**Getting to know your Raspberry Pi**

Welcome to the beginning of robotics / python section of intro to engineering. In the next couple of weeks you will learn about coding in python, working with a Linux system, using sensors and more. But do not fear you can do this.

Turning on the Raspberry Pi, & connecting via VNC

To begin grab a raspberry pi (aka raspi). On the raspi you will find a label with a number (ranging for 1-10) on the raspi; actually you will find it three separate places. The first should be the bottom of the body, the second should be on the sd card and the last should be on the Wi-Fi dongle. All these number should / need to be the same.

Now that you have the raspi, you need to know how to turn it on and connect to it. Some of you may be thinking that you need to grab a keyboard and mouse, if so you are wrong. We are going to connect to the raspi remotely via the internet with the use of a VNC sever.

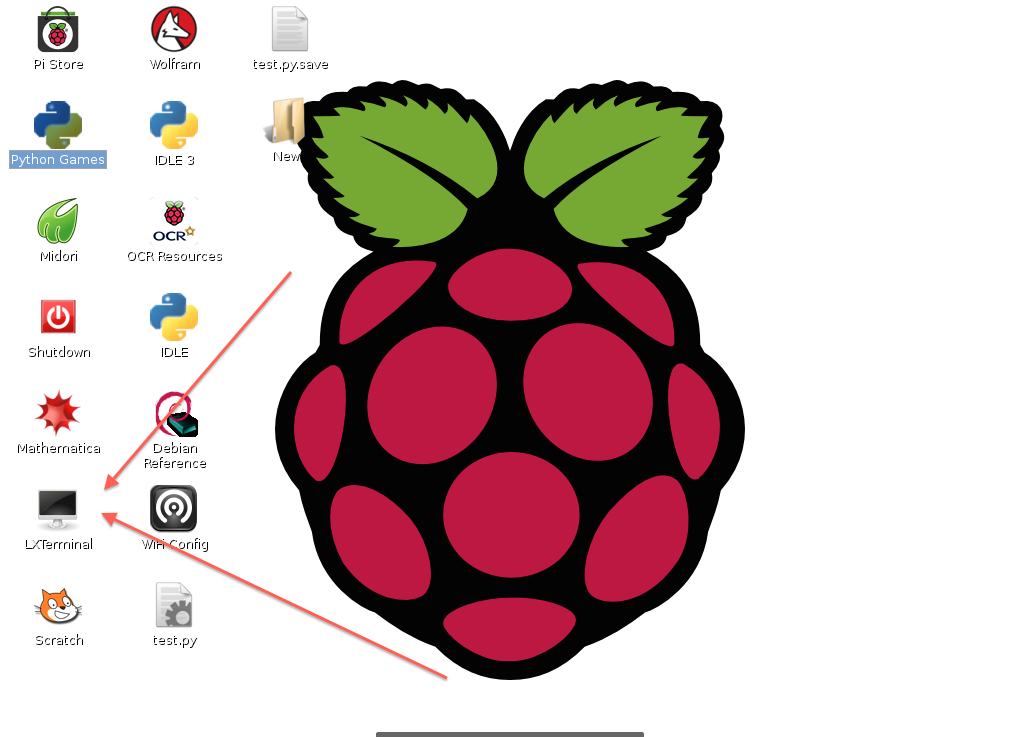
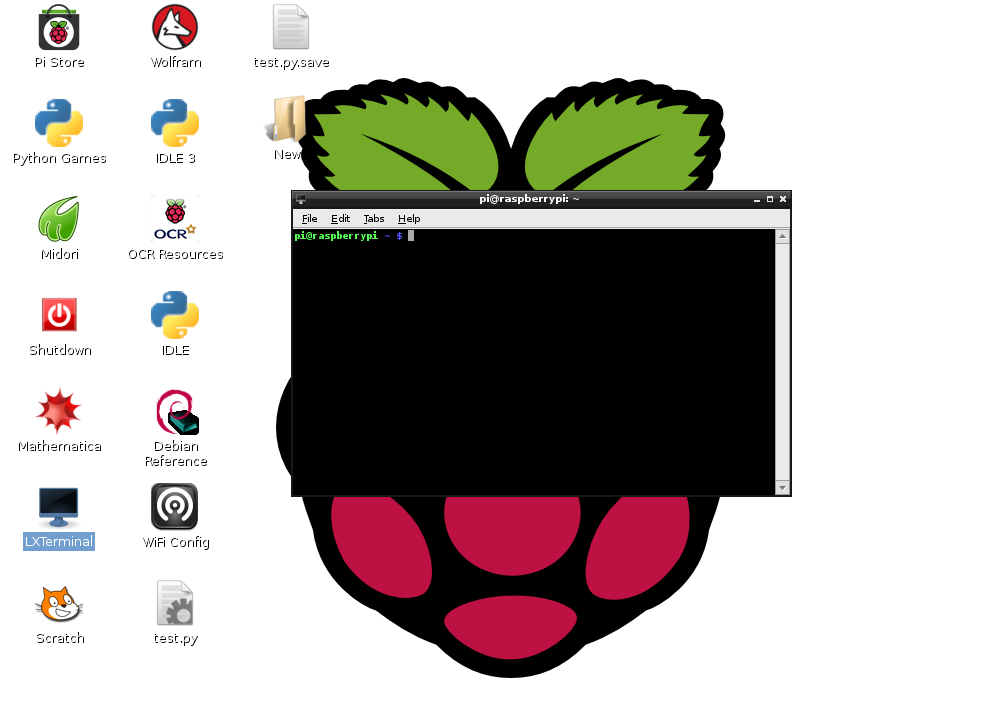
To use a VNC server, you need to download a VNC viewer on a computer you wish to program from .(either a lab computer or your personal one will work even if it’s a mac) Google has a great app for this, just search for “ chrome VNC viewer” in a Google search and it should pop up. Once you have downloaded and open the VNC viewer, you will notice that it requires an ip address to connect to anything, you will get to this next.

Now that you have a VNC viewer, you need to grab a power cable and look up the ip address assigned to the raspi number. The ip address should either be written on the pi itself or on a piece of paper that has all the ip address.

With the ip address in hand, plug in the raspi and wait until the green flashing light is only occasionally blinking (this takes about 20 seconds). Now on the VNC viewer type in the ip address with “:1” at the end. For example < 10.106.47.75:1 >. If you type in the address correctly a password prompt should pop up. The password is simple “ **raspberry** “. You should now have complete remote access to the raspi.

Introduction to the command screen and navigation:

After staring up your raspi and are on the desktop screen, click on the *LXTerminal* to open up the command window. Your screen should look something like the pictures below.



Now that you are in the command window it is time to introduce you to some simple Linux commands and terminology for navigating around your raspi.

Call Directory – This is essentially a folder on a graphical display. A call directory is not a program or a function but instead a place that stores them. In order to run any program you must first be in the call directory containing that program. We will cover this in more detail later on.

Macintosh HD:Users:Christian:Desktop:Screen Shot 2014-07-13 at 1.07.41 PM.png**ls** – You can think of this as “list” because that is essentially what it does. When you type this into the command line and hit enter, a list of files in your current directory will pop up. (See picture below) The blue names are other directories inside your current ones. If any names appear as white they are programs that can be run in your current directory.

**cd “NAME\_OF\_DIRECTORY”** –This command allows you to go into directories, just like clicking on the folder in a graphical display. You can go through multiple directories in one command using something like < cd name1/name2/name3 >. This command would get you into the “name3” directory. Below is an example of using the command.

Macintosh HD:Users:Christian:Desktop:Screen Shot 2014-07-13 at 1.38.37 PM.png

Your turn:

Using <cd>, and <ls> commands, find a directory called “ Sensor\_Examples “ and copy down the path. Hint: one of the directories has python in the name and it’s not python-games.

Back To Common Commands.

**rm “NAME\_OF\_FILE”** – The removal command, which allows you to delete a file in your current directory. For example < rm test.py >. Only use it on files that you have created.

**cp “Name\_Of\_File\_To\_Copy” “Name\_Of\_The\_New\_File”** – This command makes a copy of a file with a different name, within the same directory. This is helpful when writing a program and you want to test something out that you are not quite sure will work. See picture below for example.

Macintosh HD:Users:Christian:Desktop:Screen Shot 2014-07-13 at 1.41.22 PM.png

**sudo reboot –** this reboots the pi.

**sudo poweroff –** this command turns off the pi, and it is one of the easiest ways to turn off the pi. **DO NOT JUST UNPLUG THE PI.**

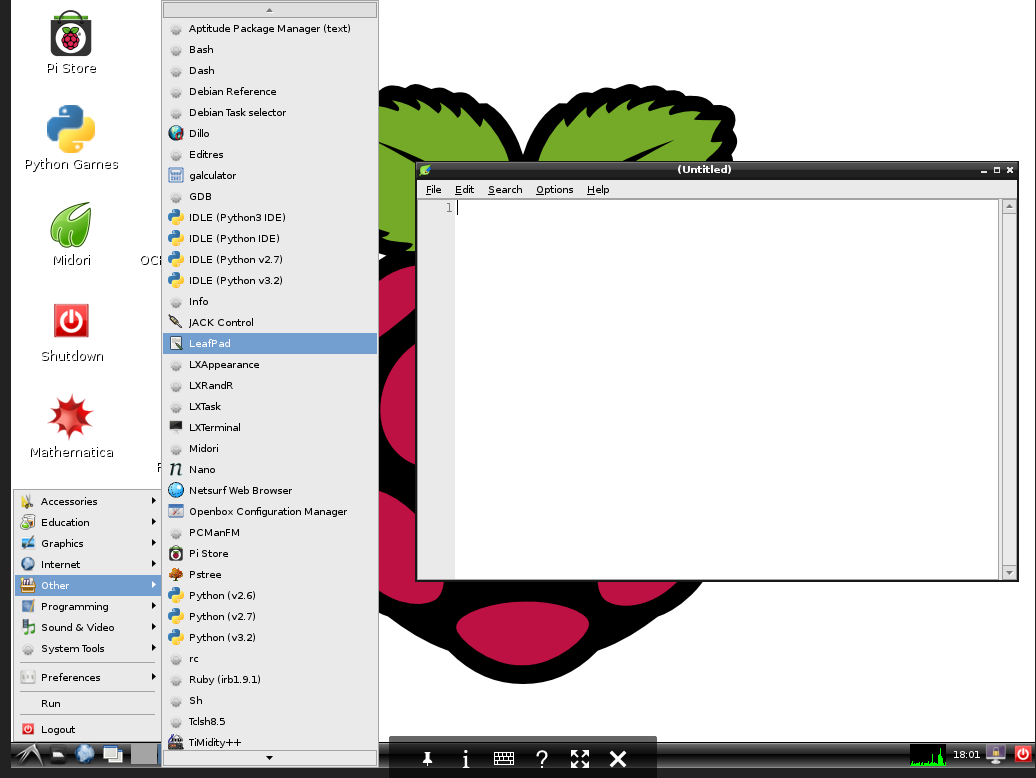
Writing a program in the command window:

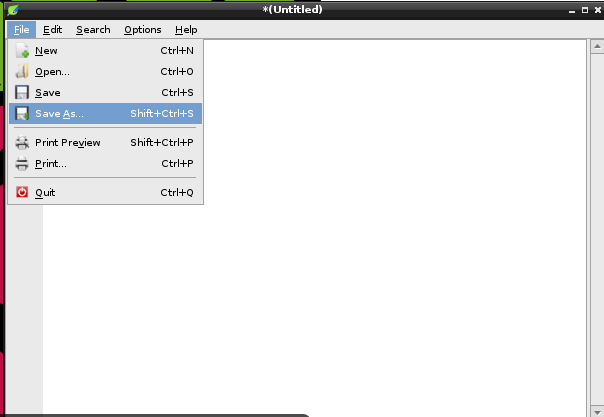
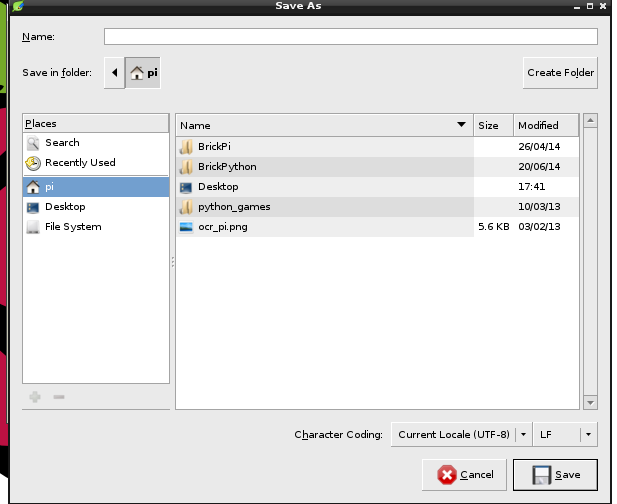
One possible way to write / edit a program is to use the command window. It is not the ideal place to write out long programs but it is nice to be able to directly edit files and run the in one location. In order to create / edit a file in the command window you use the command **< sudo nano “name of file” >.** When naming your program you must end it with < .py > in order for it to run as python code.

Your turn:

Now that you know how to look into programs, find your way back to *Example Projects* and go into a file and take a screen shot of it.

Writing programs in LeafPad:

Leafpad is a linux editor that works great for writing longer programs due to the ability to move around quicker. You can find it by clicking the bottom left corner of the screen and going under the *other* tab. When you open Leafpad, a screen like the one below should pop up.

After Leafpad has opened you can start writing your program directly into it. You can also go to the option tab and add line numbers plus auto spacing. When you are ready to save (and it is a new file) click on the *file* tab then click on *save as.* After this a screen will pop up that will allow you to name the program and navigate your way to your own storage device. Remember to end your name with < .py > in order for it to work.

One downside to working in Leafpad is the fact that you cannot run your program there. You will still have to go to the command window, which we will explain next.

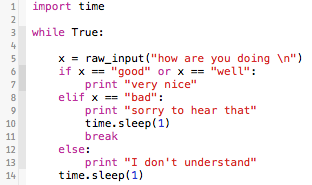
How to Run a Program:

This is really quite simple. First navigate to the directory that the program you wish to run is located. Once you are in the right directory, you simply need to type in < sudo python “name of file” >. The *sudo* allows for the program to operate as a root function enabling you to use your whole raspi. An example is below. This would run the file named *test*.

Macintosh HD:Users:Christian:Desktop:Screen Shot 2014-07-13 at 2.23.52 PM.png

Learning Python Commands:

There are a good number of similarities between Matlab and Python, as you will see once you start to write programs. But before you dive into writing your own programs, let’s first examine a sample program to see what some of the common python commands are. It might be helpful to write this program yourself to see how it works personally. (Note: the color coding will be different)



This is a simple program that will ask you how you are doing and respond based on what you type in. Here is a list of the commands I used.

1. **import** – In order to run certain functions you will have to import libraries into your code. For this example I have imported *time* functions into my program in order to use the *sleep* (lines 10 & 14) function that pauses my program for a given amount.
2. **while** loop – On line 3 you can see the familiar while loop function. This function, just as in Matlab will keep the program running until one of three things happens. First, I can interrupt it manually in the terminal. Second, if the statement after the *while* is no longer true. Third, if the program brakes from the loop. Note: that after the statement the line ends in a **colon**. This is a common trait for python and will pop up constantly, keep an eye out for it. Also you will notice the **indention**: this is important in Python. You will always have to indent the lines of code you want to be in the loop or the if clause in order for the program to run.
3. **raw\_input** – This is a common easy way to get keyboard input into your program. You can have it provide a prompt the way I did above. You may have also noticed all of the quotation marks. Just as in Matlab, you must distinguished between variables and strings; quotation marks are an easy way to make something a string. (There are other ways, feel free to look them up.) In addition, you can see that I am assigning what I input to the variable *x.* This process works exactly like it does in Matlab.
4. **if** case: The if clause is an extremely important function in both Matlab and python, however there are a few differences between the two. It still is a function such that if something is true it will run the code underneath it, if it is false it will do something else, as you will see. So the way the code above works is it first checks to see if the x == “good” or “well”. If this is a true statement it will print “very nice.” However, if this is a false statement it will check the *elif* next to see if it has a true statement. And finally, if the *elif* statement is also false it will precede to do the *else:* case. To summarize here is a nice diagram.
5. **print** – Another useful function. It will simply print out whatever you want into the command line when it is run. This makes it useful when testing out code. If you wish to print words you must make sure that it is a string.
6. **time.sleep** – This function will pause your program for a given amount of time. This is helpful when you want to slow down a while loop to keep the raspi from over processing.
7. **and, or** – These work the same as in Matlab.

This is far from a complete list of python commands, there are many more out there for you to learn and use, so please feel free to go out and find some useful ones. Now that you know a bit more about some common python functions, it is time for you to start writing your own code.

**Lab 1: My Change Please.**

For this lab please write a program that will act as a cash register. To do this you will need to generate a random number between 1 and 20 that will be the amount you need to pay. Then have the program ask if you are going to pay with a $5, $10, or $20. After answering this question manually it should give you the amount of change needed in the sets of $1 and $5. The command to create a random number in python is < random.randint(1,20) >. You will have to import the set of functions called *random* at the beginning of your program in order for it to work.

**Lab 2: Convert to Binary.**

Write a program that will take any decimal integer between 1 and 10, and then convert it to binary. Getting to know your Raspberry Pi.