Robotics Curriculum Module 2.2

*Advanced Python Programming*

***Introduction***

After completing module one, students should be able to craft a basic python program. Students will have gained the knowledge necessary to successfully start writing python code, but there are several items which need to be addressed before their skills can really be put to use.

Module two will see students explore larger, more complex topics. These range from the extremely useful (arrays, methods) to the niche, but important (dictionaries, recursion).

***Timeframe***

Unlike module one, module two will not have a strictly enforced timeframe. While each “lesson” is expected to take roughly an hour, students can spend more time / less time on topics as necessary. In addition, this module will have advanced topics. In addition, the module will only have 3 lessons, however 4 weeks are available. Use this time to review as necessary, or to make sure students understand the new material.

***The Online Textbook***

This course will be provided with access to a free online textbook, “How to Think Like a Computer Scientist”. You can access the book through this link:

<https://runestone.academy/runestone/books/published/thinkcspy/index.html>

The book has built-in runners for python code, has exercises, and great examples in the reading. This online textbook is great for coming up with additional learning exercises and freshening up on python material if you haven’t programmed in a while. You may need to make an account to access the book, but the book *is free*, so don’t pay for anything!

***Discord Channel***

There is currently a Discord channel for Innovate-IT and the robotics venue. Please email [lfoster1@iastate.edu](mailto:lfoster1@iastate.edu) for an invite link.

***Workflow***

The curriculum has pages available with teaching information and recommended materials to cover. These materials are just a suggestion, and merely a place to start when teaching students the material covered. Generally, every week should follow the same format:

1. Read the learning materials provided.
2. Watch any videos (if linked).
3. Ensure that you can answer all of the “learning outcomes” questions presented that week.
4. Work on the technical task for the remainder of free time.

***Extra Materials***

Module 2’s extra materials (python technical challenge solutions) can be found at the following box address:

<https://iastate.box.com/s/oo31hatg05vb5u7t6q3iqm2wwcow88zr>

The solutions are in the format M2\_W(lesson number).py

Don’t look unless you want the challenge spoiled!

***Lesson 1: Lists***

Lesson one is dedicated to one of the most important data structures available to the programmer - the array. Without arrays, many programming solutions would be messy, inefficient, and poorly written. Arrays are one of the most powerful tools a programmer has available, and mastering the art of the array is vital for writing advanced code.

Lesson one will see students getting familiar with lists and generating them by hand. In lesson two, students will begin using lists *as* lists, through their list data structure functionality.

***Reading Materials***

*I highly recommend watching the beginning of this code.org video. It has a nice, high level overview of what a list is. The video is javascript, so it is only relevant until ~1:45.*

[*https://www.youtube.com/watch?v=KFy7u3Rhozs*](https://www.youtube.com/watch?v=KFy7u3Rhozs)

**Chapter 10: Lists**

10.1 - 10.6, 10.17

***Learning Outcomes for Lesson 1:***

* What is a list?
* How is a list indexed?
* How can I access the first item in the list?
* How can I manipulate lists? How do I join them together?
* How do I get the length of a list?
* What happens when I access a list element outside of the list’s length?

***Technical Task***

Write a program which takes in a series of strings (names, places, items, etc) from the console input and prints them out in reverse order. Your program should always take 5 inputs. After 5 inputs, the program should print out the strings it received in a reverse order.

Here are some questions to help you get started:

1. How do I get user input?
2. How can I use an array to store my inputs?
3. How do I iterate through an array backwards?

If you have extra time, try making the program work with a number of inputs specified by the user before they start inputting any strings. Use [] \* num\_names to make the array.

***Lesson 2: Advanced Lists***

Lesson two is dedicated to the advanced topics associated with lists. Previously, we used lists with hardcoded lengths. However, the list becomes far more useful as we use its *methods*, or functions dedicated to a certain type of data. In the case of lists, we can create an “infinite” length list, which is incredibly useful, as you will see.

***Reading Materials***

**Chapter 10: Lists**

**Required:** 10.14, 10.16, 10.18, 10.19 - 10.22 (pure functions optional but helpful)

**If you are comfortable with lists:** 10.23 (2D lists)

***Learning Outcomes for Lesson 2:***

* What is a method?
* What methods can I use on a list? How do I call them?
* How can I use lists with functions?
* How can I make a list have “infinite” length?
* What is a list comprehension? Why are they useful?

***Technical Task***

Modify your program from technical task 1 to continually take in inputs until the string “END” is taken in. An example of input (red) and output (blue) is on the right.

Some questions to get you started:

1. How do I check for the “END” string?
2. How do I add into the end of the array?
3. How do I make my “reverse” for loop run for any length of array?

***Lesson 3: Dictionaries***

Lesson 3 will move away from lists and into a similar topic: dictionaries. Dictionaries are a more specific, but more complex version of a list. Instead of using *integers* to index the list, we will use *keys*. Dictionaries are exciting, and they will be a fun addition to your programming toolbox.

***Reading Materials***

**Chapter 12: Lists**

12.1 - 12.3

***Learning Outcomes for Lesson 3:***

* What is a dictionary?
* How is a dictionary different from a list?
* What are some practical uses for dictionaries?

***Technical Task***

For this lesson’s technical task, you need to create a program which takes a textual input (on the command line) and collects information about the letters used. The program should create a list of each letter in the alphabet, along with the number of times each letter is used. For example:



***Lesson 4: Extra Topics***

As stated at the beginning, arrays and dictionaries are important topics which may take some time to digest along with the other new materials. However, if you have some extra time feel free to look into some of these deeper, but smaller topics.

***Reading Materials***

**Chapter 16: Recursion**

Recursion is a complicated, but useful concept to understand. Take a quick look at the chapter provided, and see if you can come up with some use cases for recursion.

**Chapter 5: Modules**

Python modules are everywhere, and will be far more useful as we get closer to competition. For now, take a quick look at the chapter and try and take advantage of the *math* and *random* libraries.

**Chapter 9: Strings**

Strings are everywhere. Take a look at the later half of the chapter to learn more about the advanced topics associated with strings.