SYSC3010 Lab 1

Getting Started

Caution: For those without previous experience, this lab will take (much) more than three hours.

OBJECTIVES

1. Teams are formed and introduced, with an agreed code of conduct.
2. Each individual has the lab kit setup at home.
    - S/he can run their Raspberry Pi standalone and headless (lab machine and/or personal laptop).
    - S/he can write a Python program is written and run on the RPi.
3. Each individual has a basic Git setup
    - S/he has their own Github account
    - S/he has their own Github repository containing their lab work. (to be marked later)

SYNCHRONOUS VERSUS ASYNCHRONOUS

The TAs will be online for the entire lab period (Connection instructions will be provided on CULearn). The lab period for Lab 1 is broken into 3 sections:

1. The first 20 minutes: An open session for all students who do NOT have a group.
    a. The TA will arbitrarily add remaining students to any groups that are not full (i.e. less than four members).
    b. That student must then immediately contact the group, to participate in their scheduled meeting.
2. The middle period: Scheduled meetings with individual group, to get acquainted
    a. Please sign up for a 20 minute meeting with a TA at this shared document.
    b. Please connect with the TA during your allotted time.

    Later in the term, we will be using this same procedure to conduct the lab exam and the project demos. We have to collectively figure out how to conduct these virtual sessions efficiently.
3. The final hour: An open session for any individuals requiring technical help.

SUBMISSION

There will be no marked submission at this time. During the individual lab exam (Lab 4), a student must be able to perform any task involved in this lab. Additionally, the student's Github repository will be checked to ensure that (1) the content for this lab is present and (2) the timestamp for that content falls within this lab period.

1 - TEAM MEETING

Before your scheduled time: Ensure that your team contract is completed, and prepare questions to be asked of the TA.

The team meeting must take place in two places, as a test of both environments.

1) On Zoom
2) On Slack – on the private team channel which must include the TA and the instructor.

The TA that you meet will become your TA for the remainder of the term. Introduce yourself to the TA and ask questions

a. Are you third or fourth year students?
b. What aspirations do you have for your project?
c. Where is your group geographically located (Are you in the same time zone?)
d. What equipment does your group have (will have)?
e. Any concerns or troubles?

2 – RASPBERRY PI

The objective in this first lab is to get you **as individuals** comfortably running your individual RPi at home. Minimally, you must write and run two Python programs on your RPi. The two programs to be written are described in the Raspberry Pi Education Manual, **Lesson 3.2 MasterPy** and **Lesson 3.3 Roman Numerals**.

Your first task is to simply get your RPi running and be able to write small sample Python programs.

- The first easy step is to run your RPI as a **standalone**. Your RPi runs just like any desktop computer. Doing so, requires that you can scavenge a HDMI monitor, keyboard and a mouse from a desktop machine at home. If you only have a laptop, you must skip the second step – it is possible, just a bit more frustratingly difficult.
    - *Tip for Pi Model 4: This model has two HDMI port connections. You must use HDM1 1 when booting. If you see a rainbow on your monitor, it means you have used HDMI 2. Simply switch ports.*
- The second step is to run your RPi **remotely** or **headless**. Now, your RPI does not have a monitor, or keyboard or mouse; instead you connect remotely to it from your laptop.
    - The easiest is to connect directly with an Ethernet wire connected directly between the RPI and your laptop.
    - You can also connect wirelessly, but this requires some initial setup of IP addresses that needs to be done beforehand (either as a standalone or directly wired).

After this first task is achieved, you are encouraged to work on the two required Python programs. In doing so, you may find that you still need to learn a few things in order to work efficiently, especially if you are working on a headless RPi. You may need to explore some more.

In this lab, in this course, I will not teach you all the things that you need to learn. You must be willing to explore. **Please do not think: "I've done everything listed. I'm done!". Instead, you should be thinking: "Okay, what's next? How do I work efficiently on this new machine?"** On the one hand, you don't have to do *everything* in this first lab – hopefully you will be stretching your limits throughout the whole term. On the other hand, you should spend some time in these initial weeks becoming proficient in working in the RPi environment – you should not expect to be told everything to do; you are expected to seek out these tasks on your own initiative.

Tip: You can use some of these learning experiments as the basis of your **technical memo.**

Some sample tasks that you should master soon are:

1. SSH is the first step in working remotely with your RPi. It's pretty slow though – you only have one window, everything is command-line driven. How about using VNC Viewer instead? You can have multiple windows.
    a. Try this Video: https://www.instructables.com/id/How-to-Use-Windows-Laptop-As-Monitor-for-Raspberry/
    b. In this video, he uses his route to find the IP of the RPi. Try using **Advanced IP Scanner** instead.
2. Transferring files between the RPi and your laptop, using WinSCP (FTP client for Windows)
    a. Physical transfer by swapping SD cards - See Appendix C
    b. Alternatively, try this video: https://www.youtube.com/watch?v=WIOpNuQc068
    c. Alternatively, try this video: https://www.youtube.com/watch?v=4P5nEH9zGDI
        i. You will need an ethernet cable between your laptop and RPi (or to a shared router)
3. Sending a SMS message
4. Sending an email message
    a. Try this tutorial: https://www.bc-robotics.com/tutorials/sending-email-using-python-raspberry-pi/

If you do not have a RPi (e.g. because it is arriving late), you are required to demonstrate the same tasks using a RPi simulator. Click here for a list of suggested simulators: https://windowsreport.com/raspberry-pi-emulators/. Please talk to the TA and/or instructor.

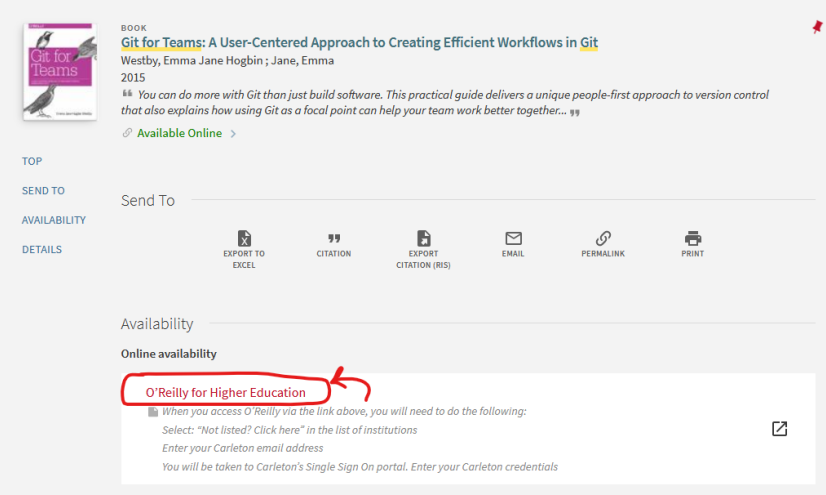
Resources

In the **Resources** listed below, it is my hope that you find the tutorials given under Resources sufficient to getting started (as well as learning about the wonderful stuff in our library available to you as students!).

And remember, you have teammates to help you as well as a TA. Sometimes it is tough getting going because of small little tricks that fool you, but remember that you are learning, and learning is hard work. Be patient with yourself.

Four books are listed below. All are electronic resources available through Carleton's library:

1. Login into Carleton's MacOdrum Library using your Carleton credentials.
2. Search for one of the books listed (or search just for *Raspberry Pi* and see lots of other resources)
3. You will be prompted to log into the O'Reilly site (and activate an account). Use your real email address, not your alias.
    - From Carleton CMAIL website: Your **real** email address is firstnamelastname@cmail.carleton.ca but you will also receive email addressed to your email **alias**: firstname.lastname@carleton.ca.



Book: Lee, Assam. *2019 Ultimate Guide to Raspberry P: Trips, Tricks and Hacks*. Packt Publishing 2019.

- Videos on Getting Started
- Chapter 1: Introduction (What is RPi)
- Chapter 2: Setting up the Hardware
- Chapter 3: SD cards (if you need to install a new O/S image)
- Chapter 4: Configuring your Raspbian OS Installation
- Chapter 5: Connecting Remotely (Minimally SSH)

Book: Donat, Wolfram. *Learn Raspberry Pi Programming with Python: Learn to Program on the World's Most Popular Tiny Computer*. Apress 2018.

- Chapter 1: Introducing the Raspberry Pi
- Chapter 2: Linux by the Seat of Your Pants
- Chapter 3: Introducing Python:
    - Make sure you are aware of Python 2 versus Python 3. A lot of RPi libraries uses Python

Book: Warren, Gay. *Advanced Raspberry Pi: Raspbian Linux and GPIO Integration*. Apress 2018**.**

- Better for theory, rather than concrete examples.

Book: Rao, Manessh. Internet of Things with Raspberry Pi 3. Packt Publishing. April 2018.

- Setting up the Raspberry Pi
- Setting up headless Raspberry Pi

3 – GIT

The objective in this first lab is to get you **as individuals** started on using Git and GitHub as a version control system. At this early stage, we simply want to keep track the successive versions of your code, stored in the cloud.

Note: All code submissions in this course will be done via GitHub.

By the end of this lab, you must have a personal Github repository called **SYSC3010_firstname_lastname.**

- Use your full name as listed in Carleton Central (i.e. your CMAIL credentials).
- Make the repository private. GitHub provides free private accounts to students.
- Add your TA and the instructor to your account. Their accounts are listed on CULearn.
- For each lab, you will add a folder called **Lab-x**. For example, for this lab, you will add a folder called **Lab-1**. This folder must contain the two Python programs that you wrote and ran on your RPi.

This GitHub repository will be used through the course. The TAs will pull all of your changes at the appropriate deadline. (Any changes after that deadline will not be marked). Please do not forget to push your changes to the repository before the deadline.

In the **Resources** listed below, you must read enough to understand the terms: **local** repository, **remote** repository, clone, add, commit, and push; you do not yet have to understand branches and merges.

As you read, it is important to know that Git and GitHub are two different pieces of software, despite having similar names.

- Git is a program which you will download onto your machine that will keep all the meta-data about the different versions of your files. You will use it in the *command-line*. On Windows, this means you will be using the Command-Prompt shell. (In the Run menu, type "cmd"). Later, you will find that Git is integrated into many IDEs such as Eclipse and Netbeans.
- GitHub is a cloud service – like Dropbox or GoogleDrive – that gives you free space to "host your remote repository". **GitHub provides free private accounts to students when you uses your Carleton credentials to register.**

Resources:

1. Nelson, Meghan. An Intro to Git and GitHub for Beginners, Oct 1, 2015.
2. Official Git Site: https://git-scm.com Focus on the following menu items:

    Downloads
    Documentation
        Book: Chacon, Scott and Straub, Ben. Pro Git
        Video: Git Basics: Get Going with Git

3. Hogbin Westby, Emma. Git for Teams, A User-Centered Approach to Creating Efficient Workflows in Git. O'Reilly Media, September 2015. Available through Carleton library
4. Lee, Assam. *2019 Ultimate Guide to Raspberry P: Trips, Tricks and Hacks*. Packt Publishing 2019.
    - Chapter 9.3 : Downloading code and Resources from GitHub

Task 1: Getting started with Git and GitHub

Each individual shall use Git by themselves. Using the **[Nelson]** tutorial, perform **Steps 1 to 6**. Defer Steps 5 and 7 (branch and merge). Step 8 onwards is for later, when working as a group.

If you are completely new to Git, you may also wish to spend some time learning the basic terms

- Watch the **Git Basics**: **Get Going with Git** on the official website
- Within the book by **[Chacon, Straub]**,
    - 1.5 Getting Started – Installing Git
    - 2.1 – 2.3 Git Basics
    - Chapter 6: GitHub

Task 2: Configuring Git

The only configuration that is mandatory is the setting of your name and email. This will play a role once you start collaborating with your team. It will allow us to know which person is committing which piece of code, to know the relative activity and contribution of each team member.

Please view the video titled: **Git Basics Episode 3: Get Going with Git**, on the official Git site.

```
> git config - - global user.name ”Dana Devops”
> git config - -global user.email “your.name@carleton.ca”
```

Every individual must submit your GitHub URL on the CULearn Lab 1 quiz so that the TAs can check your work.

Task 3 - Git Cheatsheet for your first repository

Create your first repository, committing the two Python programs that you wrote.

Creating a New Repository

1. In browser on your laptop,
    a. Login into GitHub
    b. Hit + on top-left to create a new repository.
        i. Give a name: SYSC3010-lab1
        ii. Select INITIALIZE THIS REPOSITORY WITH A README
2. In shell (or command-prompt) on your laptop or your RPi (wherever your Python programs are located)

    > cd xxx (Navigate to the root folder containing your work)
    >git remote add origin https://github.com/your-name/SYSC3010-lab1.git`
    >git add .
    >git commit -m "initial commit"
    >git push -u origin master

    If you have troubles, try

    >git remote set-url origin https://github.com/your-name/ SYSC3010-lab1.git`
    >git push -f -u origin master

To push a updated version to existing repository (do this at least at the end of every work session)

> git add . (The . means add all files)
> git commit -m “message” (See guidelines below for writing a proper message)
> git push origin master

GIT Guidelines – Expected to be used all term (Will be part of your marks)

Each commit requires a message. Below are guidelines for these messages to be followed in this course.

```
[author] type: subject
body
footer
```

The title consists of the **type** of the message and subject.

The Type is contained within the title and can be one of these types:

- **feat:** a new feature
- **fix:** a bug fix
- **docs:** changes to documentation
- **style:** formatting, missing semi colons, etc; no code change

- **refactor:** refactoring production code
- **test:** adding tests, refactoring test; no production code change
- **chore:** updating build tasks, package manager configs, etc; no production code change

The Subject should be no greater than 50 characters, should begin with a capital letter and do not end with a period.

- Use an imperative tone to describe what a commit does, rather than what it did. For example, use **change**; not changed or changes.

The Body

Not all commits are complex enough to warrant a body, therefore it is optional and only used when a commit requires a bit of explanation and context. Use the body to explain the **what** and **why** of a commit, not the how.

When writing a body, the blank line between the title and the body is required and you should limit the length of each line to no more than 72 characters.

The Footer

The footer is optional and is used to reference issue tracker IDs.

Example Commit Message

```
feat: Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72
characters or so. In some contexts, the first line is treated as the
subject of the commit and the rest of the text as the body. The
blank line separating the summary from the body is critical (unless
you omit the body entirely); various tools like `log`, `shortlog`
and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you
are making this change as opposed to how (the code explains that).
Are there side effects or other unintuitive consequenses of this
change? Here's the place to explain them.

Further paragraphs come after blank lines.
 - Bullet points are okay, too
 - Typically a hyphen or asterisk is used for the bullet, preceded
   by a single space, with blank lines in between, but conventions
   vary here

If you use an issue tracker, put references to them at the bottom,
like this:
Resolves: #123
See also: #456, #789
```

Appendix A : Installing a Clean Version of the Raspbian Operating System

If you are re-using your RPi from a different course or from a friend, you may want to start with a clean version.

URL for downloading image of the Raspbian Operating System:
https://www.raspberrypi.org/downloads/
URL for in-depth instructions for installing
https://www.raspberrypi.org/documentation/installation/installing-images/windows.md

1. Format your SD card

   Download, install and run the SD Card Formatter from https://www.sdcard.org/downloads/formatter_4/

2. Download the image for Raspbian onto a local folder (not on the SD card). Unzip.

3. Download and install the **Win32DiskImager** utility from SourceForge .
   a. URL https://sourceforge.net/projects/win32diskimager/
   b. Run-as-administrator .
   c. Point its browser to the unzipped Raspbian image.

4. Put the SD card in the RPi(without the power plugged in)

APPENDIX B - RUNNING A HEADLESS RPI USING REMOTE ACCESS

A Headless RPi is one without a monitor, keyboard or mouse. Instead, you connect it to your laptop, so you can simply use its monitor, keyboard and mouse.

URL: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-6-using-ssh/overview
Alternate URL: http://www.modmypi.com/blog/tutorial-how-to-give-your-raspberry-pi-a-static-ip-address
Alternate URL: https://pihw.wordpress.com/guides/direct-network-connection/

Prerequisite: You need a static IP address for your RPi. (See previous Appendix)

1. Wire the desktop machine to the RPi using the Ethernet cable. Verify the connection by **pinging your RPi**. Verify that you have a connection.
2. On your desktop machine, in a Command Prompt window, type "ping 10.0.0.2" (substitute your IP)
3. On the desktop machine, download **putty** from http://www.putty.org/
4. Run putty. Enter the IP address of your RPi. (and make sure that Port is 22). Wait and hit enter (maybe a couple of times, be patient).
5. Login using the default username **pi** and password **raspberry**.
6. From here, you can run any Linux commands including Python (as if in a single terminal window)

APPENDIX C: TIPS FOR DEVELOPING PROGRAMS ON YOUR RASPBERRY PI.

You can write your programs directly on the Raspberry Pi, using either Python or Java. You may find this a bit cumbersome (with the funny small RPi keyboard) or slow (remotely from your laptop). Alternatively, you can – with a little cleverness – write the bulk of the programs on your laptop, fix up most errors, even run some programs (that don't depend on the hardware, for instance) … and then send the file to the RPi for execution.

Let's say you have a program that you have edited while working on your Windows laptop. You now want to run it on your RPi.

1. Turn the power off on your RPi.
2. Eject the micro-SD card
3. Put the micro-SD card into an adapter, and put the card into your laptop.
4. Use Window's File Explorer to browse the SD card.
5. Create a Folder called Programs, for instance.
6. Store your files in this folder.
7. Eject the SD card
8. Put the SD card back in the RPi (with the power turned off)
9. Power up your RPi. Let the system start up.
10. Open up the **File Manager**. Where are your files?
11. Browse to the root directory "/". It should be down at the bottom of the folders listed.
12. Find the Boot folder. Your folder will be in this folder.
13. You cannot run your programs from this sub-folder of Boot. It is a write-protected directory. You must move your files to a user directory. Copy it somewhere under **/pi/Documents/**
14. Now use Python or Java to run your file.