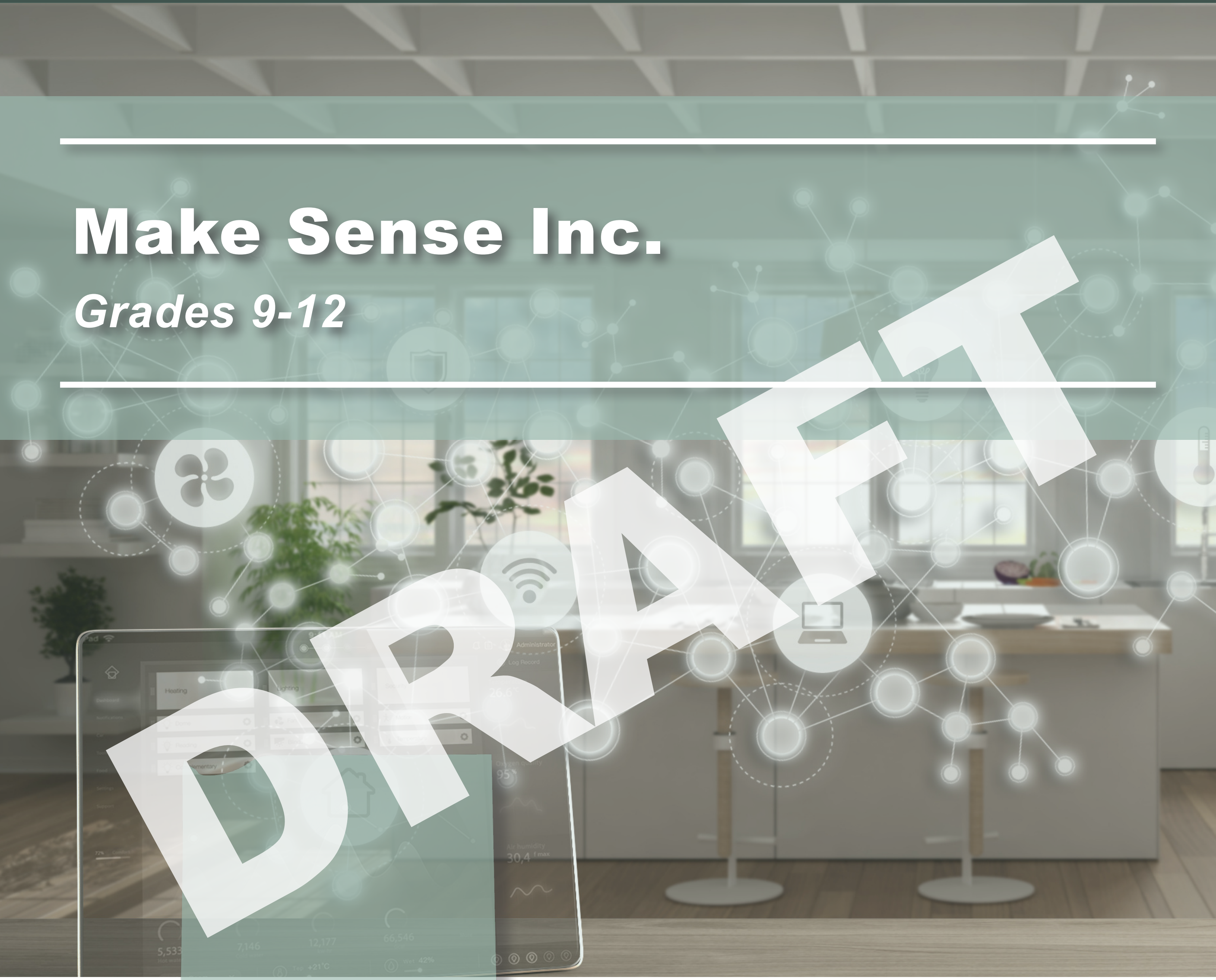
Make Sense Inc.

Grades 9-12



SCALE K-12

SCalable Asymmetric Lifecycle Engagement

Precollege Microelectronics Workforce Development

INSPIRE

Research Institute for Pre-College Engineering

PURDUE UNIVERSITY

Cover Information

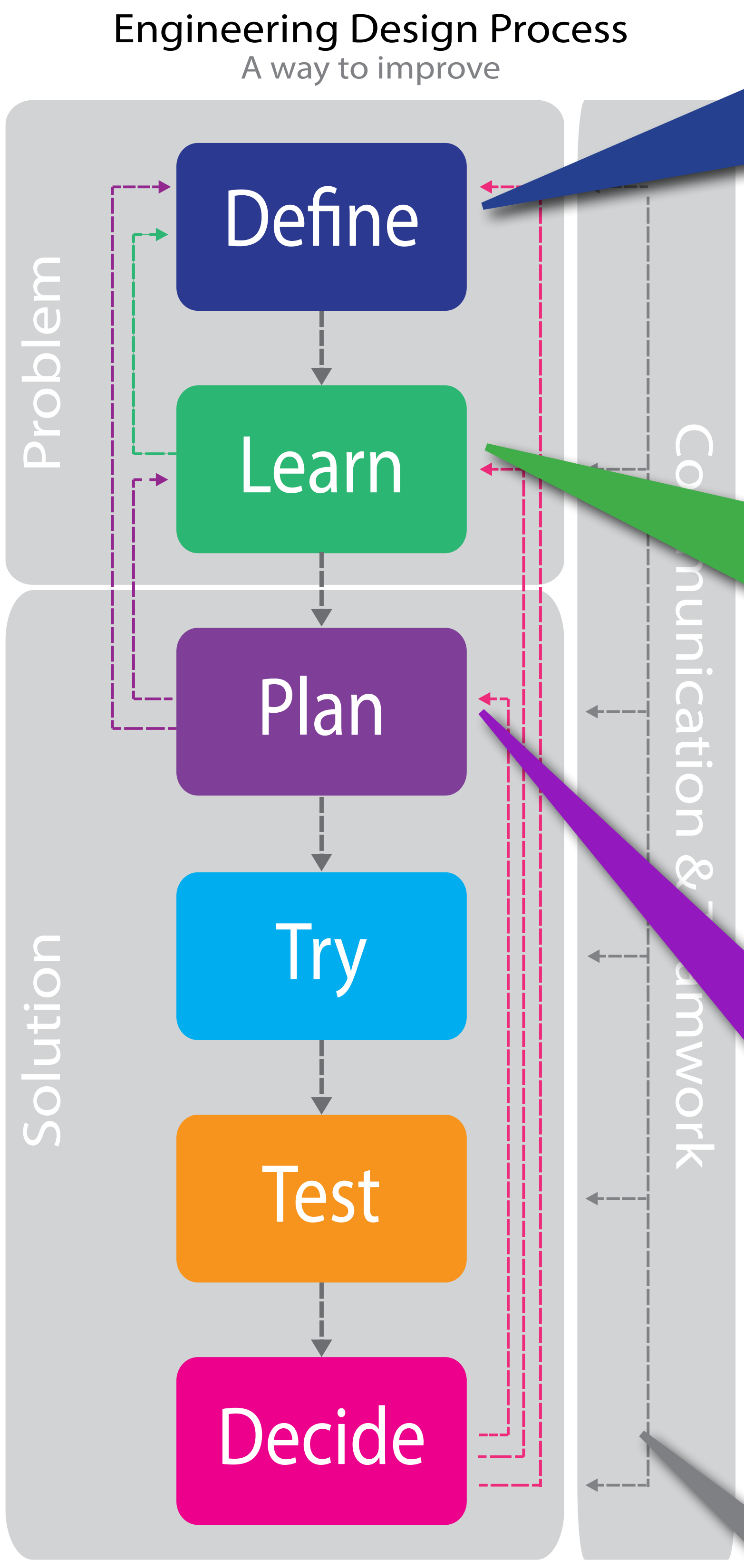
Unit Title:	Make Sense Inc.
Grade Level Range:	Grades 9-12

Acknowledgments

Teacher Authors
Jonathan Woessner
Tracie N McAtee

Program Authors
Azizi Penn
Breejha Quezada

Overview: Engineering Design Process



Copyright © 2015 PictureSTEM- Purdue University Research Foundation

DEFINE THE PROBLEM

- Who is the **client**?
- What does the client need?
- Why does the client need it?
- Who is the **end user**?
- Why might the end user want it?
- What are the **criteria** (requirements)and **constraints** (limits) of the solution?

Problem Scoping:
WHO needs **WHAT** because **WHY**

LEARN ABOUT THE PROBLEM

- What kind of background knowledge is needed?
  - What science/math knowledge will be needed?
  - What materials will be needed?
- What has already been done to solve the problem?
- What products fill a similar need?
- How should we measure success and improvement?

PLAN A SOLUTION

- Continue to specify the criteria/ constraints
- Generate ideas of possible solutions
- Develop multiple solution paths
- Consider **constraints, criteria**, and **trade-offs** (criteria that compete with one another)
- Choose a solution to try
- Develop plans (blueprints, schematics, cost sheets, storyboards, notebook pages, etc.)

COMMUNICATION

- Communicate the solution clearly and make sure it is easily understandable
- Use **evidence** to support why the client should use your solution

Overview: Engineering Design Process

TRY A SOLUTION

- Put the plan into action
- Consider risks and how to optimize work
- Use criteria/constraints and consider trade-offs from the problem/plan to build a **prototype** (a testable representation of a solution), **model**, or **product**

TEST A SOLUTION

- Consider testable questions or hypotheses
- Develop experiments or rubrics to determine if the solution is meeting the stated criteria, constraints, and needs
- Collect and analyze data

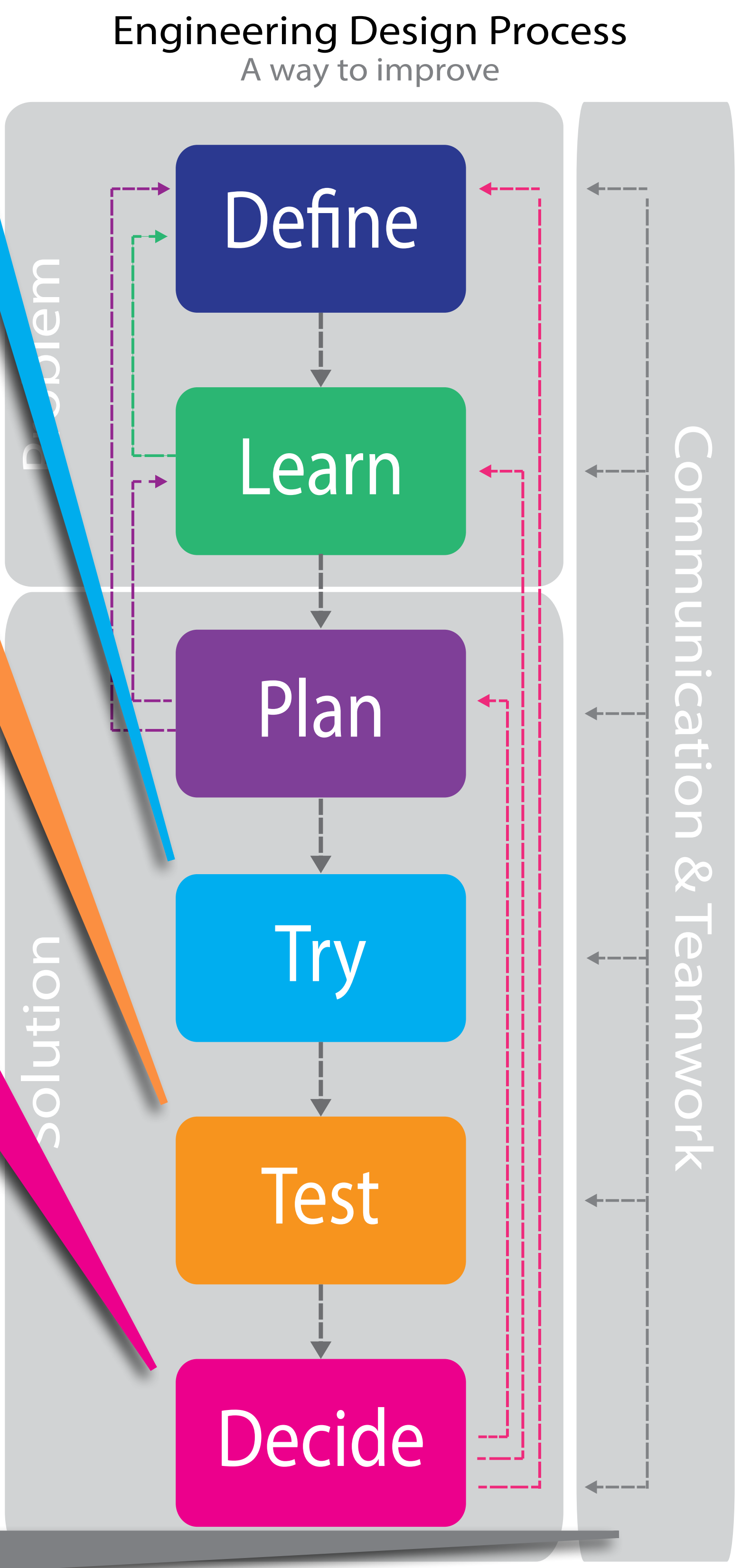
DECIDE IF THE SOLUTION IS GOOD ENOUGH

- Are users able to use the design to help with the problem?
- Does the design meet the criteria and constraints?
- How could the design be improved based on test results and feedback from the client/user?

Iterative nature of design: Always consider which step should be next!

TEAMWORK

- Discuss in teams how the solution meets the criteria and needs of the client
- Consider different viewpoints from each teammate



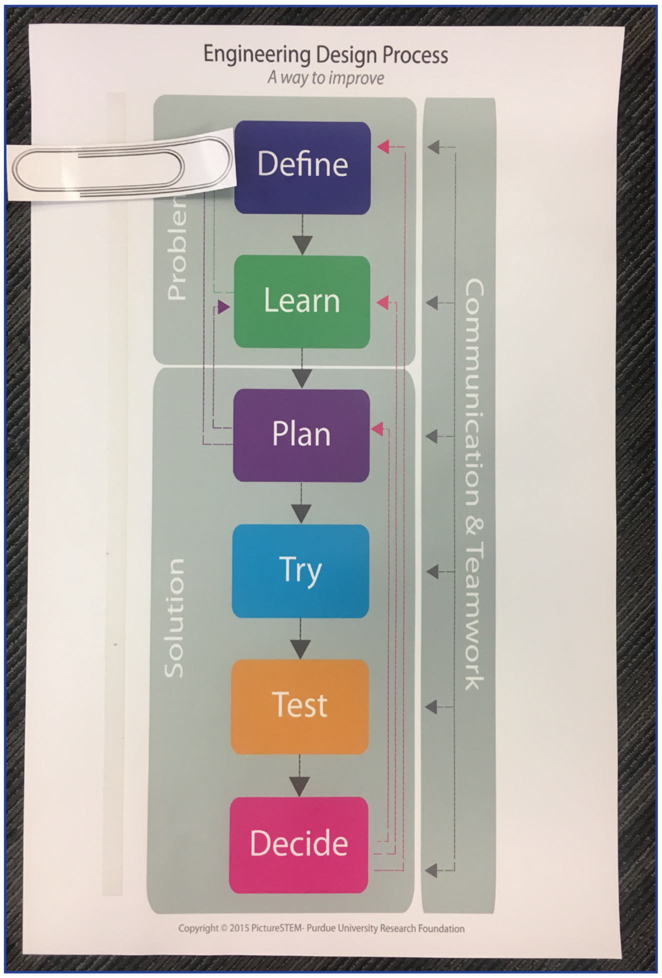
Overview: How to make EDP sliders

How to create the poster

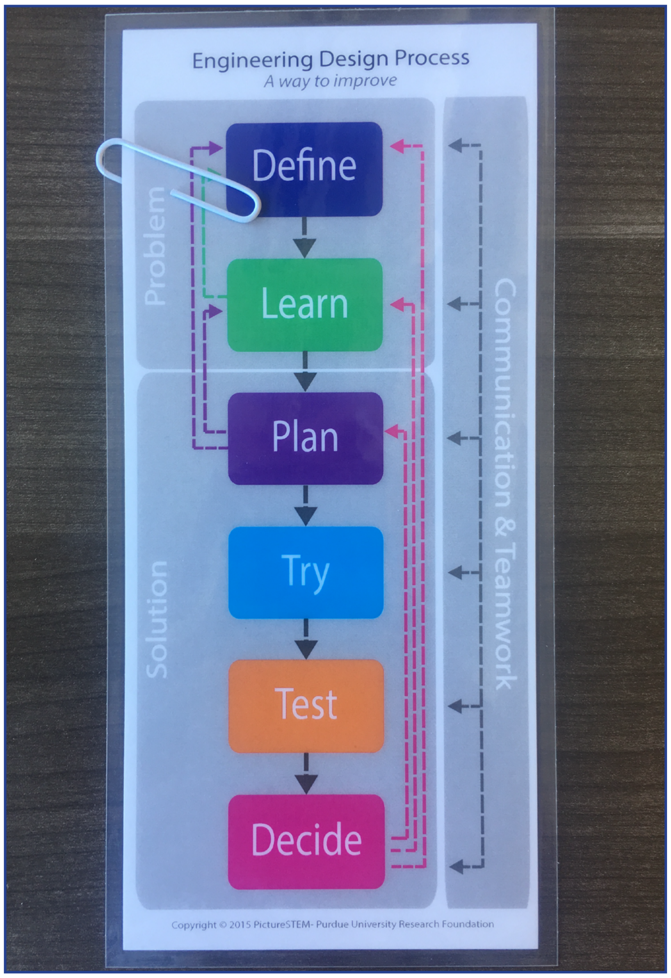
1. Download the high-quality PictureSTEM Slider Poster and the paper clip images from PictureSTEM.org.
2. Print the poster and the paper clip on poster-sized paper and cut to size. High-gloss or semi-gloss paper is the best choice.
3. Use self-sticking Velcro on the back of the paper clip and down the side of the poster so that the paper clip can be placed to point at all 6 sections of the slider.

How to create individual sliders

1. Print the sliders on the opposite page - enough for one slider per student in your class.
2. Cut the sliders apart.
3. Laminate the sliders individually.
4. Use a jumbo paper clip as the pointer for each slider.


Engineering Design Process
A way to improve
Problem
Define
Learn
Plan
Try
Test
Decide
Solution
Communication & Teamwork
Copyright © 2015 PictureSTEM- Purdue University Research Foundation

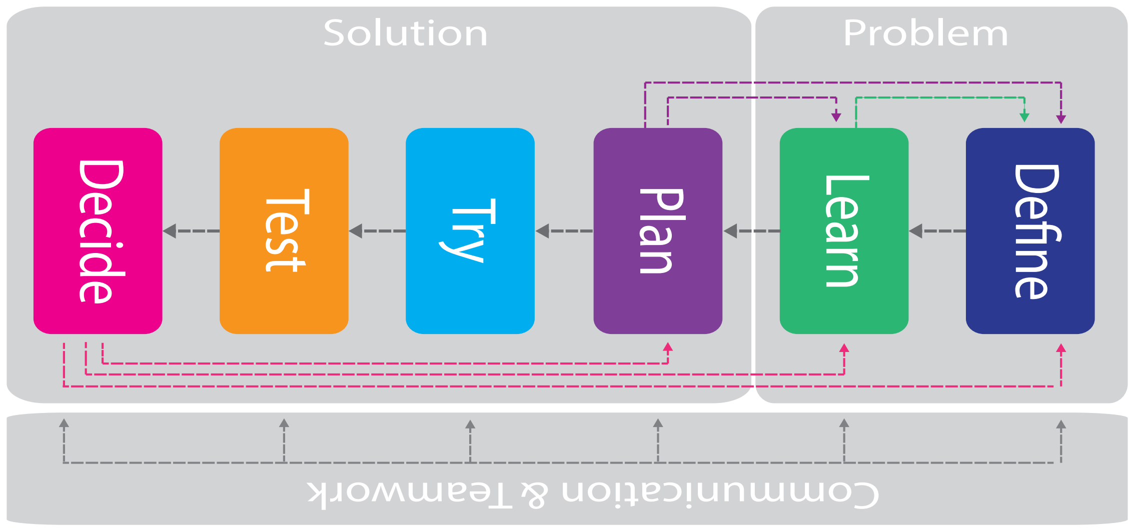
Poster


Engineering Design Process
A way to improve
Problem
Define
Learn
Plan
Try
Test
Decide
Solution
Communication & Teamwork
Copyright © 2015 PictureSTEM- Purdue University Research Foundation

Individual slider

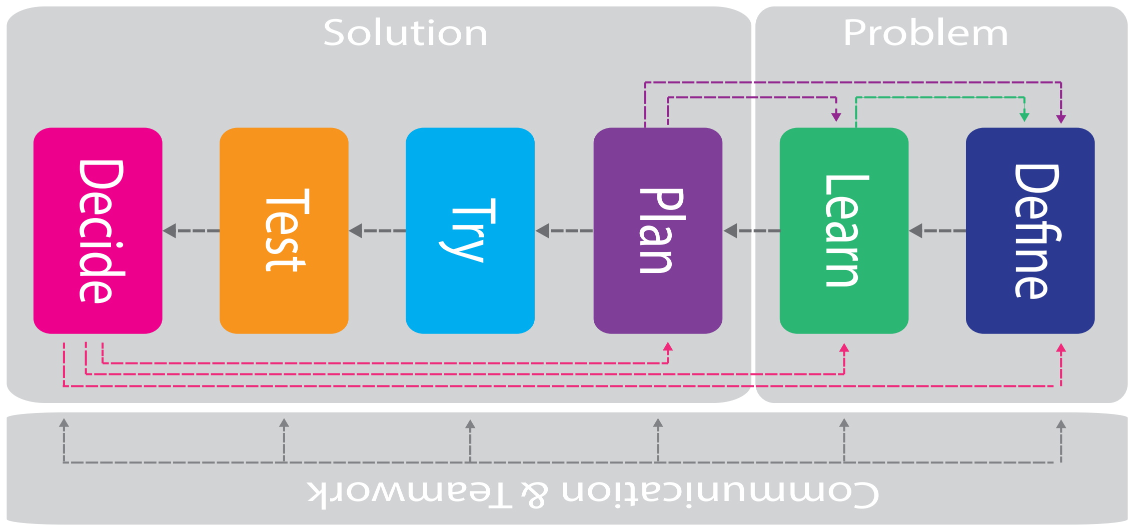
Engineering Design Process

A way to improve



Engineering Design Process

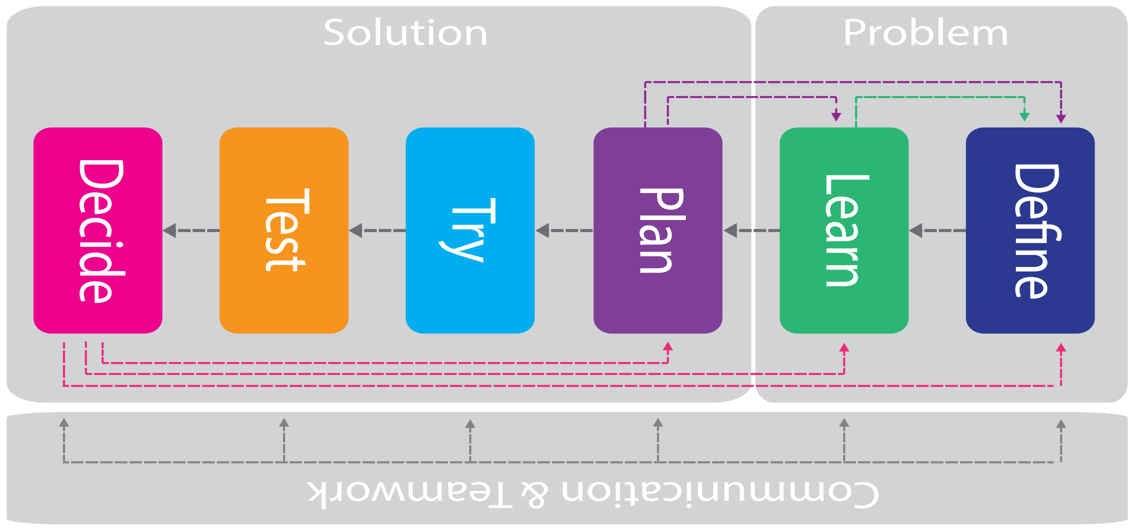
A way to improve



Copyright © 2015 PictureSTEM- Purdue University Research Foundation

Engineering Design Process

A way to improve



Copyright © 2015 PictureSTEM- Purdue University Research Foundation

Unit Overview

Lesson Summaries

Lesson 1: Make Sense Inc.
Make Sense Inc. is a smart home development company looking to build new and improved homes in Indiana. They want to take advantage of the new microelectronics industry that is growing in Indiana, and wants your help. As computer scientists and engineers, your job is to learn about the sensors and the computer code that we will use to build homes and satisfy customer needs

Lesson 2: What Makes a Computer a Computer?
Do we need computers in our smart homes? To answer this question we want you to explore what characteristics determine a computer, and be able to classify all types of electronics as either computers and not computer. To best bring you up to speed on current technology, we would also like you to learn about microcontrolers and how the fit into what our industry needs to do.

Lesson 3: Function-al Design
Functions are one of the most important elements of computer science, and because of that a key part of our favorite apps and other devices. This lesson will explore how to set up functions that will help us program our smart homes.

Lesson 4: Make Some Sense
Summary
Teacher will model assembling the temperature sensor and creating a function using the sensor.

Students will be assigned groups along with a sensor. Each group will assemble their sensor and create a function using the assigned sensor. Each group will share out their findings with the class

Lesson 5: Plan It Out
Plan - Drawing Pictures, Creating Flowcharts, Writing Pseudocode

Students will choose sensors and actuators to accomplish their chosen smart home goals. They should identify inputs, conditions, and desired outputs. They should then plan out how they will use sensor data to make decisions and control the outputs, using flowcharts and pseudocode. They will design based on a standard mock home from the client. They should choose which sensors to use, as well as where to place them in the home and what to use them for. Continue to reference the client letters to insure you are meeting their needs.

Lesson 6: Try It Out
Creating an algorithm, Using a common model to try the code.

Using their model home, sensors, and microbit, assemble your smart home layout. Use your pseudocode and flowchart to create python code to control the smart home.

Lesson 7: Test It Out
Summary
Test your design against a standard rubric. Evaluate how well your design meets needs such as usability, features, efficient sensor use, etc

Lesson 8: Decide/Redesign
Client letter suggests/asks for more complex triggers. We anticipate initial student designs will be very 1-dimensional (ex: push button, turn light on/off). The client will request solutions that have additional complexity or inter-related-ness. Examples: change the temperature set point based on time of day or outside weather or human presence, change light brightness or temperature based on brightness outside or time of day, allow user to wirelessly control parts of the house (with a second microbit).

Standards Addressed

7183.D2.1 Identify the standard documentation tools of displaying algorithms such as pseudocode, flowchart symbols and UML
7183.D2.3 Apply truth tables, Boolean logic, control structures, relational and logical operators to program algorithms
7183.D2.5 Document and express code and algorithms in an easily understandable manner using tools such as data flow diagrams, flowcharts, use case diagrams, activity diagrams, and state tables.
7183.D2.6 Develop a simple program and/or script using a compiled, object-oriented scripting language like Python.
7183.D2.15 Describe the components of a computer architecture
7183.D2.17 Successfully identify and debug errors in applications produced by themselves or others.
7183.D2.19 Apply critical thinking and problem-solving methodologies
7183.D2.20 Show the ability to delegate tasks into user defined procedures for the purpose of efficiency.

Unit Description (Make Sense Inc.)

In this unit that pairs computer science with engineering
design, students are contracted by “Make Sense Inc.,” a fictional smart home development company aspiring to incorporate Indiana’s emerging microelectronics industry into their smart home designs. Students do this by learning about Python functions, sensors, computers, and microelectronics. Ultimately, they use their functions and coding knowledge to program a microcontroller kit with interactive sensors to simulate a potential smart home system design.

Page title

Page title

LESSON ONE:

Lesson Objectives
Students will be able to:
- Identify the problem from a client.
- Identify background knowledge needed to develop a solution.
- Explain the criteria and constraints.

Time Required
One 50-minute lesson

Standards Addressed
NGSS HS ETS1-1 Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.

ICS-2.1 Use the design process to iteratively develop a computing artifact.

ICS-5.3 Utilize a problem-solving approach to develop a solution using technology.

Key Terms
Client, engineering design process, criteria, constraints, microelectronics, smart home

Lesson Materials
Per classroom
- EDP Poster
- Chart paper

Per Student
- EDP slider and paperclip
- Laptop/Chromebook/ Tablet
- Engineering notebook

Lesson Summary

Students are introduced to the Engineering Challenge by their client which will serve as the context within which students can learn about microelectronics and coding functions in Python. Students will learn about the Engineering Design Process and then take part in iterative class and group discussions to identify criteria, constraints, and knowledge gaps needed to successfully solve the client's challenge.

Teacher Background

Teamwork

Students should be teamed strategically and may or may not be assigned jobs within their team. When forming student teams, consider academic, language, and social needs. In place of strategic teaming, a random teaming can be substituted. Students will work in these teams of 3 or 4 throughout the unit. Effective teamwork is essential in this unit as well as in engineering in general. The teams will operate as software engineers with each team working together to develop an application for a smart home sensor and code the function to control it.

Engineering Design Process

NOTE: If students are familiar with the engineering design process (EDP) before beginning the unit, the teacher can skip this (EDP) introduction.

The engineering design process (EDP) is an iterative, systematic process used to guide the development of solutions to engineering problems. There is no single engineering design process, just like there is no one scientific method. However, the various engineering design processes have similar components. The engineering design process (EDP) involves understanding the problem, learning background information necessary to solve the problem, planning, trying, testing the solution, making changes based on the tests, and communicating their ideas. Students will use an engineering design process slider throughout the unit to help them understand where they are in the design process. For more information about the steps of the engineering design process presented in this unit, see the front matter section about it.

Some common misconceptions about engineering

- Engineers do not have to learn anything new when they are working on a project. **In reality:** Engineers need to continually learn throughout their lives.
- Engineers come up with solutions that are just “good enough” and do not take risks. **In reality:** Engineers strive to create the best solution possible through optimization. It is normal to experience failure when solving engineering problems.
- Engineers work alone to solve a design problem. **In reality:** Engineers collaborate with people in different disciplines and fields to best solve a problem. Engineering problems often require a wide range of content knowledge.

Some common misconceptions about the EDP

- The engineering design process is linear, and you never need to go back to previous phases. **In reality:** The EDP is a cyclical process that requires many iterations.
- Once the project is done, it is considered complete and not revisited. **In reality:** The engineering design process is never really “done” and it is revisited so engineers can improve projects and make changes.

Criteria and Constraints

One difficulty that students might experience is distinguishing between criteria and constraints. Criteria are the things required for a successful design, or goals of the designed solutions. They help engineers decide whether the solution has solved the problem. Another way of thinking about criteria are that they represent anything that the client and the engineers will use to judge the quality of a solution. Constraints are a specific type of criteria; they are those criteria that limit design possibilities, or the ways that that problem can be solved. If constraints are not met, the design solution is by default not a viable solution to the problem. The relationship between criteria and constraints is represented in the figure. It may be helpful to post the definitions with the figure somewhere in the classroom for future reference.

Problem Scoping

In this lesson, students will be in the problem scoping section of the engineering design process, specifically on the define the problem step. Define the problem and learn about the problem combine to make problem scoping. In this stage, students will be

Duplication Masters
1.A Python Installation Instructions
1.B Content Pre-Assessment
1.C Client Letter
1.D Problem Scoping
1.E Client Response

Educator Resources
1.F Content Pre-Assessment Key
1.G Problem Scoping Key

Assessment

Pre-Activity Assessment
Assess students’ prior knowledge by listening to their responses to 1.D Problem Scoping. Use students’ answers to 1.B Content Pre-Assessment as baseline data about the students’ current level of understanding and background knowledge. Do not assess 1.B as right or wrong.

Activity Embedded Assessment
Observe students’ discussions and written responses to 1.D Problem Scoping. Check students’ brainstorming lists to see if they can identify the content they will be expected to master by the end of the unit.

Post-Activity Assessment
Use the 1.G Problem Scoping Key to evaluate students’ answers to the notebook prompts.

first introduced to the engineering problem through a client letter and then be given a chance to ask questions to the client to receive more information about the problem. The problem statements given in the client memos purposefully do not provide all the information necessary to solve the problem. Students are tasked with generating questions about the problem to try to fill in this missing information. Based on all information from the client, students will then define the problem in terms of: what the problem is and why it is important, who are the client and end users, what are the criteria and constraints, and what other information they may need to learn about in order to solve the problem. This process of generating ideas and questions for the client is an important skill on its own both in engineering and in other fields, but it also helps to ensure that the students fully understand the problem and their task in the engineering design challenge.

Solution Generation

The Solution Generation section of the engineering design process includes plan the solution, try out the plan of the solution, test the solution, and decide whether the solution is good enough. When engineers are generating solutions, they will use iteration as a means to continually improve their solution, reflect back on the problem definition and what they have learned about the problem, and consider criteria, constraints, and trade-offs. Trade-offs involve having to make compromises about which criteria to emphasize because they compete with one another in terms of making the solution effective. For example, cost could be a trade-off for durability.

Engineering Notebook

Throughout the unit students will be recording information in an engineering notebook, and they will need the notebook immediately in Lesson 1. The engineering notebook is digital set of documents which includes writing prompts, blank space to take notes or upload pictures of work, and digital copies of the duplication masters that are listed in each lesson. The engineering notebook is offered as a google doc but can be adapted to your classroom needs. Students' engineering notebooks will support their communication of ideas and should be used consistently throughout the unit.

Vocabulary

Students will be introduced to many new science and engineering vocabulary terms throughout the unit. It may be helpful to create a vocabulary section in their notebook with term definition and memory clue. Additionally, the class could maintain a word wall.

Before the Activity

- Assemble the Engineering Design Process Sliders and post the EDP poster in the classroom (see the front matter for how to assemble them). If your students do not want to use the sliders, simply hanging the poster achieves the same result. Make sure you and your students can refer to the EDP sliders and/or poster throughout the unit.
- Determine student teams of three or four. These teams should be their teams throughout the rest of the unit.
- Python should already be installed on classroom and student computers. The instructions for installation can be found in 1.A Python Installation Instructions
- Print and make copies of the following worksheets in the labeled amounts:
    - (1 per student) 1.B Content Pre-Assessment, 1.C Client Letter, 1.D Problem Scoping (or they can put the 1.D answers directly in their notebook – recommended)
    - (Optional) Prepare the pre-assessment activity in the form of a survey, kahoot, etc. using the questions on 1.B Content Pre-Assessment

Classroom Instruction

Introduction

1. **Complete the content pre-assessment activity.** The students will participate in a more formal pre-assessment to assess their current level of knowledge and understanding regarding the topics of the curriculum, microelectronics, and the engineering design process. Using the questions on the 1.B Content Pre-Assessment, distribute hard copies or have students respond to a digital version of the survey. Make sure to tell students that this is just to assess any prior knowledge, so it is okay to not know the answers.
2. **Review prior knowledge.** Lead a discussion with the class in which students are able to share their prior knowledge on the topics of engineering, coding, and microelectronics. Prompts

may include the following: *What do engineers do? What kinds of industries do engineers work in?*

3. **Introduce and set up engineering notebooks. Say:** *Engineers use notebooks to document their design process and keep notes. We will also be using engineering notebooks throughout our engineering challenge. Each day, you'll use the notebooks to take notes and record what you are learning. In addition, there are questions that you'll be asked to answer.* NOTE: You can have your students type in their notebooks in two different colors – one for thoughts and prompts that are individual and one for thoughts and prompts that they discuss in their teams. This will help both you assess, and the students recognize, where ideas came from. You also may want to have students start a table of contents.
4. **Form teams.** After students have finished the prompts, explain that for the rest of the unit they will start the day with a review of the engineering design process, and then look at a specific problem that will require the use of that process. Explain that students will be working in small teams to solve a problem being brought to them by the client. Divide students into teams of 3s and 4s.

Activity

5. **Introduce the problem.** Let the students know that they are being asked to work for a company called Make Sense Inc. Provide copies of the 1.C Client Letter to the students so that they can read it. Encourage them to add information to their notebooks as they read to keep track of important items. Give students time to discuss in small teams what information they read in the letter.
6. **Complete problem scoping section 1.** The above sets up the need to discuss engineers and engineering. Direct students to the 1.D Problem Scoping in their engineering notebooks. Have students individually answer the prompts from section 1. Make sure to let them know that it is okay if they do not know very much about engineers or engineering – just have them answer these questions to the best of their ability.
7. **Discuss engineers and engineering.** Allow students to share their answers from 1.D Problem Scoping section 1. Define engineers and engineering and take some notes for students to type in their notebooks. As a class create a list of the different types of engineering and have students brainstorm careers that may fall within each type of engineering in their notebooks. Explain that the problem they will be solving falls under the

category of engineering design and draws on microelectronics to understand the context and generate a solution.

8. **Introduce the Engineering Design Process.** Display the Engineering Design Process poster and pass out individual EDP Sliders and a paper clip to each student. **Say:** *Engineers use an engineering design process, along with mathematics, science, and creativity, to understand a problem and come up with a solution. Since we are working as engineers during this unit, we will be using this engineering design process as a guide while we come up with a solution for our engineering problem.* Go through the EDP Slider and ask the students what they think each stage involves. Be sure to clarify any misconceptions and elaborate where needed. There is a detailed description of the EDP Slider in the front matter of the unit.
9. **Identify the problem from the client and develop questions for the client.** Have the students reread the letter, if necessary, to identify the problem and add it to their notebooks. Have them brainstorm questions for the client using 1.D Problem Scoping Section 2. **Ask:** *What does the client need from you? What are some possible constraints and criteria?*
10. **"Send a response" to the client with the brainstormed questions.** Create a list of the questions from your students and "send" a response from the class asking the questions the students came up with.
11. **Identify the problem from the client.** Have the students reread the original letter and response as needed to identify the problem and add it to their notebooks.
12. **Identify required information.** Have students work together to brainstorm a list of "required information" in order to help the client with their request. Encourage them to highlight/underline the things on their list they already know. Then as a class create an anchor chart that will be revisited throughout the unit. As students learn information you can check content off of the anchor chart or add to it if they think of some other information that they will need to help the client.
13. **Provide the client response to the students.** You can tell the students that you now have a response from the client (1.E. Client Response). It is best if you can make this feel as real as possible. It may be good if you can display this to the class rather than make copies so that if feels like it came as an email response. You can provide them with hard copies at a later time if you want them to be able to come back and refer to it later.
14. **Complete problem scoping section 3.** Now that you have all of the information from the client, direct students to section 3 of

1.D Problem Scoping Prompts. They can do this individually or in teams. Discuss their responses. Update your anchor chart as needed based on the conversations happening in class

Closure

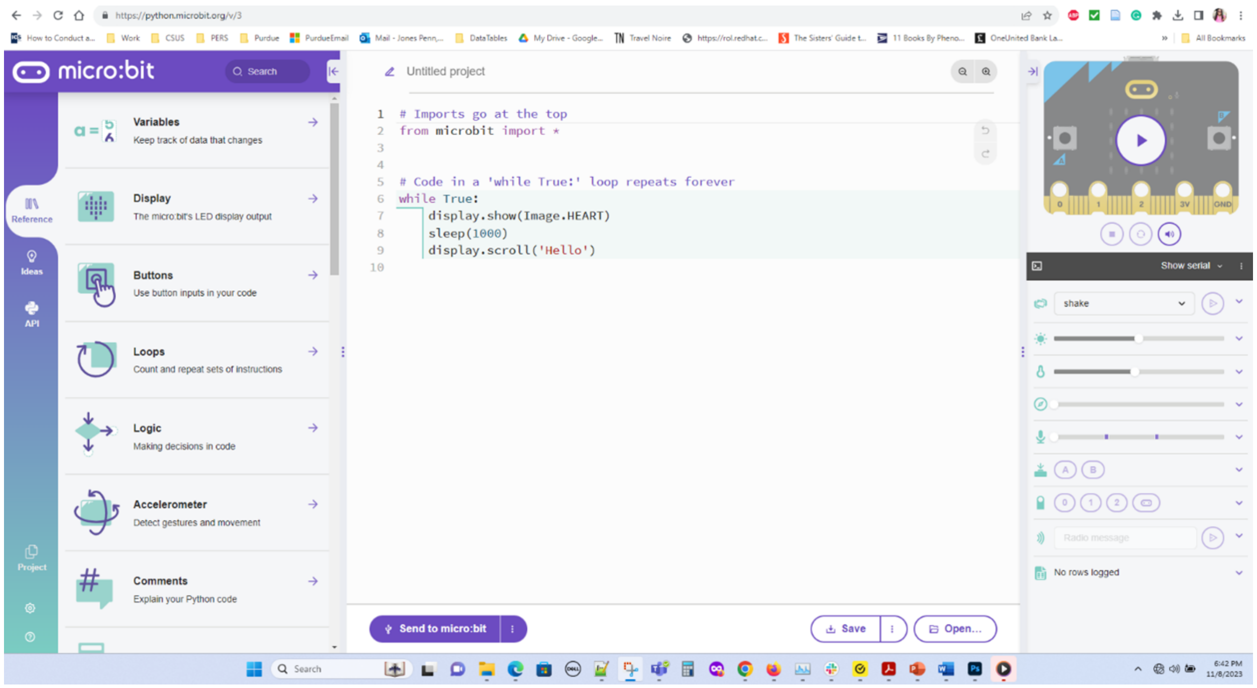
15. **Revisit the problem.** Give the students a chance to revise their list of questions or required information they composed for the engineering challenge.
16. **Discuss the engineering design process. Ask:** *Which phase of the engineering design process did we focus on related to our challenge today? Why is this important?* Ex: Students need to understand the root problem from the perspective of the client and other stakeholders before attempting a solution.

1.A Python Installation Instructions

We intend to use python.microbit.org for the programming environment.
https://python.microbit.org/
This is a completely online editor for coding in MicroPython for the micro:bit that does not need to be installed on individual student computers.
MicroPython is a subset of Python that is specifically for the microcontroller. The editor is specifically for the micro:bit microcontroller. The editor has online documentation, code hints and a linter (debugging tool). It also has a micro:bit simulator. Browse the micro:bit Python site to become more familiar with the environment.

https://microbit.org/get-started/user-guide/python-editor/

1. Students will need to be able to access the python.microbit.org site with their computers. They will also need to have a USB-A port on their computer so that they can flash the code to the micro:bit.



```
# Imports go at the top
from microbit import *


# Code in a 'while True:' loop repeats forever
while True:
    display.show(Image.HEART)
    sleep(1000)
    display.scroll('Hello')
```

Figure 1: Python editor for the micro:bit

Name______________________________________ Date______________ Period __________

1.B Content Pre-Assessment

1. Write a definition for the word, “computer”.

2. What is a central processing unit (CPU)?

3. What is the difference between random access memory (RAM) and hard drives?

4. What does an actuator do?

5. List three examples of coding languages.

6. Within programming, what is a variable?

7. What is the difference between software and hardware?

8. What does the term, “microelectronics” mean?

9. How are microelectronics used in the field of computer science?

10a. What jobs would you be interested in that use microelectronics?

10b. Provide one example of how microelectronics is used in that job.

1.C Client Letter

Dear Software Engineers,

Make Sense Inc. is company that designs smart homes in Indiana. With the recent boom of industries coming to Indiana, we are looking to capitalize on the new microelectronics coming from these industries. We would like to tie smart home sensors that are currently being developed to the new chips being manufactured here to offer our smart home customers many features to make their homes more efficient, easier to manage, more fun, etc. We need to develop code that will allow our new sensors to work with the chips as they are manufactured. We are also looking to expand how the sensors are used in our smart home applications.

The chips and sensors are not yet designed or manufactured, but we want to be ready for them when they begin to be available. So, what we need is a set of working code for already existing microprocessors that can be used as the base code for the sensors we are creating to work with the new chips as they are manufactured. So we need you to develop code that is modifiable and modular that makes our planned sensors work for smart home applications. The code to run our sensors should be able to be put easily into other code as needed for the applications intended. Some of our sensors include: a light sensor, a sound sensor, and a temperature sensor.

We will provide you with the main code that runs the whole smart home system. Your code should patch into that code and run your chosen sensor in the way you intend it to run.

I am sure that this has set up many questions for you. Please respond with questions and I will do my best to get back to you with the additional information you need.

Sincerely,

Sammy H. Sensier
Make Sense Inc.

Name____ Date_______________ Period __________

1.D Problem Scoping

Section 1:
Directions: Please answer the following questions.

1. What do engineers do?

2. How do engineers solve problems?

Section 2:
Directions: Please answer the question after hearing about the engineering design challenge.

3. What questions do you want to ask the client?

Section 3:
Directions: Please answer these questions after you have been able to ask questions about the challenge. First, complete each prompt on your own. Then write your revised answer (if different) to the prompt, based on the discussion with your team. You may use a different color writing utensil or font color to distinguish your answer and how it changed after talking with teammates.

4. The client is:

5. The client's problem is:

6. The problem is important to solve because:

7. The end-users are:

Name____ Date_______________ Period __________

8. An effective solution for the client will meet the following criteria:

9. The constraints (or the limits) of the solution are:

10. Think about the problem of designing a smart home. In terms of designing a program to help solve this problem, what are at least 2 things you need to learn in order to make an evidence-based recommendation? Make sure to consider all important aspects of the problem. Be specific.

1.E Client Response Template

Dear Software Engineers,
Thank you for your excellent questions. Here are answers to most of your questions.

[INSERT ANSWERS TO QUESTIONS]

If you have more questions for me or my team, give them to your teacher who will pass them along to us.
Sincerely,
Sammy H. Sensier
Make Sense Inc.

[DO NOT INCLUDE THIS SECTION -- THESE ARE DIRECTIONS FOR THE TEACHER] TEACHER NOTE: Before distributing this to students, fill in answers to their questions. Students will come up with a variety of questions, but good problem scoping questions help clarify the problem, identify what needs to be done, and identify the acceptable ways it can be done. Here are some sample questions/responses.
[This will need to be added later… some ideas are:
What kind of things do you want the smart home sensors to do exactly?
What's going to power these sensors, and how long do they need to last before changing batteries or charging?
Do you have any size or shape requirements for the code? Like, does it need to be super small or able to run really fast?
Is there a certain way the sensors need to talk to the main smart home system or to each other?
Lastly, are there any special features you already know you want to include?]

1.F Content Pre-Assessment w/Answer Key

1. Write a definition for the word, “computer”.

   A computer is a programmable electronic device that stores and processes data.

2. What is a central processing unit (CPU)?

   The CPU is the brain of a computer. It interprets, processes, and executes instructions.

3. What is the difference between random access memory (RAM) and hard drives?

   RAM is temporary storage that is lost when the power is turned off. A hard drive maintains long-term storage and keeps data safe even when power is turned off.

4. What does an actuator do?

   An actuator converts energy into mechanical force.

5. List three examples of coding languages.

   There are many answers to this question, but examples include C, C++, HTML, Java, JavaScript, PHP, Python, R, Ruby, and SQL.

6. Within programming, what is a variable?

   A programming variable is value or set of values that store information.

7. What is the difference between software and hardware?

   Hardware is any physical component of a computer. Software is the programming that tells the hardware what to do and how to do it.

8. What does the term, “microelectronics” mean?

   Student answers may vary, but the formal definition of microelectronics is the design, manufacture, and use of microchips.

9. How are microelectronics used in the field of computer science?

   There are many answers to this question, but examples include the design and manufacturing of computing devices (such as CPU, memory, GPU, sensors, etc.). Microelectronics are foundational in computer science as they are the hardware that allow for programmable automation.

10. What jobs would you be interested in that use microelectronics? Provide one example of how microelectronics is used in that job.

    Students’ answers will vary based on interest. Credit may be given as long as at least one job example is provided with their logic behind how that job uses microelectronics.

1.G Problem Scoping Prompts Key

Section 1:
Directions: Please answer the following questions.

1. What do engineers do?
   Engineers solve problems by designing and building things like machines, systems, and structures to fulfill a specific purpose or address a particular need.

2. How do engineers solve problems?
   Engineers solve problems by identifying the issue, brainstorming possible solutions, testing their ideas, and then refining the designs until they work well.

Section 2:
Directions: Please answer the question after hearing about the engineering design challenge.

3. What questions do you want to ask the client?

   This will vary.

Section 3:
Directions: Please answer these questions after you have been able to ask questions about the challenge. First, complete each prompt on your own. Then write your revised answer (if different) to the prompt, based on the discussion with your team. You may use a different color writing utensil or font color to distinguish your answer and how it changed after talking with teammates.

4. The client is:
   Make Sense Inc., a company that specializes in designing smart homes and is looking to integrate new smart home sensors with newly developed microelectronics in Indiana.

5. The client's problem is:
   The client needs modular and adaptable code developed for current microprocessors that can serve as a foundation for the operation of future smart home sensors in conjunction with new chips, ensuring efficiency, ease of management, and expanded functionality within smart homes.

6. The problem is important to solve because:
   Indiana is going to have a big microelectronics production industry, and so using microelectronics in homes will be popular and valuable

7. The end-users are:
   Homeowners in Indiana who are looking to purchase smart homes with the latest technology that offers increased efficiency and new features.

8. An effective solution for the client will meet the following criteria:
   The solution must be adaptable to work with future chips and sensors that are not yet designed.
   It should allow for easy integration with the main smart home system.
   The code should be modular, allowing for updates and changes without disrupting the overall system.
   It needs to be efficient in terms of processing speed and power consumption.

9. The constraints (or the limits) of the solution are:
   The current unavailability of the new chips and sensors for direct testing and development.
   Potential limitations in computational power and energy usage.
   The need to adhere to the specifications and communication protocols of the future chips.

10. Think about the problem of designing a smart home. In terms of designing a program to help solve this problem, what are at least 2 things you need to learn in order to make an evidence-based recommendation? Make sure to consider all important aspects of the problem. Be specific.
    I would need to know:
    The programming best practices for creating modular and adaptable code, particularly in environments where hardware specifications might change or be updated frequently
    And
    About all different types of sensors

LESSON TWO:

Lesson Objectives
Students will be able to:
- Identify electronic devices as a computer or not a computer
- Describe the components of a computer and their basic function

Time Required
One 50-minute lesson

Standards Addressed
7183.D2.15 Describe the components of a computer architecture

ICS-4.1 Demonstrate understanding of the hardware and operating systems of computers.

Key Terms
Computer, microcontroller (uC), system-on-chip (SoC) Input, Output , Operating System (OS), Random Access Memory (RAM), Hard Drive, Central Processing Unit (CPU)

Lesson Materials
Per classroom
- EDP Poster
- 3-5 everyday items slightly disassembled to show the microcontroller or system-on-chip inside

Per Student
- EDP slider and paperclip
- Laptop/Chromebook/ Tablet
- Engineering notebook

Lesson Summary
Students will explore the question "Do we need computers in our smart homes?" To answer this question, students will explore what characteristics determine a computer, and be able to classify all types of electronics as either computers or not computers. Students will also explore microcontrollers and system-on-chip and their relationship to computers. All of this will be related to the problem of designing smart homes.

Teacher Background
This lesson explores the differences between computers, microcontrollers, and systems on a chip. If you need additional background in these areas, there are many videos on the internet that will give some background. Here are a few that might get you started, but there are many more:

Compare microprocessor, microcontroller, and system-on-chip:
- https://www.youtube.com/watch?v=9Rrt0n1oY8E&ab_channel=NilabhNayanBorthakur

What is a computer?:
- https://www.youtube.com/watch?v=Cu3R5it4cQs&ab_channel=GCFLearnFree
- https://www.youtube.com/watch?v=-HCJDWfCI-M&ab_channel=AaronWissner
- https://www.youtube.com/watch?v=HB4I2CgkcCo&ab_channel=GCFLearnFree

What is a microcontroller?:
- https://www.youtube.com/watch?v=jKT4H0bstH8&ab_channel=MicrochipTechnology
- Much more detailed: https://www.youtube.com/watch?v=CmvUY4S0UbI&ab_channel=SolidStateWorkshop

What is a system-on-chip?:
- https://www.youtube.com/watch?v=dokgLSAhqHI&ab_channel=Arm%C2%AE

Before the Activity

The teacher should prepare by reviewing the original client letter, setting up the class EDP chart, and ensuring the PowerPoint activity and 2.1 Components of a Computer System worksheet are ready for use. Prepare the materials that are partially torn apart to explore microelectronics.

Classroom Instruction

Introduction

1. **Remind students of the engineering problem. Ask:** *What is our engineering design problem?* Students may need to revisit the original client letter to remind themselves of the engineering design challenge. **Ask:** *What were some of the things we decided we needed to learn about in order to develop a solution to this problem?* Remind students to refer to their notes from the previous lesson. Teachers may encourage students to add to the class anchor chart if you made one.
2. **Begin to build a common definition of "computer." Say:** *Our goal today is to begin to answer the question, "Do we need computers in our smart homes?" In order to begin to answer this, we really need to have a deep understanding of computers.* **Ask:** *What is a computer? How do you define it?* Write some of the student responses up on the board. **Then ask:** *What are some items that are close to a computer, but aren't actually a computer?*
3. **Identify where they are in the engineering design process (Learn). Say:** *So far, we have defined the problem with help from our client.* Point out the "Problem" block of the Engineering Design Process (EDP) poster and have students look at their EDP sliders. **Say:** *Before we can start designing solutions, we need more information.* **Ask:** *What step of the engineering design process are we in?* The students should identify that they are in the "Learn" stage.

Activity

4. **Identify computers vs not computers.** As a warm up activity, pull up this Google Sheets activity "What is a Computer?" https://rebrand.ly/computeractivity. Make a copy for your use: File > Make a Copy > Entire Presentation. Then use your editable copy to move the images on the slide to the "computer" or "not a computer" side of the slide. Discuss each image and how the students know that it is or is not a computer. Use their

Duplication Masters
2.1 Components of a Computer System

Assessment
Pre-Activity Assessment
Use the student responses to the introduction questions about the definition of a computer to gauge their background knowledge.

Activity Embedded Assessment
Use the student individual answers on 2.1 Components of a Computer System worksheet to understand what students already know about computer systems.

answers to understand any misconceptions the students may have about computers.

5. **Exploring the components of a computer system.** Distribute the 2.1 Components of a Computer System worksheet with the diagram of a basic computer system. The diagram should have labels missing. Students should fill in the blanks with words from the word bank. This can be used as an individual formative assessment to understand the current level of students' background knowledge, or the students can work in teams to complete if an exploration is more appropriate. Review 2.1 Components of a Computer System worksheet as a class and clarify any misunderstandings. The following are simple definitions of the components that may be used to guide the discussion:
    - *Input:* Explain that computers need a way to receive data. Show examples like keyboards, mice, touchscreens, etc.
    - *Processing:* Introduce the Central Processing Unit (CPU) and its role. Briefly explain that the CPU is the brain of the computer – so it processes inputs, calculations, storage, and outputs. Also discuss how it processes data based on a set of instructions (programs).
    - *Storage:* Discuss the need for memory and storage. Differentiate between RAM (temporary, volatile storage) and hard drives or SSDs (long-term, non-volatile storage).
        i. RAM (Random Access Memory): RAM is a temporary storage that quickly reads and writes data the CPU uses during operation, but loses all information when the power is off.
        ii. Hard Drives (HDDs) and Solid State Drives (SSDs): HDDs and SSDs provide long-term storage, keeping data safe even when the computer is turned off, with SSDs offering faster access to this data.
    - Output: Describe how computers present data after processing, e.g., through monitors, speakers, printers.
    - Operating System: Briefly introduce the role of OS in managing all these components and tasks. Discuss how the OS and the CPU work together.
        i. The Operating System (OS) manages the computer's hardware and software resources, coordinating tasks and ensuring efficient operation, while the CPU executes instructions provided by the OS to perform user-directed tasks.

Have students take notes on each item in their notebooks near their computer system worksheet or have them sketch components.

6. **Discuss the nuances of being a computer or not.** Have the students discuss in small groups the question: "Can a device lack any of these qualities and still be considered a computer?" Bring the class back together and discuss some of the points that came up in the groups.
7. **Discussion of microcontrollers, system-on-chip (SoC), and computers.** Use the Powerpoint slides (link here) or develop your own discussion.
   - **Microcontollers in everyday life. Ask:** *What are things you encounter in your life that contain microcontrollers or SoCs?* Take answers and use the discussion to further build on the differences between microcontrollers, SoCs, and computers.
   - **Activity:** Show the students several items that have been partially deconstructed to be able to see the microcontroller or SoC inside. Assign each team to one of the devices. Have the students:
     i. find the microcontroller or SoC in the device and record the device ID number
     ii. use a search engine to look up the ID number to find out what the device does
     iii. record the device name, ID number, and function in their notebooks

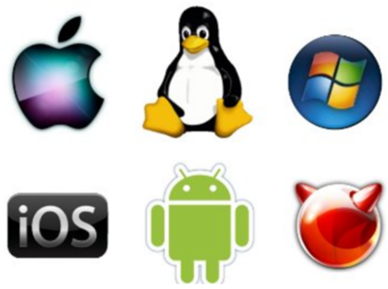
     Share out the results with the class and lead a discussion on the findings. Use the discussion to tie into the engineering design challenge.
8. Erase when ppt is complete and don't need the comment here. :

Closure

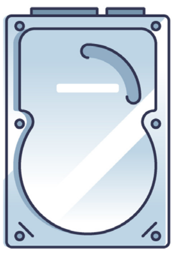
9. **Summary and reflection on the lesson.** Summarize the primary components and the qualities that define computers, microcontrollers, and SoCs. Have students reflect on the following prompt in their notebooks: "How has your understanding of computers, microcontrollers, or SoC changed?" Have them identify one thing they didn't know was a computer, microcontroller, or system on chip in their notebooks.
10. **Connect the activity to the engineering design challenge. Ask:** *What did we learn today that will help us with our solution for the client? What else do we still need to learn? What is your answer to the question "Do we need computers for our smart homes?* Use this discussion to clear up misconceptions and set up the next lesson. Let them know that they will continue this during the next class.

Name________________________________ Date______________ Period __________

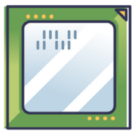
2.1 Components of Computer System



1. ________________

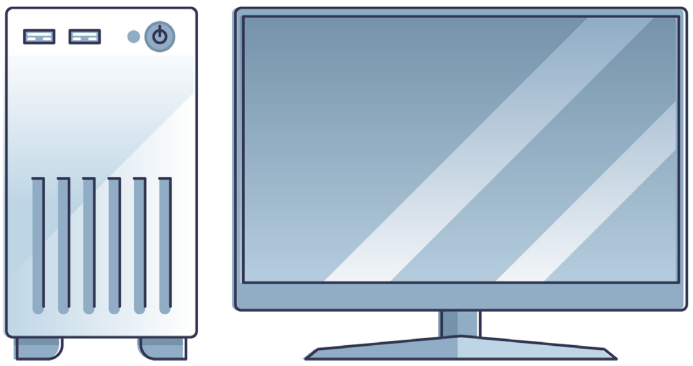


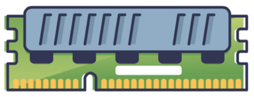
2. ________________



3.

____

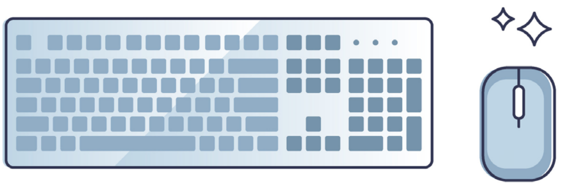




4.

____

5. ________________



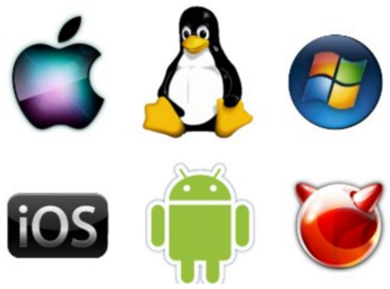
6. ________________

Word Bank:

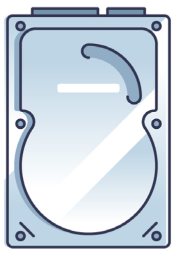
- Input
- Output
- Operating System (OS)
- Random Access Memory (RAM)
- Hard Drive
- Central Processing Unit (CPU)

Name________________________________ Date______________ Period __________

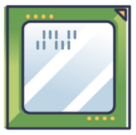
Answer Key



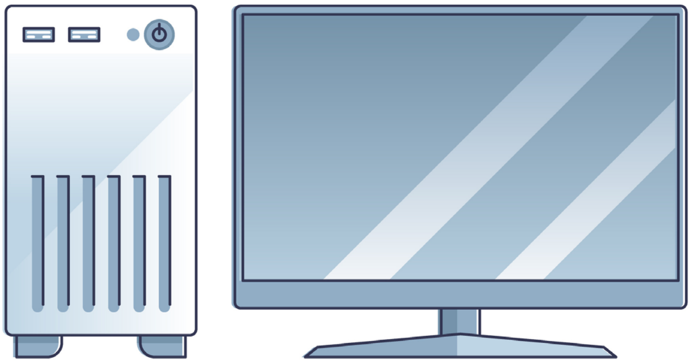
1. Operating System (OS)

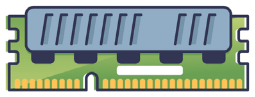


2. Hard Drive



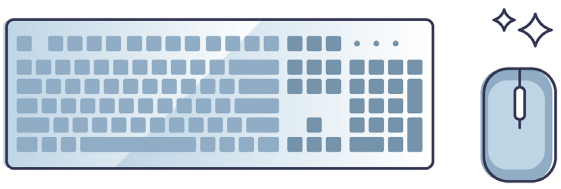
3. CPU





4. RAM

5. Output



6. Input

LESSON THREE:

Lesson Objectives
Students will be able to:
- Create programs that are clear and organized
- Reduce code complexity (optimize) by writing functions
- Understand how to work with the micro:bit
- Understand, at a high level, the paradigm of physical computing

Time Required
Three 50-minute lessons

Standards Addressed
7183.D.2.1 Develop a simple program and/or script using a compiled, object-oriented scripting language like Python

7183.D2.3 Apply truth tables, Boolean logic, control structures, relational and logical operators to program algorithms

7352.D1.3 Analyze and explain behavior of simple programs utilizing variables, expressions, assignments, I/O, control structures, functions, parameter passing, preconditions, postconditions, and invariants.

Key Terms
Function, Variables, Control Structure, Python

Lesson Summary
This lesson introduces the concept of physical computing. It distinguishes between writing code for software only applications and writing code for applications that interact with physical devices. This lesson discusses functions and has students practice creating functions based on redundant code that they identify. This lesson also introduces coding the micro:bit.

Teacher Background
The purpose of this lesson is two-fold: 1. to introduce the concept of functions, 2. To introduce the concept of physical computing. Students will gain hands-on experience creating functions from redundant code. Students will also gain experience coding the micro:bit device while becoming familiar with the Python for micro:bit online editor: https://python.microbit.org/v/3/.

This lesson assumes that students have done some programming in a text-based (Python) or block-based (Scratch) language.

Vocabulary:
Microcontroller, physical computing, micro:bit

Before the Activity
Task 1: Open the Python for micro:bit editor. Open the Ghost_Story.py code in the editor. Flash the Ghost_Story.py code to a micro:bit. Test that the micro:bit performs the code. Unplug and set the micro:bit aside, but have the battery pack ready to plug in during Activity 1.

Task 2: Assemble and hang up the code poster for the non-functionalized code. Have the functionalized code poster on standby.

Classroom Instruction
Introduction
1. **Identify where they are in the Engineering Design Process. (Learn) Ask:** *What phases of the engineering design process have we used so far?* **Say:** *In the last lesson we learned about microcontrollers.* **Ask:** *What more do you think we'll need to learn in order to meet the design challenge?*

2. **Identify what students need to learn about: Say:** *The client has asked us to create modular code for their microcontroller. So we will need to learn about how to create modular code. We will also need to learn how to code microcontrollers.*

Activity

Activity 1: Laminated Poster code

3. **Ask:** *What languages have you written code in? What did you write the code to do?*
   a. Students may say Scratch or Python, etc. The object is to get students to identify that different "types" of code are used for different purposes.
2. **Direct students' attention to the large post containing the code that has not been broken into functions (Ghost_Story-main.py). Say:** *This is Python code for the micro:bit microcontroller.* **Ask:** *What do you notice about this code?* Have small groups discuss what they notice about the code and then report out.
3. **Explain physical computing:** Even though this is Python, a language you may have seen before, the code is a little different. This is actually MicroPython. It's a subset of Python designed specifically for microcontrollers. Since microcontrollers don't have as much memory as our laptop or desktop computers, languages used to code them must be small enough to fit in the memory that's available. **Say:** *The type of coding that we'll be doing falls into the category of physical computing. In physical computing, the code you write doesn't just run on a computer screen, it interacts with the "real-world" through devices like sensors or motors.*
4. **Equate to smart home:** The code we'll write for our smart home design will make things happen in the "real-world" too, so we'll need to learn something about how to write code for physical computing. That means we will have to think a little differently about how we code, even though the language is the same.
5. **Flow Chart the code. Say:** *Let's try to understand what this code does. This code tells a story. Let's figure out what the story is.* On the board, Co-construct the story with students by creating a flow chart of the code with the whole class. **Say:** *Look at this block of code (the first block). What can we decipher about what it's doing?* Draw a flow graph based on what students say for each block of code by writing on the board.
   a. What does the code tell us happens first?

Lesson Materials
Per classroom
- Projector/Way to Display Computer Screen

Per Group (3 per group)
- Computer with Internet access

Posters
1. GhostStory-main
2. GhostStory-functions

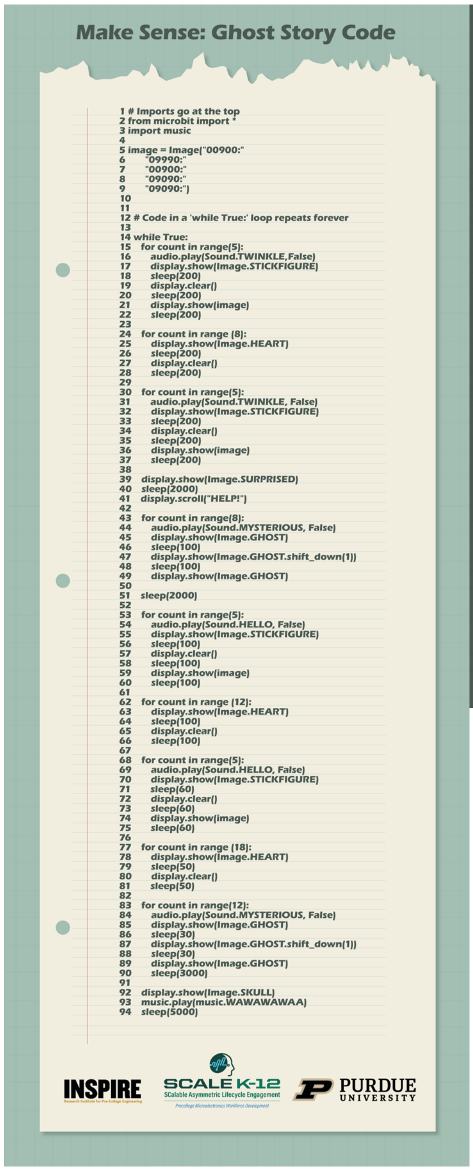
Duplication Masters
3A. Activity 2: function templates
3B. Activity 2: Code Skeleton
3C: Activity 3 Walk_ Heartbeat Code
3D. Available Images for the Microbit

Python Codes
GhostStory-comments.py
GhostStory-functions.py
GhostStory-main.py
Tortoise_Hare-functions.py
Tortoise_Hare-main.py
Tortoise_Hare-template.py
Walk_Heartbeat.py

Assessment
Activity Embedded Assessment
Students correctly divide code into functions.
Students flash code to the micro:bit

b. What happens next?

8. **Run the simulator in the editor. Say:** *Okay let's see if we were right.*
   i. Open the micro:bit Python editor.
   j. **Say:** *Here is the editor that was used to write the code. It's a special editor for using Python with the micro:bit.*
   k. Upload the Ghost_Story code. **Say:** *I've uploaded the code here. Point out the code.* **Say:** *This editor has a simulator that will preview for us what the code will do once flashed to the micro:bit. Let's see if we were right about what the code does.* Press play on the simulator.
9. **Briefly discuss the differences between what the co-constructed flow chart said and what the simulator showed. Say:** *This code tells a ghost story.*
10. **Flash the micro:bit. Say:** *Now let's flash the micro:bit with this code and see it run on the physical device.* Show students how to plug in the micro:bit using the USB cable. Press the "Send to micro:bit " button. Show students the code running on the micro:bit.
    a. Note: Sometimes this process requires the microbits to update firmware first, go to https://microbit.org/get-started/user-guide/firmware/ for instructions if the code does not download/flash to the mico:bit right away.
2. **Functions:** Post the functionalized poster (GhostStory-functions.py) next to the un-functionalized poster of the Ghost Story. **Say:** *"Here's another piece of code for the micro:bit microcontroller." Do you think this code does more work or less work than the other code (from the long poster.)* After students answer **say**, *"Let's see what this code does."* Open the functionalized Ghost_Story code in the editor. Simulate it. **Say:** *It does the same exact thing! But it's much shorter. Why is that? (Instead of repeating the same blocks of code, this code is using functions.) This makes the code more modular like our client is asking for. Let's learn more about making code more modular by using functions.*
3. **Discuss the structure of the function declaration:**
   a. How it has a "def" keyword, a ":" and indented code body
   b. How it has parameters
   c. How the function is called in the main body of the code
   a. How the parameters are passed to the function and how they are used to make the person walk faster or slower and how many steps are taken.



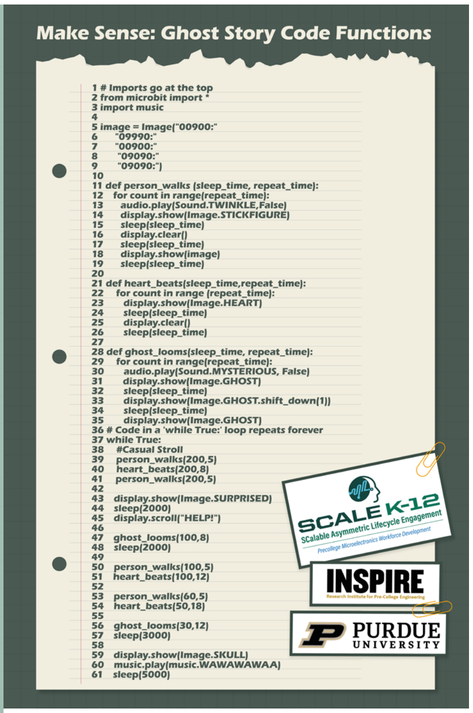
Make Sense: Ghost Story Code

```
# Imports go at the top
from microbit import *
import music

image = Image("00900:"
      "09990:"
      "00900:"
      "09090:"
      "09090:")


# Code in a 'while True:' loop repeats forever

while True:
    for count in range(5):
        audio.play(Sound.TWINKLE,False)
        display.show(Image.STICKFIGURE)
        sleep(200)
        display.clear()
        sleep(200)
        display.show(image)
        sleep(200)

    for count in range (8):
        display.show(Image.HEART)
        sleep(200)
        display.clear()
        sleep(200)

    for count in range(5):
        audio.play(Sound.TWINKLE, False)
        display.show(Image.STICKFIGURE)
        sleep(200)
        display.clear()
        sleep(200)
        display.show(image)
        sleep(200)

    display.show(Image.SURPRISED)
    sleep(2000)
    display.scroll("HELP!")

    for count in range(8):
        audio.play(Sound.MYSTERIOUS, False)
        display.show(Image.GHOST)
        sleep(100)
        display.show(Image.GHOST.shift_down(1))
        sleep(100)
        display.show(Image.GHOST)

    sleep(2000)

    for count in range(5):
        audio.play(Sound.HELLO, False)
        display.show(Image.STICKFIGURE)
        sleep(100)
        display.clear()
        sleep(100)
        display.show(image)
        sleep(100)

    for count in range (12):
        display.show(Image.HEART)
        sleep(100)
        display.clear()
        sleep(100)

    for count in range(5):
        audio.play(Sound.HELLO, False)
        display.show(Image.STICKFIGURE)
        sleep(60)
        display.clear()
        sleep(60)
        display.show(image)
        sleep(60)

    for count in range (18):
        display.show(Image.HEART)
        sleep(50)
        display.clear()
        sleep(50)

    for count in range(12):
        audio.play(Sound.MYSTERIOUS, False)
        display.show(Image.GHOST)
        sleep(30)
        display.show(Image.GHOST.shift_down(1))
        sleep(30)
        display.show(Image.GHOST)
        sleep(3000)

    display.show(Image.SKULL)
    music.play(music.WAWAWAWAA)
    sleep(5000)
```

INSPIRE Research Institute for Pre-College Engineering

SCALE K-12 SCalable Asymmetric Lifecycle Engagement Precollege Microelectronics Workforce Development

PURDUE UNIVERSITY



Make Sense: Ghost Story Code Functions

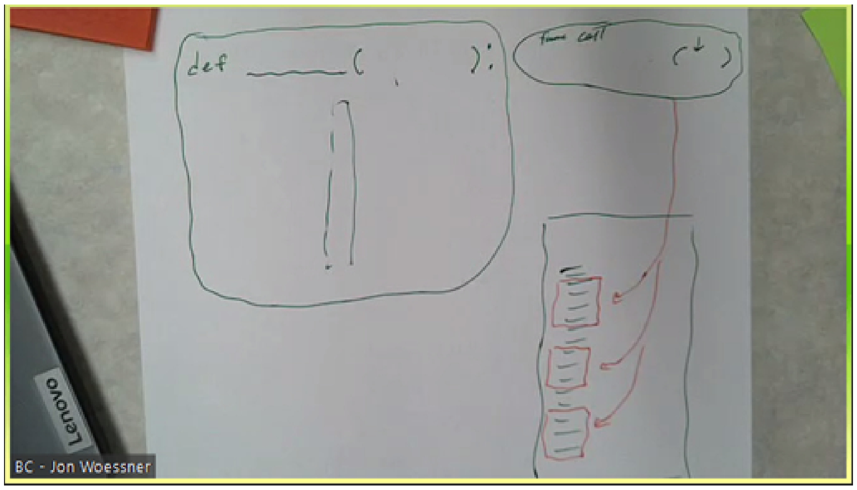
```
# Imports go at the top
from microbit import *
import music

image = Image("00900:"
      "09990:"
      "00900:"
      "09090:"
      "09090:")

def person_walks (sleep_time, repeat_time):
    for count in range(repeat_time):
        audio.play(Sound.TWINKLE,False)
        display.show(Image.STICKFIGURE)
        sleep(sleep_time)
        display.clear()
        sleep(sleep_time)
        display.show(image)
        sleep(sleep_time)

def heart_beats(sleep_time,repeat_time):
    for count in range (repeat_time):
        display.show(Image.HEART)
        sleep(sleep_time)
        display.clear()
        sleep(sleep_time)

def ghost_looms(sleep_time, repeat_time):
    for count in range(repeat_time):
        audio.play(Sound.MYSTERIOUS, False)
        display.show(Image.GHOST)
        sleep(sleep_time)
        display.show(Image.GHOST.shift_down(1))
        sleep(sleep_time)
        display.show(Image.GHOST)
# Code in a 'while True:' loop repeats forever
while True:
    #Casual Stroll
    person_walks(200,5)
    heart_beats(200,8)
    person_walks(200,5)

    display.show(Image.SURPRISED)
    sleep(2000)
    display.scroll("HELP!")

    ghost_looms(100,8)
    sleep(2000)

    person_walks(100,5)
    heart_beats(100,12)

    person_walks(60,5)
    heart_beats(50,18)

    ghost_looms(30,12)
    sleep(3000)

    display.show(Image.SKULL)
    music.play(music.WAWAWAWAA)
    sleep(5000)
```

SCALE K-12 SCalable Asymmetric Lifecycle Engagement Precollege Microelectronics Workforce Development

INSPIRE Research Institute for Pre-College Engineering

PURDUE UNIVERSITY

Activity 2 – On paper
For this activity, students will identify repeated code in a program, turn that code into functions and "re-write" the code using functions. This is an offline activity that utilizes paper and pencil.



Story for Activity 2:
Build basic circuits from Schematic Diagrams
Say: *Look at the code for Activity 2. This code tells the story of the Tortoise and the Hare. With your team, you will identify portions of repeated code and turn that code into one or more functions. Then you will re-arrange the code to tell the same story using functions.*
Steps:

1. Identify repeated sections of code and draw a box around those sections. Encourage students to use colored pencils, etc, and use one color for each new repeated code chunk/ function. For example, use blue for all the Tortoise code and red for all the Hare code.
2. Create a function definition that will contain that piece of code.
   a. Use the function definition template and write the statements of code inside that you identified should be converted into a function. (Students hand write the code in inside of the function def template).
2. Repeat step 2 for any new functions.
3. Call the functions. Look at the skeleton code. Place calls to your new functions inside the skeleton code to replace the code that you made into functions.

Activity 3: Walk_Hearbeat code and getting to know the micro:bit.

Students type in the Walk_Hearbeat code to the python.microbite online editor.
Students use the simulator to simulate the actions of the micro:bit.
Students download code to the micro:bit to view the physical device.
Challenge: Students change the code to make the heart beat faster, and the person walk faster, going from strolling to running.

Activity 4 – Tell a story with functions
Write your own story using functions. Make sure you include functions.
Map out the story first, using a flow chart then execute the map.
The python micro:bit editor has built-in documentation that can help students discover images to code.
The online reference documentation located at: https://microbit-micropython.readthedocs.io/en/latest/image.html also refers to images and image manipulation.
The Micropython codes for this lesson are located here: **https://purdue.app.box.com/folder/234476685537**

Closure

1. **Discuss usefulness and need for writing clear functions.** This is for motivation for students to understand why they should use functions. A usual hangup for students at this stage is: Why should I do this 'extra' work to write a function, when I could just put the code in the main section? This should help show that functions make code more modular, easier to read, and easier to modify and fix.
2. **Discuss with students the differences in physical computing and software computing. T**heir code will "manifest" itself in the 'real world', on a physical device. The language, Python, is the same but there are different considerations that the programmer must take into account. Ask students to name a few of those considerations (see Note if students ask questions like what some other considerations could be. This distinction between coding for software applications and coding for physical devices will help prepare students for lesson 4, where they will go to a deeper level in communicating with the sensors and actuators.
   a. Note: Other considerations could be: processor speed, low power or energy conservation requirements (say to conserve battery power), the need to interact with lower-lever hardware functions like turning an output on/off, or communicating with sensors that need specific signal timing.

Activity 2 Code: Tortoise and Hare Code

```
# Imports go at the top
from microbit import *
import music

# Code in a ‘while True:’ loop repeats forever
while True:
#Hare
display.show(Image.RABBIT)
sleep(1000)
audio.play(Sound.SPRING, False)
display.clear()
sleep(100)
display.show(Image.RABBIT)
sleep(1000)
display.scroll(“FAST”)
for count in range(15):
audio.play(Sound.SPRING, False)
display.show(Image.RABBIT)
sleep(100)
display.clear()
sleep(100)
display.show(Image.RABBIT.shift_left(1))
sleep(100)
display.show(Image.RABBIT.shift_left(2))
sleep(100)

#Tortoise
display.show(Image.TORTOISE)
sleep(1000)
audio.play(Sound.HELLO,False)
display.clear()
sleep(100)
display.show(Image.TORTOISE)
sleep(1000)
display.scroll(“SLOW”)
for count in range(5):
audio.play(Sound.HELLO,False)
display.show(Image.TORTOISE)
sleep(200)
display.clear()
sleep(200)
display.show(Image.TORTOISE.shift_left(1))
sleep(200)
display.show(Image.TORTOISE.shift_left(2))
```

```
sleep(200)

#Hare
display.show(Image.RABBIT)
sleep(1000)
audio.play(Sound.SPRING, False)
display.clear()
sleep(100)
display.show(Image.RABBIT)
sleep(1000)
display.scroll(“FAST”)
for count in range(15):
audio.play(Sound.SPRING, False)
display.show(Image.RABBIT)
sleep(100)
display.clear()
sleep(100)
display.show(Image.RABBIT.shift_left(1))
sleep(100)
display.show(Image.RABBIT.shift_left(2))
sleep(100)
display.scroll(“SLEEP”)
display.show(Image.ASLEEP)
music.play(music.POWER_DOWN)
sleep(2000)

#Tortoise
display.show(Image.TORTOISE)
sleep(1000)
audio.play(Sound.HELLO,False)
display.clear()
sleep(100)
display.show(Image.TORTOISE)
sleep(1000)
display.scroll(“SLOW”)
for count in range(5):
audio.play(Sound.HELLO,False)
display.show(Image.TORTOISE)
sleep(200)
display.clear()
sleep(200)
display.show(Image.TORTOISE.shift_left(1))
sleep(200)
display.show(Image.TORTOISE.shift_left(2))
```

Activity 2 Code: Tortoise and Hare Code

```
sleep(200)

display.scroll("WIN!")
display.show(Image.TARGET)
sleep(2000)
display.show(Image.HAPPY)
music.play(music.ENTERTAINER)
sleep(2000)

display.show(Image.RABBIT)
music.play(music.POWER_UP)
sleep(100)
display.scroll("WAKE")

#Hare
display.show(Image.RABBIT)
sleep(1000)
audio.play(Sound.SPRING, False)
display.clear()
sleep(100)
display.show(Image.RABBIT)
sleep(1000)
display.scroll("FAST")
for count in range(15):
audio.play(Sound.SPRING, False)
display.show(Image.RABBIT)
sleep(100)
display.clear()
sleep(100)
display.show(Image.RABBIT.shift_left(1))
sleep(100)
display.show(Image.RABBIT.shift_left(2))
sleep(100)

display.scroll("LOST")
display.show(Image.SAD)
music.play(music.CHASE)
sleep(2000)

display.scroll("END")
```

Activity 2: Tortoise and Hare Define Functions – See duplication Masters

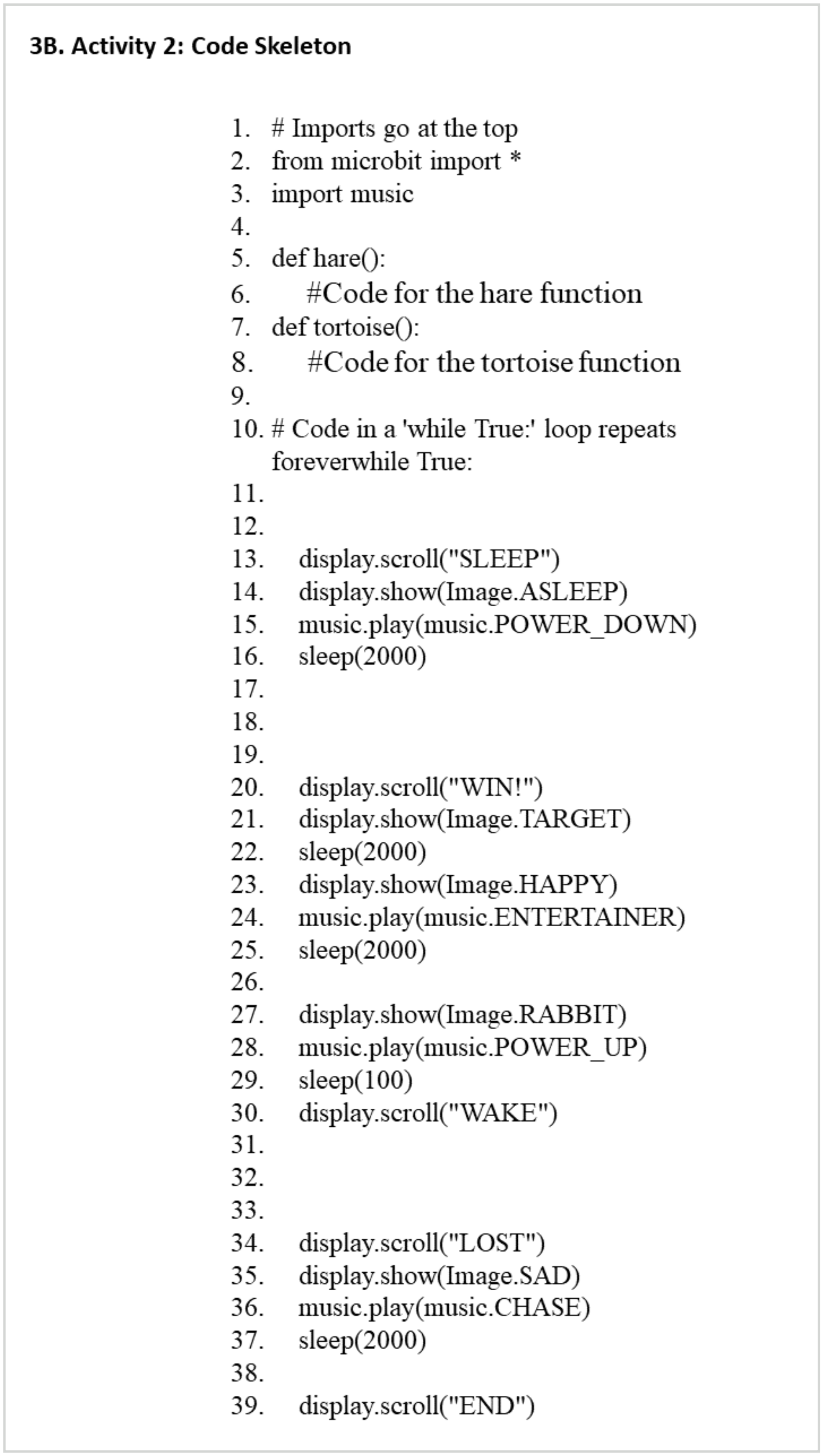
1. Define the function and place the code inside the function

```
def tortoise():
    #Place tortoise code below
```

```
def hare():
    #Place hare code below
```

2. Make function calls inside the Skeleton Code
   a. tortoise()
   b. hare()

Activity 2: Tortoise & Hare Functions Code Skeleton



3B. Activity 2: Code Skeleton

```
# Imports go at the top
from microbit import *
import music

def hare():
    #Code for the hare function
def tortoise():
    #Code for the tortoise function

# Code in a 'while True:' loop repeats foreverwhile True:


    display.scroll("SLEEP")
    display.show(Image.ASLEEP)
    music.play(music.POWER_DOWN)
    sleep(2000)



    display.scroll("WIN!")
    display.show(Image.TARGET)
    sleep(2000)
    display.show(Image.HAPPY)
    music.play(music.ENTERTAINER)
    sleep(2000)

    display.show(Image.RABBIT)
    music.play(music.POWER_UP)
    sleep(100)
    display.scroll("WAKE")



    display.scroll("LOST")
    display.show(Image.SAD)
    music.play(music.CHASE)
    sleep(2000)

    display.scroll("END")
```

Answer Key: Tortoise and Hare with Functions

```
# Imports go at the top
from microbit import *
import music

def hare():
display.show(Image.RABBIT)
sleep(1000)
audio.play(Sound.SPRING, False)
display.clear()
sleep(100)
display.show(Image.RABBIT)
sleep(1000)
display.scroll(“FAST”)
for count in range(15):
audio.play(Sound.SPRING, False)
display.show(Image.RABBIT)
sleep(100)
display.clear()
sleep(100)
display.show(Image.RABBIT.shift_left(1))
sleep(100)
display.show(Image.RABBIT.shift_left(2))
sleep(100)

def tortoise():
display.show(Image.TORTOISE)
sleep(1000)
audio.play(Sound.HELLO,False)
display.clear()
sleep(100)
display.show(Image.TORTOISE)
sleep(1000)
display.scroll(“SLOW”)
for count in range(5):
audio.play(Sound.HELLO,False)
display.show(Image.TORTOISE)
sleep(200)
display.clear()
sleep(200)
display.show(Image.TORTOISE.shift_left(1))
sleep(200)
display.show(Image.TORTOISE.shift_left(2))
sleep(200)
```

Answer Key: Tortoise and Hare with Functions

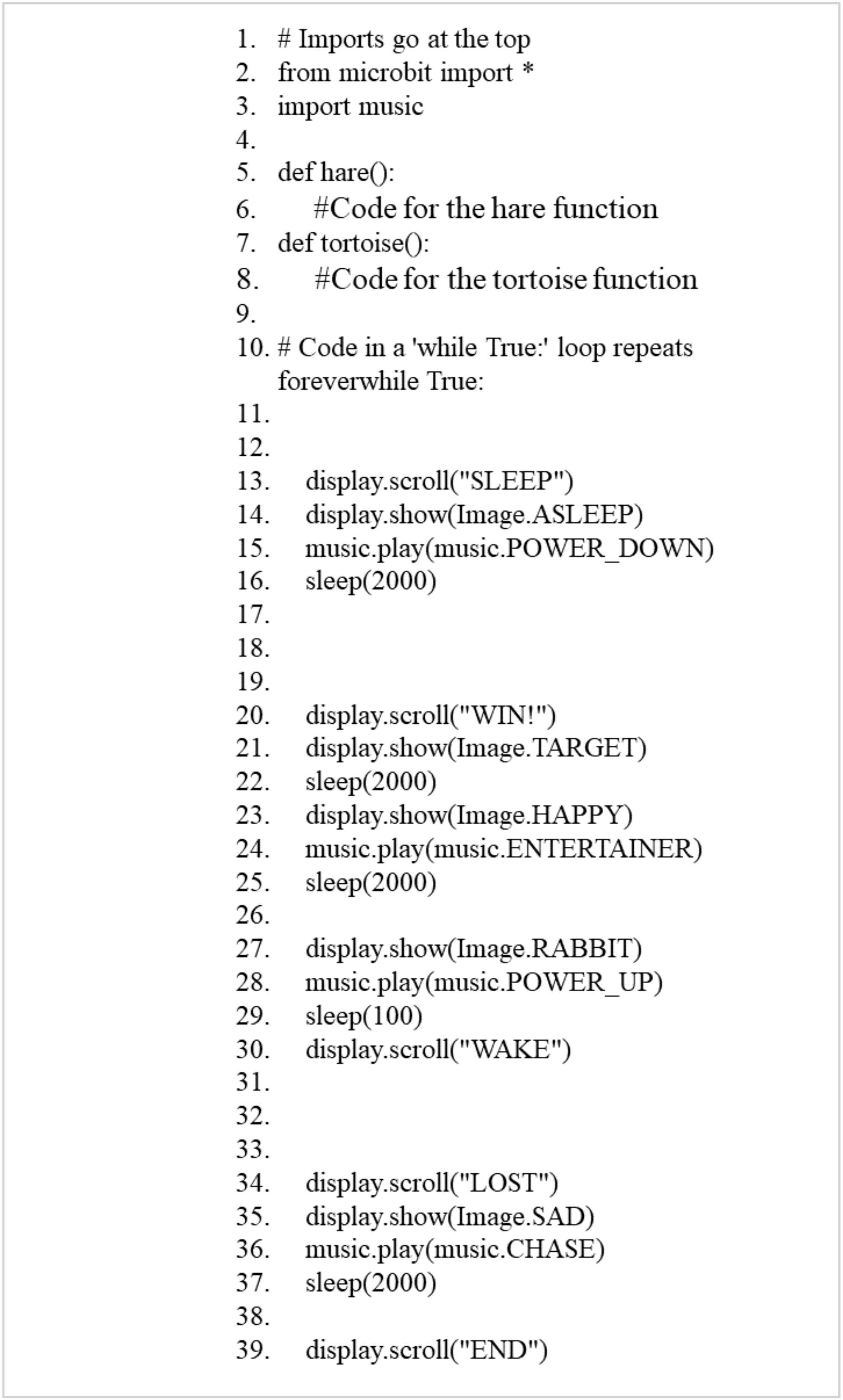
```
# Code in a ‘while True:’ loop repeats forever
while True:
hare()
tortoise()
hare()

display.scroll(“SLEEP”)
display.show(Image.ASLEEP)
music.play(music.POWER_DOWN)
sleep(2000)

tortoise()

display.scroll(“WIN!”)
display.show(Image.TARGET)
sleep(2000)
display.show(Image.HAPPY)
music.play(music.ENTERTAINER)
sleep(2000)

display.show(Image.RABBIT)
music.play(music.POWER_UP)
sleep(100)
display.scroll(“WAKE”)

hare()

display.scroll(“LOST”)
display.show(Image.SAD)
music.play(music.CHASE)
sleep(2000)

display.scroll(“END”)
```

3A. Activity 2: function templates

```
def tortoise():
        #Place tortoise code below
```

```
def hare():
        #Place hare code below
```

3B. Activity 2: Code Skeleton



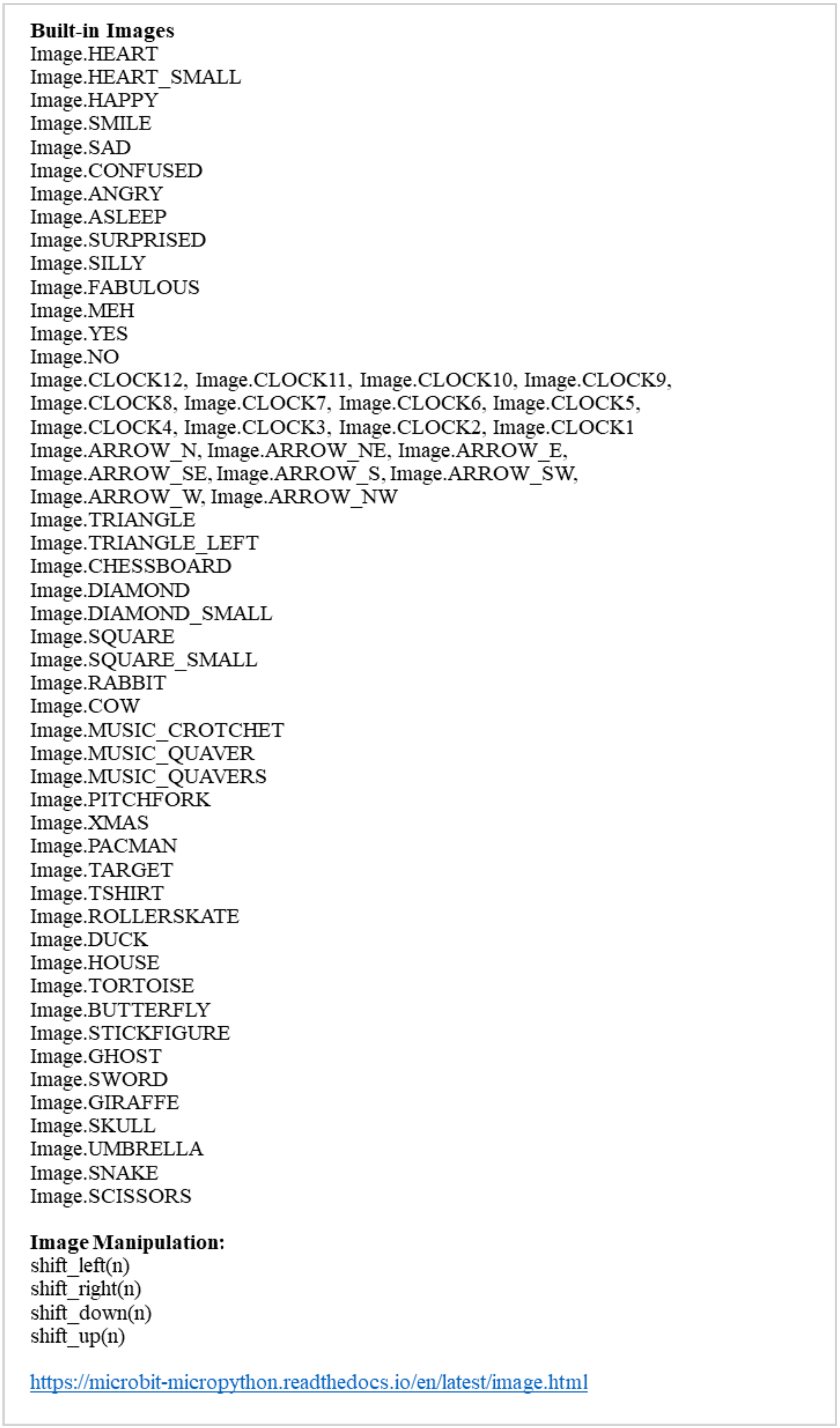
```
# Imports go at the top
from microbit import *
import music

def hare():
    #Code for the hare function
def tortoise():
    #Code for the tortoise function

# Code in a 'while True:' loop repeats foreverwhile True:


    display.scroll("SLEEP")
    display.show(Image.ASLEEP)
    music.play(music.POWER_DOWN)
    sleep(2000)



    display.scroll("WIN!")
    display.show(Image.TARGET)
    sleep(2000)
    display.show(Image.HAPPY)
    music.play(music.ENTERTAINER)
    sleep(2000)

    display.show(Image.RABBIT)
    music.play(music.POWER_UP)
    sleep(100)
    display.scroll("WAKE")



    display.scroll("LOST")
    display.show(Image.SAD)
    music.play(music.CHASE)
    sleep(2000)

    display.scroll("END")
```

3C. Activity 3: Walk_Heartbeat Code

```
# Imports go at the top
from microbit import *

image = Image("00900:"
        "09990:"
        "00900:"
        "09090:"
        "09090:")

# Code in a 'while True:' loop repeats forever

while True:
    #Casual Stroll
    for count in range(5):
        audio.play(Sound.TWINKLE,False)
        display.show(Image.STICKFIGURE)
        sleep(200)
        display.clear()
        sleep(200)
        display.show(image)
        sleep(200)

    #Normal Heartbeat
    for count in range (8):
        display.show(Image.HEART)
        sleep(200)
        display.clear()
        sleep(200)
```

3D. Activity 4: Tell a story using functions

https://microbit-micropython.readthedocs.io/en/latest/image.html



Built-in Images
Image.HEART
Image.HEART_SMALL
Image.HAPPY
Image.SMILE
Image.SAD
Image.CONFUSED
Image.ANGRY
Image.ASLEEP
Image.SURPRISED
Image.SILLY
Image.FABULOUS
Image.MEH
Image.YES
Image.NO
Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9,
Image.CLOCK8, Image.CLOCK7, Image.CLOCK6, Image.CLOCK5,
Image.CLOCK4, Image.CLOCK3, Image.CLOCK2, Image.CLOCK1
Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E,
Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW,
Image.ARROW_W, Image.ARROW_NW
Image.TRIANGLE
Image.TRIANGLE_LEFT
Image.CHESSBOARD
Image.DIAMOND
Image.DIAMOND_SMALL
Image.SQUARE
Image.SQUARE_SMALL
Image.RABBIT
Image.COW
Image.MUSIC_CROTCHET
Image.MUSIC_QUAVER
Image.MUSIC_QUAVERS
Image.PITCHFORK
Image.XMAS
Image.PACMAN
Image.TARGET
Image.TSHIRT
Image.ROLLERSKATE
Image.DUCK
Image.HOUSE
Image.TORTOISE
Image.BUTTERFLY
Image.STICKFIGURE
Image.GHOST
Image.SWORD
Image.GIRAFFE
Image.SKULL
Image.UMBRELLA
Image.SNAKE
Image.SCISSORS

Image Manipulation:
shift_left(n)
shift_right(n)
shift_down(n)
shift_up(n)

https://microbit-micropython.readthedocs.io/en/latest/image.html

LESSON FOUR:

Lesson Objectives
Students will be able to:
- Implement control structures (sequence, logic, and selection) within their program
- Implement Boolean logic
- Understand sensor, threshold, trigger and actuator relationship.
- Understand how to plug the sensor in
- Understand how to write code that uses information from the sensor.
- Write code that reads data from a sensor and print the values out to the terminal.
- Use conditional logic to decide what to do with sensor input.
- (How do the sensors behave – experiment with what the data from the sensors means)

Time Required
Two-three 50-minute lessons

Standards Addressed
ICS-2.2 Demonstrate competencies of programming constructs, including: use of data types and variables, control structures (sequencing, looping, branching), and modularity (such as a function)

Lesson Summary

Students learn about sensors and actuators. Students examine and use code for a sensor and actuator. Students create a program that uses the output from a sensor to trigger the action of an actuator.

All code, slides and full-size duplication masters are located on Box: https://purdue.app.box.com/folder/217818657940?s=pyptq46jq6e1dixvfoh5fretcb2r7r0c Under the MicroPythonCode and Resources folders.

Standards

7183.D2.6 Develop a simple program and/or script using a compiled, object-oriented scripting language like Python.

7183.D2.19 Apply critical thinking and problem-solving methodologies

Background

Teacher Background

Sensors:
See Duplication Masters 4C & 4D

Before the Activity
- Set up sensor for class demonstration using slide 3 – Crash Sensor Demo Circuit
- Set up sensor/actuator for class demonstration using 4.F Sensor/Actuator Demo Circuit
- Have the code for both setup on hand and have the Micropython micro:bit editor ready to upload the code and flash the micro:bit.
- Have the slide deck ready to display the slides and duplication masters as indicated.

Making Some Sense (of the Sensors)

Classroom Instruction

Introduction

1. **Connect to the engineering challenge. Ask:** *What is our engineering challenge from the customer?* Potential answer is: Write modular code for a smart home.
2. **Revisit the EDP.** Have students identify where in the EDP we are: Learn.
3. **Introduce sensors: Say:** *The client has asked us to understand sensors and how they can be used in a smart home. Let's explore what sensors do and how they are used in different contexts.* Show the video: https://www.youtube.com/watch?v=ht-_RmhLD7k
   Say: *Sensors are used any many places when we want to detect some properties in the physical environment and take an action based on those properties.* **Ask:** *Where have you seen sensors used in your everyday environment?*
   Ask: *For the sensors that we've mentioned, what property in the environment are they actually reading?* (For example, if someone mentions a thermostat, the sensor is reading temperature. If someone mentions an automatic door, this sensor is reading the distance of an object from the door or the presence of weight on a touchpad embedded in the ground in front of the door.)
4. **Example scenario: Say** *Let's use an example to better understand how sensors work. In the video we saw many different types of sensors. For our example, let's take a smoke sensor.* **Ask:** *Who here has a smoke detector at home? Has the smoke detector ever sounded an alarm at your house? What made the smoke alarm go on? What if there's smoke coming from a cigarette, does the alarm sound? What if there's a small amount of smoke from someone cooking food on the stove does the alarm sound?*
   The point of these questions is to get students to think about a threshold level. Not ALL smoke will trigger the alarm, the smoke needs to reach a certain level to trigger the alarm.
5. **Trigger/Threshold: Say:** *Okay, so we've determined that not just any amount of smoke will trigger the alarm. There has to be a lot of smoke. So I'll introduce a new word – threshold. The threshold describes how much the monitored element needs to deviate from the norm in order to trigger the sensor to take action.*

Key Terms
Sensor, Micro:bit, Pins, Raw data, Library, Analog signals, Serial protocol, Binary

Lesson Materials
Per classroom
- Projector

Per Group (3 per group)
- 1 micro:bit
- 1 set of external sensors

Per Student
- Notebook
- computer

Slides
1. Human Sensor Actuator Trigger
2. Human & Electronic Sensor Actuator Trigger
3. Crash Sensor Demo Circuit

Also, several of the duplication masters will be used as slides.

Duplication Masters
4A. Human Sensor Actuator Trigger
4B Human Sensor Actuator Trigger - If – Then Logic
4C. Sensors and Actuators built into the micro:bit
4D. External Sensors and Actuators
4E. Micro:bit pins
4F. Sensor/Actuator Demo Circuit
4G. Sensor/Actuator Demo Code
4H. Code Snippets
4H(b). Code Snippets

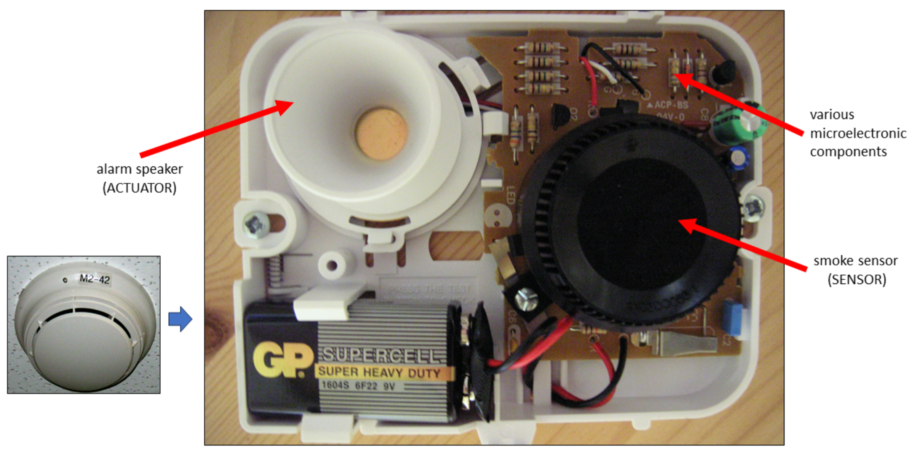
Python Codes
Builtin_Light_Sensor.py
Buttons_Sensor.py
Capacitive_Touch.py
Crash_Sensor.py
Demo_Sensor_Actuator_Code.py
Neopixel_Actuator.py
Servo_Actuator.py
Sound_Sensor.py
Temperature_Sensor.py

Assessment
Activity Embedded Assessment
Students can discuss the functionality of sensors and actuators.

Post-Activity Assessment:
Students have written code that uses sensor data to trigger an actuator.

6. **Actuator: Say:** *Ok when the smoke sensor inside the smoke detector senses that there's too much smoke, what happens?* Students answer, "An alarm goes off." **Say:** *Correct: But the sensor is not the alarm. Even though the smoke detector in your house looks like its all one piece, inside of it there is both a sensor and an alarm. The alarm only sounds when the sensor detects a level of smoke above a set threshold. The alarm is an actuator. An actuator is a device that takes input from the sensor and performs a physical action based on that input. In the case of a smoke detector, the alarm actuator makes a loud noise and continues making a loud noise until the smoke sensor ceases to detect smoke or is reset.* Show **Slide 1 Say:** *Here's a picture of the inside of a smoke detector. It shows that the detector has both a sensor and an actuator.*

Inside a smoke Detector

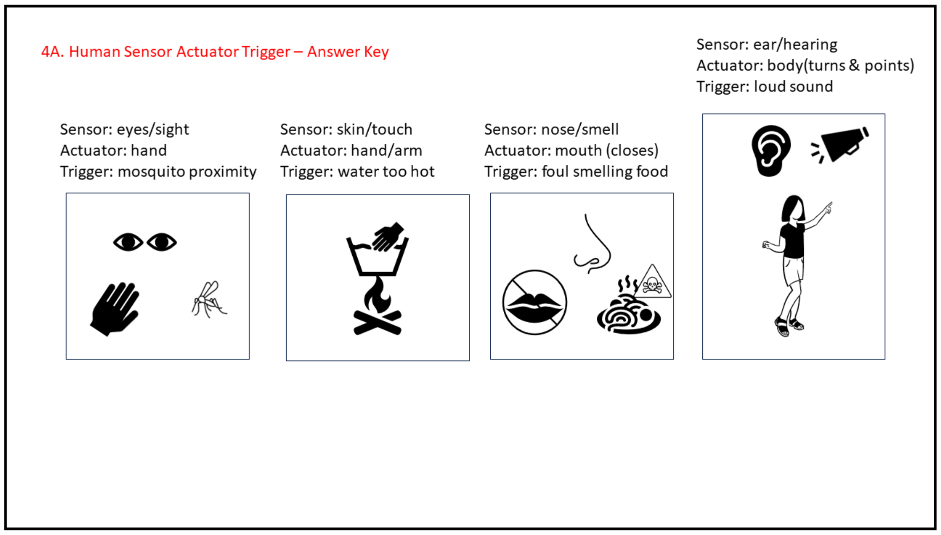


Slide 1

Public domain licensed images from kids.kiddle.co

Activity 1: Human sensor, threshold, actuator examples (Day 1)

7. Human sensor, threshold, actuator examples. **Say:** *Let's use this idea of sensor, actuator and trigger with a human body. Remember from the video that humans have sensors too. We have eyes, ears, a tongue, hands, and a nose. Humans also have actuators that will do something in response to what we sense. The body parts that we can voluntarily move, are our actuators.*
   a. Have students view **Duplication Master 4A**
   b. **Say:** *On this sheet, there is a sensor, an actuator, and a trigger. Identify each.* [give students time to write]. Review answers with the students.



4A. Human Sensor Actuator Trigger – Answer Key

Sensor: eyes/sight
Actuator: hand
Trigger: mosquito proximity

Sensor: skin/touch
Actuator: hand/arm
Trigger: water too hot

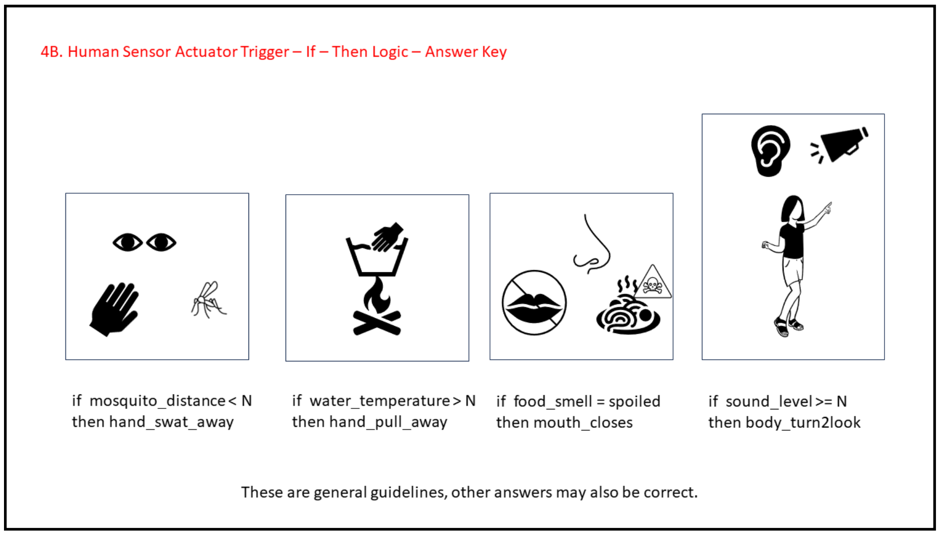
Sensor: nose/smell
Actuator: mouth (closes)
Trigger: foul smelling food

Sensor: ear/hearing
Actuator: body(turns & points)
Trigger: loud sound

c. **Say:** *Okay so for the hand, eye, mosquito example – let's identify the threshold that must be crossed for the sensor to signal the actuator to do something. How close will the mosquito have to get to you before your hand slaps it away?* Students name a distance. **Say:** *Let's call this distance the threshold.* Translate this into conditional logic on the board.

d. For example if students say the threshold is 7 inches.
   i. Write "if mosquito_distance < 7 then slap_it_away"

b. Have students identify thresholds for the other scenarios and translate those into conditional logic statements: Heat of the water before removing hand, smell of the food before refusing to eat it, loud noise before it gets attention. Students use **Duplication Master 4B** to write their answers.



4B. Human Sensor Actuator Trigger – If – Then Logic – Answer Key

if mosquito_distance < N
then hand_swat_away

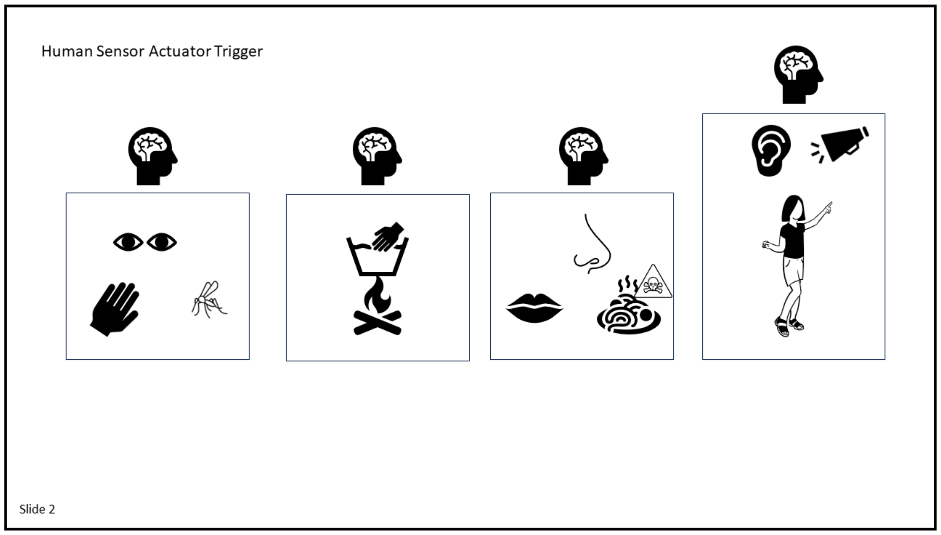
if water_temperature > N
then hand_pull_away

if food_smell = spoiled
then mouth_closes

if sound_level >= N
then body_turn2look

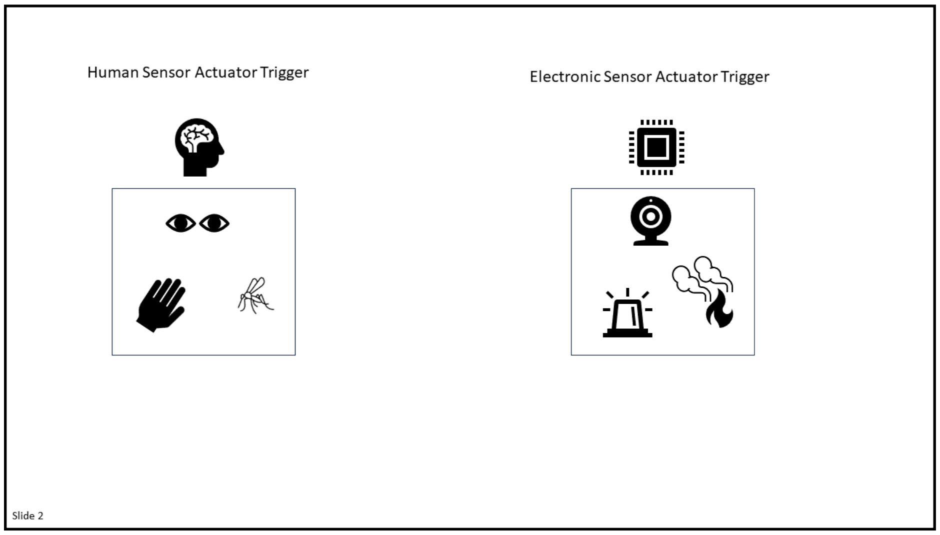
These are general guidelines, other answers may also be correct.

8. **Say:** *So we know that as humans, we have sensors (our five senses) and actuators, our body parts that can move.* **Ask:** *What is the key part of humans that connects our sensors to our actuators?* The idea is that students will identify that our brain needs to take the input from the sensor and activate the actuator based on that input. Once students identify the brain – show the slide that has the brain with all of the scenarios from **Slide 2**.

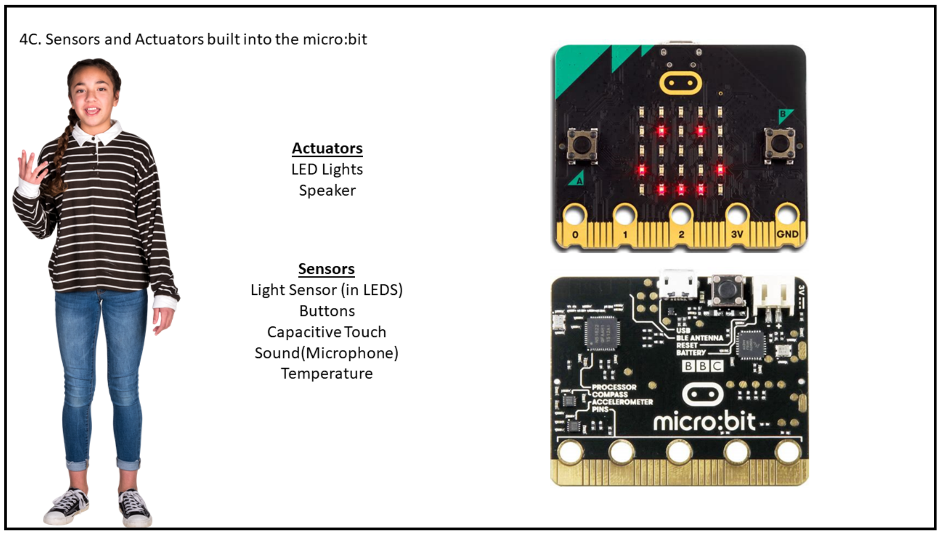

Human Sensor Actuator Trigger

Slide 2

9. **Say:** *Now let's translate this idea to microelectronics.* **Show Slide 3** *We have electronic sensors like smoke sensors. The electronic sensor senses the environment and translates that information into numeric data. Electronic actuators like the alarm receive an electronic signal that triggers them to activate. An electronic brain coordinates between the sensor and the actuator. It takes the input from the sensor and controls the action of the actuator. That electronic brain is a microcontroller like the micro:bit. A human must code the microcontroller to get environmental data from the sensor, detect whether a threshold has been crossed, and activate the actuator.*


Human Sensor Actuator Trigger
Electronic Sensor Actuator Trigger
Slide 2

10. Show the first micro:bit slide that lists the built in sensors & actuators., **4C. Sensors and Actuators**. Discuss briefly.

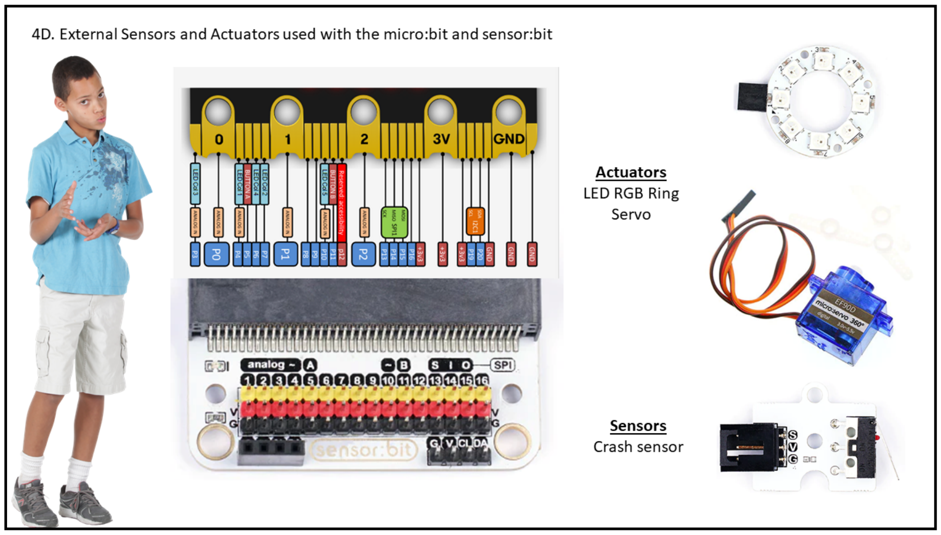


4C. Sensors and Actuators built into the micro:bit

Actuators
LED Lights
Speaker

Sensors
Light Sensor (in LEDS)
Buttons
Capacitive Touch
Sound(Microphone)
Temperature

11. Show the second micro:bit slide that lists the external sensors & actuators. **4D. External Sensors and Actuators.** Discuss briefly. **Say:** *The micro:bit allows us to attach external devices and use the micro:bit as the brain for those devices. An extension device like the sensor:bit allows us to access all of the micro:bit pins. Without the sensor:bit, we would have limited access to any micro:bit pin except 0, 1, 2, 3V and GND. With the sensor:bit, we expand our access to include micro:bit pins 4 – 20.*

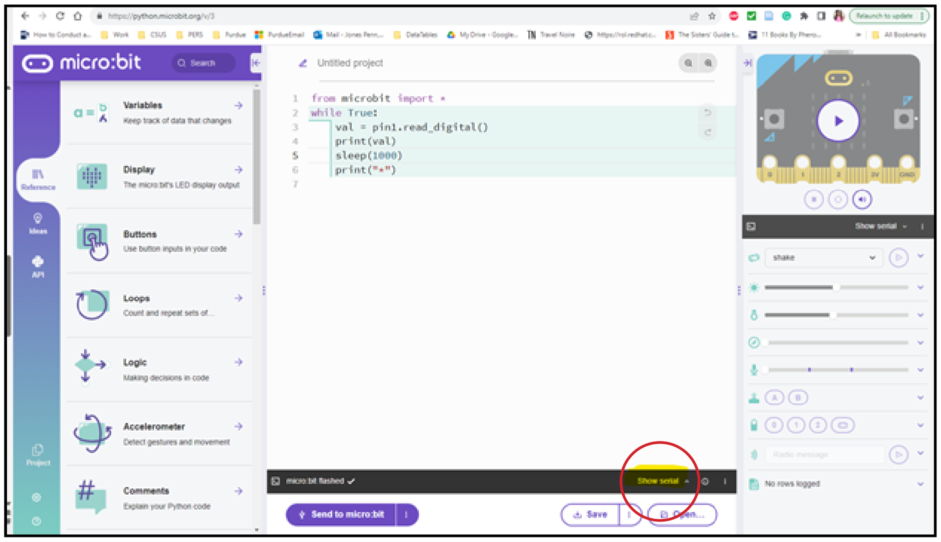

4D. External Sensors and Actuators used with the micro:bit and sensor:bit
Actuators
LED RGB Ring
Servo
Sensors
Crash sensor
sensor:bit

12. **Day 1 Discuss the concepts as a class.** Encourage students to share information they found interesting with the class. **Ask:** *Based on what we've discussed today, how do sensors work? After writing down student responses, summarize the thoughts of the class.* **Ask:** *What are the differences between sensors and actuators?* Again, write down student responses then summarize the thoughts of the class. Students should be able to identify that sensors take in information and produce output that can trigger a reaction.
13. Have each team choose two items: one internal sensor, one internal actuator and/or one external device (sensor or actuator) that they will utilize for Day 2 lesson. Be sure that each sensor/actuator is chosen by at least one team.

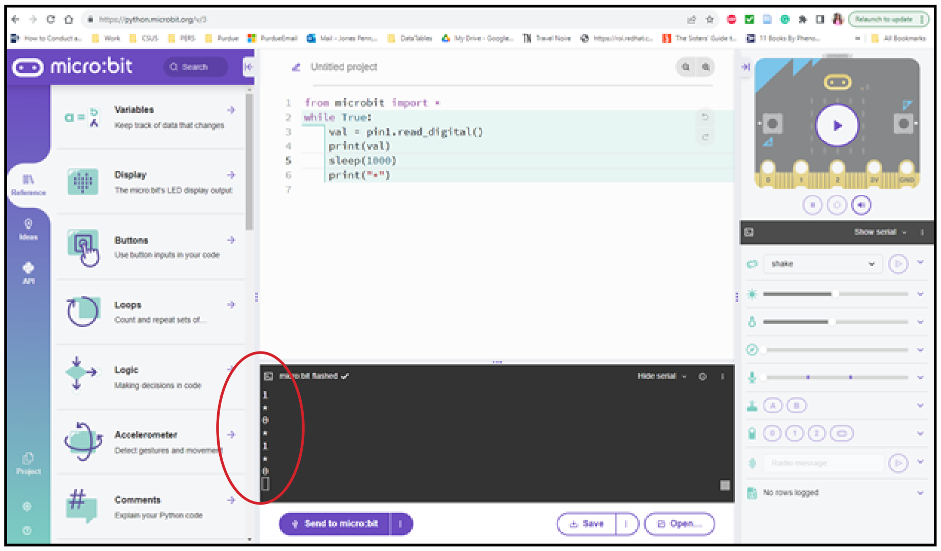
Activity 2: Coding sensors & actuators Day 2

1. **Demonstrate an example of a sensor with digital data, serial monitor.** Set up the crash sensor using the diagram on **Slide 3: Crash Sensor Demo Circuit**. Load the code for this sensor (Crash_Sensor.py) into the Micropython micro:bit online editor. Explain the following aspects of the functioning sensor using the following information:
   b. Sensor circuit: The demo uses an external crash sensor. The external crash sensor is connected to the microcontroller via pin1.
   c. Sensor code: Show the Micropython editor with the code loaded. Connect the micro:bit to the computer via a USB cable.When you flash the micro:bit, you will see that a serial monitor becomes available. Press "Show serial" to see what the serial monitor produces. Press down on the crash sensor

and the value will change from 1 to 0. Explain that the code is reading a signal from the sensor. Since it's binary, either the value will be 1 for open or 0 for closed. The serial monitor is a way to view the values sent by the sensor so that you can write logic to test for the values you want to act upon.



```
from microbit import *
while True:
    val = pin1.read_digital()
    print(val)
    sleep(1000)
    print("*")
```

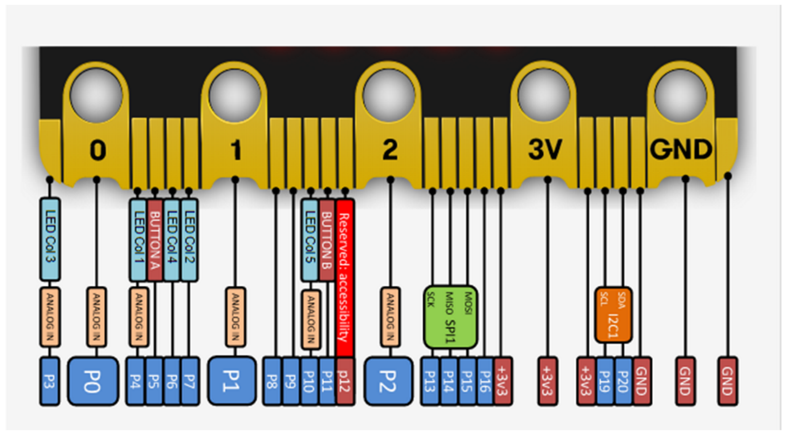


```
from microbit import *
while True:
    val = pin1.read_digital()
    print(val)
    sleep(1000)
    print("*")
```

```
micro:bit flashed ✓
1
*
0
*
1
*
0
```

2. **Micro:bit pins** Show the micro:bit pin schematic on **4E. Micro:bit pins.**

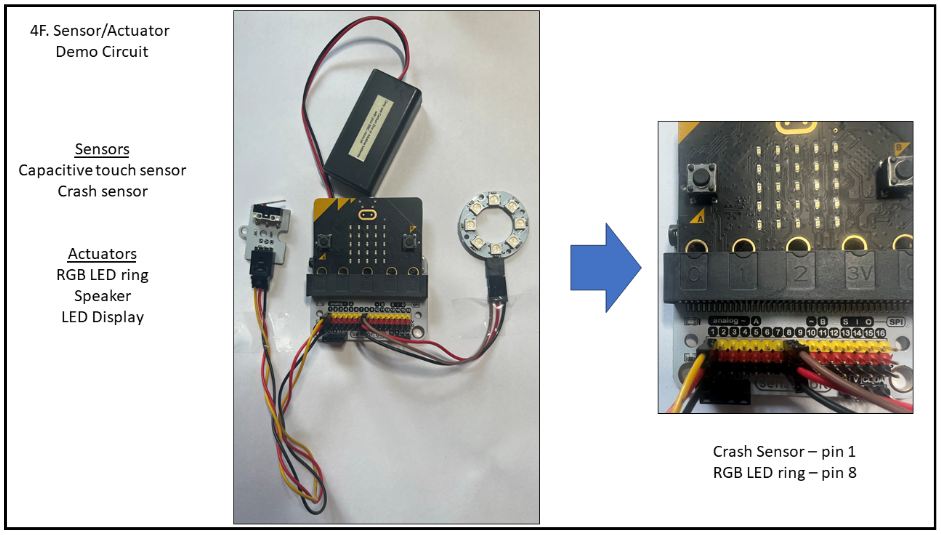
4E. Micro:bit pins



Discuss: The micro:bit microcontrollers read the sensor data digitally (or convert analog to digital). Data is represented in binary, and is usually a raw voltage number that the programmer has to convert to more useful units. Only certain pins can read analog signals (like those from a sensor) and some pins are shared with other things like the physical capabilities on the micro:bit (e.g. pin 5 and 3.) All sensors send data to the micro:bit in a form that is NOT readable for humans. This data must be translated to be "human-readable".

a. A "library" contains code others have written. These libraries can be used to read and convert the data from sensors. These sensors communicate with a serial protocol. This means that someone else has written the code needed to "talk" over serial to these sensors.
b. Direct attention back to the demo sensor setup. Show that the line pin1.read_digital() calls a library that someone else has written that converts the data from the sensor to either a 0 or 1, which your program then displays on the serial monitor.

3. **Demonstrate an example of a functional sensor/actuator setup.** Set up the sensor/actuator using the diagram in **4F Sensor Demo** Resource.



4F. Sensor/Actuator Demo Circuit

<u>Sensors</u>
Capacitive touch sensor
Crash sensor

<u>Actuators</u>
RGB LED ring
Speaker
LED Display

Crash Sensor – pin 1
RGB LED ring – pin 8

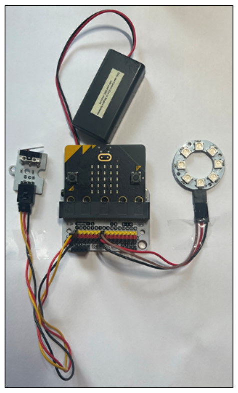
Explain the following aspects of the functioning sensor using the following information:

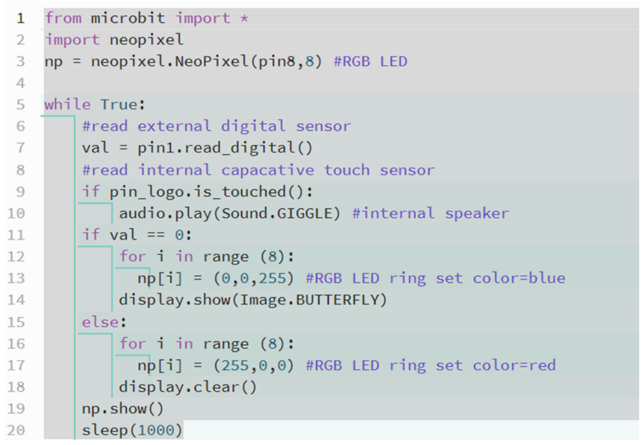
a. Inputs/sensor: The demo uses two sensors: the internal capacitive touch sensor and the external crash sensor. The external crash sensor is connected to the microcontroller via pin1.
b. Outputs/actuators: The demo uses three actuators: the internal actuators are the LED display and the microphone. The external actuator is the RGB LED ring. The RGB LED ring is connected to the microcontroller on pin 8.
c. Demonstrate the operation of the circuit. When the crash sensor is pressed, the RGB LED changes color and the micro:bit LED matrix displays a butterfly. When the pin logo is touched, the speaker plays the giggle sound.
d. Have students identify the actuators and sensors in this circuit.

4. Demo Schematic and Code: Show **4F. Sensors Demo Circuit.** Point out how the external devices are connected to pins on the sensor bit, which correlate to the micro:bit pins from the schematic. Say: “See how we are using the sensor:bit to gain access to the small pins on the micro:bit. The micro:bit can communicate with the external sensor and actuator using the pins that you saw in the schematic we looked at earlier.

5. Show **4G Sensors Demo Code**.

4G. Sensor/Actuator Demo Code





```
from microbit import *
import neopixel
np = neopixel.NeoPixel(pin8,8) #RGB LED

while True:
    #read external digital sensor
    val = pin1.read_digital()
    #read internal capacative touch sensor
    if pin_logo.is_touched():
        audio.play(Sound.GIGGLE) #internal speaker
    if val == 0:
        for i in range (8):
          np[i] = (0,0,255) #RGB LED ring set color=blue
        display.show(Image.BUTTERFLY)
    else:
        for i in range (8):
          np[i] = (255,0,0) #RGB LED ring set color=red
        display.clear()
    np.show()
    sleep(1000)
```

Say: *Let's look at the code that makes this circuit work and how the code interacts with the sensors and actuators.* Ask students to examine the code and offer their opinions about how/what the code does.

6. **Provide students an opportunity to ask questions.** After demonstrating how the sensor works together with the micro:bit to trigger the output, allow students time to ask questions about the device and do some additional research if necessary.

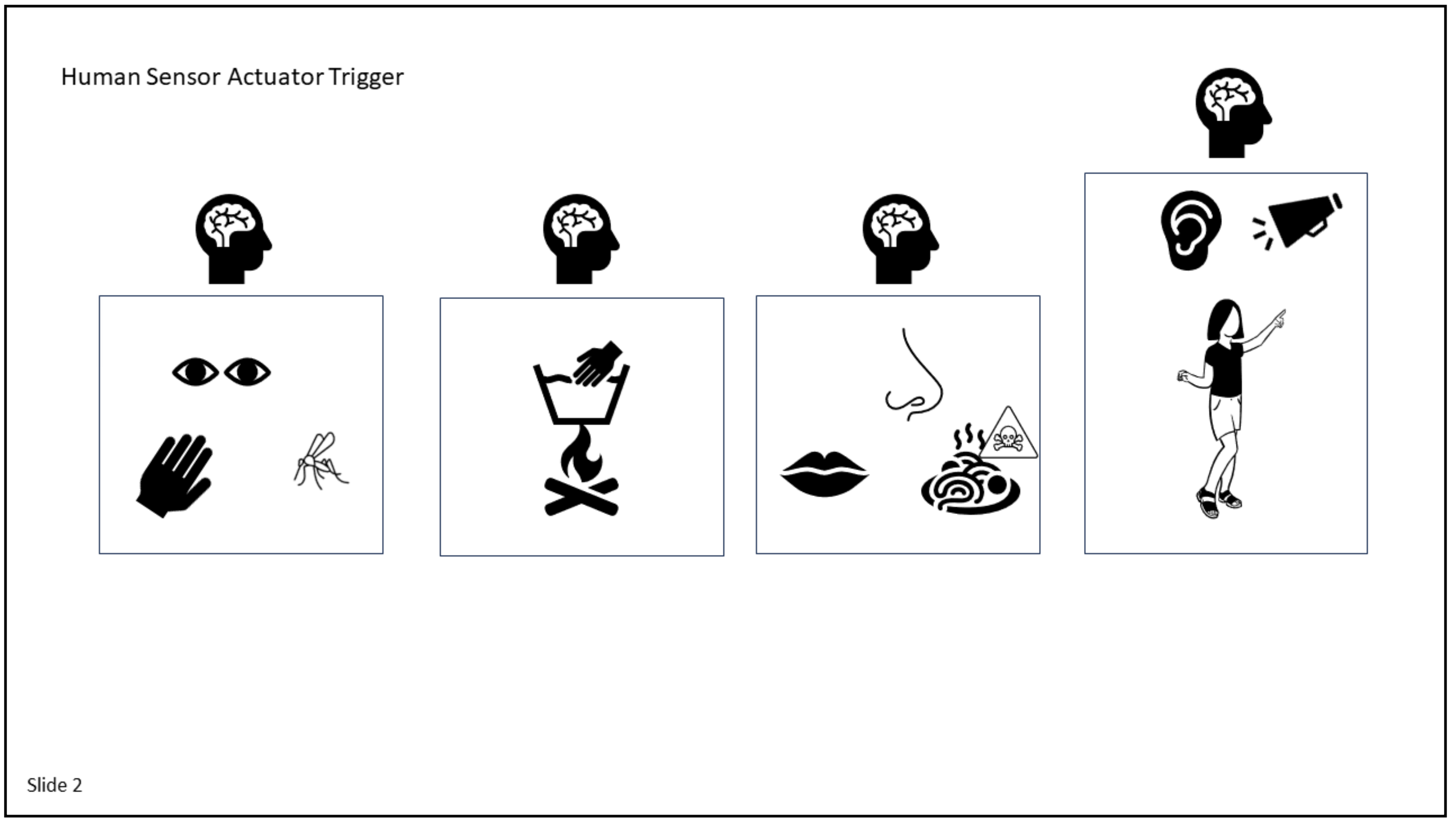
Activity: Group exploration of sensors. (Day 2-3)

7. **Day 2 Group exploration of sensors and actuators. Say:** *Now you, as a team, will explore the sensors and actuators that you chose yesterday. You have been given a small snippet of code for your sensor and actuator (**4H & 4H(b): Code Snippets.**) Use this code as a starting place to explore how to get input from the sensor and send signals to the actuator. Use the Micropython editor to enter the code, flash it to your micro:bit and see what it does. Test each individually, then pair the sensor with the actuator by deciding on a threshold that will trigger action. Write this code and flash it to your micro:bit.*
8. **Day 2 Have groups share out information.** After all groups have been explored their sensors and actuators, walk through one or two devices as a class. Give students the chance to share what they have learned and discuss any challenges they experienced.
Ask: *What does the data from the sensor mean? How did you associate the sensor output to the action of the actuator?*

NOTE: It may be beneficial to reflect using a jigsaw-style activity instead of sharing out to the class. Have one student from each team collaborate with two students from different teams. The groups for this reflection should be able to have enough information to fill out the entire guidebook (all six sensors).

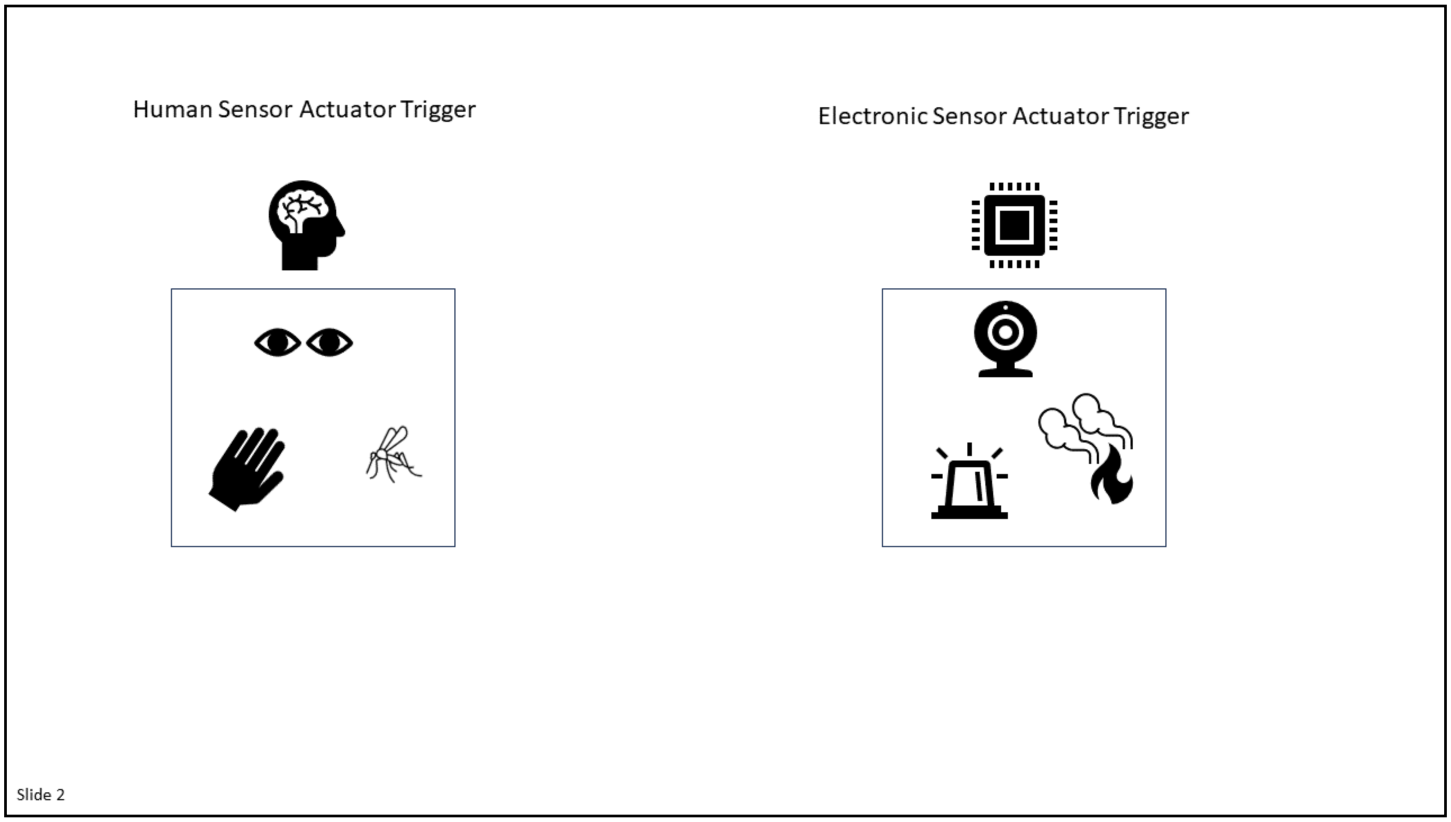
Closure

9. **Summarize: Say:** *Between each team, we have written code to interact with all of the sensors and actuators available to us in this lesson. So, as a class we have all of the sensors and actuators available to use for our smart home design.*
10. **Connect to the engineering challenge.** What phase of the engineering design process do you think we'll be in next time? Do you feel you have learned enough to be able to work through the design challenge? What further questions do you have.



Human Sensor Actuator Trigger

Slide 2

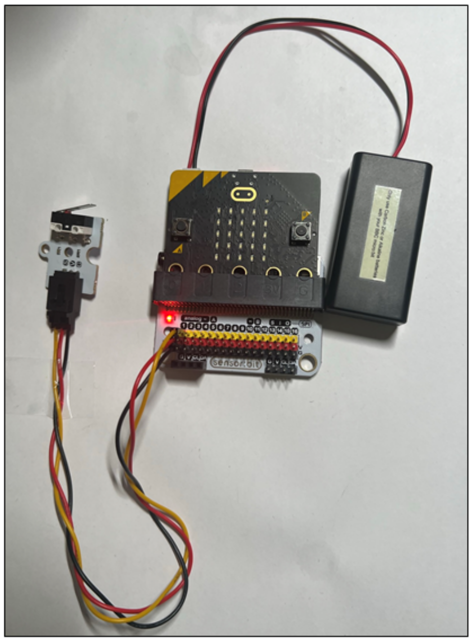


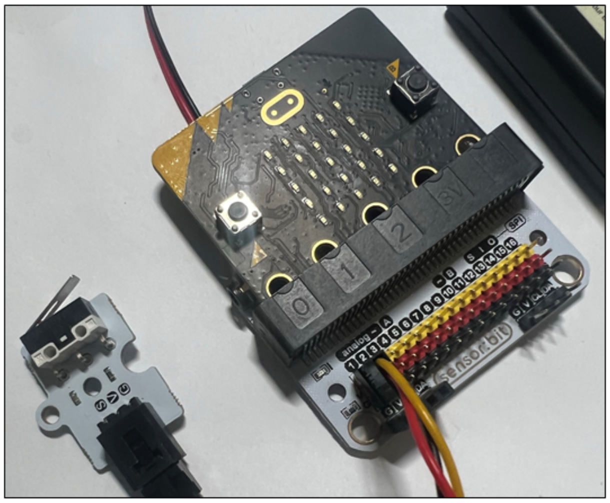
Human Sensor Actuator Trigger

Electronic Sensor Actuator Trigger

Slide 2

Crash Sensor Demo Circuit

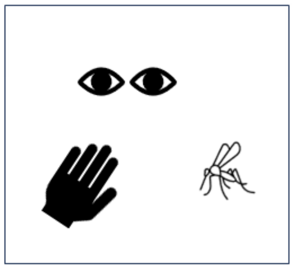




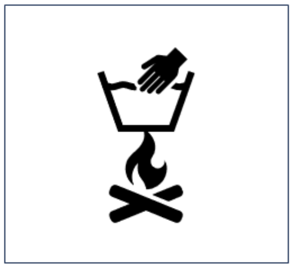
Slide 3

4A. Human Sensor Actuator Trigger

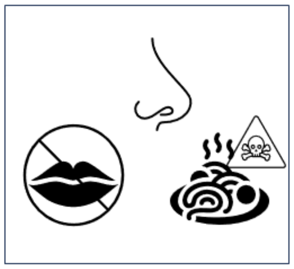
Sensor: ________________
Actuator: ______________
Trigger: ________________



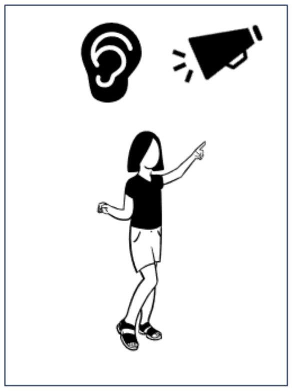
Sensor: ________________
Actuator: ______________
Trigger: ________________



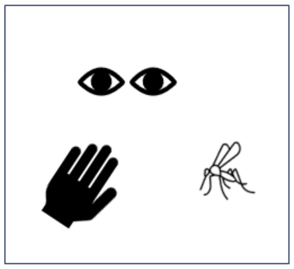
Sensor: ________________
Actuator: ______________
Trigger: ________________



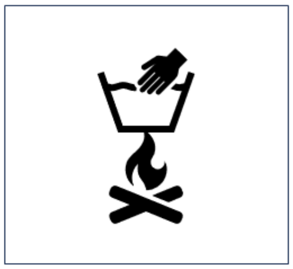
Sensor: ________________
Actuator: ______________
Trigger: ________________



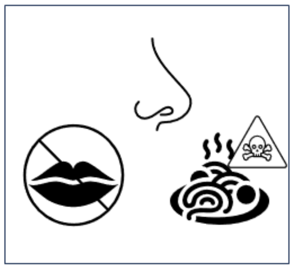
4B. Human Sensor Actuator Trigger – If – Then Logic



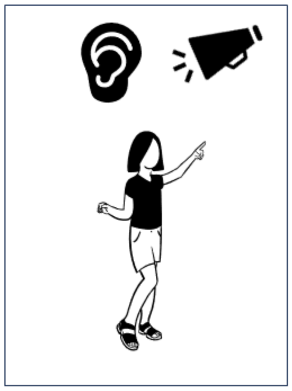
if ________________
then ________________



if ________________
then ________________

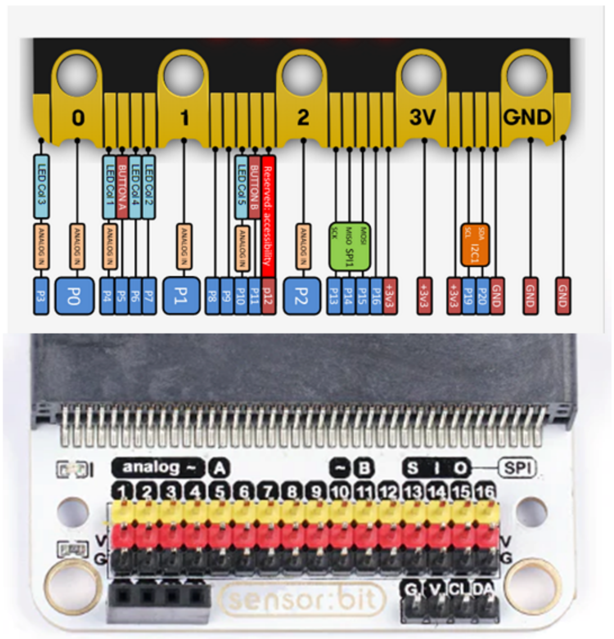


if ________________
then ________________

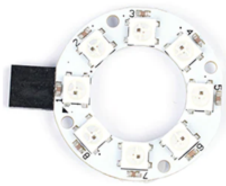


if ________________
then ________________

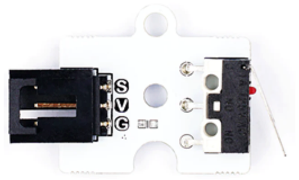
4D. External Sensors and Actuators used with the micro:bit and sensor:bit



Actuators
LED RGB Ring
Servo



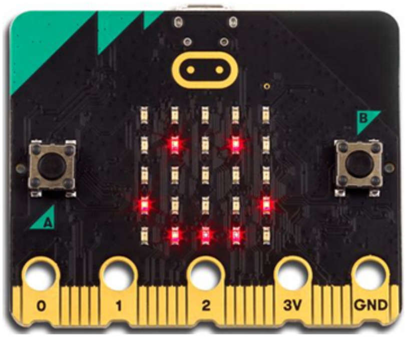
Sensors
Crash sensor

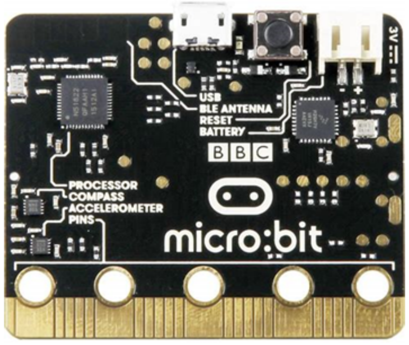


4C. Sensors and Actuators built into the micro:bit

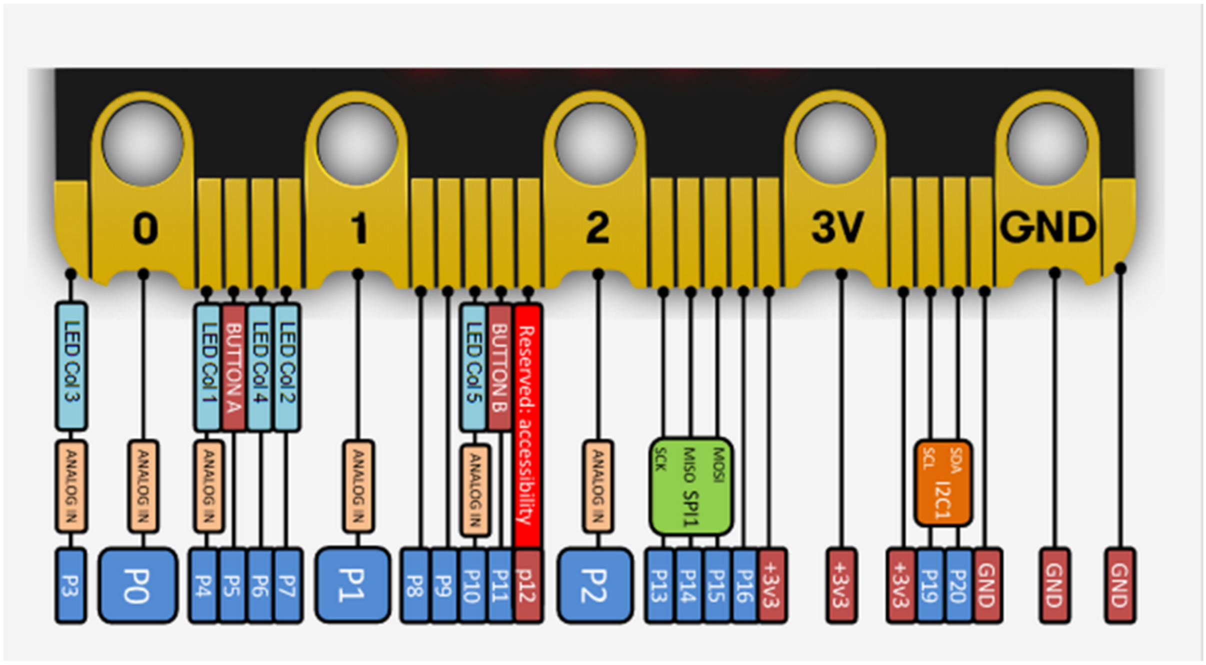
Actuators
LED Lights
Speaker

Sensors
Light Sensor (in LEDS)
Buttons
Capacitive Touch
Sound(Microphone)
Temperature

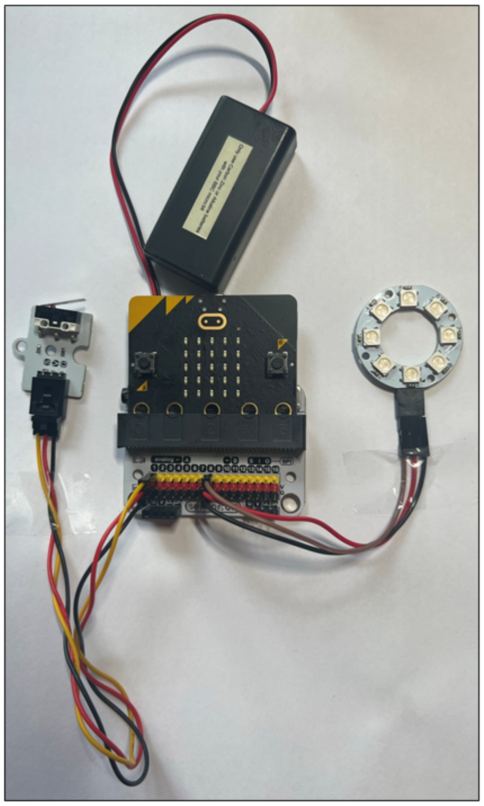


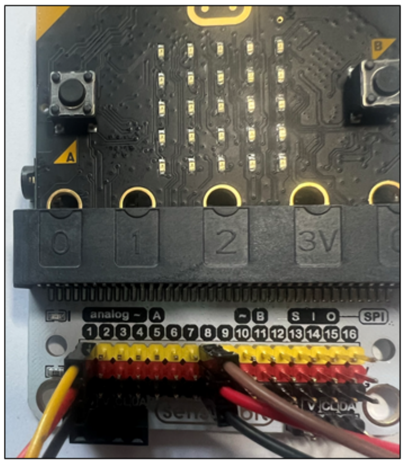


4E. Micro:bit pins



4F. Sensor/Actuator Demo Circuit



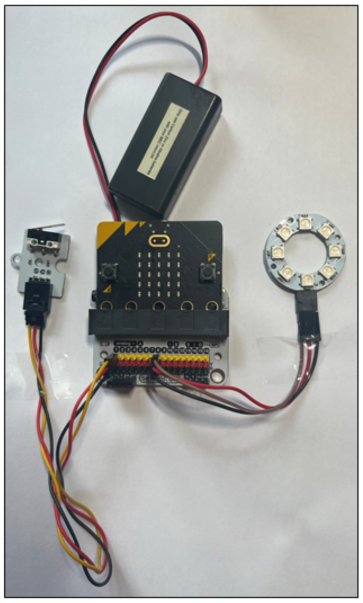


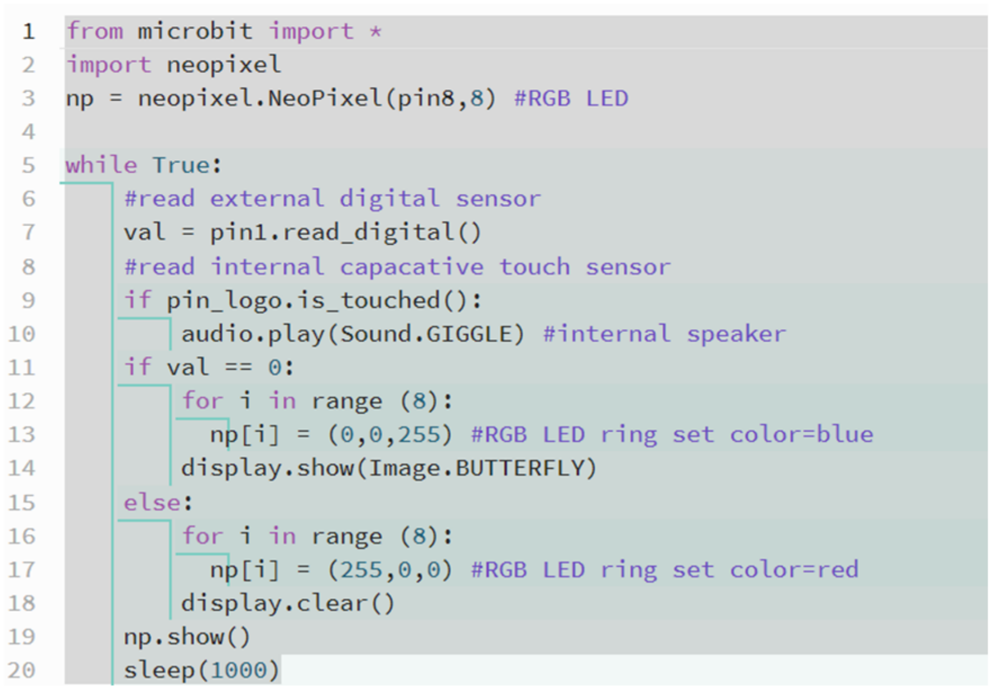
Sensors
Capacitive touch sensor
Crash sensor

Actuators
RGB LED ring
Speaker
LED Display

Crash Sensor – pin 1
RGB LED ring – pin 8

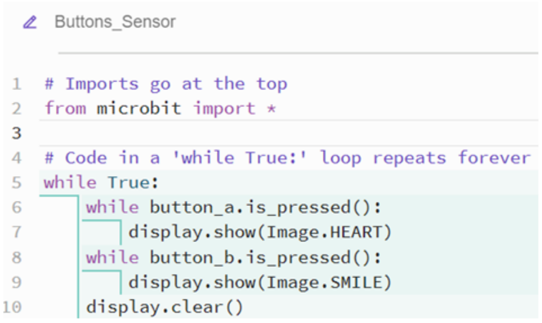
4G. Sensor/Actuator Demo Code





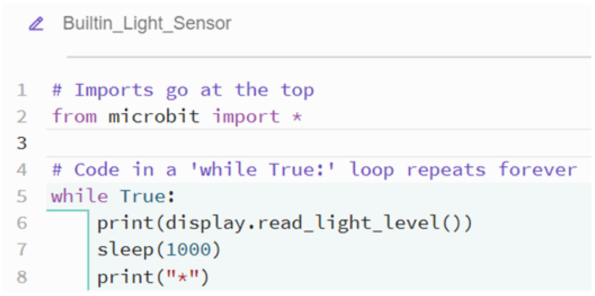
```python
from microbit import *
import neopixel
np = neopixel.NeoPixel(pin8,8) #RGB LED

while True:
    #read external digital sensor
    val = pin1.read_digital()
    #read internal capacitive touch sensor
    if pin_logo.is_touched():
        audio.play(Sound.GIGGLE) #internal speaker
    if val == 0:
        for i in range (8):
          np[i] = (0,0,255) #RGB LED ring set color=blue
        display.show(Image.BUTTERFLY)
    else:
        for i in range (8):
          np[i] = (255,0,0) #RGB LED ring set color=red
        display.clear()
    np.show()
    sleep(1000)
```

4H. Code Snippets



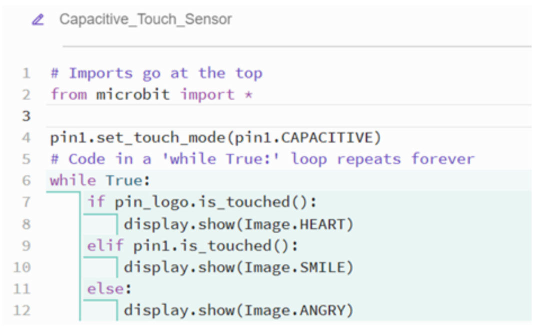
Buttons_Sensor

```python
# Imports go at the top
from microbit import *

# Code in a 'while True:' loop repeats forever
while True:
    while button_a.is_pressed():
        display.show(Image.HEART)
    while button_b.is_pressed():
        display.show(Image.SMILE)
    display.clear()
```



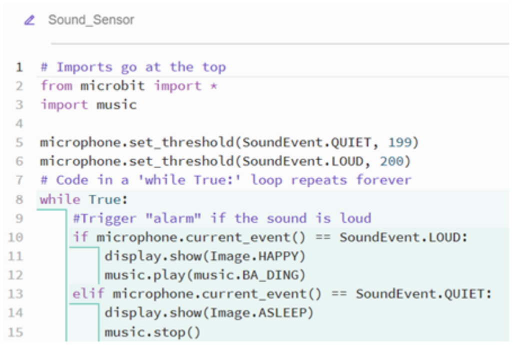
Builtin_Light_Sensor

```python
# Imports go at the top
from microbit import *

# Code in a 'while True:' loop repeats forever
while True:
    print(display.read_light_level())
    sleep(1000)
    print("*")
```



Capacitive_Touch_Sensor

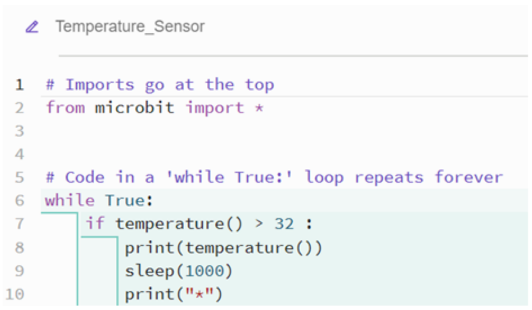
```python
# Imports go at the top
from microbit import *

pin1.set_touch_mode(pin1.CAPACITIVE)
# Code in a 'while True:' loop repeats forever
while True:
    if pin_logo.is_touched():
        display.show(Image.HEART)
    elif pin1.is_touched():
        display.show(Image.SMILE)
    else:
        display.show(Image.ANGRY)
```



Sound_Sensor

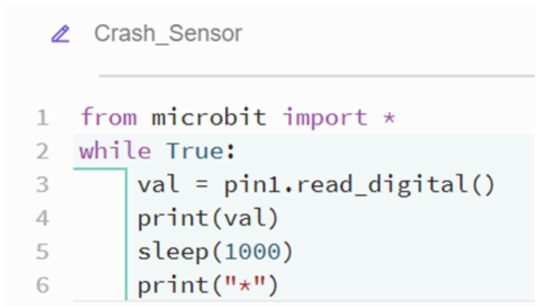
```python
# Imports go at the top
from microbit import *
import music

microphone.set_threshold(SoundEvent.QUIET, 199)
microphone.set_threshold(SoundEvent.LOUD, 200)
# Code in a 'while True:' loop repeats forever
while True:
    #Trigger "alarm" if the sound is loud
    if microphone.current_event() == SoundEvent.LOUD:
        display.show(Image.HAPPY)
        music.play(music.BA_DING)
    elif microphone.current_event() == SoundEvent.QUIET:
        display.show(Image.ASLEEP)
        music.stop()
```

4H(b). Code Snippets



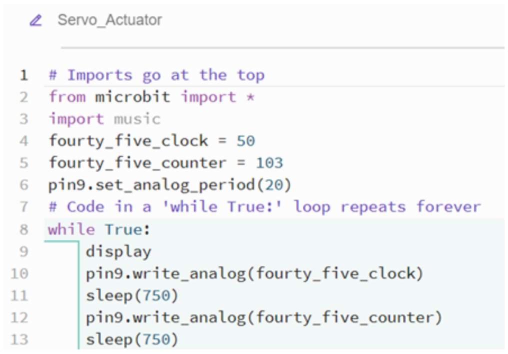
Temperature_Sensor

```python
# Imports go at the top
from microbit import *


# Code in a 'while True:' loop repeats forever
while True:
    if temperature() > 32 :
        print(temperature())
        sleep(1000)
        print("*")
```



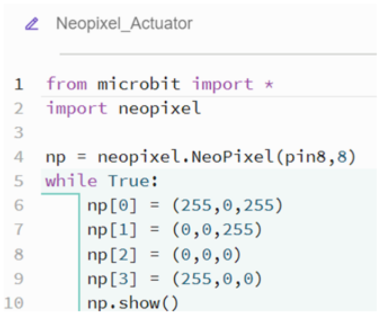
Crash_Sensor

```python
from microbit import *
while True:
    val = pin1.read_digital()
    print(val)
    sleep(1000)
    print("*")
```



Servo_Actuator

```python
# Imports go at the top
from microbit import *
import music
fourty_five_clock = 50
fourty_five_counter = 103
pin9.set_analog_period(20)
# Code in a 'while True:' loop repeats forever
while True:
    display
    pin9.write_analog(fourty_five_clock)
    sleep(750)
    pin9.write_analog(fourty_five_counter)
    sleep(750)
```



Neopixel_Actuator

```python
from microbit import *
import neopixel

np = neopixel.NeoPixel(pin8,8)
while True:
    np[0] = (255,0,255)
    np[1] = (0,0,255)
    np[2] = (0,0,0)
    np[3] = (255,0,0)
    np.show()
```

LESSON FIVE:

Lesson Objectives
Students will be able to:
- Brainstorm potential solutions
- Generate Flowcharts
- Write Pseudocode

Time Required
One-two 50-minute lesson(s)

Standards Addressed
ICS-2.5 Formulate algorithms using programming structures to decompose a complex problem.

Key Terms
Flowcharts, pseudocode

Lesson Materials
Per Classroom
- EDP Poster

Per Student
- EDP slider and paperclip
- Laptop/Chromebook/Tablet
- Engineering notebook

Duplication Masters
5.A, 5.B, 5.C

Classroom Instruction

Introduction

1. **Tie in the engineering problem. Ask:** *What is our engineering design problem?*
2. **Identify where they are in the engineering design process (Plan). Say:** *So far, we have defined the problem with help from our client.* Point out the "Problem" block on the EDP poster and have students look at their EDP sliders. **Say:** *Before we can start designing solutions, we need more information.* **Ask:** *What step of the engineering design process are we in?* The students should identify that they are in the "Plan" stage.
3. **Connect back to Prior Knowledge. Say:** *In the previous lesson, you learned about coding, functions, and sensors. Now, you will use that knowledge to help solve the client problem.*

Activity

4. **Discuss tools to generate ideas for a solution. Say:** *There are different ways that engineers generate ideas to solve a problem. Brainstorming, and sketching are tools that we can use to work toward solving the challenge. Brainstorming is the generation of ideas to work toward a goal. Sketching your ideas will help visualize the electric expansion pack so your teammates can ask questions about how it will work.*
5. **Brainstorm potential solutions to the client's problem. Say:** *We are going to brainstorm individually, and then work as a group to continue brainstorming solutions to the client's problem.* Hand out Duplication Master 7.C Brainstorming Worksheet and allow a few minutes for the students to brainstorm individually.
6. **Gather into pre-determined design teams to brainstorm. Say:** *Now that you have come up with some ideas on your own, with your team to review your ideas and pick 2-3 of the best sensors and operations.* Allow the students a few minutes to discuss.
7. **Sketch top design choices.** Hand out Duplication Master 7.D Design Sketching. **Say:** *Now that you have a final design choice to pursue, sketch it out! Each student in your group should make a sketch so you can see if your visualizations are similar or different.* Allow students some time to sketch their primary idea.
8. **Create Flowcharts.** Show Slides with flow chart components and explain how these work like a choose-your-own-adventure game. Decisions, or in this case conditions indicate which

path your program will take and what will happen next. Write Flowcharts for each of the chosen functions

9. **Write Pseudocode.** Using their flow charts, have students write out lines of pseudocode for them to reference when they start their final program in the next session.

Closure

Connect the activity to the engineering design challenge. Ask: *What did we learn today that will help us provide a recommendation to the client? Why is brainstorming important? How did sketching help explain your idea?* **Say:** *We will move to the Try stage during our next class.*

Name______________________________________ Date_______________ Period __________

5.A Brainstorming Worksheet

For each idea, ask yourself the following questions:
What sensor does it use?
Where in the house would it go?
How does it help solve the Client Problem?
How does it help the end users?

When brainstorming with your teammate(s), you get to ask the questions! Try to figure out what they are thinking about and how it will help our client with the challenge.

Idea #1:

Idea #2:

Idea #3:

Idea #4:

Idea #5

Idea #6

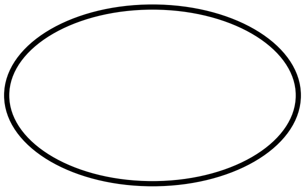
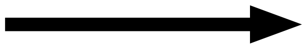
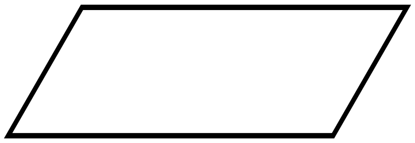
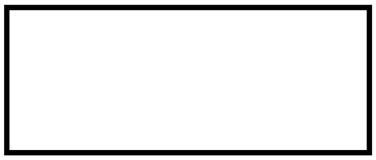
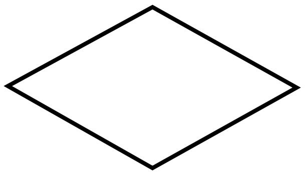
Name______________________________________ Date______________ Period __________

5.B Design Sketching

Now that you have chosen which idea will be proposed as your solution to the engineering design challenge, it is time to draw it! Make a sketch of your design in the box below. What components will there be? Where in the house will they be? Ensure you label components clearly.

Name____ Date_______________ Period __________

5.C – Flowchart Symbols and Pseudocode Tips

Symbol	Name	Function
 Oval	Start/End	Start or End of Code
 Arrow	Arrow	Connects other Shapes
 Parallelogram	Input/Output	Data that is provided to or produced by the code
 Rectangle	Process	Action that the code performs
 Diamond	Decision	If/Else code block, or decision between two or more options

Pseudocode Tips:

1. Have a goal in mind. Sometimes it is easiest to start with the beginning and end of your process. Having these clear will help you not get confused for the middle part of the process.
2. Use (Mostly) Plain English – we want this to be understandable to someone with very limited understand of the project and even programming
3. Use Control Structures: Introduce standard control structures such as if/else, while, for, etc. These should be written in a way that clearly shows the flow of the program. Use your flowcharts to figure out where these structures should be written.

Name____ Date________________ Period __________

LESSON SIX:

Lesson Objectives
Students will be able to:

- Write functions to control sensors
- Use flowcharts and pseudocode to build algorithms, and evaluate that the algorithms are accomplishing their goals

Time Required
One-two 50-minute lesson(s)

Standards Addressed
ICS-2.1 Use the design process to iteratively develop a computing artifact

ICS-2.2 Demonstrate competencies of programming constructs, including: use of data types and variables, control structures (sequencing, looping, branching), and modularity (such as a function).

ICS-2.3 Understand how abstractions hide implementation details when used in everyday objects.

ICS-2.4 Use abstraction to manage program complexity (such as a function to create recallable code).

ICS-2.5 Formulate algorithms using programming structures to decompose a complex problem

Lesson Summary

Lesson Summary: Creating an algorithm and using a common model to try the code. Using their model home, sensors, and microbit, assemble your smart home layout. Use your pseudocode and flowchart to create microbit code to control the smart home.

Classroom Instruction

Introduction

1. **Identify Where Students are In the Engineering Design Process. Say** *Last time we Planned our solution all out, well today we get to try!*
2. **Discuss the Worksheets from last lesson, t**hey will use these documents to try out their designs. Have students also have their notes and/or code from lessons 3 and 4

Activites

3. **Assembling the Smart Home Layout.** Using Duplication Masters 6.1 as a template, cut out and construct the house models from cardboard.
4. **Programming the Sensors.** Have students begin translating their pseudocode into code for their microbits. Encourage students to work with each other to debug and solve problems as the come up. Occasionally remind students by saying *"This is a prototype and most likely will not work perfectly. Engineers and programmers often have to try over and over again to get things right."*

Conclusion

5. **Wrap Up and Reflection**
    a. Reflect on the challenges and successes of the implementation phase.
    b. Briefly introduce the next lesson's objective of evaluating their designs

Try It Out!

Key Terms
Functions, debugging

Lesson Materials
Per classroom
- EDP Poster

Per Student
- EDP slider and paperclip
- Laptop/Chromebook/Tablet
- Engineering notebook

Per Group (3 per group)
- Sensor 1
- Sensor 2
- Sensor 3
- Microbit

Duplication Masters
6.1

Duplication Masters 6.1

LESSON SEVEN:

Lesson Objectives
Students will be able to:
- Demonstrate their code's functionality by testing it
- Communicate the solution to their peers

Time Required
One 50-minute lesson

Standards Addressed
ICS-2.1 Use the design process to iteratively develop a computing artifact

ICS-2.6 Assess a program by testing to verify correct behavior.

ICS-5.4 Analyze the work of peers and provide feedback.

Key Terms
Functions, debugging

Lesson Materials
Per classroom
- EDP Poster

Per Student
- EDP slider and paperclip
- Laptop/Chromebook/ Tablet
- Engineering notebook

Per Group (3 per group)
- Sensor 1
- Sensor 2
- Sensor 3
- Microbit

Duplication Masters
7.1
7.2

Before the Lesson

Review Duplication Masters 7.1, teacher may want to decide to give students the rubric early so that they can start refining the projects for a grade. It is possible that this presentation of rubric becomes the main focus of students however, and overshadows the lesson activities. Teachers should use their best judgement when it comes to their students.

Classroom Instruction

Introduction (5 minutes)

1. **Identify where students are in the Engineering Design Process. Say:** *Today we will focus on testing and presenting your smart home sensor coding solutions.* Emphasize how testing validates the functionality and reliability of their solutions.

Activity (40 minutes)

2. **Testing Preview** (5 minutes): Distribute Duplication Masters 7.2. Explain that students are first going to have a chance to test their own functions, then demonstrate their solutions to their peers. **Optional:** Also pass out Duplication Masters 7.1, explain to students that they will ultimately be graded on these standards, but not today.

3. **Hands-On Testing** (10 minutes): Students use their model homes to test their sensor solutions. Encourage them to make notes of any issues and improvements needed.

4. **Peer Presentations** (20 minutes): In pairs or small groups, students present their tested solutions to each other. Use Duplication Masters 7.2 Peers provide constructive feedback, focusing on functionality and problem-solving aspects.

Conclusion (5 minutes)

5. **Reflection and Learning Points:** Guide students in reflecting on what they learned from the testing process. Discuss the value of feedback and iteration in developing effective solutions.
   a. Final Remarks and Q&A: Sum up the session, highlighting the importance of thorough testing in engineering projects.
   b. Address any final questions or concerns from the students.

Assessment
Activity Embedded Assessment:
Peer Feedback During the Testing Stage

Name____ Date_______________ Period __________

Duplication Masters 7.1

Rubric

Concept	Potential Scores	Student Score
Use of the Design Process (ICS-2.1)	**Exemplary (4 points):** Demonstrates a thorough understanding of the design process with clear, iterative development steps. **Proficient (3 points):** Adequately uses the design process with some iterative development. **Basic (2 points):** Shows minimal use of the design process with limited iteration. **Needs Improvement (1 point):** Little to no evidence of using the design process.	
Programming Constructs (ICS-2.2)	**Exemplary (4 points):** Excellent use of data types, variables, control structures, and modularity in programming. **Proficient (3 points):** Good use of programming constructs with minor errors or inefficiencies. **Basic (2 points):** Basic use of constructs but with notable errors or misconceptions. **Needs Improvement (1 point):** Limited or incorrect use of programming constructs.	
Abstraction in Everyday Objects (ICS-2.3)	**Exemplary (4 points):** Demonstrates a deep understanding of abstraction and its application in hiding implementation details. **Proficient (3 points):** Adequate understanding and use of abstraction in the project. **Basic (2 points):** Shows basic understanding with some misconceptions or limited application. **Needs Improvement (1 point):** Poor understanding or application of abstraction concepts.	
Managing Program Complexity (ICS-2.4)	**Exemplary (4 points):** Effectively uses abstraction to create clean, recallable code, significantly managing complexity. **Proficient (3 points):** Manages complexity well, though there may be minor issues in code organization. **Basic (2 points):** Some effort to manage complexity, but the approach is rudimentary or flawed. **Needs Improvement (1 point):** Little effort or success in managing program complexity.	

Name____ Date_______________ Period __________

Algorithm Formulation (ICS-2.5)	**Exemplary (4 points):** Excellent formulation of algorithms to decompose complex problems efficiently and creatively. **Proficient (3 points):** Good formulation with effective problem decomposition, albeit with some inefficiencies. **Basic (2 points):** Basic algorithm formulation, but lacking in efficiency or creativity. **Needs Improvement (1 point):** Limited or flawed approach in algorithm formulation.	
Program Assessment and Testing (ICS-2.6)	**Exemplary (4 points):** Thorough testing with comprehensive verification of correct behavior. **Proficient (3 points):** Adequate testing with most behaviors verified correctly. **Basic (2 points):** Basic testing performed, but some aspects of behavior remain unverified or incorrectly assessed. **Needs Improvement (1 point):** Minimal or no testing, with little verification of program behavior.	
	Total	

Name____ Date_______________ Period __________

Duplication Masters 7.2

Peer Observation Report

What Sensor(s) does the team use in this solution?

What does the team claim the code can do?

Does the code Function as explained? If not, where does it fail, or what does it do instead?

What is one or more recommendations you would make to this team to improve their code?

Name____ Date_______________ Period __________

LESSON EIGHT:

Lesson Objectives
Students will be able to:
- Refine their code with peer feedback and observed issues
- Communicate their revised solutions to the client
- Reflect on the project as a whole

Time Required
Two 50-minute lesson

Standards Addressed
ICS-2.1 Use the design process to iteratively develop a computing artifact

ICS-2.6 Assess a program by testing to verify correct behavior.

ICS-5.4 Analyze the work of peers and provide feedback.

Key Terms
Functions, debugging

Lesson Materials
Per classroom
- EDP Poster

Per Student
- EDP slider and paperclip
- Laptop/Chromebook/ Tablet
- Engineering notebook

Per Group (3 per group)
- Sensor 1
- Sensor 2
- Sensor 3
- Microbit

Classroom Instruction

Introduction (5 minutes)

1. **Identify Where Students Are In the Engineering Design Process. Say:** *Now that you have feedback from your peers, the client needs you to prepare a final report with your recommended solutions to them*

Activity (2 days)

2. **Refinement Session** (15 minutes): Students work individually or in groups to incorporate the feedback they received into their code. Encourage them to focus on improving functionality, usability, and reliability.

3. **Preparation of Client Report** (20 minutes day 1, all but 10 mins of day 2): Using Duplication Masters 8.A, students prepare a report that outlines their solution, how it meets the client's needs, and any unique features or considerations.

Conclusion (10 minutes)

4. **Final Remarks and Q&A:** Have students reflect on the project as a whole, spend a few minutes to have a class discussion about what they learned, and what they liked/didn't like about this project.

5. **Give Post Assessment**

Report it Out!

Duplication Masters

8.A

8.B

8.C

8.D

Name____ Date_______________ Period __________

Duplication Masters 8.A

Preparing Your Make Sense Project Report

Section 1: Understanding the Client's Needs
1. Describe the Client's Problem:
   - What is the problem or need addressed by your project?
   - Why is it important to the client?

2. Client's Expectations:
   - What are the specific requirements or expectations of the client?

Section 2: Your Solution
1. Overview of Your Solution:
   - Briefly describe the solution you have developed.

2. Technologies Used:
   - List the programming languages, tools, and technologies you used.

3. Implementation Process:
   - Outline the steps you took to develop the solution.

Section 3: Meeting the Client's Needs
1. Solution Alignment:
   - Explain how your solution addresses the client's problem.
   - How does it meet or exceed the client's expectations?

2. Benefits of Your Solution:
   - What are the advantages of your solution?
   - How does it improve upon existing solutions?

Section 4: Reflections on Redesign
1. Explain how your solution was refined over time. How did those changes come about?
2. Challenges and Learning:
   - Describe any challenges faced during the project and how you overcame them.
   - What did you learn in the process?

Section 5: Effective Communication
1. Clarity and Conciseness:
 Use clear and concise language.
   - Avoid technical jargon, or explain it if necessary.

2. Structure and Organization:
   - Organize your report logically.
   - Use headings, bullet points, and numbered lists for clarity.

Name____ Date_______________ Period __________

3. Visual Aids:
   - Include diagrams, charts, or screenshots to support your text.

4. Proofreading:
   - Check for spelling and grammatical errors.
   - Ensure technical accuracy.

Final Checklist:

☐ Report complete with all sections.
☐ Technical details accurately explained.
☐ Report proofread and formatted.
☐ Visual aids included where necessary.

Teacher's Comments:

____

____

____

Name______________________________________ Date_______________ Period __________

Duplication Masters 8.B

Rubric

Concept	Potential Scores	Student Score
Use of the Design Process (ICS-2.1)	**Exemplary (4 points):** Demonstrates a thorough understanding of the design process with clear, iterative development steps. **Proficient (3 points):** Adequately uses the design process with some iterative development. **Basic (2 points):** Shows minimal use of the design process with limited iteration. **Needs Improvement (1 point):** Little to no evidence of using the design process.	
Programming Constructs (ICS-2.2)	**Exemplary (4 points):** Excellent use of data types, variables, control structures, and modularity in programming. **Proficient (3 points):** Good use of programming constructs with minor errors or inefficiencies. **Basic (2 points):** Basic use of constructs but with notable errors or misconceptions. **Needs Improvement (1 point):** Limited or incorrect use of programming constructs.	
Abstraction in Everyday Objects (ICS-2.3)	**Exemplary (4 points):** Demonstrates a deep understanding of abstraction and its application in hiding implementation details. **Proficient (3 points):** Adequate understanding and use of abstraction in the project. **Basic (2 points):** Shows basic understanding with some misconceptions or limited application. **Needs Improvement (1 point):** Poor understanding or application of abstraction concepts.	
Managing Program Complexity (ICS-2.4)	**Exemplary (4 points):** Effectively uses abstraction to create clean, recallable code, significantly managing complexity. **Proficient (3 points):** Manages complexity well, though there may be minor issues in code organization. **Basic (2 points):** Some effort to manage complexity, but the approach is rudimentary or flawed. **Needs Improvement (1 point):** Little effort or success in managing program complexity.	

Name____ Date_______________ Period __________

Algorithm Formulation (ICS-2.5)	**Exemplary (4 points):** Excellent formulation of algorithms to decompose complex problems efficiently and creatively. **Proficient (3 points):** Good formulation with effective problem decomposition, albeit with some inefficiencies. **Basic (2 points):** Basic algorithm formulation, but lacking in efficiency or creativity. **Needs Improvement (1 point):** Limited or flawed approach in algorithm formulation.	
Program Assessment and Testing (ICS-2.6)	**Exemplary (4 points):** Thorough testing with comprehensive verification of correct behavior. **Proficient (3 points):** Adequate testing with most behaviors verified correctly. **Basic (2 points):** Basic testing performed, but some aspects of behavior remain unverified or incorrectly assessed. **Needs Improvement (1 point):** Minimal or no testing, with little verification of program behavior.	
	Total	

Name______________________________________ Date______________ Period __________

8.C Content Post Assessment

1. Write a definition for the word, “computer”.

2. What is a central processing unit (CPU)?

3. What is the difference between random access memory (RAM) and hard drives?

4. What does an actuator do?

5. List three examples of coding languages.

6. Within programming, what is a variable?

7. What is the difference between software and hardware?

8. What does the term, “microelectronics” mean?

9. How are microelectronics used in the field of computer science?

10a. What jobs would you be interested in that use microelectronics?

10b. Provide one example of how microelectronics is used in that job.

Name______________________________________ Date______________ Period __________

8.D Content Post Assessment Key

1. Write a definition for the word, “computer”.

    A computer is a programmable electronic deivce that stores and processes data.

2. What is a central processing unit (CPU)?

    The CPU is the brain of a computer. It interprets, processes, and executes instructions.

3. What is the difference between random access memory (RAM) and hard drives?

    RAM is temporary storage that is lost when the power is turned off. A hard drive maintains long-term storage and keeps data safe even when power is turned off.

4. What does an actuator do?

    An actuator converts energy into mechanical force.

5. List three examples of coding languages.

    There are many answers to this question, but examples include C, C++, HTML, Java, JavaScript, PHP, Python, R, Ruby, and SQL.

6. Within programming, what is a variable?

    A programming variable is value or set of values that store information.

7. What is the difference between software and hardware?

    Hardware is any physical component of a computer. Software is the programming that tells the hardware what to do and how to do it.

8. What does the term, “microelectronics” mean?

    Student answers may vary, but the formal definition of microelectronics is the design, manufacture, and use of microchips.

9. How are microelectronics used in the field of computer science?

    There are many answers to this question, but examples include the design and manufacturing of computing devices (such as CPU, memory, GPU, sensors, etc.). Microelectronics are foundational in computer science as they are the hardware that allow for programmable automation.

10. What jobs would you be interested in that use microelectronics? Provide one example of how microelectronics is used in that job.

    Students’ answers will vary based on interest. Credit may be given as long as at least one job example is provided with their logic behind how that job uses microelectronics.