Patron Training: Code Week – Getting Started with Programming using the Raspberry Pi

Class Outline

“Discover the foundations of computational thinking and computer programming with the popular Raspberry Pi and Scratch. No experience needed. All ages welcome.”

**Learning Objectives**

At the end of this program learners will be able to:

* Understand what computational literacy is
* Be able to identify the parts of a computer and Pi
* Able to use scratch and understand its uses an purpose
* Able to identify python and what it is used for
* Be able to follow-up with self-guided learning after the session is completed

**Time**: 2 hours

**Materials**:

* 1 staff
* Raspberry Pi activity sheet
* Learning further Handout
* 6 Pis and adaptors
* Evaluation forms

**Location**: Computer Training Room

**Outline**

|  |  |  |
| --- | --- | --- |
| **Time** | **Content** | **Materials** |
| 10:30 – 10:35 | **Introductions & Outline**   * Who I am: * What the class is for:   + Learning about coding and why it is considered an important skill   + Trying out a type of hardware that is cheap and popular for those learning to code   + Learning about some basic concepts around coding that will give you an idea of the shape and scope of the field   + Gain a bit of a vocabulary around coding * You won’t be an expert of anything after this class, and we are aiming at beginners. |  |
| 10:35 – 10:45 | **Intro: Why Code?**  There are more jobs than there are skilled workers in the computational science / tech industry, and there is a push for young people to gain those skills.  But we aren’t kids, so why do we care?  Because in order for there to be young people interested in code, etc. There need to be enough people out there who understand what problems code can solve.  When we talk about activities around coding, around Pis, and around other activities, the skills that we are working on, and the traits we are trying to develop are often referred to as computational thinking.  Let’s unpack this a bit more.  The greatest contribution programs that are about “learning code” bring, aren’t people who get jobs as programs, but it changing the way people think – this is computational thinking part of it.  Computational thinking is a bit like: looking at a number of ingredients in your cupboard and all you can see is- onions, lentils, some spices and thinking “soup!” Instead of “takeout”.  Mentally sketching a path from raw ingredients to a meal is computational thinking. Coding is like that – taking some specific building blocks and seeing a way to a solution. Like being able to use downtime effectively (making a salad while the soup is cooking)  Other skills in the computational toolkit are– iteration: testing to see when things are working, and adjusting and improving all the time. < adding a bit more spice, turning down the heat. When you next try to make soup, adding in another ingredient.  These pieces of computational thinking might be using new words, but shouldn’t sound too different.  And really, part of this little session today and programs like it, aren’t about making more programs, especially as adults, but being able to look at a problem and see that there might be a technological solution to it.   * Example Fire Hydrants from article.   *A few years ago some software engineers were in Boston before a terrible snowstorm for a special program. Their plans changed a bit when the storm hit.*  *They discovered that a problem during big storms is that fire hydrants are buried, and firefighters in an emergency have to spend time digging them out.*  *Boston had a list of the locations of all 13,000 hydrants. So they got out their laptops.*  *Now, Boston has adoptahydrant.org, a simple website that lets residents "adopt" hydrants across the city. The site displays a map of little hydrant icons. Green ones have been claimed by someone willing to dig them out after a storm, red ones are still available—500 hydrants were adopted last winter.*  *This has an ongoing cost of $9 a year.*  This type of thinking – using technology to solve everyday problems can help in other ways as well. People are trying to increase diversity in tech fields – if we are thinking about problems that can be solved with programming that matter to us – as library folk and people in a female-dominated profession, will attract young women to enter the field. Programming skills are more than just games, making money, and “math” but also being able to make a difference in society.  But enough about the why’s of computational thinking, how about we get into today’s activity! |  |
| 10:45 – 10:50 | **Why Pi?**  What makes the Raspberry Pi so exciting as a learning tool?   * Cheap! If you break it, it isn’t the end of the world base cost about $30 * You have to set it up – try some command line, use a less fancy user interface, making people learn and think about what happens behind the pretty interfaces * And – if you *really* mess up, it is such a lightweight system, that reformatting and starting from scratch isn’t a big deal. The Pi is a great sandbox. * Lots of plug-and-play options for building computers, cameras, touchscreens, time lapse cameras, well documented and there is a huge community of sharing build around them. * Lots of tutorials for setting up home servers, weather stations, home security devices, TOR nodes, little virtual libraries, jukeboxes, automatic cat feeders, - if you can think of it, someone has built it. * Ask:   + “Does anyone here have a Raspberry Pi at home?”   + “Has anyone here used Scratch before?”   + “Has anyone ever tried out some other coding programs? “ * If yes – get them to help * Ground rules with the Pis:   + be gentle with the pis (treat them like pup-pis. Haha).   + be patient with the pis (they are about 30 times cheaper than a regular computer so they will be slower!!)   + work with your buddy and make sure you take turns “driving” or using the keyboard/mouse |  |
| 10:50 – 10:55 | **Computer Insides**  Gather around the opened computer. Discuss the parts and functions. Some tips:   * Ask if *they* know what the parts are before telling them. In our experience, there is always at least one person who knows every part of the computer.   After you’ve looked at the computer, take out a Raspberry Pi and do a side-by-side comparison. See if they can find the “big” computer parts on the Pi. |  |
| 10:55 – 11:05 | **Scratch Demo**  Open Scratch on the teaching computer so kids can see examples of projects before they get started. Explain that the Level one projects are at a degree of difficulty that most people would be able to achieve by the end of the program.  **Level 1:**  SuperCat:  <http://scratch.mit.edu/projects/43335338/>  Super Scratchy:  <http://scratch.mit.edu/projects/43034668/>  **Level 2:**  Water the Superhero:  <http://scratch.mit.edu/projects/43270376/>  Super Dog:  <http://scratch.mit.edu/projects/41680046/>  **Level 3:**  Super Alpaca:  <http://scratch.mit.edu/projects/43117134/> |  |
| 11:05 – 11:10 | **Hook up and Start-Up the Raspberry Pis**   * Hand out RPis * Split into 6 groups * Help groups plug Pis in using this sequence:   + 2 USB ports (mouse, keyboard)   + HDMI   + Power cord * Log on to RPi: * Username: pi * Password: raspberry   Then – startx |  |
| 11:10 – 11:15 | **Introduction to Scratch –**   * Describe how the blocks work to create commands * The blocks fit together like puzzle pieces but they have to be in the right order to work * Walk through the Scratch interface   + Difference between the Blocks Palette and Scripts Area and Stage   + Green flag and red flag | Open scratch (screenshots?) |
| 11:15 – 11:20 | **Scratch Activity 1**  Don’t give too many instructions, help when asked. | Doc #724747 Activity 1 |
| 11:20 – 11:30 | **Scratch Activity 2**  Swap who has control over the mouse  Complete activity  \*If pairs finish early get them to try a challenge card | Doc #724747 Activity 2 |
| 11:30 – 11:35 | **Review of Concepts**  The purpose of scratch is to help instill ideas of blocks of code – of how loops work, how different pieces of code interact with one another, but also how to improve on projects, how to remix and collaborate. | Scratch website |
| 11:35 – 11:40 | **Python Intro** (5)  Python is a popular coding language also used for games (among other things).  Like in Scratch you are using a specific vocabulary to make things happen. Programming. One of the things that make Python a good coding language is that it is open source. [ask if anyone knows what open source is]  Open Source means that the core of the language is made open so that anyone can see it, build off of it, improve it. As a result of this, and the popularity of python, there are hundreds of what are referred to as “libraries” that anyone can use in their own Python-based projects.  Some things to remember as we get started. Certain parts of the vocabulary are predefined “True” vs “true”  But a lot of it is what you define – especially what we’ll be playing with – variables. More on that in a minute.  Another note – this is a beginner exercise that due to limits of time and (my) expertise doesn’t necessarily give you all of the building blocks that you will need to be | Whiteboard |
| 11:40 – 11:55 | **Python Activity 1** (15) – Tip Calculator  Open up Python 3  Save the empty file as **“script.py”**  Python allows you to set up variables – variables are blocks with values. They can be numbers, combinations of other variables, math, and other things.  Our first exercise is going to be to try to create a simple Tip Calculator.  What is the first thing that you need to know when you are thinking about a tip? [cost of meal]  Our first variable is going to be to define the cost of the meal.  **meal = 44.50**  What else do you need to know? [tax] In this case the tax is 6.75% But we can’t write it is way – how else could we write it?  **tax = 0.0675**  And finally we need to decide what % of tip to assign the meal  **tip = 0.15**  Now we need to do the math!  First we “redefine” meal cost as meal with tax  **meal = meal + meal \* tax**  and from this new cost we create a total  **total = meal + meal \* tip**  and we’ll now tell it to print this new total.  **print("%.2f" % total)**  Save and run.  This looks a lot like math class doesn’t it?  Let’s try changing a variable to change the total – remember to save | Handout #  Python 3 |
| 11:55 – 12:15 | **Python Activity 2** (20) - Interactive tip calculator  So we can define our own variables as we saw above, but there are functions like “print” that are built-in functions. One that will allow us to have a more interactive Tip Calculator is called “input”  Create a new file, “**script2.py”**  To create an interactive tip calculator we need to make three changes to our script   1. Create dialogues for us to input the costs and percentages ourselves   input() with “” serves this purpose.  **meal = input("What was the cost of the meal?")**  **tax = input("What is the tax %?")**  **tip = input("What % tip would you like to leave?")**   1. Tell python that the inputs are integers so that we can do some math to them   Wrapping input in int() tells python that the content being typed in is an integer and not a string.  **meal = int(input("What was the cost of the meal?"))**  **tax = int(input("What is the tax %?"))**  **tip = int(input("What % tip would you like to leave?"))**   1. Create a variable that will let us transform the tax and tip percent into those smaller numbers to multiply by   Decimal allows us to convert the tip and tax percent into a number since we cannot divide a variable by 100.  **decimal = 0.001**  Then we transform the variables from percentages into decimals.  **tax = tax \* decimal**  **tip = tip \* decimal**  And the rest is the same as previous.  **meal = meal + meal \* tax**  **total = meal + meal \* tip**  **print("%.2f" % total)** | Python3 |
| 12:15 –  12:20 | **Review what the Pi is and the Process**  We’ve just used these Raspberry Pis to work on some programming activities, but they can be used for so much more!  There are projects at the library that we have used Raspberry Pis for – the video display in October and November was powered by a Pi, and recently we changed the “brain” inside of the daily events kiosk from a regular computer to a Pi.  They are great pieces of hardware to tinker with, try new things, and learn new skills. |  |
| 12:20 – 12:30 | **Where to go from here?**  This is a jumping off point! I hope that you have been inspired to learn more – or even just reassured yourself that coding is a skill like any other, on  Handout  Review what we have at the library  Answer questions | Handout |