This course acts to continue the content learned in the previous course, and build on the foundation supplied to the students with some much more complicated coding concepts and a proper introduction to using Python both inside and outside of Minecraft. If you are planning to join the course from this stage, some knowledge of computer science and the logic of coding should definitely be had, or the student should be at the developmental equivalent to a year ~8 student. If you are new and wondering if your child is capable of joining this section instead of starting at the first one, a quiz and homework project can be supplied to test their ability. The following is an introduction to the benefits of Minecraft coding in general, and my way of thinking about these lessons and how I try to go about teaching them. The paragraphs below are very similar to those that are on the first course syllabus, so if you have already read through that or have a returning student, you may want to skip to the Lesson 1 itinerary.

The purpose of this course is to give students a foundation in computer science and a computational way of thinking, while remaining appealing and engaging to the young target audience. Enable scripting with makecode provides an amazing platform to teach the foundations of coding, as it lets the students see what their code is doing quickly and in a game they definitely love. Minecraft is the most popular game in the world, and having the ability to teach a student while effectively letting them “play” it makes for a class that they will be excited for and pay attention in.

Every student learns at a different pace and has different strengths. I designed this course with that in mind, and left a lot of room for instructor improvisation to make sure all of the students spend their time in class learning. I also tried to give the students as many creative projects as I realistically could, such as choosing their own superpower to make in Minecraft, or the final assignment of using code to make a complex functional minigame that is playable. Giving the students this creative freedom allows for the students to challenge themselves to whatever level they are at. If a student is rather skilled and picking up on the topics quickly, they can be more ambitious with a project like this so that every one is learning to their fullest potential. Having the students choose their projects and designs also makes them more invested in the work they are doing, and it undoubtedly helps get them excited about all the new concepts they will learn later on - notably once they are no longer working with Minecraft. It is crucial that no student feels that coding “isn’t meant for them” or “it’s too hard”. I am a strong believer that everyone should learn at least a basic competency of computer science and coding in a rapidly automating world, and having a course that hides behind the facade or a student’s favourite video game ensures that they are getting the best introduction to computer science possible. This course maximizes the chance that a young mind will take to coding and continue to learn on their own merit.

This course covers a beginner to intermediate competency of Python through lessons taught in Minecraft. It also provides the students with a basic computer science knowledge, far ahead of what the average adult knows. The course also uses Minecraft building assignments to teach the students about historical architecture, and how the laws of physics affect creating a structure.

Below is a formal lesson plan of what subjects are covered in each of the 10 lessons, and the homework that is assigned after each.

NOTE: This course plan is a basic workflow, and all subjects listed will be covered. The course may end up being more accelerated andhaving extra content taught - depending on the competency level and speed of the given class. The learning goals will remain the same, but the course structure is subject to change.

| Lesson # | Class itinerary | Learning goals covered | Homework assigned |
| --- | --- | --- | --- |
| 1 | * Course introduction * What to expect from this course, and what will be covered * Introduction to competition coding, and how we can prepare for them * Easy story competition problem - “How many bowls?” challenge * Hard story competition problem - “10th Shortest?” * Installing IDLE for Python 3.7 terminal use * Introduction to real Python and “Hello World!” program * Quick introduction to using variables and if statements in Python (IDLE) with “5 greater than 2” lesson * Connecting to and setting up Minecraft class server, and getting new private locations for server building (same server as previous section will likely be used) * “Raining cats and dogs” Minecraft Python assisted challenge   + “Conditional rain” challenge problem * Homework and what to expect next class | * What are coding competitions near me, and how do I enter them? * What types of questions will be on these competitions? * How do I win a coding competition? * What is IDLE? * How do I load a new project and run a command line in my IDE? * How do I print to the terminal? * How do I use “if” statements in Python? * How do I use variables in Python? * How do you comment in your Python code? * How do I load a Python project in makecode? * What are coding modules and why do we use them? | * “Aquarium Maker” Minecraft Python code problem * Weekly coding competition problem (self timed to 10 minutes) * IDE problem - “Animal Printer” |
| 2 | * In depth summary of last week’s homework problems * Easy story competition problem - “Rare Mushrooms” challenge * Hard story competition problem - “Market Exchange” * IDLE “Variable Printer” tutorial * Different types of variables (callback), and the “type()” function   + Interpreting strings   + Interpreting integers   + Interpreting floats * Naming conventions for functions and variables * Taking a user input in Python * Joining two python strings * Simple Python arithmetic * “Variable Typer” IDE challenge * Class Minecraft server project introduction - “Double piston extender” * “Leaping Salmon” Minecraft Python tutorial * “Salmon Killer” Minecraft Python challenge (building on “Leaping Salmon” project) * Homework and what to expect next class | * How to go about solving algebra coding problems? * How to go about solving basic calendar problems in a contest? * How to use different variable types in Python? * What are the primary variable types? * How can you find the type of a variable? * How do you properly name your variables and functions? * How do you join two strings together? * How to perform simple arithmetic in Python? * How do you take a user’s input in Python? * How do you make a double piston extension system in Minecraft? * How do you run code “on animal killed” in Python? | * Make a double piston extension contraption on the class server * Weekly coding competition problem (self timed to 10 minutes) * “Variable type?” IDE program modifier |
| 3 | * In depth summary of last week’s homework problems * Easy story competition problem - “Bird Watching” challenge * Hard story competition problem - “Moving Packages” * IDE for statement and loops introduction * “Var number printer” IDE assisted challenge * “Even number printer” IDE challenge * Flying Machine introduction on class server * How to use observers in Minecraft * Minecraft Makecode “Wipeout Deletion” Python challenge * “Var number printer” Minecraft Python tutorial * “Procedural Wipeout” semi-assisted Python challenge * Homework and what to expect next class | * How do you think like a computer? * How to solve long word problems in competitions (and helpful tricks to go faster)? * How to use a “for” statement properly in Python * How does a “for” statement work? * When do we use a “for” statement * How to use the built in functionality of a “for” statement to our advantage * How to combine conditionals with nested “for” loops * What is a “flying machine” * How do I use observers in Minecraft? * Why can’t I delete a really big area in makecode? * How can I use for statements to increase the complexity of deletion functions? * How can you offload computational work to allow for bigger job ontake? | * Make a project on the class server using Observers (could be a flying machine, or anything else * Weekly coding competition problem (self timed to 10 minutes) * “Procedural platform spawner” Minecraft code assignment |
| 4 | * In depth summary of last week’s homework problems * Very hard “Weighing Boxes” coding challenge problem * Very hard “Museum Tour” code challenge problem * Very hard “Jumping Kangaroo” code challenge problem * Defining clump variables in the IDE * Accessing clumped variables * Introduction to lists   + List vs Array * How to define a list * List access functions and how to use them   + len() BIF   + [] Access point   + .append() BIF   + .remove() BIF * Using lists with user inputs * “Zoo game” list challenge IDE * Using lists in makcode Minecraft * How to set a list variable in makecode * “Rainbow beacon” code challenge * Homework and what to expect next class | * How to handle extremely hard conceptual problems in contests? * How can you manage your time while dealing with problems? * How can you use procedural thinking to break a large harder problem into smaller ones? * Why do we always need a pen and paper even on easy questions? * What is a list? * How does a list differ from an array? * When do we use lists? * Why do we use lists? * How to declare a list in Python? * What are the primary list access built in functions, and how do we use each of them? * How do we declare a list in Makecode? * When do you need to use a list in Minecraft? | * List functions worksheet. *This is a 2 page homework page, meant to give some practice to remembering and properly using the list access commands.* * Weekly coding competition problem (self timed to 10 minutes) * “Number looper” Minecraft makecode Python challenge |
| 5 | * In depth summary of last week’s homework problems * In class practice coding contest   + Using the content learned in the first class, and as a midway check-in, the students will partake in a 30 minute practice coding contest with minimal teacher assistance. This will be administered over Kahoot so students can see the correct answer after each question and then they can be discussed. This is meant to simulate a smaller real coding contest, which is usually 45 minutes in reality. * “Mob Cave” Python Minecraft code assignment   + This is a long coding assignment with many complex steps. It is designed as practice for coding something more substantial than a few lines, as it becomes harder to track what you are doing with more complexity and it’s an important skill to have. This project is meant to take close to the remainder of class time, and will make use of every single concept taught so far in this course and the one prior. The end product should be approximately 40 lines of code, which is extremely substantial compared to our usual 5 or 6 for a challenge problem. I (the teacher) will still provide significant assistance throughout this project if it is needed by the students, and will walk them through some of the harder parts if necessary. * Homework and what to expect next class | * What does a real computing contest look like? * How do I pace myself and successfully eliminate answers in a stressful situation? * What do you do if you aren’t sure of an answer? * How to work around trick questions, and how do you spot them? * How do you use an iterative list in makecode? * How do you procedurally spawn in a sphere in makecode * How can you create a lighting system in Minecraft without the use of light emitting blocks? * How do you save a player position as a variable, and what type of variable is this? * How do you interact with and alter a list variable in Minecraft makecode? * How do you increase a magnitude recursion formula using an “on chat” variable? | * “Signal Receiver” Minecraft Python challenge (hard) * Weekly coding competition problem (self timed to 10 minutes) * Finish the “mob cave” assignment, if not completed in class time |
| 6 | * In depth summary of last week’s homework problems * Easy story competition problem - “Sisters and brothers” challenge * Hard story competition problem - “It’s about time” * Very hard story competition problem - “Racing track” * “Time Master” partially assisted Python code challenge * “God of Weather” code tutorial * Discussion of the futile nature of programming, and how there can be thousands of solutions to any given problem. *This relates to the previous problem as they have extremely different solutions, but do very similar tasks and one solution is vastly more simple.* * “Leap of faith” unassisted code challenge * Introduction to passive redstone loops * Redstone looper in-class building project on class server * Homework and what to expect next class | * How to handle clock problems in a competition? * How do you go about family tree problems in a contest? *These are a lot more common than you might think.* * How do you use a logic circuit to add binary numbers? * How do you use a logic circuit to subtract binary numbers? * How do logic circuits work with binary numbers? * Could a logic circuit work with decimal numbers? * How does redstone background circuitry relate to microcontrollers and computers? * How can you tell if you have the best coding solution? * How can you improve a program? What steps do you need to take? | * “Participle tracker” IDE challenge * Weekly coding competition problem (self timed to 10 minutes) * “Rain or fall” Minecraft Python challenge |
| 7 | * These classes (7 and 8) are intentionally left undesigned until the course has started, and I am able to gauge the skill level of the class. If there is too much content in the other 8 classes, and the student’s go through slower than expected, these spaces are used as a buffer to expand out some of the work from each of the other classes. If students are aligned with the expected pace, these classes will be filled with other learning content of a difficulty based on what I believe the student’s are able to handle from my experience in the first few weeks with them. I have 5 concepts of different difficulties that I am prepared to insert some or all into this course, but which ones and when will be determined once I have gathered more information. If you are really curious about what concepts *may* end up being taught in these classes, get in contact and I can fill you in. | **TBD** | * “Particle movement” IDE assignment * Weekly coding competition problem (self timed to 10 minutes) * “Weather you won’t!” Minecraft Python challenge * TBD |
| 8 | * These classes (7 and 8) are intentionally left undesigned until the course has started, and I am able to gauge the skill level of the class. If there is too much content in the other 8 classes, and the student’s go through slower than expected, these spaces are used as a buffer to expand out some of the work from each of the other classes. If students are aligned with the expected pace, these classes will be filled with other learning content of a difficulty based on what I believe the student’s are able to handle from my experience in the first few weeks with them. I have 5 concepts of different difficulties that I am prepared to insert some or all into this course, but which ones and when will be determined once I have gathered more information. If you are really curious about what concepts *may* end up being taught in these classes, get in contact and I can fill you in. | **TBD** | * “Iterative Calculator” IDE assignment * Weekly coding competition problem (self timed to 10 minutes) * “Octopus friend” Minecraft Python challenge * TBD |
| 9 | * In depth summary of last week’s homework problems * Full coding contest practice. This test will be administered on a simulated competition website (as a regular one would be), and the students are given 45 minutes to complete as many problems as they can. * Extensive review of the coding contest, and common mistakes that may have been made to avoid on the real one. * Introduction to final class redstone projects, and examples of what to aim for, as well as partial work period on this assignment if time is had in the class * Homework and what to expect next class |  | * Complete the final redstone assignment on the class server. |
| 10 | * “Show and tell” of student’s final redstone projects * Final course test on computer science knowledge and contest preparedness   + This will be about 15 minutes long and administered over google forms. The marks will be supplied immediately after submitting, and wrong answers will be discussed in detail in a private breakout room with the student * Introduction to final coding assignment, and work period on it. I will call students into breakout rooms routinely to have private check ins   + This project is to be done in makecode instead of the IDE, as it makes it more engaging for the students, and the content is nearly identical. This project serves as a summary of all of the work done in the course so far as well as the previous one, and will test the student’s knowledge in both what was learned in our IDE and makecode.   + The project is similar in concept to the previous course’s final project , but significantly more advanced and difficult.   + It will still be rather open ended so the student’s have the liberty to use some creativity on the project, but a list of mandatory constraints will be supplied so every student demonstrates the necessary learning goals. | * How do we use the | * Finish the final coding assignment and submit it for grading. *This may be skipped if the student has already done this during class time.* |