# Coding using Microbits - Python — Reflections

## Module 2:  Software & Hardware (Algorithms)

This module introduces a conceptual framework for thinking of a computing device as something that uses code to process one or more inputs and send them to an output(s). Questions to be answered include: What is a computer? What is a microbit and what can it do? Students will be making projects that utilize the sensors and screen output of the micro:bit.

What computers are in your house?

What are the parts of a computer? List examples of each.

|  |  |  |
|---|---|---|
|  |  |  |

**Blackbox -** What are the parts in a Blackbox?

|  |  |  |
|---|---|---|
| ⇒ |  | ⇒ |

What are some examples of Blackboxes in society?

### 02.2a Sensors Temperature Activity

Pseudocode for Sensors Temperature program

### Button A Press Event

What is the code for a button A pressed event?

What comments should be included at the top of a program?

How could your microbit be calibrated to reflect the current room temperature?

What are some other **sensor** inputs on the microbit?

### 02.2b Sensors Temperature & Compass Activity

Add a button B pressed event to your Sensor code to find the compass direction. Pseudocode to add a compass to the  Sensors Temperature program

### 02.2c Accelerometer Tilt Activity

Code different tilt events and display an arrow or a word showing the tilt direction. Pseudocode to create a tilt events program. (See Python Microbit Notes)

### 02.3 Project: Blackbox

In this project you will plan, design, and create your own Blackbox using different events, microbit sensors, output to LEDs, and MakeCode's block programming. You will also use a maker elements as part of you design and construction.

Brainstorm Ideas ____________________________________________

## 2.0 Ideas, Sketches, Planning, Notes, & Reflections —

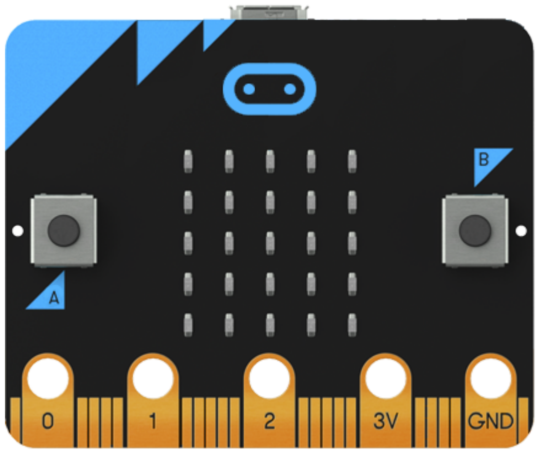
### Coding & Innovation using Microbits - Python

Project: ______________________________

Description: ______________________________

______________________________

Project Sketch:



Blackbox Algorithm & Pseudocode

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

# 2.0 Ideas, Sketches, Planning, Notes, & Reflections —

**Coding & Innovation using Microbits - Python**

Materials Needed: ______________________________

______________________________

Coding Plan: ______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

## Notes & Reflections

Beta Testing: ______________________________

______________________________

Revision Ideas: ______________________________

______________________________

Photos:

## Assessment Rubric - **Competency scores**

| **Competencies** | **4** | **3** | **2** | **1** |
|---|---|---|---|---|
| **Inputs** | At least 4 different inputs are successfully implemented. | At least 3 different inputs are successfully implemented. | At least 2 different inputs are successfully implemented. | Fewer than 2 different inputs are successfully implemented. |
| **Outputs** | At least 4 different outputs are successfully implemented. | At least 3 different outputs are successfully implemented. | At least 2 different outputs are successfully implemented. | Fewer than 2 different outputs are successfully implemented. |
| **Micro:bit Program** | micro:bit program:<br>1) uses event handlers in a way that is integral to the program<br>2) compiles and runs as intended<br>3) includes meaningful comments | micro:bit program lacks 1 of the required elements. | micro:bit program lacks 2 of the required elements. | micro:bit program lacks all of the required elements. |
| **Collaboration Reflection** | Reflection piece includes:<br>1) brainstorming ideas<br>2) construction<br>3) programming<br>4) beta testing | Reflection piece lacks 1 of the required elements. | Reflection piece lacks 2 of the required elements. | Reflection piece lacks 3 of the required elements. |

## Notes

## Python Microbits Code Notes

### Events Code

Here is a link to all the possible button and gesture events associated with the microbit. Any section(s) can be copied and used. This code is meant to make it is easier to get started using events on the microbit.

```
# 2.0 Event Structures
# by C Lyman
# July 2019
# Module 2 of Coding & Innovation using Microbits - Python
# Structures for different events using Microbits

from microbit import *

# forever loop for Events
while True:
     # Event - button A pressed?
     if button_a.is_pressed():
          # action when A is pressed
          display.show("A")

     # Event - button B pressed?
     if  button_b.is_pressed():
          # action when B is pressed
          display.show("B")

     # Event - buttons AB pressed?
     if  button_a.was_pressed() and button_b.was_pressed():
          # action when A&B are pressed
          display.scroll("AB")

     # Event - pin0 touched?
     if  pin0.is_touched():
          # action when pin0 & ground are touched
          display.show("0")

     # Event - pin1 touched?
     if  pin1.is_touched():
          # action when pin1 & ground are touched
          display.show("1")

     # Event - pin2 touched?
     if  pin2.is_touched():
          # action when pin2 & ground are touched
          display.show("2")

     # Event gesture face up
     faceUp = accelerometer.was_gesture("face up")
          if faceUp:
          display.scroll("UP")
```

```
# Event gesture face down
     faceDown = accelerometer.was_gesture("face down")
     if faceDown:
          display.scroll("DWN")

     # Event gesture shake
     shake = accelerometer.was_gesture("shake")
     if shake:
          display.scroll("SHK")

     # Event gesture up
     up = accelerometer.was_gesture("up")
     if up:
          display.scroll("^")

     # Event gesture down
     down = accelerometer.was_gesture("down")
     if down:
          display.show("v")

     # Event gesture right
     right = accelerometer.was_gesture("right")
     if right:
          display.show(">")

     # Event gesture left
     left = accelerometer.was_gesture("left")
     if left:
          display.show("<")

     # Event - freefall?
     freefall = accelerometer.was_gesture("freefall")
     if freefall:
          # action when microbit is in freefall
          display.scroll("FF")

     # Event - 3g?
     threeG = accelerometer.was_gesture("3g")
     if threeG:
          #6g & 8g are also options
          # action when microbit is accelerated at 3G
          display.scroll("3G")
```

## Sensors Code

There a many sensors that can be used as input from the microbit. Below are snippets of code that can be used to access the sensors using MicroPython. For the most part they return a numeric value that can be displayed on the LED screen or used in a calculation.

```
# 2.0 Sensors
# by C Lyman
# April 2019
# Module 2 of Coding & Innovation using Microbits - Python
# Code for different sensors using Microbits

from microbit import *

while True:
# Sensor code lines and examples with value stored in a variable

# Temperature in Celsius
# Code: temperature()
# Example:
    temp = temperature()

# Light level from the display 0-255
# Code: display.read_light_level()
# Example:
    light = display.read_light_level()

# Acceleration x - tilting left - right +
# 0 when flat facing up
# Code: accelerometer.get_x()
# Example:
    accelX = accelerometer.get_x()

# Acceleration y - tilting forward + back -
# 0 when flat facing up
# Code: accelerometer.get_y()
# Example:
    accelY = accelerometer.get_y()

# Acceleration z - moving up + down -
# -1024 when flat face up (Gravity acting downwards)
# 1024 when face down
# vigorous movement will get values +-2048
# Code: accelerometer.get_z()
# Example:
    accelZ = accelerometer.get_z()

# Acceleration all axes
# Code: accelerometer.get_values()
# Example:
    accelXYZ = accelerometer.get_values()
```

```
# Compass Calibrate
# The compass must be calibrated before it can be used.
# The microbit asks you to "tilt until the screen is filled".
# When that is completed a smiley face shows on the screen.
# Then the compass will work.
# Code: compass.calibrate()
# Example:
    compass.calibrate()

# Compass Heading
# Gives a compass degrees for the direction top of the microbit
# (away from the pins) is pointed. 0 or 360 North, 90 East,
# 180 South, and 270 West.
# Code: compass.heading()
# Example:
    compassHeading = compass.heading()

# Compass x gives a magnetic field strength reading in nano tesla
# Code: compass.get_x()
# Example:
    magnetismX = compass.get_x()

# Compass Strength gives an indication of the magnitude
# of the magnetic field strength around the device in nano tesla
# Code: compass.get_field_strength()
# Example:
    magnetismStrength = compass.get_field_strength()
```