# of Days")
plt.ylabel("Gene Expression (FPKM)")
plt.title(”Expression of YFG and YFHKG Over Time")
plt.show()
```

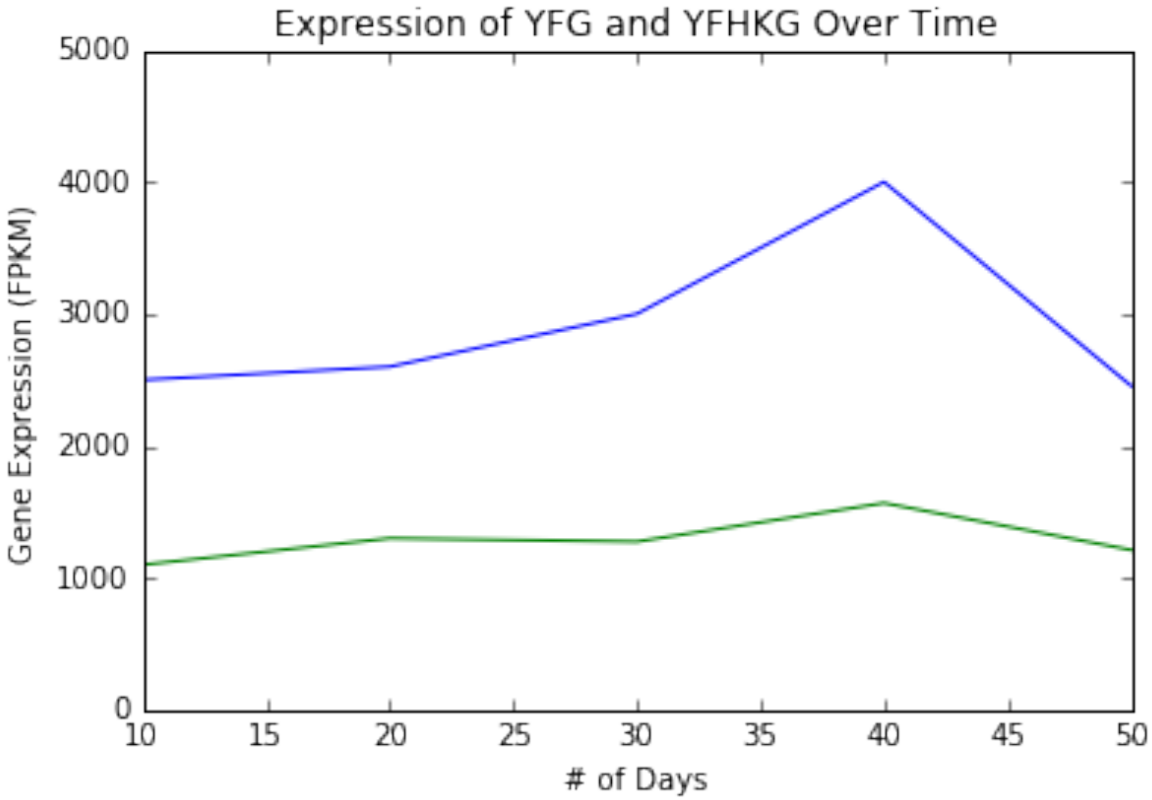
Example Result



Example: Changing Axes

```
%matplotlib inline
import matplotlib.pyplot as plt

days = [10, 20, 30, 40, 50]
YFG = [2500, 2600, 3000, 4000, 2455]
YFHKG = [1100, 1300, 1275, 1567, 1212]

plt.plot(days, YFG)
plt.plot(days, YFHKG)
plt.xlabel("# of Days")
plt.ylabel("Gene Expression (FPKM)")
plt.title(”Expression of YFG and YFHKG Over Time")
plt.axis([10,50, 0, 5000])
plt.show()
```

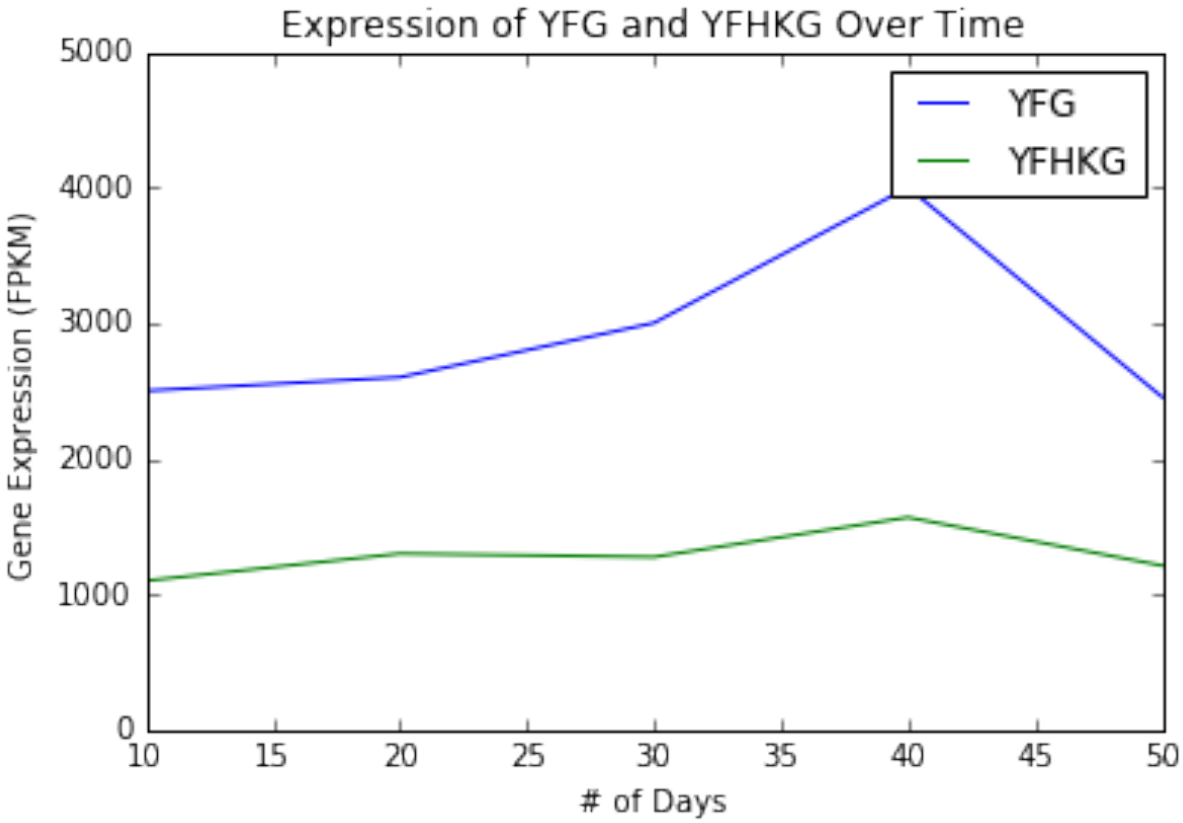
Example Result



Example: Changing Axes

```
%matplotlib inline
import matplotlib.pyplot as plt

days = [10, 20, 30, 40, 50]
YFG = [2500, 2600, 3000, 4000, 2455]
YFHKG = [1100, 1300, 1275, 1567, 1212]

plt.plot(days, YFG, label="YFG")
plt.plot(days, YFHKG, label="YFHKG")
plt.xlabel("# of Days")
plt.ylabel("Gene Expression (FPKM)")
plt.title("Expression of YFG and YFHKG Over Time")
plt.axis([10,50, 0, 5000])
plt.legend()
plt.show()
```

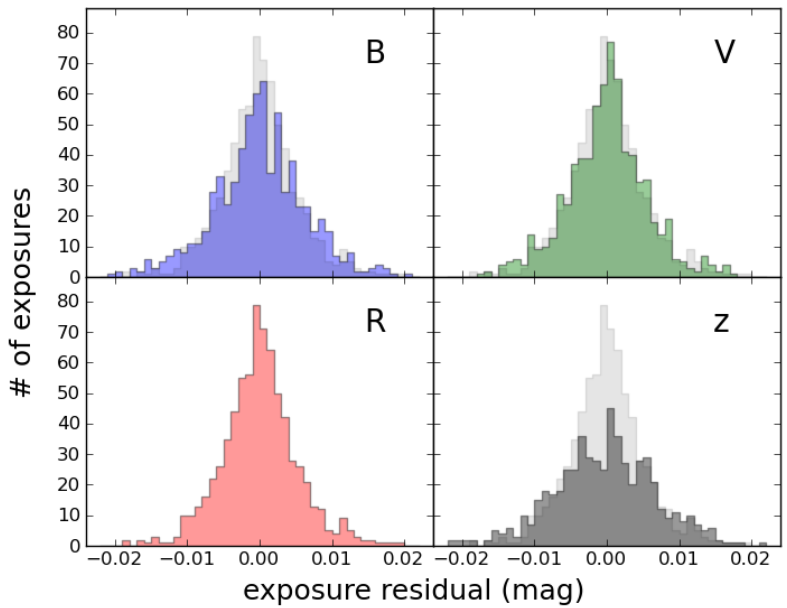
Example Result

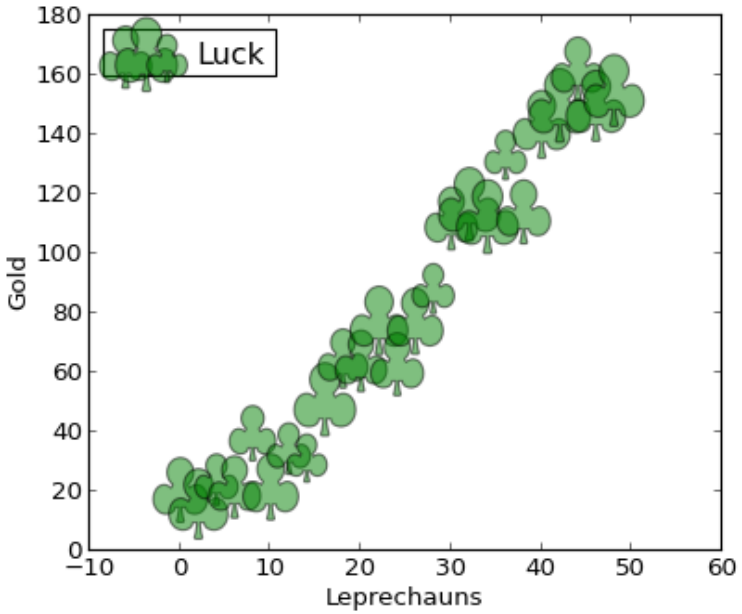


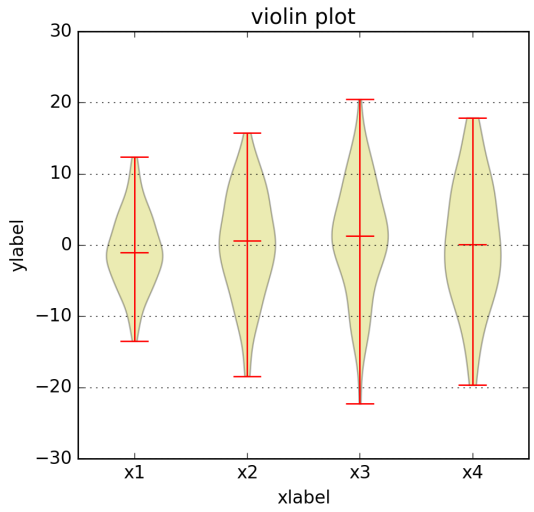
Notes on matplotlib

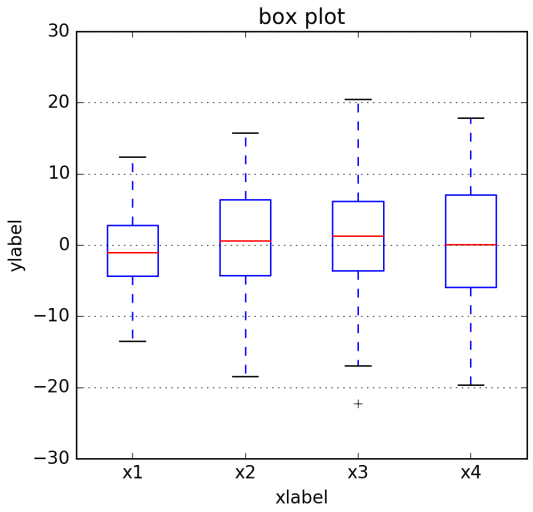
- Visualization is highly customizable
  - Line color, type, width
  - Error bars
  - Legend attributes
  - Axes attributes
  - Double y-axes, etc
- Examples of specific types of figures and their implementation are available online
- Can create beautiful figures fast
- Plotting with code is *impressive*

Matplotlib examples









2. optparse

optparse

Purpose: provides flagged input functionality to scripts

1. Input arguments can be in any order
2. Resulting command is human read-able
3. Allows the user to query input arguments and "help"

optparse

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option("-f", "--file", dest="filename",
                  help="write report to FILE”)
parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose”,
                  default=True,
                  help="don't print status messages")

(options, args) = parser.parse_args()
```

optparse

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option("-f", "--file", dest="filename",
                  help="write report to FILE”)
parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose”,
                  default=True,
                  help="don't print status messages")

(options, args) = parser.parse_args()
```

optparse

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option("-f", "--file", dest="filename",
                  help="write report to FILE")
parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose",
                  default=True,
                  help="don't print status messages")

(options, args) = parser.parse_args()
```

optparse

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option("-f", "--file", dest="filename",
                  help="write report to FILE”)
parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose”,
                  default=True,
                  help="don't print status messages")

(options, args) = parser.parse_args()
```

optparse

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option("-f", "--file", dest="filename",
                  help="write report to FILE”)
parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose”,
                  default=True,
                  help="don't print status messages")

(options, args) = parser.parse_args()
```

Example optparse

- **Problem:**
  - You have a script that computes gene lengths of set a genes given an annotation file that contains genomic positions of both introns and exons. You want to include a parameter that would change the functionality to only include exons in gene length when desired.

optparse exmple: CDS length

```
from optparse import OptionParser

parser = OptionParser()
```

optparse exmple: CDS length

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option(”-g", "--geneAnnotation",
                  dest=”annot",
                  help=”UCSC gene annotation file”)
```

optparse example: CDS length

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option(”-g", "--geneAnnotation",
                  dest=”annot",
                  help=”UCSC gene annotation file”)
parser.add_option(”-c", "--coding",
                  action="store_true", dest=”coding”,
                  default=False,
                  help=”flag to only inlude exons in gene
   length computation")
```

optparse exmple: CDS length

```
from optparse import OptionParser

parser = OptionParser()
parser.add_option(”-g", "--geneAnnotation",
                  dest=”annot",
                  help=”UCSC gene annotation file”)
parser.add_option(”-c", "--coding",
                  action="store_true", dest=”coding”,
                  default=False,
                  help=”flag to only inlude exons in gene
   length computation")

(options, args) = parser.parse_args()
```

optparse: interaction (help)

```
python compute_length.py --help
```

or

```
python compute_length.py --help
```

Usage: compute_length.py [options] arg1 arg2

Options:

-h, --help	show this help message and exit
-g, --geneAnnotation	UCSC gene annotation file
-c, --coding	flag to only include exons in gene length computation

optparse: interaction (inputs)

```
python compute_length.py –-geneAnnotation
UCSC.gtf --coding
```

or

```
python compute_length.py –g UCSC.gtf –c
```

Usage: compute_length.py [options] arg1 arg2

Options:

-h, --help show this help message and exit

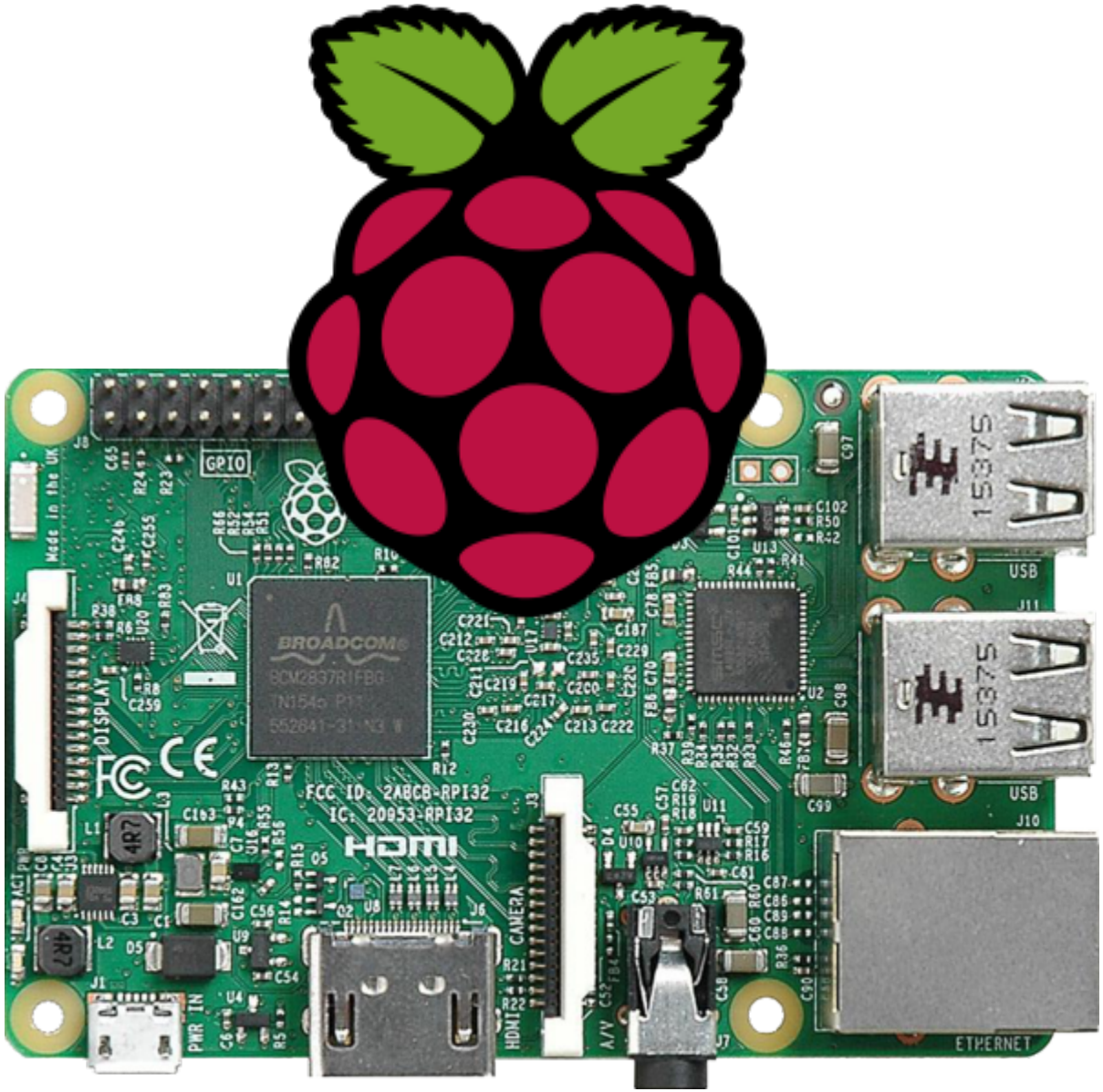
-g, --geneAnnotation UCSC gene annotation file

-c, --coding flag to only include exons in gene length computation

optparse: using inputs

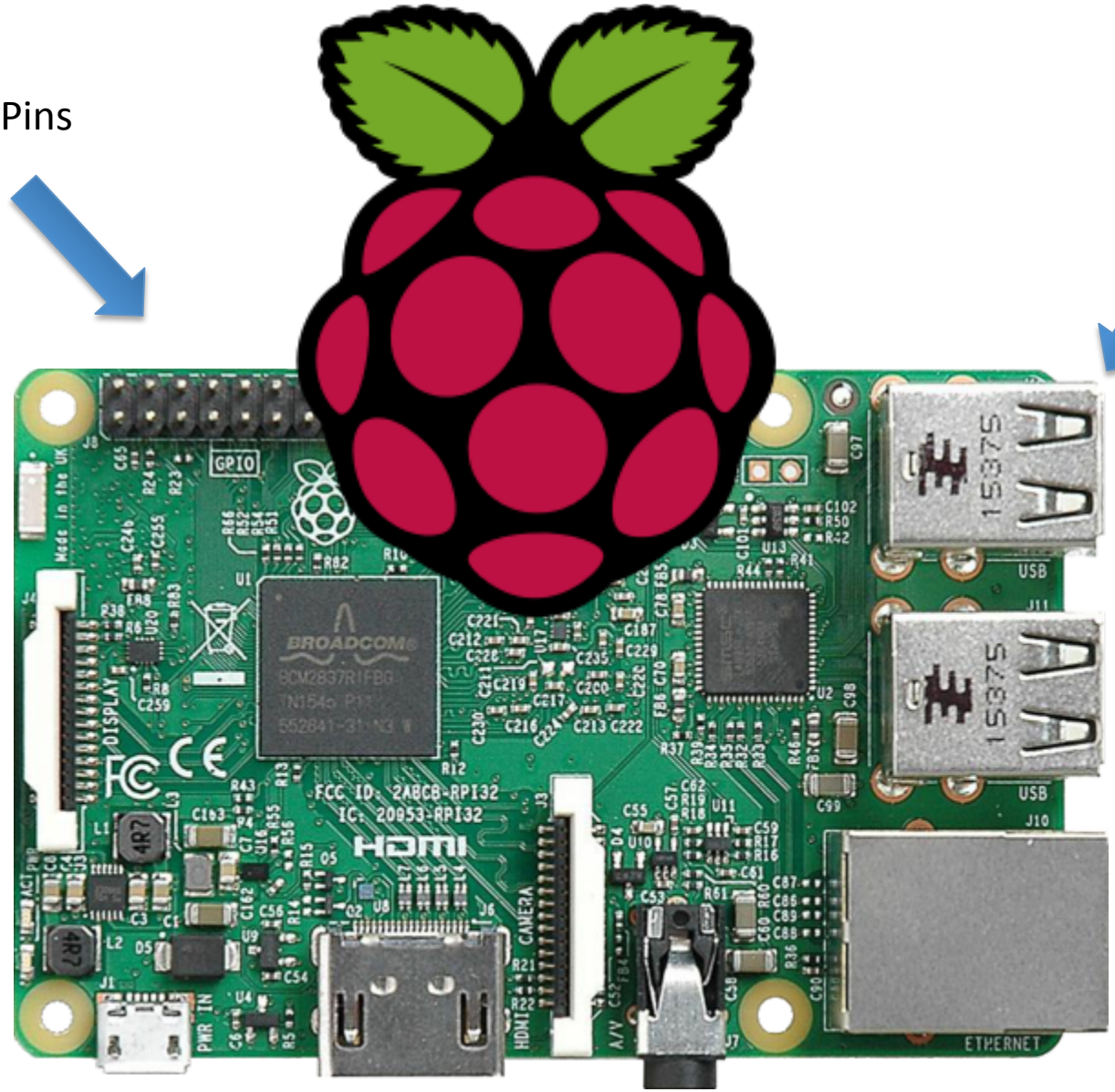
```
(options, args) = parser.parse_args()

gene_file = options.annot

if options.coding == True:
        print “Exclude introns from computation”
```

3. Learning with RaspberryPi

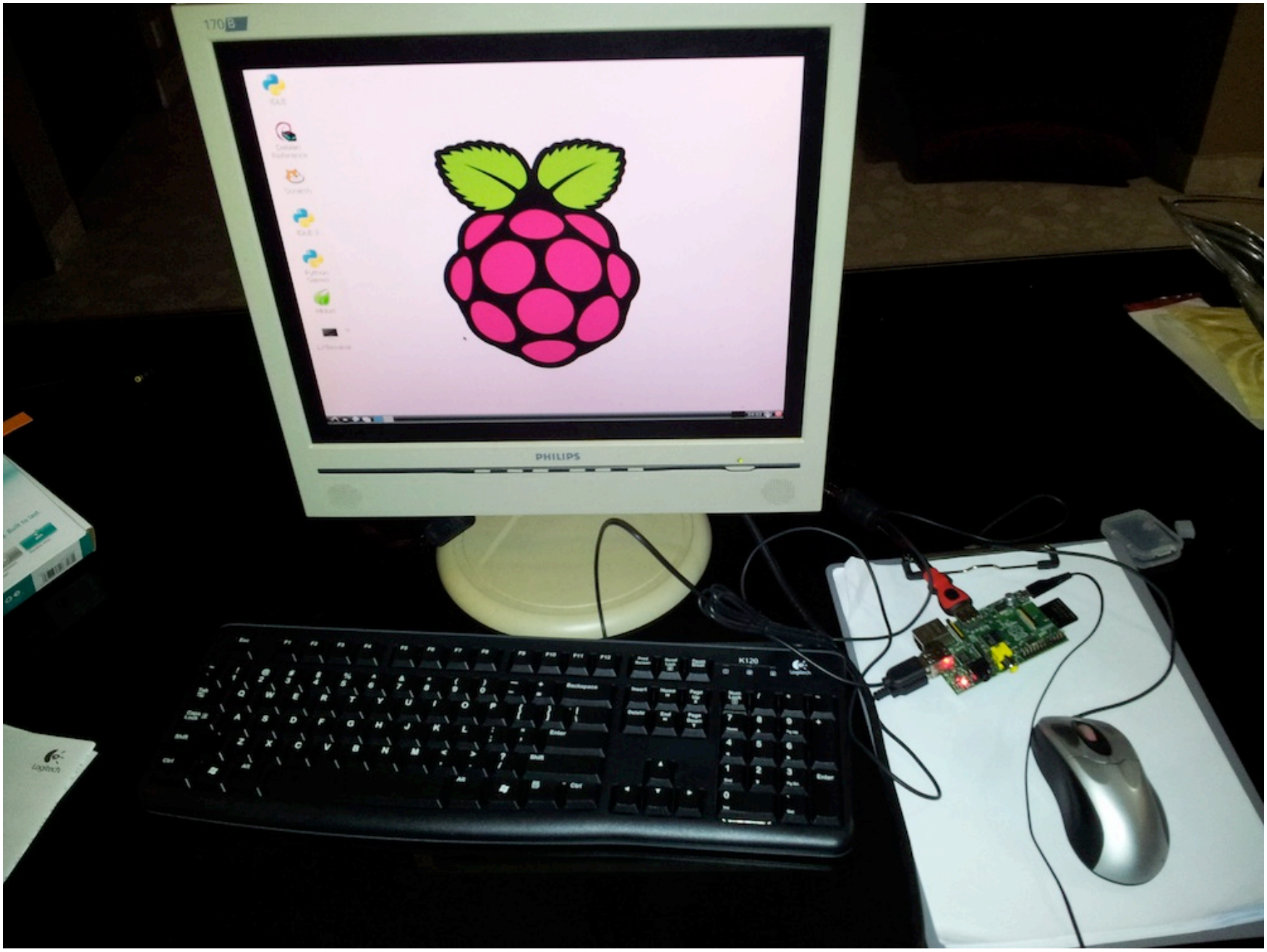

GPIO
BROADCOM
BCM2837B0IFBG
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI
CAMERA
DISPLAY
USB
USB
ETHERNET
PWR IN
A/V
HDMI

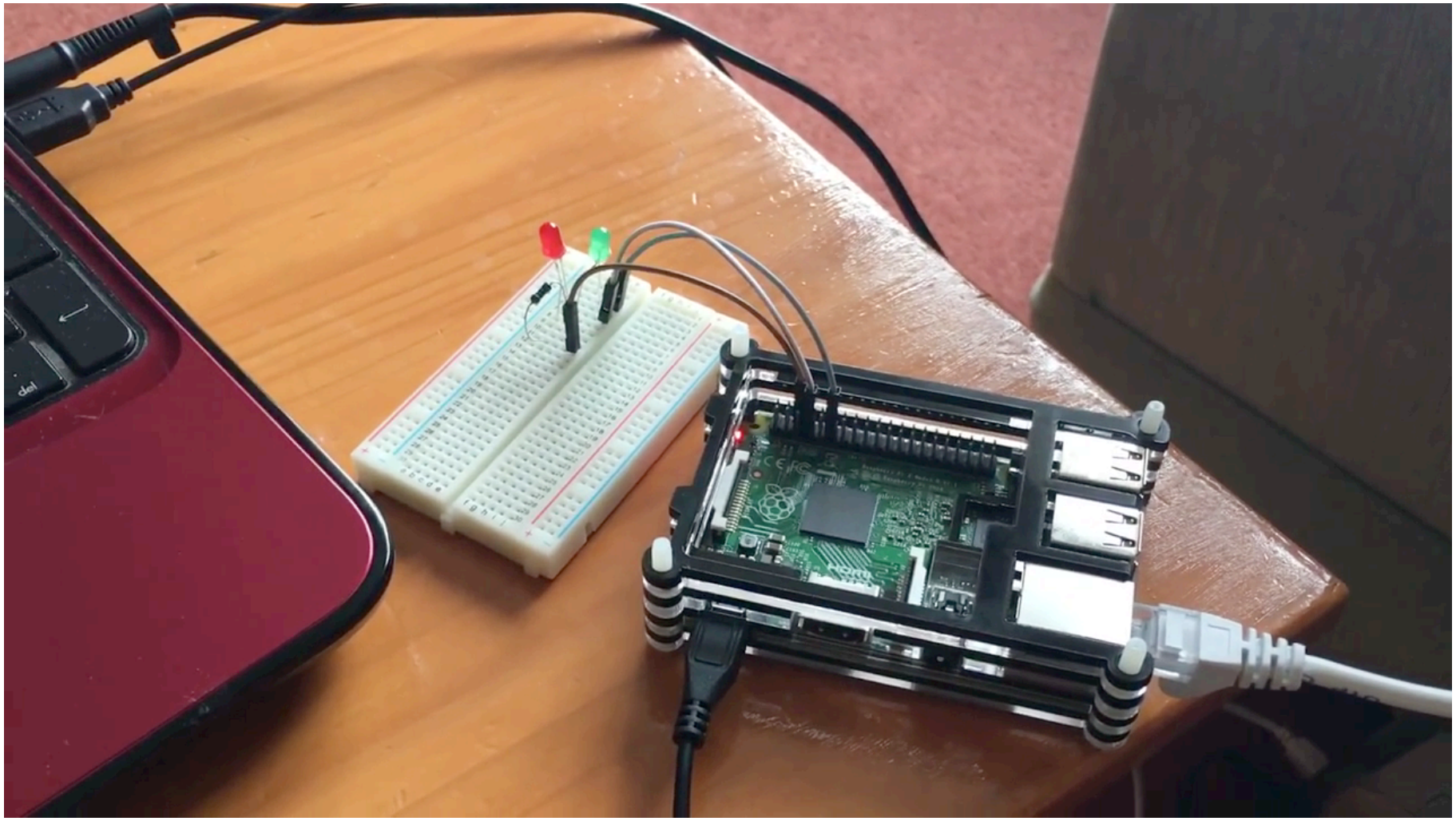
GPIO Pins

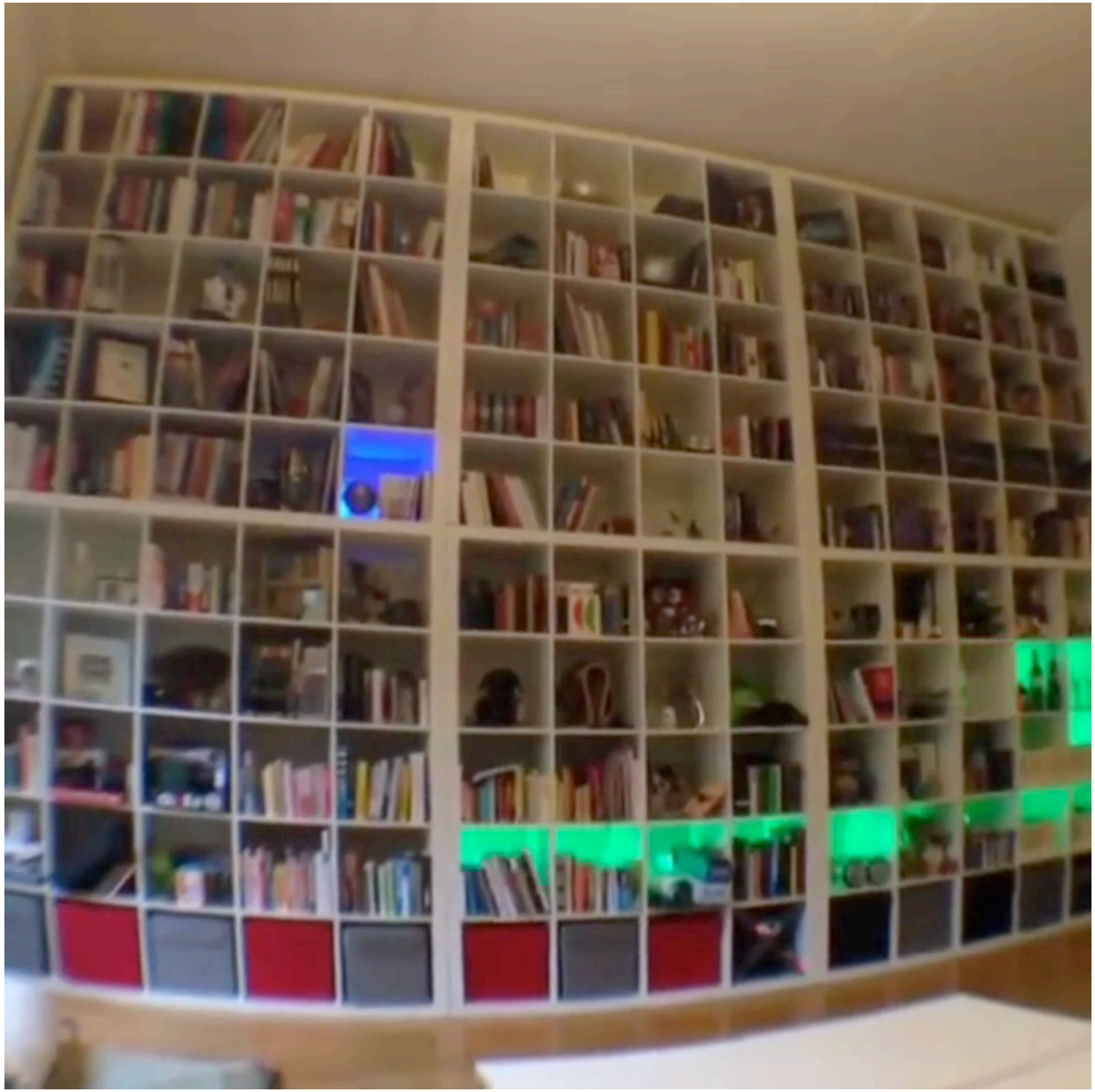
USB Ports



Ethernet
WiFi
Bluetooth


170B
PHILIPS
K120





Raspberry Pi

- 35$ for computer itself
- ~50-75$ for beginner kits (breadboard, wires, accessories, etc)
- Multitude of lesson plans and tutorials online to help learn python and to build cool projects