Exercise 1: Intro to Python and Raspberry Pi*

EENG350: Systems Exploration, Engineering, and Design Laboratory

Tyrone Vincent

Department of Electrical Engineering
Colorado School of Mines

Fall 2022

1 Introduction

The purpose of this exercise is to get familiar with a Raspberry Pi computer, the hardware features, and how to interact with it using a PC/laptop. These basic exercises will set up the foundation to be able build a much more complex system later.

Exercise Objectives:

- Use the Python IDE to run a python program on the Raspberry Pi
- Write a Python program that uses one or more of the following Python elements
  - Variables and Types
  - Lists
  - Functions
  - Methods
  - Packages
  - NumPy arrays
  - Flow Control (if, for, break, continue, etc.)
  - File input and output
  - Errors and Exceptions

1.1 Approach to the assignment

This handout (and the ones that follow) may be structured in a different way than a typical lab course. The intent is to provide you with resources that you can use to either solve a problem, or gain proficiency on a tool that you may use later to solve a problem. While there are some demonstration exercises and other deliverables listed below, if you approach this assignment with the goal of merely completing "what needs to be turned in", you will probably find that later in the semester, your proficiency on the material covered in this handout is poor, and you will end up wasting more time later than you saved by "just figuring out what I need to get this assignment done". Your goal for this assignment should be to gain proficiency. On the other hand, there is a lot of material, and it can quickly become overwhelming. As part of the course, you can develop some skills in finding and learning new material when you do not have an instructor to lay it all out for you. The ability to do this is one of the hallmarks of life-long learning.

We will discuss our approaches to learning new software in an upcoming discussion assignment. In the mean time, here is one way to approach it.

- Build up some fundamentals. Go through some introductory tutorials and write some simple programs.
- Work on a project - decide on a task you want to accomplish. Think about what do you need to know in order to accomplish it. Scan through documentation and look for key words and key examples.
- Build up expertise. If you get your project working, go back and research the functions or concepts you used in more detail. Try different implementations - is there a different way to accomplish the same thing? Learn about other methods, data structures, or advanced concepts.
- Pace yourself! Set a schedule for learning. It is hard to overstate how much better your retention will be if you work 1 hour a day for a week rather than 7 hours on one day.

2 The Hardware

The Raspberry Pi is a fully functionally computer. It has a dedicated processor, memory, and a graphics driver for output through HDMI. It runs a specially designed version of the Linux operating system. You have been provided with the Raspberry Pi 3 with Raspbian (OS) preinstalled on it and ready to go. However, if you want to re-install the operating system, you can obtain the version of the operating system that we are using from this link. To perform the install, follow these instructions except for the part about downloading NOOBS, as you will use the file you downloaded from the previous link instead.

3 Python

For this course, Python will be used as the default scripting language. Python is already installed on the lab computers and on the Raspberry Pi. **You will be using Python 3 for this class.** You can use Spyder to run Python 3 on the lab computers and Thonny or IDLE 3 on the Raspberry Pi. (C/C++ can be used if preferred. Personal experience suggests Python as a good choice for the beginner when it comes to computer vision.).

You have already started the tutorial at `https://www.learnpython.org`. Continue this tutorial with Numpy Arrays (Pandas Basics is optional) and the lessons listed under Advanced Tutorials up through and including Code Introspection. (the other items are also optional).

For further resources, you can use the tutorial from the python software foundation at `https://docs.python.org/3.5/tutorial/`. Go through the tutorial chapters 1-5, plus chapter 7. There are also many (non-free) on-line courses for Python, such as through data camp or code academy

4 Raspberry Pi Setup

Hook up the USB keyboard and mouse to the Raspberry Pi board. Hook up the monitor using the HDMI cable. Power up the board and check the monitor to see the OS load. One of the first things to do is to make sure that the keyboard layout is US (English). This can be done from Menu (under Preferences). Also ensure that the Pi is connected to the internet. Connect to "CSM Wireless" and register the computer. The wireless connection will be slow but is still helpful.

Although the OS comes with a pretty decent GUI, it may be necessary to work at command line quite a bit for the course of the project. Some basic commands and how to use them can be found here. Here is the Raspberry Pi technical documentation.

Create a directory for your projects using `mkdir`. Code for future exercises and projects should be saved in this directory.

Use the Python integrated development environment (IDE) (Thonny or IDLE 3) on your Pi to try out code mentioned in the tutorials. (Click on the raspberry in the upper right corner, and then on the programming menu to find links

to these.) You should bookmark the tutorials in case there are other topics that you will need for future projects (e.g. Section 5 on Data Structures may be useful). You should also bookmark the Python Standard Library documentation.

5 Remote Desktop

You may find it more convenient to access your Pi using a remote desktop from your laptop/desktop rather than having to plug in the keyboard, mouse, and monitor into the Pi. Later in the course, this will also be very useful if you want your teammates to be able to use the Pi remotely. You can follow these instructions to setup a VNC server on the Pi, and use the RealVNC app to connect to it (VNC should already be installed, so skip that step). In order to make the access independent of the Pi's IP address, you will want to set up a cloud connection. Scroll down on the page for the instructions **Establishing a cloud connection**.

6 State Machines

When developing code for hardware you may find that you need to organize the code so that different code is executed at different times. For example, suppose you have a robot that needs to get from one place to another. To do this, the robot may need to complete a variety of tasks in a particular sequence. For example, the robot may need to first localize it self (find its location and orientation), rotate (rotate to the correct orientation), move forward, and then stop (when at desired location). One way to create a program that executes different code for each task is by using a state machine. We can identify a state with each of these different behaviors. While in a particular state, the robot executes the desired behavior, and also keeps track of when it needs to move to a new state.

Read about finite state machines here. On this page, find the example of a coin-operated turnstile. There are a multitude of ways that a state machine can be implemented. One of the most straightforward is to include a variable for the current state, and use logic to execute code depending on the value of the state variable. For readability, it is helpful to collect the code for a particular state in a function. Copy the following code into the IDE, and run it to see how it operates. (Note: if you are viewing this assignment pdf in the Canvas pdf viewer, you may lose the carriage returns when copying text. You may find it easier to download the pdf to the raspberry pi. You may also lose tabs - you will have to fix the tabbing in order for the program to run.)

```
# The following code demonstrates a very simple idiom for a state machine.
# Each of the state functions below performs some action and then implements
# logic to choose next state.

# define state variable, and initialize state 0
state = 0

# create a dictionary to describe the states
state_dictionary = {
    0 : "locked",
    1 : "unlocked"
    }

# function for state 0
def state0(activity):
    # you can put any code here that should run in state 0
    # athough this example doesn't have any
    #
    # At the end, we decide what the next state should be
    if activity == "push":
        print("turnstile was pushed, turnstile remaining locked")
        return 0
    elif activity == "coin":
```

```
        print("coin was entered, turnstile becoming unlocked")
        return 1
    else:
        return None

# function for state 1
def state1(activity):
    if activity == "push":
        print("turstile was pushed, turnstile becoming locked")
        return 0
    elif activity == "coin":
        print("coin was entered, turnstile remaining unlocked")
        return 1
    else:
        return None

# initalization
state = 0    # initial state
print("Type coin or push to run state machine")
print("Anything else will end the program")
# main loop
while state is not None: # Run until state is None
    print("Current state is "+state_dictionary[state])
    # Get current activity
    activity = input("input: ")
    if state==0:
         state=state0(activity);
    elif state==1:
         state=state1(activity);
    else:
        Exception("Invalid state has been entered");
print("Done with state machine")
```

You can see in the main loop how an if - elif -else statement was used to check the current state and decide which code should be run. When there are a lot of states, this can become kind of annoying. In other languages, a case statement might be used, but Python doesn't have a native case statement. However, there is a slick way of eliminating the if statement. In python, the function name can be used as a pointer to the function, and thus passed to other functions, or returned from a function. Carefully compare the code below to the code above.

```
# The following code demonstrates a very simple idiom for a state machine
# Each of the state functions below performs some action and then implements
# logic to choose next state.  Each state function returns the next state.

def state0(activity):
    if activity == "push":
        print("turnstile was pushed, turnstile remaining locked")
        return state0
    elif activity == "coin":
        print("coin was entered, turnstile becoming unlocked")
        return state1
    else:
        return None

def state1(activity):
```

```
    if activity == "push":
        print("turstile was pushed, turnstile becoming locked")
        return state0
    elif activity == "coin":
        print("coin was entered, turnstile remaining unlocked")
        return state1
    else:
        return None

# create a dictionary to describe the states
state_dictionary = {
    state0 : "locked",
    state1 : "unlocked"
    }


# initalization
state = state0    # initial state as pointer to state0 function
print("Type coin or push to run state machine")
print("Anything else will end the program")
while state is not None: # Run until state is None
    print("Current state is "+state_dictionary[state])
    # Get current activity
    activity = input("input: ")
    new_state = state(activity)  # launch state machine
    state = new_state   # update the next state
print("Done with state machine")
```

7 Demonstration Exercises

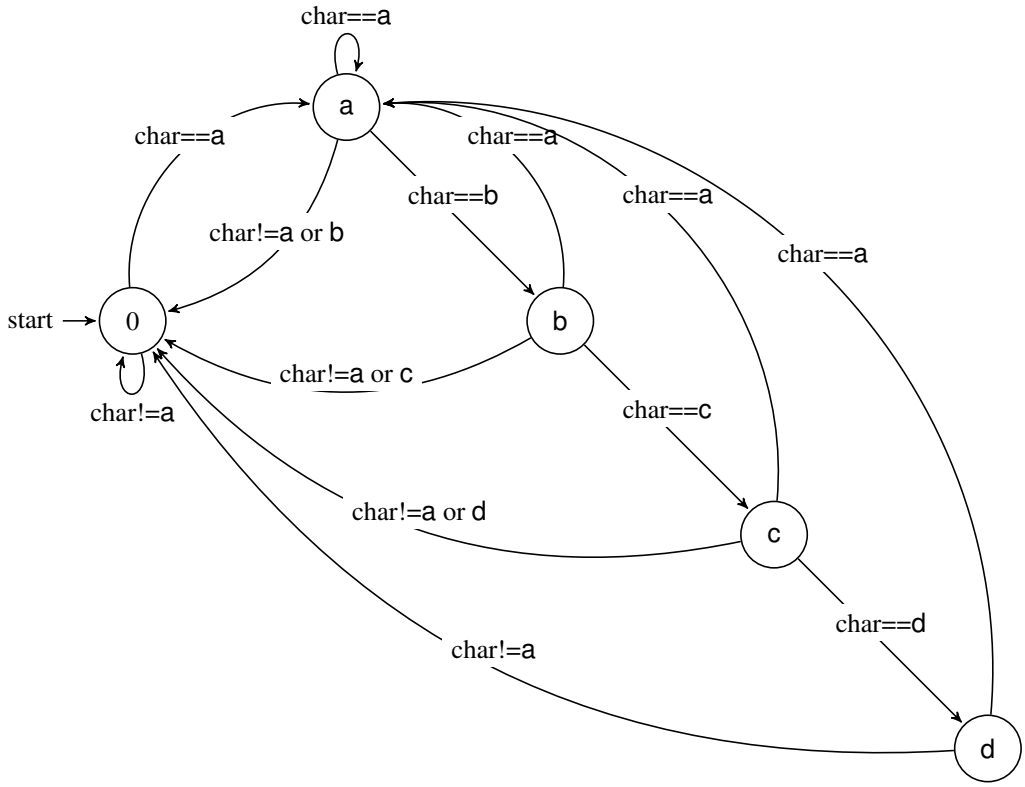
- Exercise 1: There is a data file on Canvas called datafile.txt. Copy this file to your Pi (use a usb drive or email or go to canvas using the internet browser on the Pi). This file generates a list of 100 integers. You can load this data using the following code:

  ```
  with open('datafile.txt','r') as f:
      b = eval(f.read())
  ```

  Create a program on your Pi in python that will display the following information about this list:

  1. The maximum
  2. The minimum
  3. The index where the number 38 is located
  4. The number or numbers that are repeated the most, and the number of times they are repeated
  5. A sorted list. (Convert to numpy array and use appropriate method)
  6. All even numbers, in order. (Use a list comprehension to extract the even numbers)

- Exercise 2: Write a program that does the following: input a string from the user, then using a state machine, determine if the sequence "abcd" is contained in the string. In this case, your program will loop through each character in the string. The state machine is the following



If you arrive at state d, you should print "abcd is contained in the string"

When you have finished an exercise, call over a TA to get the exercise graded. You should be able to obtain the correct results, and also go through the code and answer questions about it.

8 Documentation

Upload a document to Canvas that contains the following:

1. Answers the following questions:
   (a) What is an IDE?
   (b) What are the key words to handle exceptions in Python?
   (c) How do you define a function in Python?
   (d) True or False - when calling a specific function, the number of arguments is fixed.
   (e) What are two methods that can be used with list objects, and what do those methods do?
   (f) Carefully describe all the differences between the example state machine implemented with if statements, and the example state machine implemented with function pointers. How do pointers make it easier to add new states?
2. Your **well commented** and easily readable code for the demonstration exercises.

Also, don't forget to upload your case study and reflection log to the appropriate assignment link.