# My Piano

# Table of Contents

## 1. Synopsis

This unit will teach students about procedures and playing music in an app. Students will create an app with piano keys that the user can press to play musical notes. The “**MyPiano**” unit emphasizes being iterative and incremental, as students will start with one note and then add more to their apps. Procedures, a key part of abstraction and modularization, will also be introduced. Students will write a procedure for playing a note and then use the same procedure for the other notes. Students will reuse the code for each subsequent key on the piano apps they create.

## 2. Learning Objectives

After completing this unit, students will be able to:

1. Code a piano app using the App Inventor Blocks Editor.
2. Apply the computational thinking practices of being incremental and iterative, developing and testing an app in stages.
3. Reuse code in an app.
4. Demonstrate abstraction by creating and implementing a procedure in App Inventor;
5. Test and debug an app using the MIT AI2 Companion.
6. Increase their positive perception of programming by creating a useful and fun app.

## 3. Mapping with the Computational Thinking Framework

These tables show the alignment of this unit with the intended learning outcomes for the computational thinking framework. The entries in the tables indicate the expected relevance of the unit to each outcome:

| | | |
|---|---|---|
| ✔✔✔ | : | High relevance |
| ✔✔ | : | Some relevance |
| ✔ | : | Low relevance |

### Computational Thinking Concepts

| Unit 2: My Piano App | | |
|---|---|---|
| 1. Sequences | ✔✔✔ | Ordering of blocks in the button click event blocks affects the result (setting source of sound file must happen before playing sound). |
| 2. Events | ✔✔✔ | Button click events correspond to different piano keys. |
| 3. Repetition | | |
| 4. Conditionals | | |
| 5. Parallelism | | |
| 6. Naming | ✔✔✔ | Naming of components is important, especially piano key buttons, for setting different wav filenames. |
| 7. Operators | | |
| 8. Manipulation of data and elementary data structure | | |

## Computational Thinking Practices

| **Unit 2: My Piano App** | | |
|---|---|---|
| 1. Reusing and remixing | ✔✔✔ | Students reuse and remix the code (playing sound) from the last unit; students also reuse and remix the code by copying/pasting blocks. |
| 2. Being incremental and iterative | ✔✔✔ | Students can add a note or two at a time, then stop and test to make sure their new notes work. They will iterate on the current app to make it more interesting by adding more notes. |
| 3. Abstracting and modularizing | | |
| 4. Testing and debugging | ✔✔✔ | Testing and debugging at each stage of creating the app is needed to make sure new code blocks work. |
| 5. Algorithmic thinking | | |

## Computational Thinking Perspectives

| L2U8.2: My Piano App | | |
|---|---|---|
| 1. Expressing | ✔✔ | Students express their creativity by creating a musical app. |
| 2. Connecting | ✔✔ | Students can relate this piano app to other similar programs they have used. Also, musical students can see the relation of programming to other interests they have. |
| 3. Questioning | ✔✔✔ | Students gain a deeper understanding of how apps are created by making their own apps. |
| 4. Computational identity | ✔✔ | By creating the app, students can see themselves as creators of technology. |
| 5. Digital empowerment | ✔✔✔ | Students feel empowered by creating an app they can share with friends and family. |

## 4. Mapping with the CSTA Standards

This table shows  alignment with the intended learning outcomes to the CSTA CS Standards. The entries in the tables indicate the expected relevance of the unit to each outcome:

| | | |
|---|---|---|
| ✓✓✓ | : | High relevance |
| ✓✓ | : | Some relevance |
| ✓ | : | Low relevance |

| | | |
|---|---|---|
| 2-CS-02 | Design projects that combine hardware and software components to collect and exchange data.<br>[C] CS: Hardware & Software [P] Creating (5.1) | Hardware (sounds) are incorporated into an app. |
| 2-AP-14 | Create procedures with parameters to organize code and make it easier to reuse.<br>[C] AP: Modularity [P] Abstraction (4.1, 4.3) | PlayNote procedure with note input is used. |
| 2-AP-17 | Incorporate existing code, media, and libraries into original programs, and give attribution.<br>[C] AP: Program Development [P] Abstraction (4.2), Creating (5.2), Communicating (7.3) | Students use template to build app. |
| 2-AP-18 | Systematically test and refine programs using a range of test cases.<br>[C] AP: Program Development [P] Testing (6.1) | Testing and debugging used at different stages to test procedure to play notes. |

## 5. Learning Prerequisites

Students should have experience with App Inventor, using the Designer and Blocks to create simple apps.

MIT APP INVENTOR

# 6. Lesson Plan ( 45 minutes x 4)

This unit consists of two 45 minutes lessons. Expanded teacher guides can be found in the appendix.

## Lesson 1

| Time | Activity |
|---|---|
| 5 min | **Introduction to MyPiano Project**<br>1. Ask how many students play musical instruments. Explain they will be creating their own piano apps.<br>2. Demonstrate the finished **MyPiano** app and tell students that they are going to make their own piano apps. |
| 10 min | **Introduction to Layout Components**<br>1. Demonstrate HorizontalArrangement and VerticalArrangement so students understand how they can determine the layout of an app. |
| 30 min | **Adding Components**<br>1. Show students how to login to App Inventor and import "**MyPiano_template**".<br>2. Show students the uploaded media files.<br>3. Ask students to add HorizontalArrangement, Buttons, and Label in the Designer and update their properties following the Student Guide: Part 1. |

## Lesson 2

| Time | Activity |
|---|---|
| 5 min | **Review and Introduction to Lesson 2**<br>1. Make sure all students have completed the layout for the app.<br>2. Explain that in this lesson, the students will start to code buttons to play musical notes. |
| 10 min | **Coding Activity**<br>1. Ask students if they  remember how to play a sound file from the HelloItsMe app.<br>2. Students follow along with the teacher to complete the code for the C and D buttons.<br>3. Ask students to stop there for discussion. Some may go ahead to complete the remaining buttons. |
| 20 min | **Introduction to Procedures**<br>1. Ask students about next steps. Most students should suggest copy/pasting the blocks for the remaining buttons.<br>2. Explain that they will be using a new type of block in App Inventor, a procedure.<br>3. Unplugged Activity: Do the “Making Pizza with a Procedure” with students to demonstrate procedures.<br>4. Demonstrate the Procedure code blocks. |
| 10 min | **Wrap-up**<br>1. Review the new concept of procedures. |

## Lesson 3

| Time | Activity |
|---|---|
| 5 min | **Review and Introduction to Lesson 3**<br>1. Review the concept of Procedures from Lesson 2.<br>2. Ask if any students used copy and paste to complete the code of the other notes. If so, what problems did they encounter? |
| 15 min | **Add PlayNote Procedure**<br>1. Ask students to identify what is the same in each block for the C and D buttons. They should identify everything except the NotePlayer.Source file and the note displayed.<br>2. Walk students through making the PlayNote procedure.<br>a. Demonstrate the join block to join text.<br>3. Test to make sure C and D notes play correctly.<br>4. Add code blocks for remaining notes and test. |
| 15 min | **Updating the Procedure**<br>1. Ask students to add a feature to the app to display all the notes played, instead of a single note. How would they change the code? (Response could be: to update the PlayNote procedure).<br>2. Emphasize the fact that students need only update the PlayNote procedure. Discuss why this is beneficial.<br>3. Students follow Student Guide Lesson 3 to add the feature to display all notes, and the Clear button. |
| 10 min | **Wrap-up**<br>1. Review the  new concept of procedures.<br>2. Have students complete the multiple choice questions. |

## Lesson 4

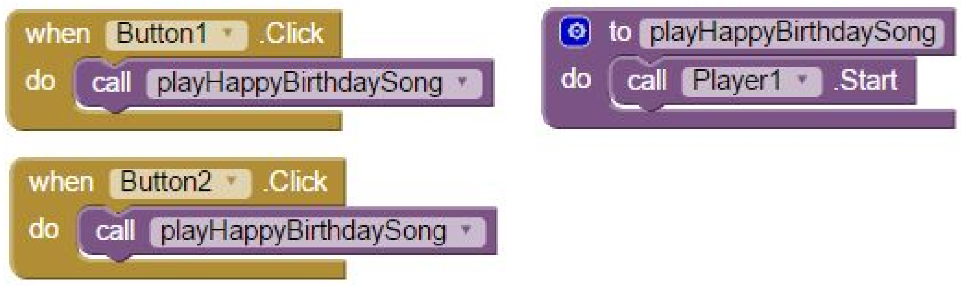
| Time | Activity |
|---|---|
| 5 min | **Introduction to Lesson 4**<br>1. Review what has been accomplished so far.<br>2. Review Procedures and their benefits.<br>3. Explain that students can work on completing the app if they haven't already, or they can use the lesson to add more notes or new features. |
| 30 min | **Add New Features**<br>Based on suggestions at the end of the Student Guide: Part 3, students can add more notes. Note files are included in the template for the sharp notes. |
| 10 min | **Sharing**<br>Any students who have added new features may share them with the class. |

# 7. Assessment

**Multiple-choice questions**

Multiple-choice questions assess key concepts of the activity as shown below:

1. If Player1 plays a 30-second long Happy Birthday song, what would happen if you press Button1 and then press Button2 1 second later, using the following blocks?



```
when Button1 .Click
do   call playHappyBirthdaySong

when Button2 .Click
do   call playHappyBirthdaySong

to playHappyBirthdaySong
do   call Player1 .Start
```

A. Player1 starts playing a Happy Birthday song. When Button2 is pressed, the song continues to play.

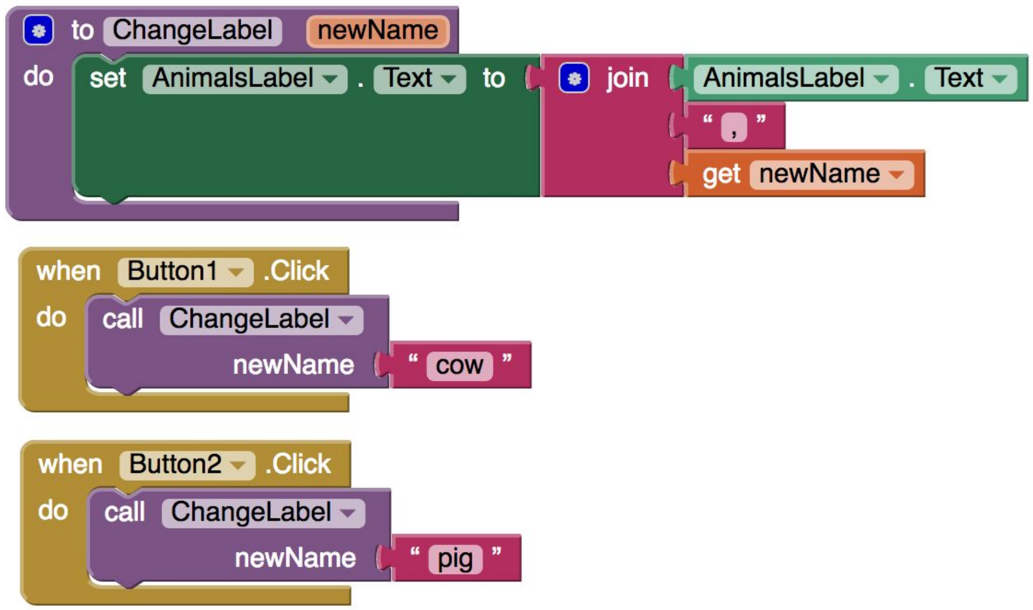
B. Player1 starts playing a Happy Birthday song. When Button2 is pressed, the song stops playing.

C. Player1 starts playing a Happy Birthday song. When Button2 is pressed, the song continues to play but Button2 plays another copy of the song from the beginning so it sounds like two people are singing with a 1-second delay.

D. Player1 starts playing a Happy Birthday song. When Button2 is pressed, the song stops playing and starts over from the beginning.

(Answer: **D**)

2. Using the following code blocks, and assuming that AnimalLabel.Text starts off blank, what will the text be on AnimalLabel after Button1 is pressed, and then Button2 is pressed?



```
to ChangeLabel newName
do  set AnimalsLabel . Text to  join  AnimalsLabel . Text
                                      " , "
                                      get newName

when Button1 .Click
do  call ChangeLabel
             newName  " cow "

when Button2 .Click
do  call ChangeLabel
             newName  " pig "
```

A. cow

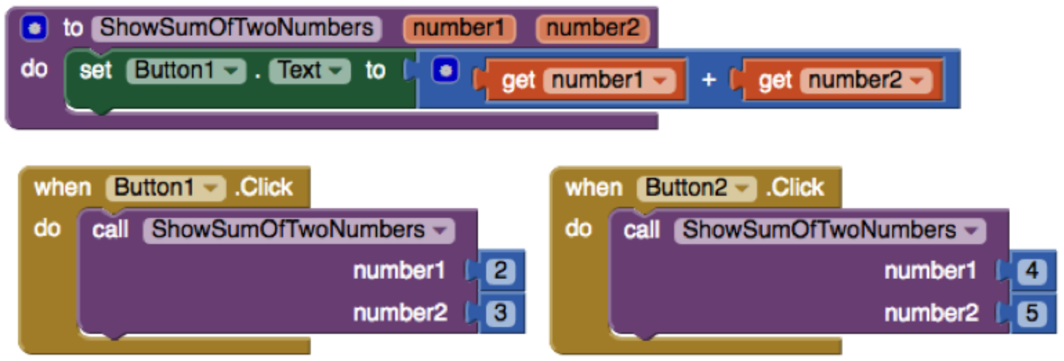
B. pig

C. cow,pig

D. ,cow,pig

(Answer: **D**)

3. What happens when Button1 is pressed, and then Button2 is pressed?



```
to ShowSumOfTwoNumbers number1 number2
do  set Button1 . Text to  get number1 + get number2

when Button1 .Click
do  call ShowSumOfTwoNumbers
                      number1  2
                      number2  3

when Button2 .Click
do  call ShowSumOfTwoNumbers
                      number1  4
                      number2  5
```

A. The text on Button1 changes to 5. Button 2 does not change.

B. The text on Button2 changes to 9. Button1 does not change.

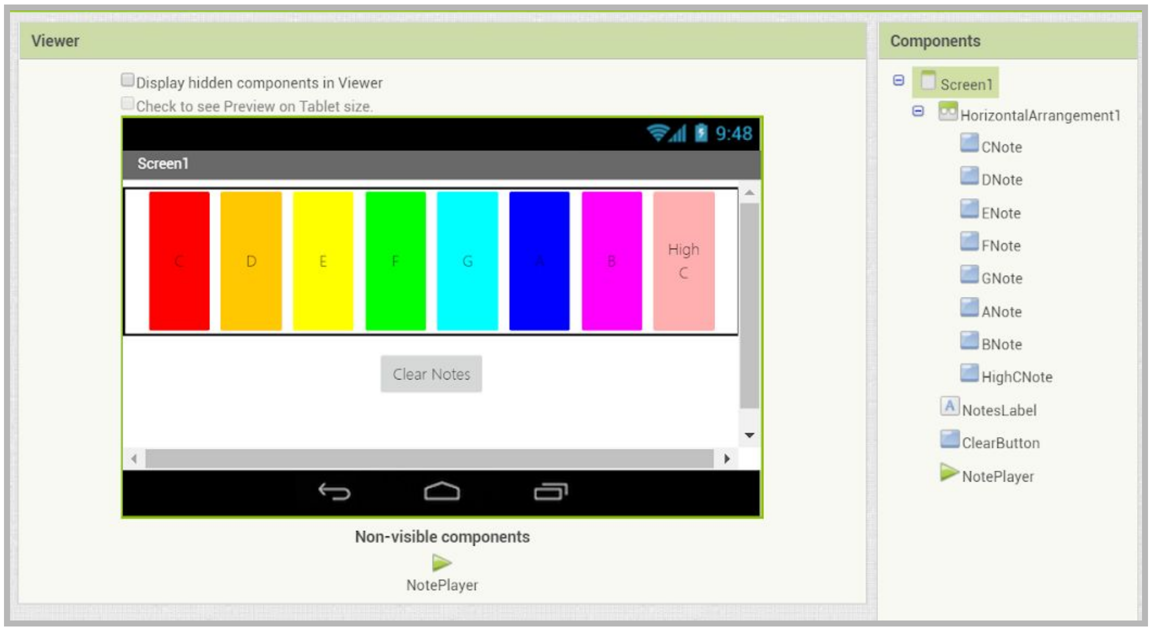
C. The text on Button1 changes to 5 and then it changes to 9. Button2 does not change.

D. The text on Button1 changes 5 and the text on Button2 changes to 9.

(Answer: **C**)

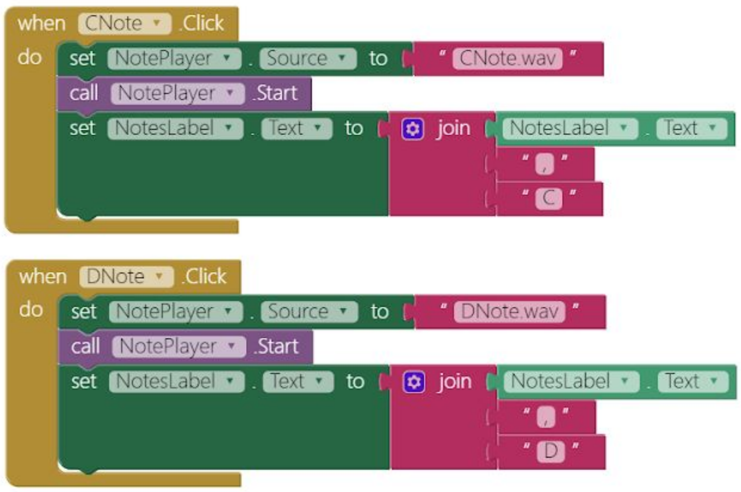
# 8. Screen Design and Code

## Designer



## Blocks Editor for Lesson 2

Only C and D notes will play.

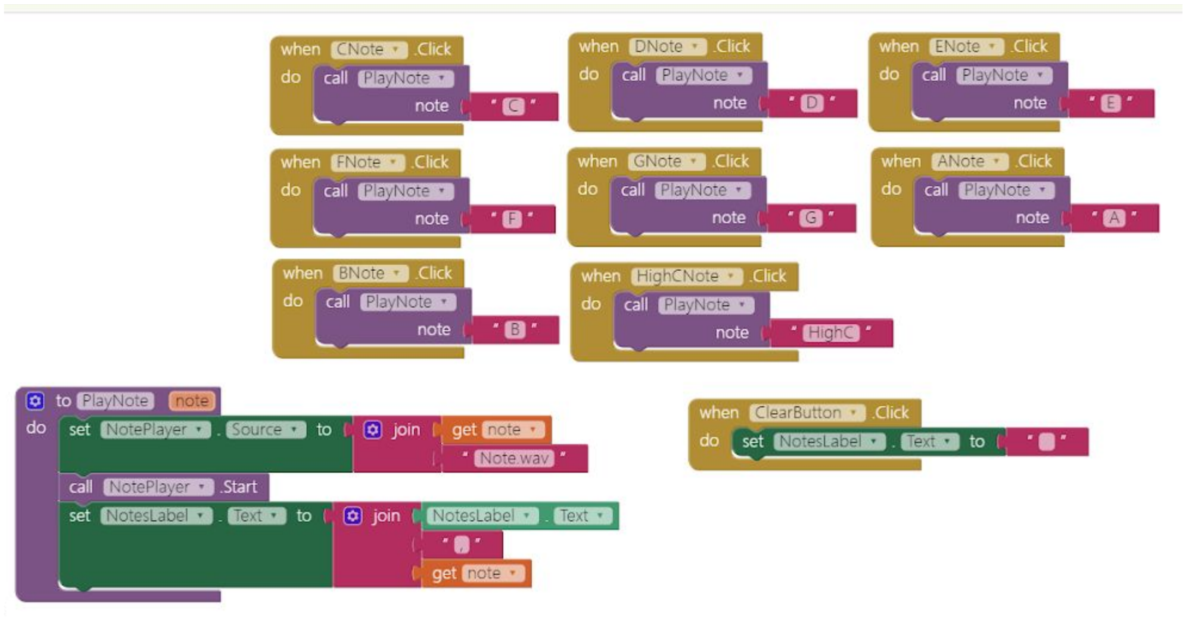


```
when CNote.Click
do  set NotePlayer.Source to "CNote.wav"
    call NotePlayer.Start
    set NotesLabel.Text to join NotesLabel.Text
                                ","
                                "C"

when DNote.Click
do  set NotePlayer.Source to "DNote.wav"
    call NotePlayer.Start
    set NotesLabel.Text to join NotesLabel.Text
                                ","
                                "D"
```

## Blocks for lesson 3

All  eight notes will sound. Use the **PlayNote** procedure to manage the blocks, and call it with a different parameter when each key is clicked.



```
when CNote.Click
do  call PlayNote
         note "C"

when DNote.Click
do  call PlayNote
         note "D"

when ENote.Click
do  call PlayNote
         note "E"

when FNote.Click
do  call PlayNote
         note "F"

when GNote.Click
do  call PlayNote
         note "G"

when ANote.Click
do  call PlayNote
         note "A"

when BNote.Click
do  call PlayNote
         note "B"

when HighCNote.Click
do  call PlayNote
         note "HighC"

to PlayNote note
do  set NotePlayer.Source to join get note
                                  "Note.wav"
    call NotePlayer.Start
    set NotesLabel.Text to join NotesLabel.Text
                                ","
                                get note

when ClearButton.Click
do  set NotesLabel.Text to ""
```

# Appendix 1
# Teacher's Guide: Lesson 1

## Learning Objectives

At the end of this lesson, students should be able to:

1. Create the user interface for the MyPiano app.
2. Use Layout components to organize user interface components within an app.
3. Be incremental and iterative by building an app in stages.

## Lesson Outline

**Introduction to MyPiano Project (10 minutes)**

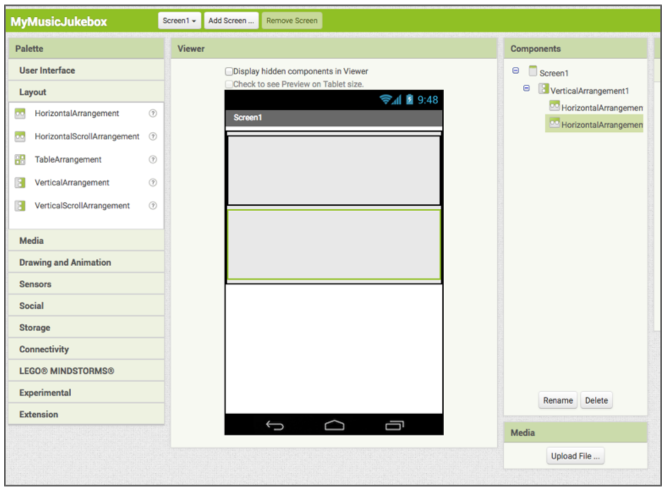
To get the students excited about the project, tie the project to music and playing a musical instrument.

1. Ask how many students play a musical instrument. Explain they will be creating their own piano apps.
2. Demonstrate the finished **MyPiano** app and tell students that they are going to make their own piano apps.

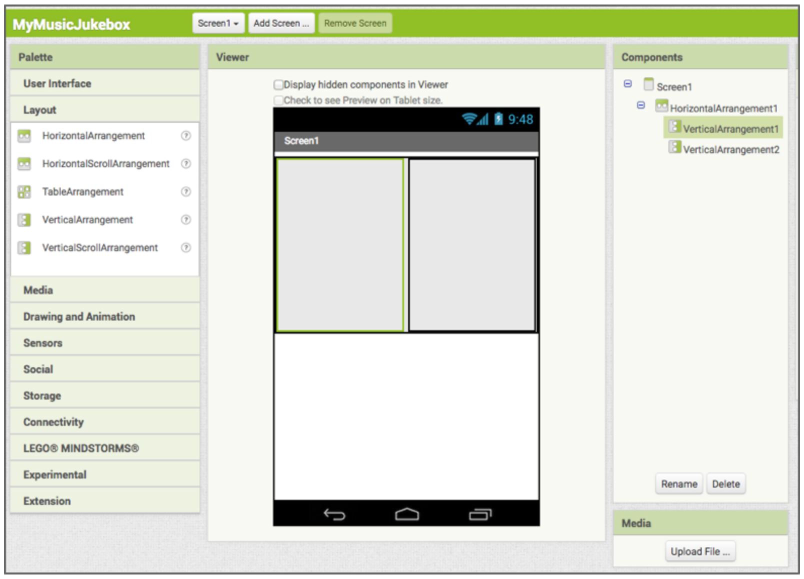
## Introduction to Layout Components (10 minutes)

1. Demonstrate HorizontalArrangements and VerticalArrangements so students understand how they can help to determine the layout of an app. Components inside VerticalArrangements are placed below each other. That is also the default behavior in App Inventor. Components placed inside HorizontalArrangements appear side by side.

   The example below is a VerticalArrangement with 2 HorizontalArrangements inside.



   The example below is a HorizontalArrangement with 2 VerticalArrangements inside.



**Adding Components (30 minutes)**

In this section, the teacher will lead students through creating the basic app, based on the project template. The template includes all of the media files for the project. Students will add all the necessary components.

1. Log in to App Inventor and ask students to import "MyPiano_template". Remind them how to do this if needed.
2. Go over the uploaded media files.
3. Ask students to add the HorizontalArrangement, Buttons, and Label in the Designer and update their properties following the Student Guide: Part 1. They can also follow the Youtube video guide (up to 5:38).
4. Point out to the students that there are many buttons, but there is only one Player component for playing notes. Explain that since only one note plays at a time, they only need one Player component.

# Appendix 2
# Teacher's Guide: Lesson 2

## Learning Objectives

At the end of this lesson, students should be able to:

1. Code buttons to play notes in an app.
2. Duplicate code blocks in an app.
3. Be incremental and iterative in building an app in stages.

**Review and Introduction to Lesson 2 (5 minutes)**

1. Make sure all students have completed the layout for the app.
2. Explain that in this lesson, the students will start to code the buttons to play musical notes. You will go through coding the C and D buttons, and then ask students to stop before proceeding further.

**Coding Activity (20 minutes)**

1. Ask students if they remember how to make a button play a sound from the HelloItsMe app.
2. Students can either follow the Student Guide: Part 2, or follow the Youtube video (5:38-8:57). Or, the teacher can lead the class through the steps below.
    1. Drag the CNote.Click and NotePlayer.Start blocks out and ask students to test if it works. They should note that no sound plays. Ask students if they can figure out what is wrong. The problem is the Source for the Player has not been set.

2. Show students how to set the NotePlayer.Source using code blocks.
3. Have students add code blocks for the D button. Demonstrate how to duplicate blocks in App Inventor.
4. Ask students if they have to change any of the duplicate code. They should note that the Source must be changed and the Text block for the note.
5. Test and Debug.

Explain the idea of being incremental and iterative. Students can add a note or two at a time, then stop and test to make sure their new notes work. They will iterate on the current app to make it more interesting by adding more notes. It is good practice to add a little, stop and test to make sure new additions work, then proceed.

1. Why might it be better to do a little coding, stop and test, then do a little more?

2. What problems might you have if you code your entire app and find it doesn't work correctly?

**Introduction to Procedures**

1. Ask students what would be next steps. Most students should say copy/pasting the blocks for the remaining buttons.
2. Explain that instead they will be using a new type of block in App Inventor, a procedure.
3. Unplugged Activity: Do the "Making Pizza with a Procedure" with students to demonstrate procedures.

**Unplugged Activity**

The unplugged activity is another way of explaining to students what a procedure is.

*Make a Pizza with a Procedure*

Requires setup: cover a book with paper and a title "Pizza Cookbook", and insert the 4 pizza recipes from the "Make Pizza with a Procedure Instructions" document in different places in the book.

1. Hold up a recipe book. Say this is a pizza recipe book. Inside are all the recipes for pizza.
2. Ask a student to read the recipe for a mushroom pizza. From the book, pull out a piece of paper with the mushroom pizza recipe (or have the recipes taped into the book so they read from the book).
3. Ask another student to read the recipe for a pepperoni pizza. Make a show of leafing through the book to find the new recipe. Hand it to the student to read.
4. Do the same for pepper and anchovy pizzas with two more students.
5. Ask the class if they notice anything about the recipes. They should respond that they are all the same, except for the topping ingredient.
6. Explain to the class that instead of having this large cookbook with many recipes for pizza, we're going to replace it with one sheet of paper with a recipe that uses the word "ingredient" to specify the topping.
7. State that after making some pizzas, you have decided the pizza needs 7 minutes to cook properly instead of 5. Using the cookbook, how many places would they have to change the recipe? They should answer - 100, or as many pizza recipes as are in the book. Hold up the single recipe sheet and talk about the other benefit of using one generic recipe. You only have to update one recipe, not many.

4. Talk about the Procedure blocks and demonstrate the Procedure code blocks.

5. Explain that Procedures are often used to organize code into blocks, according to a particular task or function. In this case, the procedure will be playing a note. It is also useful if similar code is being used in multiple places. In this case, we have 8 code blocks that are similar, one for each note button. In the next lesson, students will implement a procedure for MyPiano.

# Appendix 3
# Teacher's Guide: Lesson 3

## Learning Objectives

At the end of this lesson, students should be able to:

1. Be incremental and iterative by adding code bit by bit to their app.
2. Use abstraction and modularization by using a procedure in an app.
3. Test and debug an app using MIT AI2 Companion.

## Lesson Outline

**Review and Introduction to Lesson 3 (5 minutes)**

1. Review the concept of Procedures from Lesson 2.
2. Ask if any students used copy and paste to complete the code of the other notes. Ask these students what problems they encountered.
2. Explain again to students that they will be using a new type of block in App Inventor, a procedure. A procedure helps you by putting code all in one place. Instead of copying and pasting the code, you can just call a procedure to run the code. That way, you just keep one copy instead of many. Relate back to the Making a Pizza with a Procedure activity.

**Addition of PlayNote Procedure (15 minutes)**

The teacher will lead students through the process of making a procedure to replace the need to duplicate code. Alternatively, students can follow Student Guide: Lesson 3, or follow the Youtube Video (from 8:57)

1. Ask students to identify what is the same in eachblock for the C and D buttons. They should identify everything except the Player.Source file and the note displayed.
2. Explain that students will use a new block called a Procedure. Procedures help by putting code blocks that appear in many places in a single place. Then you "call" it to place that code where you want it. Now make a procedure to play a note.
3. Walk students through making the PlayNote procedure.
    a. Demonstrate the join block to join text for the NotePlayer.Source file.
4. Test to make sure C and D notes play correctly.
5. Add code blocks for remaining notes and test.

**Updating the Procedure (15 minutes)**

Students will change the code blocks for the PlayNote procedure to display all notes played in a string. This exercise should help to reinforce the idea that using a single block of code in a procedure is an easier way to organize and maintain code in an app.

1. Ask students to add a feature to the app to display all notes played, instead of a single note. How would they change the code? (Students should respond that they need to update the PlayNote procedure).
2. Emphasize the fact that students need only update the PlayNote procedure. Without a procedure they would have to change 8 blocks of code. Discuss why this is beneficial.

3. Ask students to follow Student Guide Lesson 2 to add the display the feature to play all notes, and the Clear button.

**Wrap-Up (10 minutes)**

1. Review the  new concept of procedures.
2. Have students complete the multiple choice questions.

# Appendix 4
# Teacher's Guide: Lesson 4

## Learning Objectives

At the end of this lesson, students should be able to:

1. Add an enhancement to their app.
2. Test and debug their apps using the MIT AI2 Companion app and live testing.
3. Share their work with their peers.

## Lesson Outline

### Introduction to Lesson 4 (5 minutes)

1. Review what has been accomplished so far.
2. Review Procedures and their benefits.
3. Explain that students can work on completing the app if they haven't already, or they can use the lesson to add more notes or new features.

### Add New Features (30 minutes)

Students may continue to work on the app if they have not yet finished.
Based on suggestions at the end of the Student Guide: Part 3, students can add more notes. Note files are included in the template for the sharp notes.

**Sharing (10 minutes)**

Any students who have added new features may share with the class.