Science Experiment:

Project: Coding
Lesson 1, Programming Language

Supplies:

Each child should have a pencil, paper and ruler.

Time:
15 minutes (quick easy ice beaker to get them thinking)

What to Do:
Explain that computers follow exact directions. Every step of a process must be “coded”.
See if the children can draw a picture from these instructions.

1. Draw a dot in the center of your page.
2. Starting at the top left-hand corner of the page rule a straight line through the dot finishing at the bottom right hand corner.
3. Starting at the bottom left-hand corner of the page rule a line through the dot, finishing at the top right hand corner.
4. Write your name in the triangle in the center of the left-hand side of the page.

Reflect:
Allow youth to volunteer to share.

What questions came to mind as you heard or read the directions?
What would happen, the end result, if a step was skipped?
How does a computer know what to do?

Apply:
How does a computer know what to do?

Resources:
http://csunplugged.org/programming-languages

Science Experiment:

Project: Coding
Lesson 1, Programming Language

Supplies:

Ahead of time prepare cards with images on them. Smiley faces, squares, triangles, etc on 3x5 note cards.
Paper
Pencils

Time:
40 minutes

What to Do:
Choose a child and give them a card with an image. The child describes the picture for the class to reproduce. The children can ask questions to clarify the instructions. The object is to see how quickly and accurately the exercise can be completed.
Repeat the exercise, but this time the children are not allowed to ask questions. It is best to use a simpler image for this exercise, as the children can get lost very quickly.
Now try the exercise with the instructing child hidden behind a screen, without allowing any questions, so that the only communication is in the form of instructions. Point out that this form of communication is most like the one that computer programmers experience when writing programs. They give a set of instructions to the computer, and don't find out the effect of the instructions until afterwards.

Reflect:
Have the children draw a picture and write down their own instructions. Try them out in pairs or as a whole class.

Apply:
Imagine the consequences of an error in the program of a computer in a space shuttle launch, a nuclear power plant, or the signals on a train track!

Resources:
http://csunplugged.org/programming-languages

Science Experiment:

Project: Coding
Lesson 2, Harold the Robot

Supplies:

Legos or wooden blocks

Time:
60 minutes

What to Do:
Explain: In this activity children simply give directions to a "robot" (either an adult or another child) and find out which instructions the robot is able to follow, and how their instructions are taken literally.
1. Place a small collection of blocks or similar objects on the bench/table.
2. One person (perhaps the teacher or other adult for a start) plays the role of Harold the Robot. Harold can only respond to particular commands. These commands are *not* given to the children, and can be made up on the fly.
3. Have a child talk Harold through making a tower out of the blocks using instructions such as "Move your hand to the left", "Pick up the block beside your hand" and so on. If the child gives an instruction that is too complex or otherwise not in Harold's vocabulary (e.g. "put the three blocks on top of each other") then Harold expresses confusions by shaking his head or burying his head in his hands.
4. The task is completed when the tower is built. At this point, discuss with the children about which commands it would be reasonable for the robot to respond to, which wouldn't make sense. Does a small vocabulary limit what can be done, or does it simply make more instructions necessary?

Reflect:
This activity is intended to expose students to the idea that computers follow instructions very precisely, which can be frustrating at times. It also raises the issues surrounding choosing instruction sets, and whether it's better to have a large complex instruction set, or a small efficient set.

Apply:
Imagine the consequences of an error in the program of a computer in a space shuttle launch, a nuclear power plant, or the signals on a train track!

Resources:
csunplugged.org

Science Experiment:

Project: Coding
Lesson 3, Debugging & Algorithms

Supplies:

Lined paper or graph paper
Pencils or crayons or markers
Chalk Board or White Board for front of room.

Time:
60 minutes

What to Do:
1.Draw a 3X3 Checker Board on a piece of paper or white board, next to it write out the Following key:

- → Move One Square Right
- ← Move One Square Left
- Down Arrow = Move one Square Down
- Up Arrow = Move one square Up
- XXX = Fill in Square

2.Ask the youth to write an algorithm (or Instructions) for drawing this image.
3.Go through some of their algorithms to see if they work.
4.Now write this sample algorithm on the white board/chalk board where all can see.

"Move right, fill-in, move right, move down
move left, move left,
fill-in, move right, move right, fill-in, move down
move left, move left
move right, fill in, move right"

5.Ask the youth to write a program code for the algorithm using the program key.
6.Go through some of their programs to see if they work.

Reflect:
Why do you think computer programmers use programs and code instead of typing out verbal algorithms?
What are some of the challenges of debagging code written by another person?

Apply:
Why are symbols useful? Can you think of some examples of useful symbols that you see everyday?

Resources:
K-8 Intro to Computer Science Course (2013). Retrieved from http://learncode.org

Science Experiment:

Project: Coding
Lesson 4, Relay Programming

Supplies:

Lined paper or graph paper
Pencils
2 tables.
Multiple copies of a 6X6 graph paper drawing, for suggestions see the reference at the bottom of the page.

Time:
30 minutes

What to Do:

1. Divide a group of youth into teams of 4-6 people
2. On one side of the gathering or open space have a station with a 6X6 graph paper drawing, a piece of paper, and pencil for each team.
3. Line up each team in a row on the opposite side of the room (think of a relay race running across the room)
4. Explain the following rules to the youth:
    a. One person from each team must hurry to the "station", across the room.
    b. The person must write the first programming symbol for the graph paper drawing.
    c. They then must hurry back across the room and tag the next person in their row.
    d. The tagged person will go across the room to the station and review the teammate's work.
        i. If correct, the will add another symbol.
        ii. If they find an error, then they will only correct the error.
    e. They then hurry back and tag the next teammate who will review and edit or add the next.
    f. This continues until a team completes their graph paper drawing programming.

Reflect:
Would it be easier or harder to have fewer in a group or a large number of individuals writing code?
What are some of the challenges of debagging code written by another person?

Apply:
How can working as a team make it easier to make errors? Or make it easier to avoid errors?
What are some of the advantages of having another person look over your work?

Resources:
K-8 Intro to Computer Science Course (2013). Retrieved from http://learncode.org

Science Experiment:

Project: Coding
Lesson 5, Flappy Bird

Supplies:

Computers or tablets (atleast one for every two club participants)
Internet Access

Purpose: Today the youth will incorporate various coding functions to create works of art.

Time:
40 minutes

What to Do:
To get started follow these simple instructions:

1. Go to studio.code/flappy/1
2. Create your own version of Flappy Bird.
3. Run the code first by clicking on the “Run” program button.
4. Click on the “Show Code” text to compare and contrast the code represented in JavaScript.
5. Click on the “Run Program” button to see if your code works. To start a puzzle over, click the “Reset” button.
6. After each puzzle you solve correctly, a prompt will pop up to congratulate you. Before clicking the “Continue” button, click on the “Show Code” text to review the blocks you have organized represented in JavaScript.

Reflect:
Trial and error is an important part of coding. What did you do when you created a faulty code for Flappy Bird?

Apply:
Give an example of a time when you may need to work through a problem using trial and error?
If you could design any video or computer game, what would you design?

Resources:
K-8 Intro to Computer Science Course (2013). Retrieved from http://learncode.org
Utah State University Extension

Science Experiment:

Project: Coding
Lesson 6, Binary Bracelets

Supplies:

String
Beads (lots of beads preferable black and white)
Copies of a Binary Decoder Key, found in multiple places on the web.

Purpose: Students will: Encode letters into binary, decode binary back to letters and relate the idea of storing their name on a bracelet or necklace to the idea of storing information in a computer.

Time:
60 minutes

What to Do:

1. Pass out the Binary Decoder Key.
2. Have each youth find the first letter in their name.
3. Their letter is represented by 8 different squares and each square is either “open” or “closed”, “black” or “white”, “filled” or “not filled”. The terminology is not important but the understanding that each letter has a specific unique code or set of blocks that are “black” or “white”.
4. Pass out a thread and have the youth place 8 beads on to the thread to match the binary code that makes up the letter of their first name.
5. Have students complete a bracelet of their first name or a necklace of their first and last name.

Reflect:
What else might you represent binary instead of boxes that are filled or not filled?

Apply:
What might happen if an error is made in the conversion of binary. How does one error change the spelling of your name?

Resources:
https://code.org/contact