Robotics Curriculum Module 1

*Introduction to Programming in Python*

***Introduction***

One of the most important outcomes of the Robotics venue is gaining good programming skills. The first two modules of the Robotics Python curriculum are designed to give you a stepping stone into programming, while also introducing more advanced topics.

Module 1: Introduction to programming in python, will begin by getting students familiar with python itself. Once familiarized with python, students will begin learning about programming itself, with major topics such as variables, control flow, iteration, arrays, and functions.

***Timeframe***

This module is designed to be covered over a single month of meetings. Assuming a single hour of availability per week, this will amount to roughly 4 hours of expected work time.

Module 1: December - January

***Iowa Computer Science Standards Covered***

* 3A-CS-03
* 3A-AP-13
* 3A-AP-23
* 3B-AP-10
* 3A-AP-15
* 3B-AP-09

***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 meeting 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 1’s extra materials (python technical challenge solutions) can be found at the following box address:

<https://iastate.box.com/s/4jqtdu364yy2orrk035up3vkyhnzts4s>

The solutions are in the format M1\_W(week number).py

Don’t look unless you want the challenge spoiled!

***Week 1: Introduction to Programming, Python, and Variables***

Week one should be an introduction to programming for students who have never programmed before, and a refresher for students who are familiar with programming.

There is a lot of material here, so picking and choosing according to your student’s needs might be important. Below are the recommended materials, with the absolutely necessary materials in bold.

***Reading Materials***

*Chapter 1: General Introduction*

**The entire chapter is required. It will be extremely helpful in the future to have a good understanding of debugging, python, and the book’s interpreter.**

*Chapter 2: Simple Python Data*

**Required: 2.1 - 2.8**

Recommended: The entire chapter

***Learning Outcomes for Week 1:***

* What is programming? Algorithms?
* How does python work?
* How do I write Hello World?
* What is a variable? A Type?
* What is a statement? An Expression?
* How do I write a program with variables, input, and output?

***Technical Task***

Write a program which does the following:

* Take in two numbers (stored in variables) from the python input.
* Add, subtract, multiply, and divide the numbers and print the output.
* Convert the two numbers into strings.
* Print both of the strings on one line.

***Week 2: Variables and Conditionals***

For week 2 you are expected to cover boolean variables, expressions, and conditional execution. This material is far deeper than the previous weeks, so less material is necessary.

***Reading Materials***

*Chapter 7: Selection*

**Required Material: 7.1 - 7.7**

If your students understand functions, feel free to cover 7.8. Functions will be covered in two weeks.

***Learning Outcomes for Week 2:***

* What is a boolean variable?
* How do I compare two variables? What is the result of a comparison?
* What does an if-statement do?
* What does ‘else’ do in a conditional statement?
* How can I use ‘elif’ to extend a conditional?
* What are the benefits of using conditional statements?
* How do I use conditional statements in a program?

***Technical Task***

Expand on the ideas you used last week to make a more interactive calculator.

Your program should do the following:

* Ask for an operation at the beginning of the program
  + “Add”, “Subtract”, “Multiply”, “Divide”
* Ask for two numbers, and store them in variables
* Print out the operation, the numbers, and the answer.
* *Example: 10 + 3 = 13*
* Bonus: Look for a divide by zero error if you are doing a divide operation.

***Week 3: Iterations and More Control Flow***

If conditional statements are the most important part of a programming language, iteration is a close second. This week will allow students to begin understanding iteration and how powerful it is. This unit will cover *while* and *for* loops, as well as useful operators and “templates” for looping. In addition, the unit will cover turtles, as they are an extremely useful extension of the programming knowledge students will have gained so far. With turtles and iteration, students will be in good shape.

As usual, understanding iteration is extremely important for the future programming skills of students. Once students understand iteration, they can begin writing far more complex programs.

***Reading Materials***

*Chapter 4: Python Turtle Graphics*

**Required: 4.1 - 4.5**

Recommended: 4.1 - 4.9

This chapter introduces turtles, and more importantly for-loops. Even if students do not understand turtles, ensure they fully understand for-loops. A useful alternative to the text is explaining some concrete examples of for loops:

* How can I add every integer from 0 to 10 most easily?
* How can I ask for user input multiple times?

*Chapter 8: More About Iteration*

**Required: 8.1 - 8.3, 8.8**

*There are no additional recommendations.*

This chapter introduces while loops along with some concrete examples. Once again, ask students to think about use cases for while loops to ensure understanding:

* How can I ask for input until I get the string “end” from the user?
* How can I make a turtle keep drawing forever?

***Learning Outcomes for Week 3:***

* What is a for-loop?
* What is a while-loop?
* When should I use a for-loop over a while-loop? Why?
* What is the difference between a *for* loop and a *while* loop?
* How do I write a loop in python?
* What types of programs use loops?

***Technical Task***

Expand even further upon your calculator program and make it work for multiple numbers and operations!

Your program should:

* Continue to ask for an operation, numbers, and print the result until “end” is entered when asked for an operation.
* Use a loop to make sure that the operation entered is valid (if the operation is not valid, notify the user and ask again)
* BONUS: Use a loop to make sure that the numbers entered are valid (similarly to above)

Since this is a complicated task, here is a sample set of input and output:

User Input

Program Output

What operation do you want to do? (Add/Subtract/Multiply/Divide)

Operation: Add

Number 1: 10

Number 2: 5

10 + 5 = 15

What operation do you want to do? (Add/Subtract/Multiply/Divide)

Operation: Divide

Number 1: 10

Number 2: 5

10 / 5 = 2

What operation do you want to do? (Add/Subtract/Multiply/Divide)

Operation: Muutiply

Sorry, that is not a correct operation.

What operation do you want to do? (Add/Subtract/Multiply/Divide)

Operation: Multiply

Number 1: 10

Number 2: 5

10 \* 5 = 50

What operation do you want to do? (Add/Subtract/Multiply/Divide)

Operation: end

Goodbye!

***Week 4: Functions***

Functions are one of the final hallmarks in learning how to program well. They save time and create more easily readable code. Functions are diverse and can accomplish almost anything you can program.

Functions are hard to wrap your head around and will appear daunting to students at first. However, a good grasp on functions will result in much better code later on.

***Reading Materials***

*Chapter 6: Functions*

**Required: 6.1-6.2, 6.4-6.10**

There are no additional recommended reading materials.

Chapter 6 is an excellent introduction to learning functions. Not only does it cover theory and how to use a function, but it also contains many templates, patterns, and examples that demonstrate functions.

***Learning Outcomes for Week 4:***

* What is a function?
* Why is using functions important?
* What is an argument? Can I have multiple?
* What does “calling” a function mean?
* What is “return”, and what does it do?
* Can functions “call” other functions?
* How can I write a function in python?

***Technical Task***

This is your final opportunity to expand on your calculator. Lets use functions to simplify the code and add new operations!

Your program should:

* Use functions to remove redundant code (move copy/paste code into a function)
* Use functions to create 3 new operations to do to the numbers provided:
  + Pow(A, B): Return A to the Bth power. To do this, multiply A by itself B times. Example: Pow(2, 5) = 2 \* 2 \* 2 \* 2 \* 2 = 32.
  + Midpoint(A,B): Return the number halfway between A and B.
  + Super(A,B): Return ((A-B) \* (A+B)) / (Pow(A,B)). This is an exciting one.
* Implement the new functions as commands you can ask the calculator to calculate.