## INTRODUCTION

Kids -

Hi! this is a document about our summer Python Programming and Machine Learning project.

The first part is Python, which is the computer language we will use. Python is a very modern and active language right now. I think Python is installed on all of your computers using a software management tool called Anaconda.

We will also be using the program "git" to do "version control" of our stuff. Git is also a very modern and active tool that I am sure you will all be using at work. Version control means you can save your files, make a change, save it again, and then get back to the previous (or any) version. Git is also famously good at maintaining different branches (or versions) of computer files and then merging them together.

As a preview, we will be using two modern and active Machine Learning libraries - SciKit Learn and Google TensorFlow. Within the TensorFlow world we will be using Keras and possibly McFly.

## LET'S GET READY

Let's check that we have the right Python. Start a command prompt and type

```
  $ conda -h
```

See explanation below

To start a command prompt you can click on the "Start" button probably at the lower left of the screen, then just type "cmd" (without the quotes) and press the "Enter" key.

A black window with text (probably white text) will appear. The last line will have a string followed by ">".
For instance, my computer says C:\Users\mdo>. This string is the "prompt", which I will usually denote with
a "$" as I did above in "$ conda -h". The $ means wait for the prompt and then type what follows.
In this window type "conda -h" (without the quotes). It should type a bunch of lines and then end with something like this:

```
optional arguments:
  -h, --help     Show this help message and exit.
  -V, --version  Show the conda version number and exit.
```

That means that Anaconda (see above) is installed on your computer. If not, tell me and we will get it installed.

Now do $ conda info --envs

You should see at least one line that says "root" and maybe some other lines. If you see a line that says py35 and it has a "*" next to it, that is what we want and you can skip to the "LET'S PROGRAM" part.

If you see a line that says py35 but no "*" next to it, skip to "SELECT PY35 ENVIRONMENT"

Otherwise we will create a py35 environment.

## Create py35 Environment

Let’s use Python 3.5.3 for now since it is known to work with all the libraries we want. In that black “cmd” or “Command Prompt” window do the following:

```
conda create --name py35 python=3.5.3 Anaconda
```

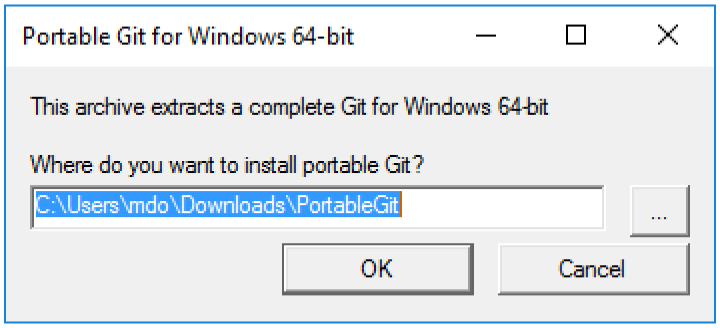
You will get a prompt Proceed ([y]/n); type y and press enter

## Get Git

Git is not an acronym; it is a program to “version control” files (programs and data). We will download portable Git, which can be moved anywhere. Carl – you will want to install this on your D: drive to save space on your C: drive. Git also includes a version of the Unix/Linux “bash” which is a powerful command line shell.

Use your web browser (Chrome) to go to [https://git-scm.com/download/win](https://git-scm.com/download/win)

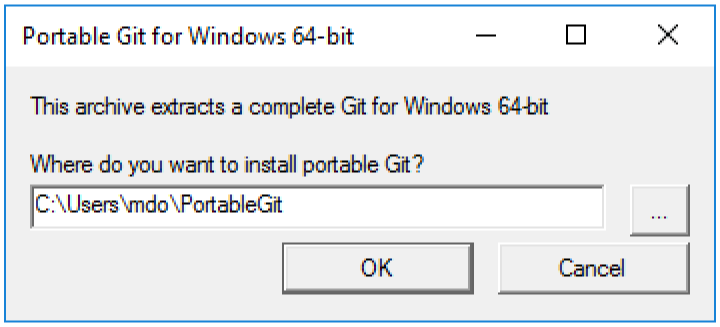
It will automatically download a file that would install Git, but we will click on “64-bit Git for Windows Portable” in “Git for Windows Portable ("thumbdrive edition")”. After this in your Downloads directory (Example: C:/Users/mdo/Downloads; on your computer it would say for example Trishie instead of mdo) you will find PortableGit-2.13.2-64-bit.7z.exe. Double-click on this and it will put up a dialog box like the following. We will edit the path by removing \Downloads


Portable Git for Windows 64-bit

This archive extracts a complete Git for Windows 64-bit

Where do you want to install portable Git?

C:\Users\mdo\Downloads\PortableGit


Portable Git for Windows 64-bit

This archive extracts a complete Git for Windows 64-bit

Where do you want to install portable Git?

C:\Users\mdo\PortableGit

You can now navigate to C:\Users\mdo\PortableGit, find git-bash.exe, right-click-drag it to the desktop, and select "Create a Shortcut". Talk to me if that seems confusing.

## LET'S PROGRAM

### Start Git Bash and Select py35 Environment

Double click on that "Git Bash" icon on the desktop.

Now in that black "cmd" window (says MINGW64) type

```
conda info --envs
source activate py35
python --version
```

Note that on several lines we are doing two "-"; Word has a tendency to change this to one special symbol. After the last line it should say something like "Python 3.5.3 :: Anaconda 4.4.0 (64-bit)"

If it is not there already let's get one other package we want; type the following and give it a "y" for yes:

```
conda install beautifulsoup4
```

It is OK if it tells you it is already installed

### Jupyter Notebook First Steps

Just a simple Python program. First make a place to store your program.

```
cd ~
pwd
mkdir python
cd python
pwd
```

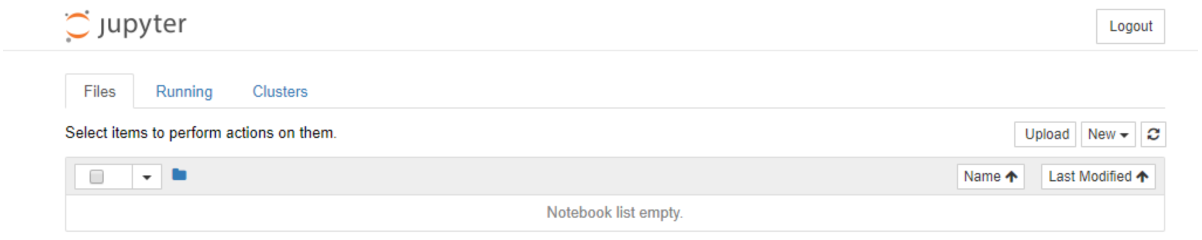
"~" is a shorthand for the home directory (for instance c:\Users\mdo). "cd" stands for change directory so it moves the "working directory" (where we are) to the home directory.

"pwd" is print (to the screen) working directory. Note that Windows thinks we are in c:\Users\mdo but our bash shell thinks we are in /c/Users/mdo; these are different ways of saying the same thing. There are advantages to using the forward slash “/” instead of the backward slash “\”; the backward slash is a specially-processed character in many situations.

“mkdir python” will make a directory named python starting at the current working directory. “cd python” moves there. “pwd” again shows where we are, “/c/Users/mdo/python”.

Now start a notebook. This will open your web browser if not already open. Note the spelling

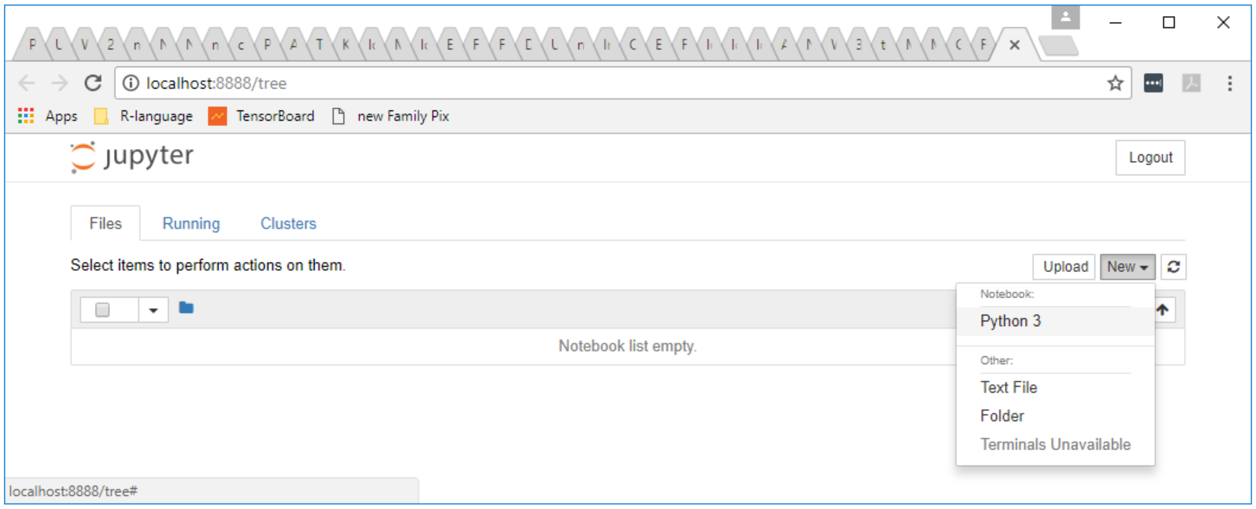
```
jupyter notebook &
```



Select items to perform actions on them.

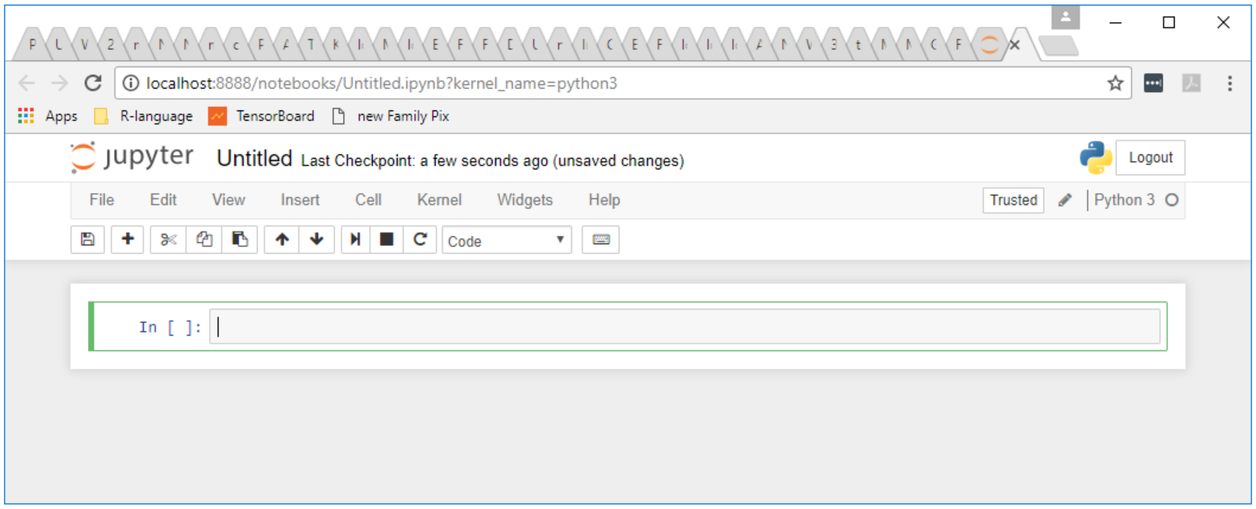
Notebook list empty.

We go to the right and click on “New”. Then select “Python 3” because we will be making a Python notebook.



Select items to perform actions on them.

Notebook list empty.

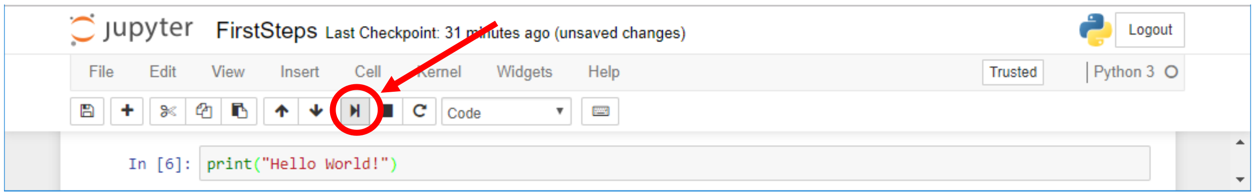


The first thing we should do is give our workbook a better name. Let’s call it FirstSteps. Double-click on the “Untitled” and in the “Rename Notebook” dialog type in FirstSteps.

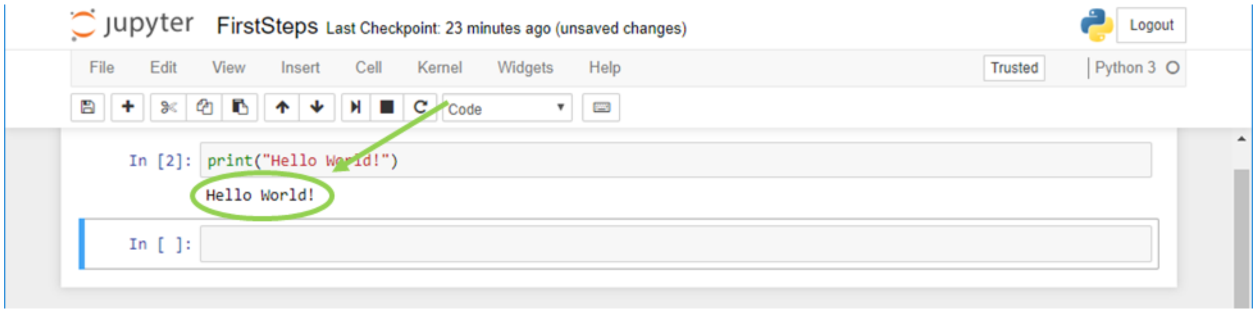
Now click inside the box labeled “In” and type

print(“Hello World!”)

To make this short program run, while still in that box, hold down the SHIFT key while pressing ENTER. Alternatively, click the “run cell” button highlighted below.

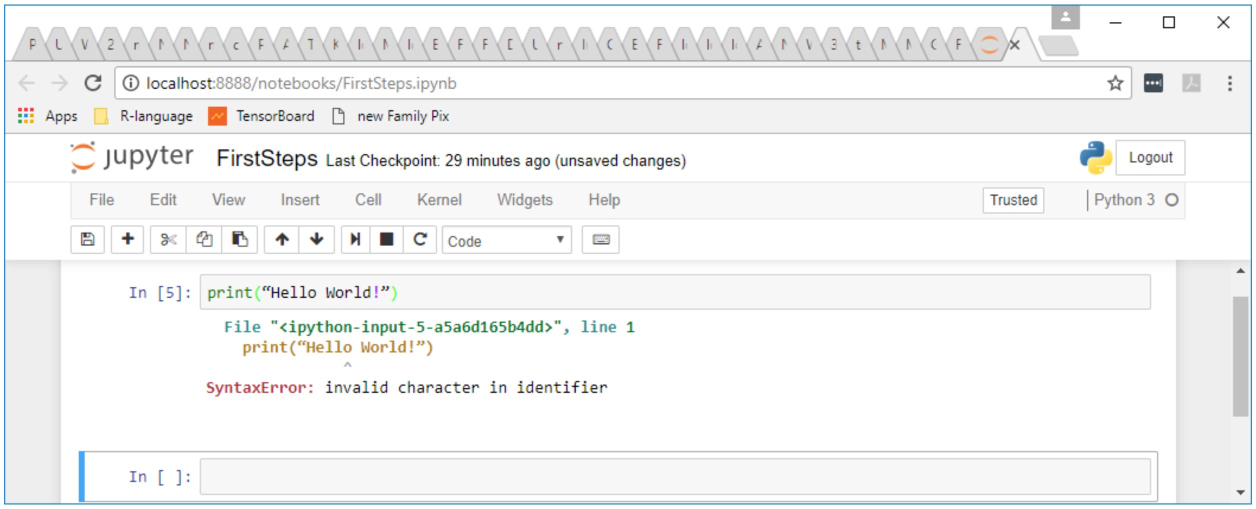

jupyter FirstSteps Last Checkpoint: 31 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Code

```
In [6]: print("Hello World!")
```


jupyter FirstSteps Last Checkpoint: 23 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Code

```
In [2]: print("Hello World!")
        Hello World!

In [ ]:
```

Note that if you copy/paste from this Word document into that cell, it will use different double-quotes (“”) and Python will not understand what you intended; see below for what this would look like:


localhost:8888/notebooks/FirstSteps.ipynb
Apps
R-language
TensorBoard
new Family Pix
jupyter FirstSteps Last Checkpoint: 29 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Code

```
In [5]: print(“Hello World!”)
          File "<ipython-input-5-a5a6d165b4dd>", line 1
            print(“Hello World!”)
                  ^
        SyntaxError: invalid character in identifier

In [ ]:
```

Python tries to identify the first character associated with the error but in this case it doesn’t give a human a clue as to the problem. The problem is the “curvy” double-quotes.

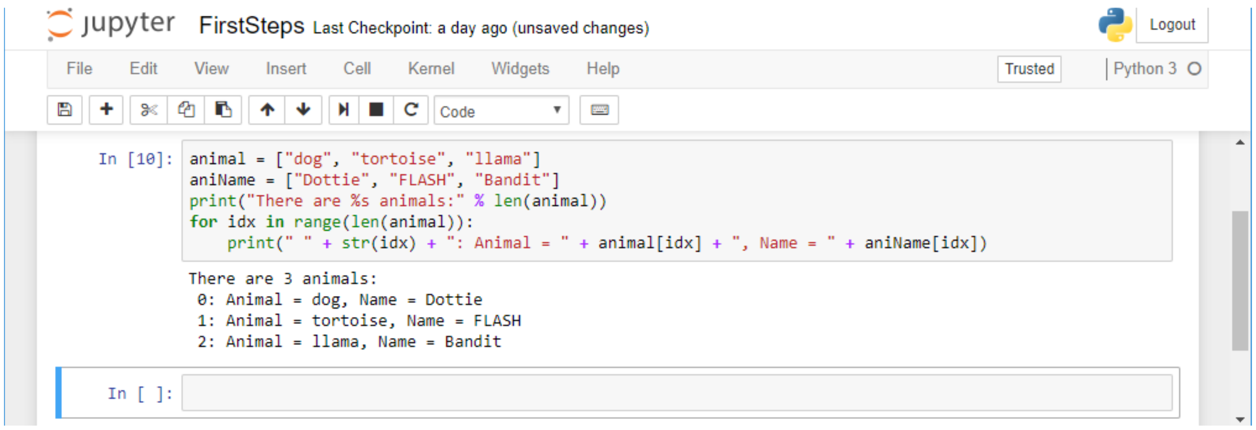
## Python List

Let’s make a couple of Python lists (same as arrays in some languages). Note that “ENTER” without shift just keeps going in a notebook input box. In the empty input box type

```
animal = [“dog”, “tortoise”, “llama”]
aniName = [“Dottie”, “FLASH”, “Bandit”]
print(“There are %s animals:” % len(animal))
for idx in range(len(animal)):
        print(“  “ + str(idx) + “: Animal = “ + animal[idx] + “, Name = “ + aniName[idx])
```

Watch out for the quote marks being changed by Word in the text above.

Type it in then run it. We will discuss this a little; there are several important points in here.

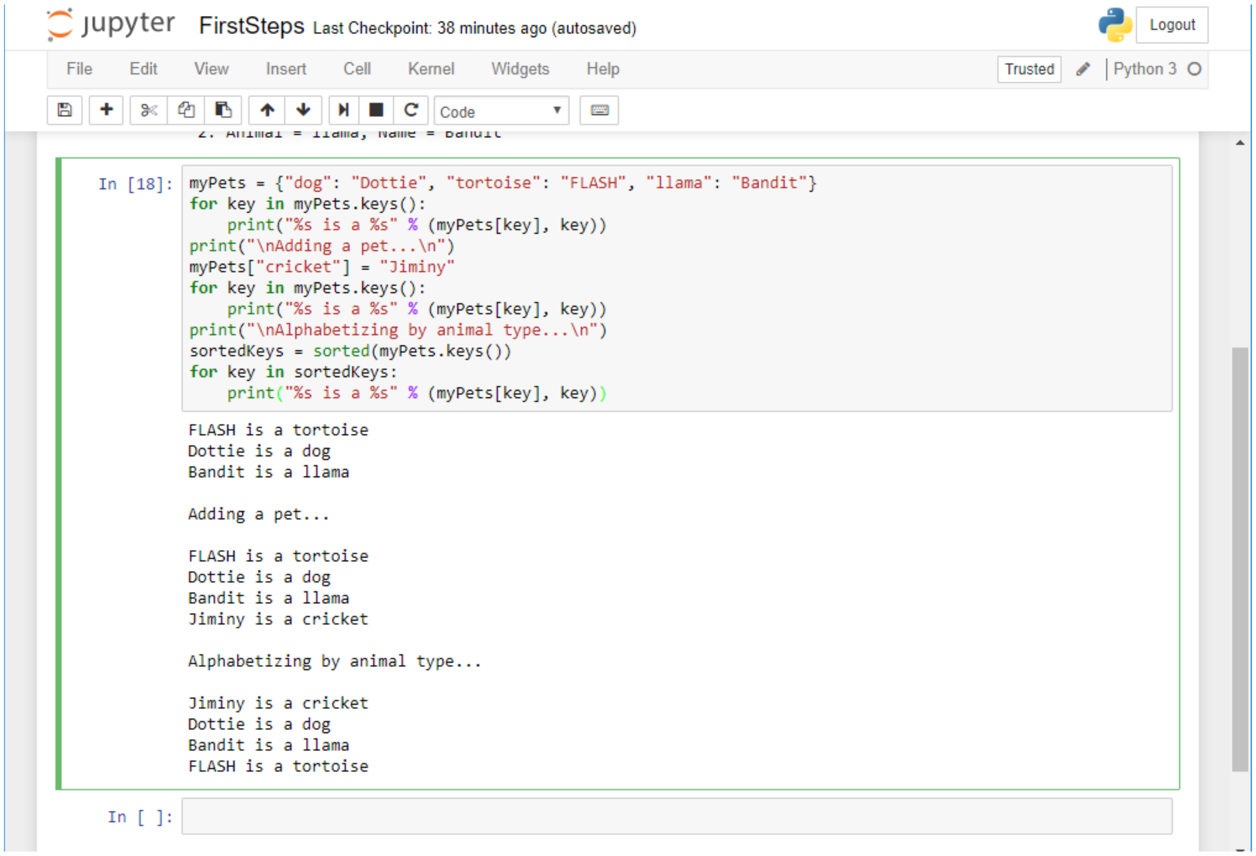


```
In [10]: animal = ["dog", "tortoise", "llama"]
         aniName = ["Dottie", "FLASH", "Bandit"]
         print("There are %s animals:" % len(animal))
         for idx in range(len(animal)):
             print(" " + str(idx) + ": Animal = " + animal[idx] + ", Name = " + aniName[idx])

         There are 3 animals:
          0: Animal = dog, Name = Dottie
          1: Animal = tortoise, Name = FLASH
          2: Animal = llama, Name = Bandit

In [ ]:
```

## Python Dictionary

A Python "dictionary" is an extremely powerful feature. A Python dictionary associates one value (a key) with another particular value, just like a language dictionary associates a word with a definition. Here is a quick demo:

```
myPets = {"dog": "Dottie", "tortoise": "FLASH", "llama": "Bandit"}
for key in myPets.keys():
   print("%s is a %s" % (myPets[key], key))
print("\nAdding a pet...\n")
myPets["cricket"] = "Jiminy"
for key in myPets.keys():
   print("%s is a %s" % (myPets[key], key))
print("\nAlphabetizing by animal type...\n")
sortedKeys = sorted(myPets.keys())
for key in sortedKeys:
   print("%s is a %s" % (myPets[key], key))
```



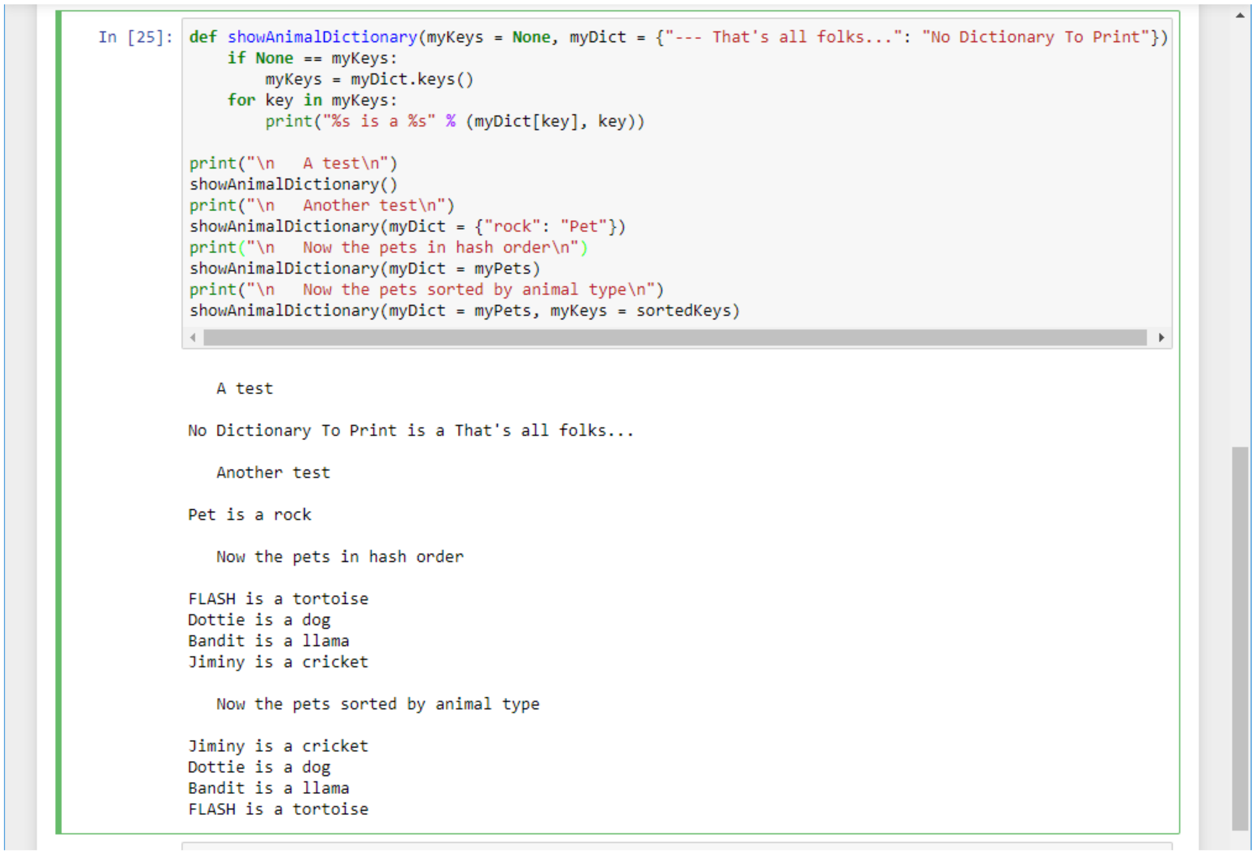
```
In [18]: myPets = {"dog": "Dottie", "tortoise": "FLASH", "llama": "Bandit"}
         for key in myPets.keys():
             print("%s is a %s" % (myPets[key], key))
         print("\nAdding a pet...\n")
         myPets["cricket"] = "Jiminy"
         for key in myPets.keys():
             print("%s is a %s" % (myPets[key], key))
         print("\nAlphabetizing by animal type...\n")
         sortedKeys = sorted(myPets.keys())
         for key in sortedKeys:
             print("%s is a %s" % (myPets[key], key))

         FLASH is a tortoise
         Dottie is a dog
         Bandit is a llama

         Adding a pet...

         FLASH is a tortoise
         Dottie is a dog
         Bandit is a llama
         Jiminy is a cricket

         Alphabetizing by animal type...

         Jiminy is a cricket
         Dottie is a dog
         Bandit is a llama
         FLASH is a tortoise

In [ ]:
```

Note that just like a language dictionary is in alphabetical order, a Python dictionary is in an order too: a "hash" order. This means that the keys are stored in a scrambled order that is impossible to predict. If you want a particular order you need to sort in that order. Also think about what you want the key to be; in our case  it is not too hard to print in order of the animal type (our key) but if we wanted to print out by the name it would be trickier.

## Python Subroutine

A subroutine can be “called” with different values. We could make a subroutine to print out info on the animal dictionary:

```
def showAnimalDictionary(myKeys = None, myDict = {"--- That's all folks...": "No Dictionary To Print"}):
  if None == myKeys:
    myKeys = myDict.keys()
  for key in myKeys:
    print("%s is a %s" % (myDict[key], key))

print("\n   A test\n")
showAnimalDictionary()
print("\n   Another test\n")
showAnimalDictionary(myDict = {"rock": "Pet"})
print("\n   Now the pets in hash order\n")
showAnimalDictionary(myDict = myPets)
print("\n   Now the pets sorted by animal type\n")
showAnimalDictionary(myDict = myPets, myKeys = sortedKeys)
```



```
In [25]: def showAnimalDictionary(myKeys = None, myDict = {"--- That's all folks...": "No Dictionary To Print"})
             if None == myKeys:
                 myKeys = myDict.keys()
             for key in myKeys:
                 print("%s is a %s" % (myDict[key], key))

         print("\n   A test\n")
         showAnimalDictionary()
         print("\n   Another test\n")
         showAnimalDictionary(myDict = {"rock": "Pet"})
         print("\n   Now the pets in hash order\n")
         showAnimalDictionary(myDict = myPets)
         print("\n   Now the pets sorted by animal type\n")
         showAnimalDictionary(myDict = myPets, myKeys = sortedKeys)
```

```

   A test

No Dictionary To Print is a That's all folks...

   Another test

Pet is a rock

   Now the pets in hash order

FLASH is a tortoise
Dottie is a dog
Bandit is a llama
Jiminy is a cricket

   Now the pets sorted by animal type

Jiminy is a cricket
Dottie is a dog
Bandit is a llama
FLASH is a tortoise
```

## LET'S START ON THE FIRST PROBLEM – HOUSING PRICES AND SCIKIT-LEARN

Originally the plan was just to do the stock market portion of this project, but when I bought “Hands-On Machine Learning with SciKit-Learn and TensorFlow” by Aurelien Geron I saw that he did such a fantastic job of showing the entire process in one example that I should use it as the first part and move the stock market stuff to the second part. Also, this allowed including a section on regression that tied in with our previous Google Causal Impact exploration.

The housing exercise is mostly taken from Chapter two of A. Geron’s book.