江西理工大学 信息工程学院
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY SCHOOL OF INFORMATION ENGINEERING

Mobile application development

移动应用开发

Lecture 012:

APP Inventor _Example

Review the example and introduce some blocks

Dr Ata Jahangir Moshayedi

Prof Associate ,
School of information engineering Jiangxi university of science and technology, China

EMAIL: ajm@jxust.edu.cn

Autumn _2021

江西理工大學 信息工程学院

JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY SCHOOL OF INFORMATION ENGINEERING

MOBILE APPLICATION DEVELOPMENT

LECTURE 012: APP Inventor _Example

Review the example and introduce some blocks

App Inventor

江西理工大學

JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Agenda

- **Review the example and introduce LIST blocks**
  - **MIT App oInventor color Blocks**
  - **MIT App oInventor procedure Blocks**
  - **Example 4**

APP inventor _ Color block

There are three main types of color blocks:

- a color box
- make color
- Split color

MIT App Inventor Color Blocks

How do colors work in App Inventor?

- Internally, App Inventor stores each color as a single number.
- When you use make color and take in a list as an argument, internally this list is then converted using App Inventor's color scheme and stored as a number.
- If you knew the numbers for the colors, you could even specify what color you wanted something to be by just setting its Color property to a specific number.

This chart gives the App Inventor numeric codes for a variety of colors. (With thanks to James Laroche)

Color Name	Color	App Inventor Color Number
AliceBlue		-984833
AntiqueWhite		-332841
Aqua		-16711681
Aquamarine		-8388652
Azure		-983041
Beige		-657966
Bisque		-6972
BlanchedAlmond		-5171
BlueViolet		-7722014
Brown		-5952982
BurlyWood		-2180985
CadetBlue		-10510688
Chartreuse		-8388864
Chocolate		-2987746
Coral		-32944
CornflowerBlue		-10185235
Cornsilk		-1828
Crimson		-2354116
DarkBlue		-16777077
DarkCyan		-16741502
DarkGoldenRod		-4684277
DarkGreen		-16751616
DarkKhaki		-4343957
DarkMagenta		-76675773
DarkOliveGreen		-11179217
Darkorange		-29696
DarkOrchid		-6737204
DarkRed		-7667712
DarkSalmon		-1468806
DarkSeaGreen		-7357297
DarkSlateBlue		-12042869
DarkSlateGray		-13676721
DarkTurquoise		-16724271
DarkViolet		-7077677

MIT App Inventor Color Blocks

basic color 

- This is a basic color block. It has a small square shape and has a color in the middle that represents the color stored internally in this block.

- If you click on the color in the middle, a pop-up appears on the screen with a table of 70 colors that you can choose from. Clicking on a new color will change the current color of your basic color block.

- Each basic color block that you drag from the Colors drawer to the Blocks Editor screen will display a table with the same colors when clicked.

MIT App Inventor Color Blocks

make color

make color takes in a list of 3 or 4 numbers. These numbers in this list represent values in an RGB code. RGB codes are used to make colors on the Internet.

- An RGB color chart is available here.
  - This first number in this list represents the R value of the code.
  - The second represents the G. The third represents the B. The fourth value is optional and represents the alpha value or how saturated the color is.
  - The default alpha value is 100. Experiment with different values and see how the colors change using this block.

split color

split color does the opposite of make color.

It takes in a color:

a color block, variable holding a color, or property from one of the components representing a color and returns a list of the RGB values in that color's RGB code.

MIT App Inventor Procedure Blocks

- An App Inventor procedure collects a sequence of blocks together into a group.
- You can then use the sequence of blocks repeatedly by calling the procedure.
- If the procedure has arguments, you specify the arguments by using name blocks.
  - When you create a procedure, App Inventor automatically generates a call block and places it in the My Definitions drawer.
  - You use the call block to invoke the procedure. move the mole to a random location on the screen.

What is a procedure?

- A procedure is a set of instructions that perform a specific task or tasks. It may also be called a function.
- **A recipe for banana bread is an example of a procedure. The baker must follow the instructions step-by-step to produce the bread.**
- We use procedures in App Inventor to create new blocks that we can use repeatedly and take up less space than all of the blocks used in the original procedure.
- If we are using the same sets of blocks more than once, these blocks are called redundant.

MIT App Inventor Procedure Blocks

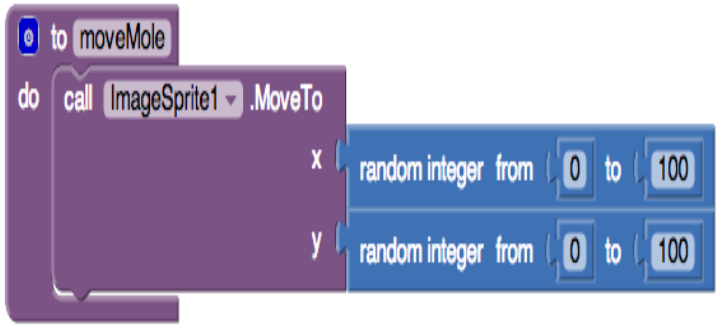
- **How its work in APP inventor**

When you create a new procedure block, App Inventor chooses a unique name automatically. You can click on the name and type to change it.

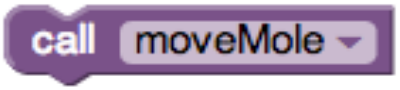
- **Some notice:**
  - Procedure names in an app must be unique.
  - App Inventor will not let you define two procedures in the same app with the same name.
  - You can rename a procedure at any time while you are building the app, by changing the label in the block.
  - App Inventor will automatically rename the associated call blocks to match.

MIT App Inventor Procedure Blocks

- **How to start ?**
  - Lets build a procedure to do the job of the redundant code blocks.
  - In App Inventor, you define a procedure in a manner similar to how you define variables.
  - From the Procedures drawer, drag out either a procedure do block or a to procedure return block.
  - Use the latter if your procedure should calculate some value and return it. After dragging out a procedure block, you can change its name from the default procedure by clicking the word procedure and typing the new name.



After you create a procedure, a block is put in the Built-In Procedures drawer that lets you call your procedure.

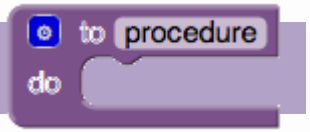


MIT App Inventor Procedure Blocks

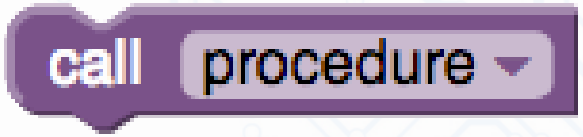
- A procedure is a sequence of blocks or code that is stored under a name, the name of your procedure block.
- Instead of having to keep putting together the same long sequence of blocks, you can create a procedure and just call the procedure block whenever you want your sequence of blocks to run.
- **In computer science, a procedure also might be called a function or a method.**
  - **procedure do**
  - **procedure result**

MIT App Inventor Procedure Blocks

procedure do

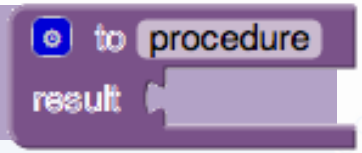


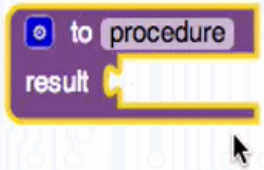
- Collects a sequence of blocks together into a group.
- You can then use the sequence of blocks repeatedly by calling the procedure.
  - If the procedure has arguments, you specify the arguments by using the block's mutator button.
  - If you click the blue plus sign, you can drag additional arguments into the procedure.
- When you create a new procedure block, App Inventor chooses a unique name automatically. Click on the name and type to change it. Procedure names in an app must be unique.
  - App Inventor will not let you define two procedures on the same screen with the same name.
  - You can rename a procedure at any time while you are building the app, by changing the label in the block.
  - App Inventor will automatically rename the associated call blocks to match
  - Remember Java kewords cannot be used as procedure names.
- When you create a procedure, App Inventor automatically generates a call block and places it in the Procedures drawer.
- You use the call block to invoke the procedure.



MIT App Inventor Procedure Blocks

procedure result





What is an argument?

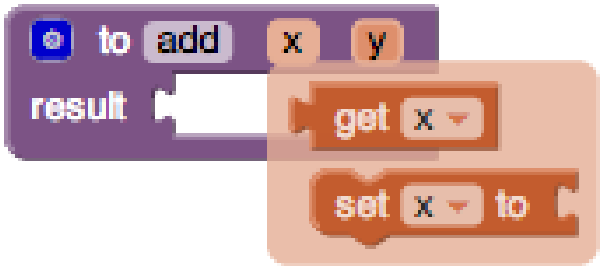
An argument is an input to our procedure. Some procedures require knowing some bits of information that change how the procedure is run.
When you create a procedure, you can use the mutator button to add arguments.

Same as a procedure do block, but calling this procedure returns a result.

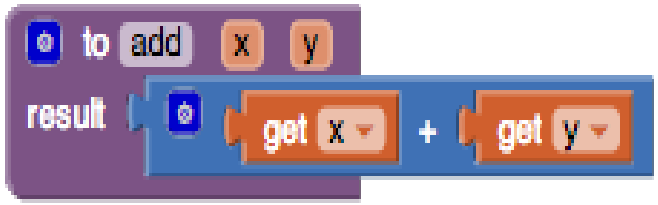
After creating this procedure, a call block that needs to be plugged in will be created. This is because the result from executing this procedure will be returned in that call block and the value will be passed on to whatever block is connected to the plug.

MIT App Inventor Procedure Blocks

- *appear. Drag these blocks onto your screen to use them.*



This procedure takes two arguments: x and y. Then it returns the result of adding x to



Summary

- A procedure is a set of blocks together in a group that can be run by calling the procedure. A procedure can return something or not have a return value.
- A procedure can take arguments.

江西理工大學信息工程学院

JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY SCHOOL OF INFORMATION ENGINEERING

MOBILE APPLICATION DEVELOPMENT

APP Inventor _Example

Review the example and introduce some blocks

江西理工大學信息工程学院

JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY SCHOOL OF INFORMATION ENGINEERING

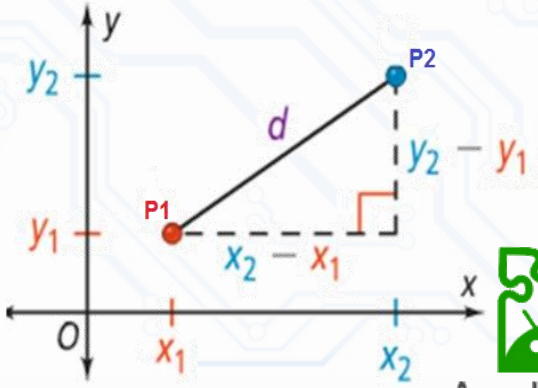
App Inventor

Example:
Distance between two points in the plane.

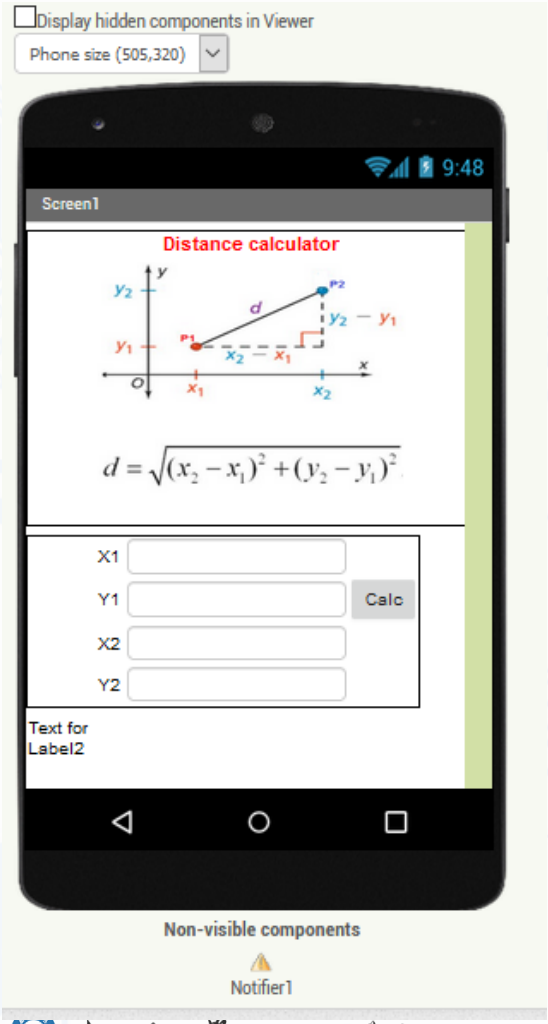
Aim:

- This we will see is a **simple example** of **Procedure**, could enter operations in the same block of Button1 and we would save the block of Procedure, it is simply a didactic example.
  - Normally **Procedures** are used when a part of the code (eg operations) **are needed in several blocks,** imagine we have **8 - Button** and each of them have to perform the operations, it would be easier to create a **common block** of operations through the **procedure** and call it from the various buttons, put all operations on all buttons.
  - This is to introduce four data corresponding to the coordinates of two points **P1(x1, y1)** and **P2(x2, y2)**, we call a **Procedure** called **distance** and we will result the distance between those two points that are in the same plane.
- The expression for calculation is:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



Distance calculator Designer section



@Screen 1

Layout arrangement

Image*2

Label*1

Layout arrangement

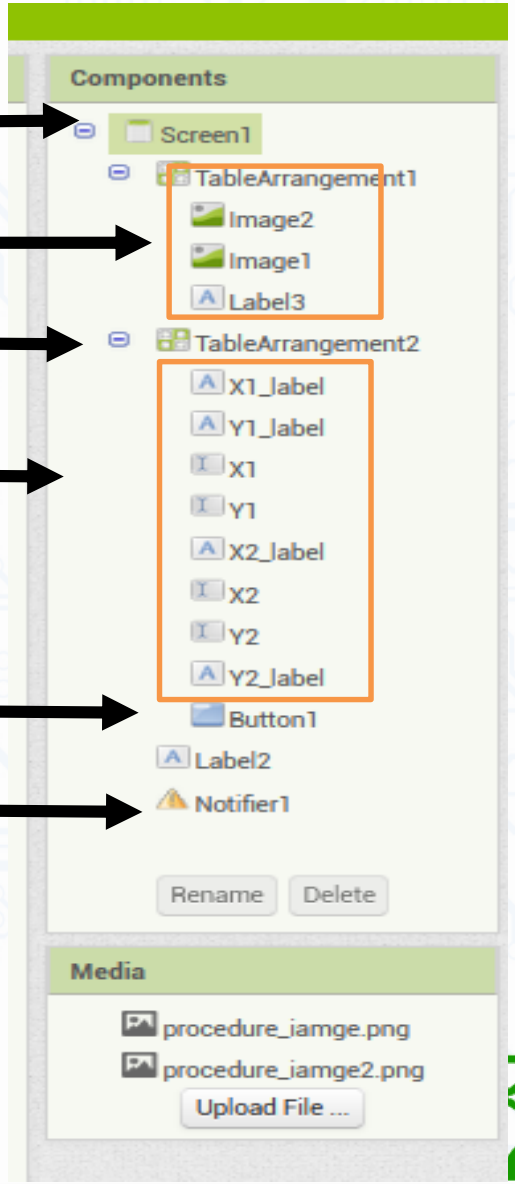
Text box *4

Label *4

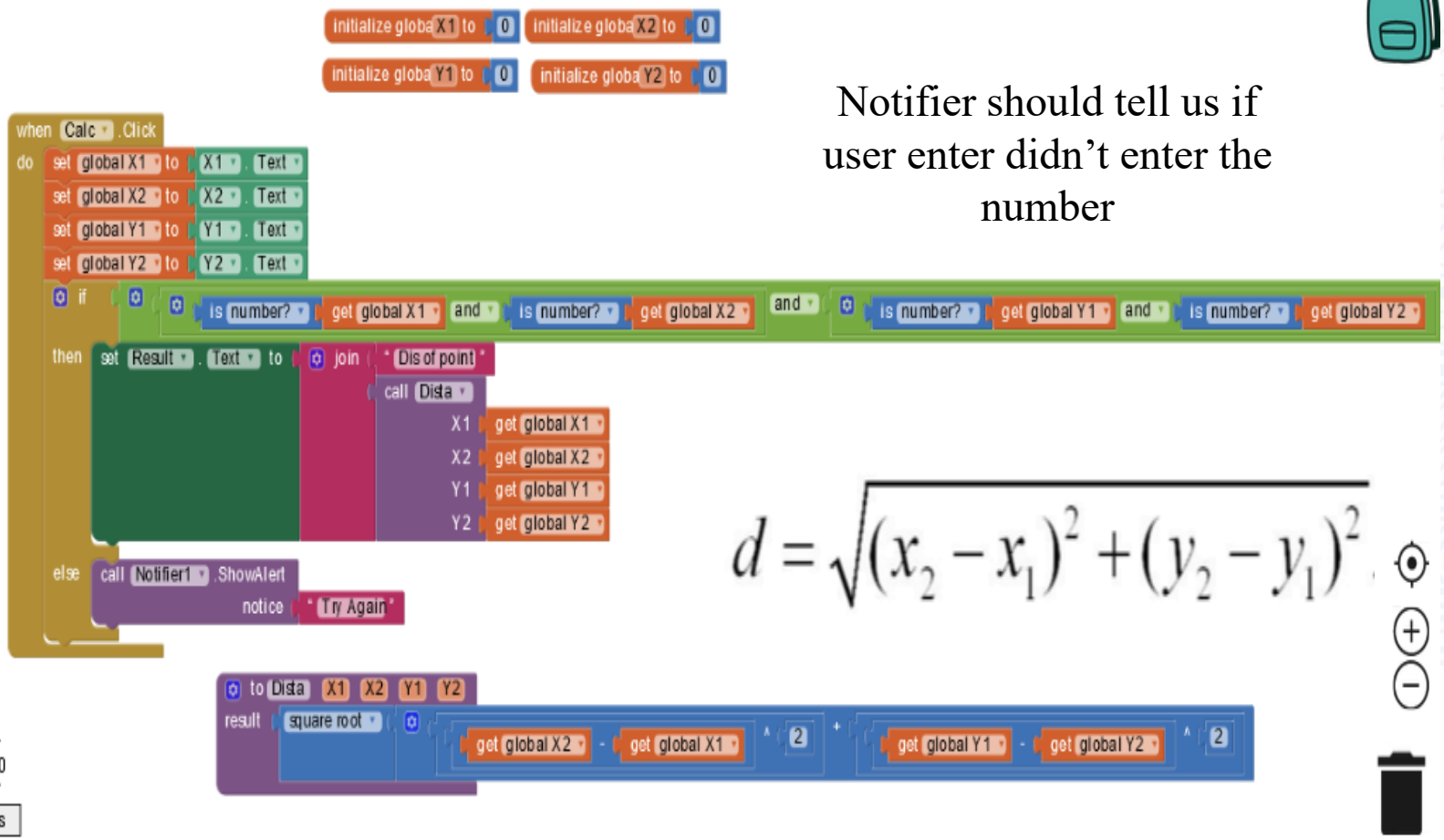
Buttton*1

Label*1

Notification section



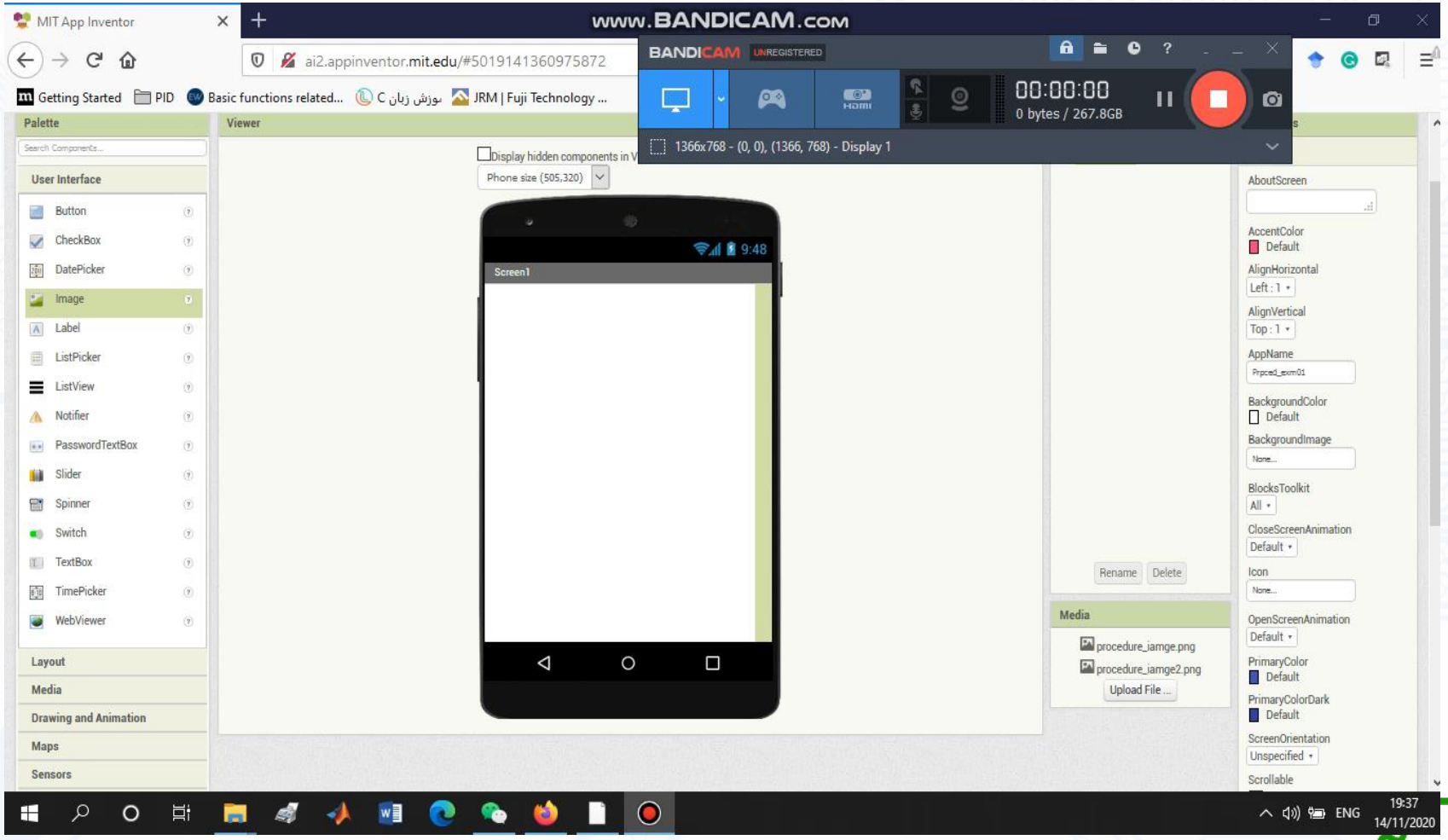
After solving the problem



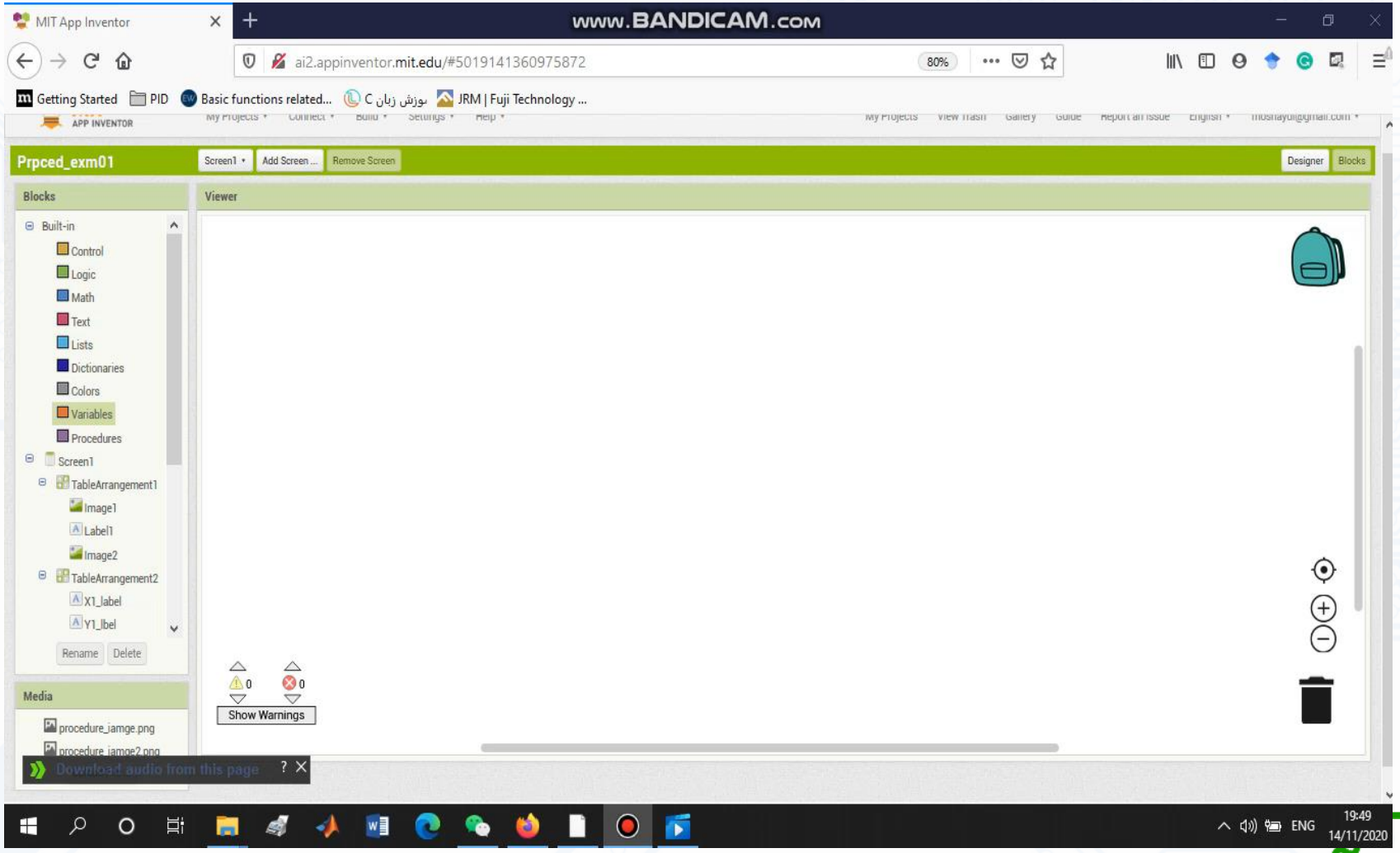
Notifier should tell us if user enter didn't enter the number

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

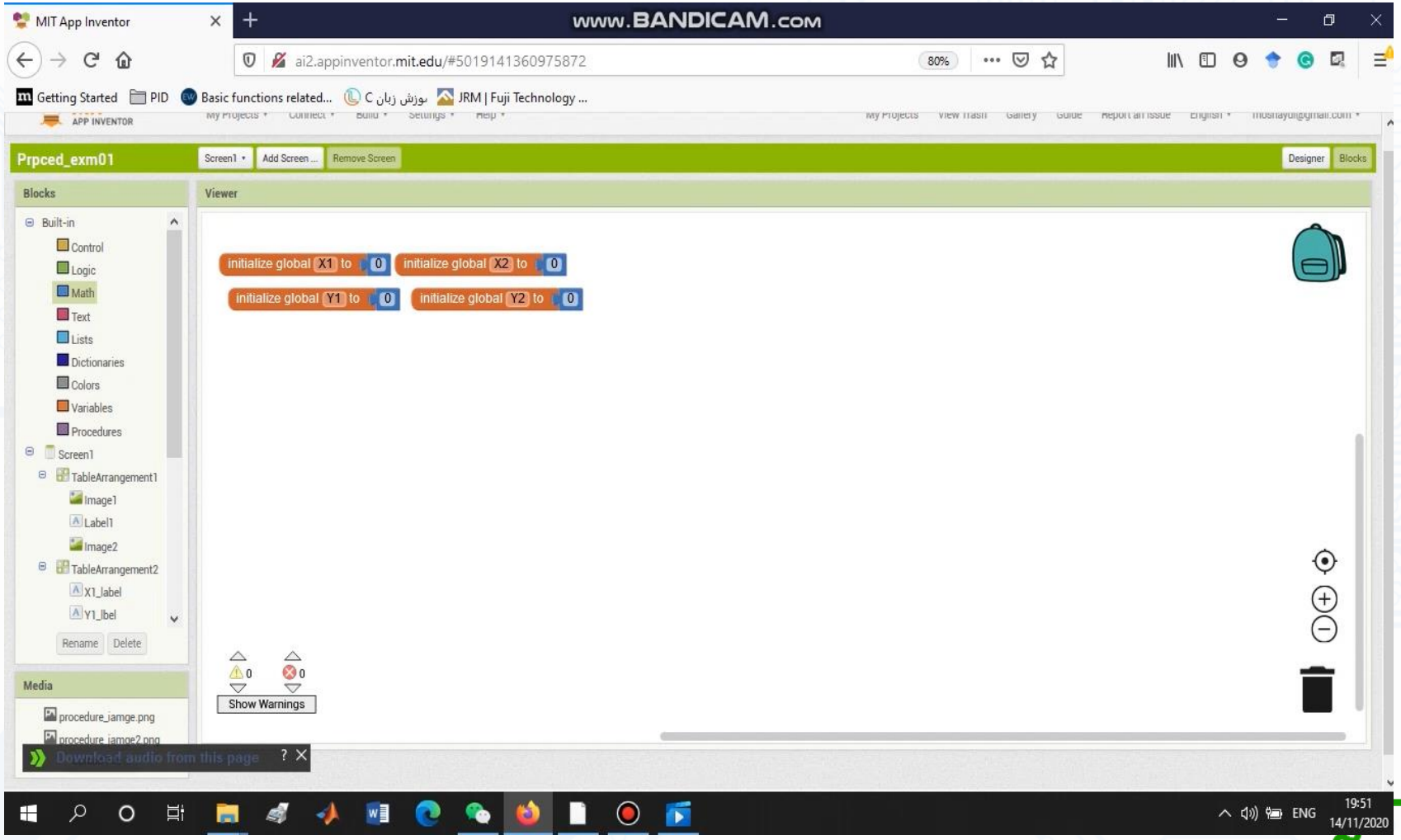
Demo and process: Designer part



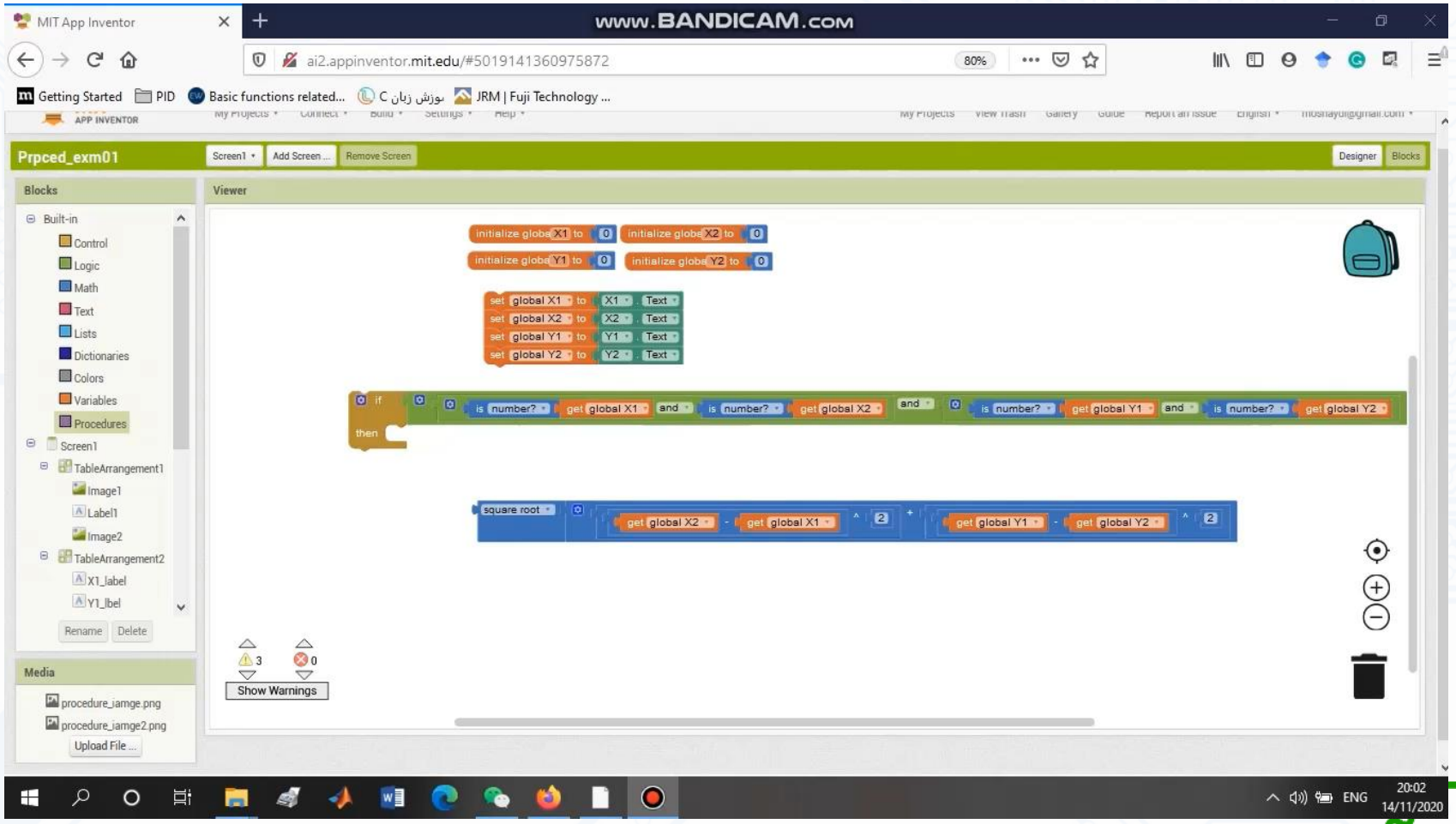
Demo and process: Block part



Demo and process: Block

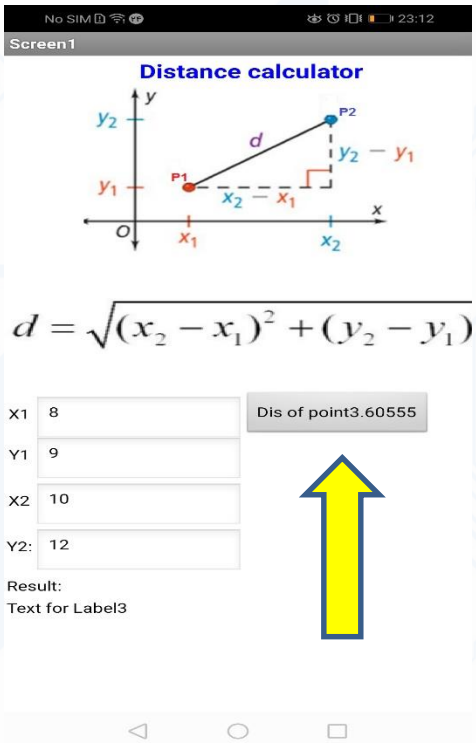


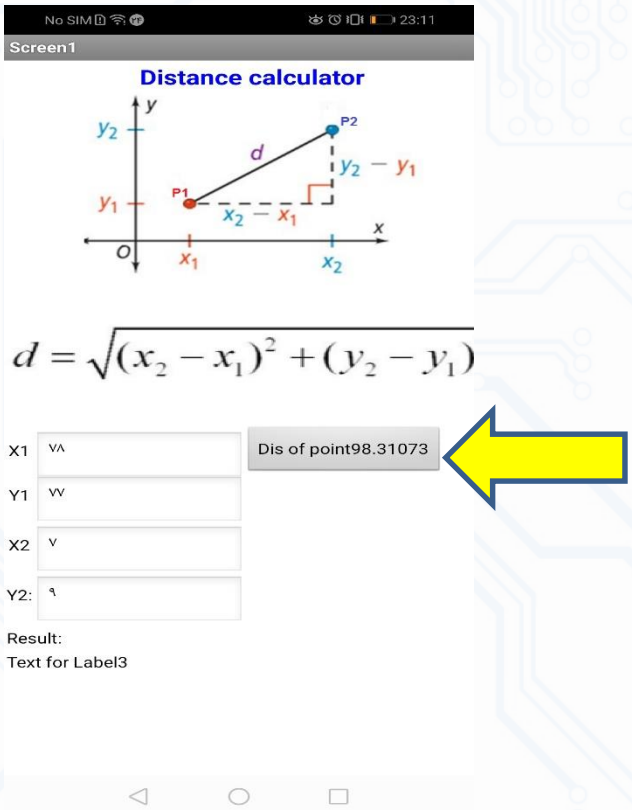
Demo and process: Block part



Demo and process: Block

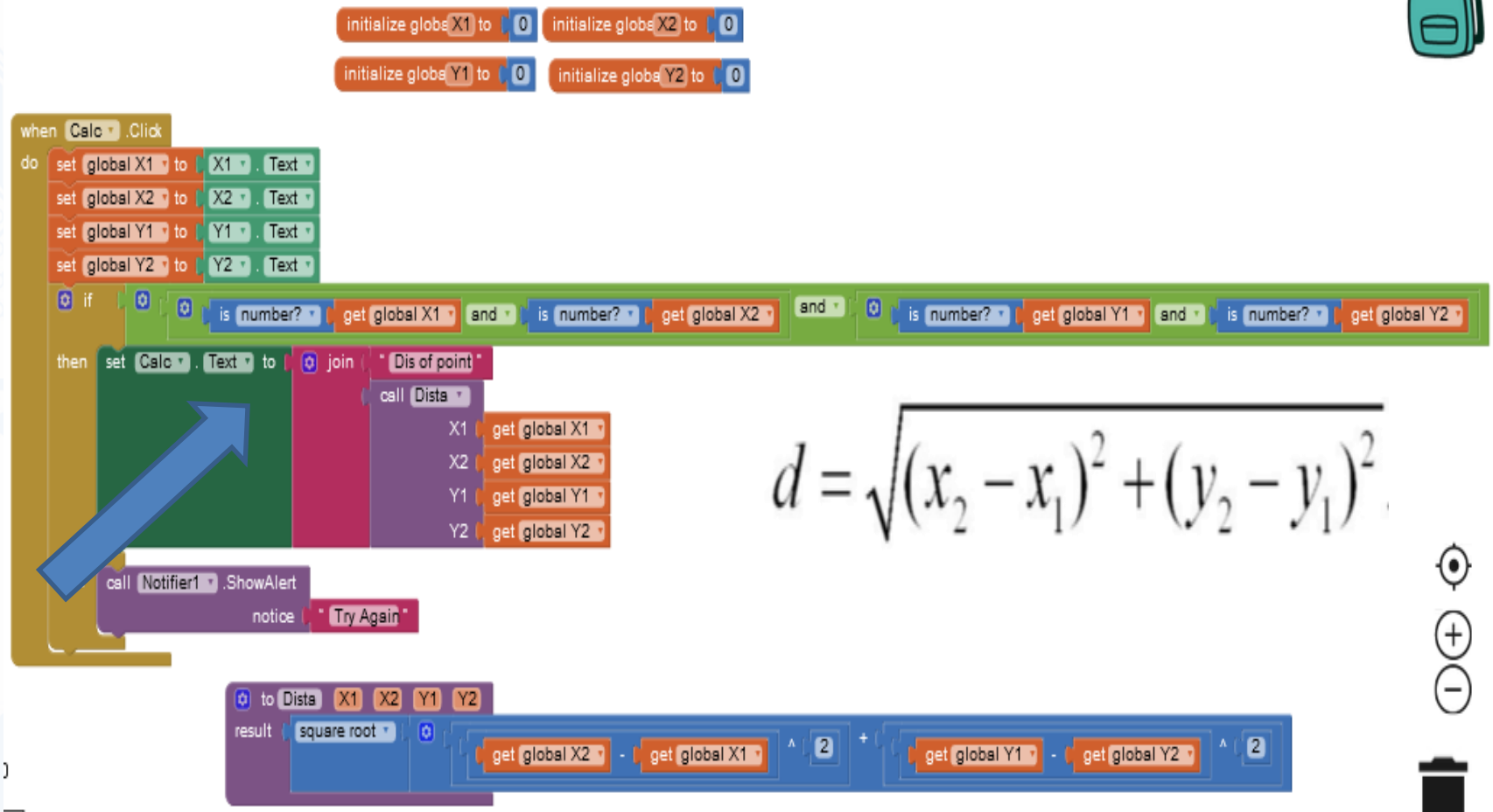
- There is one problem





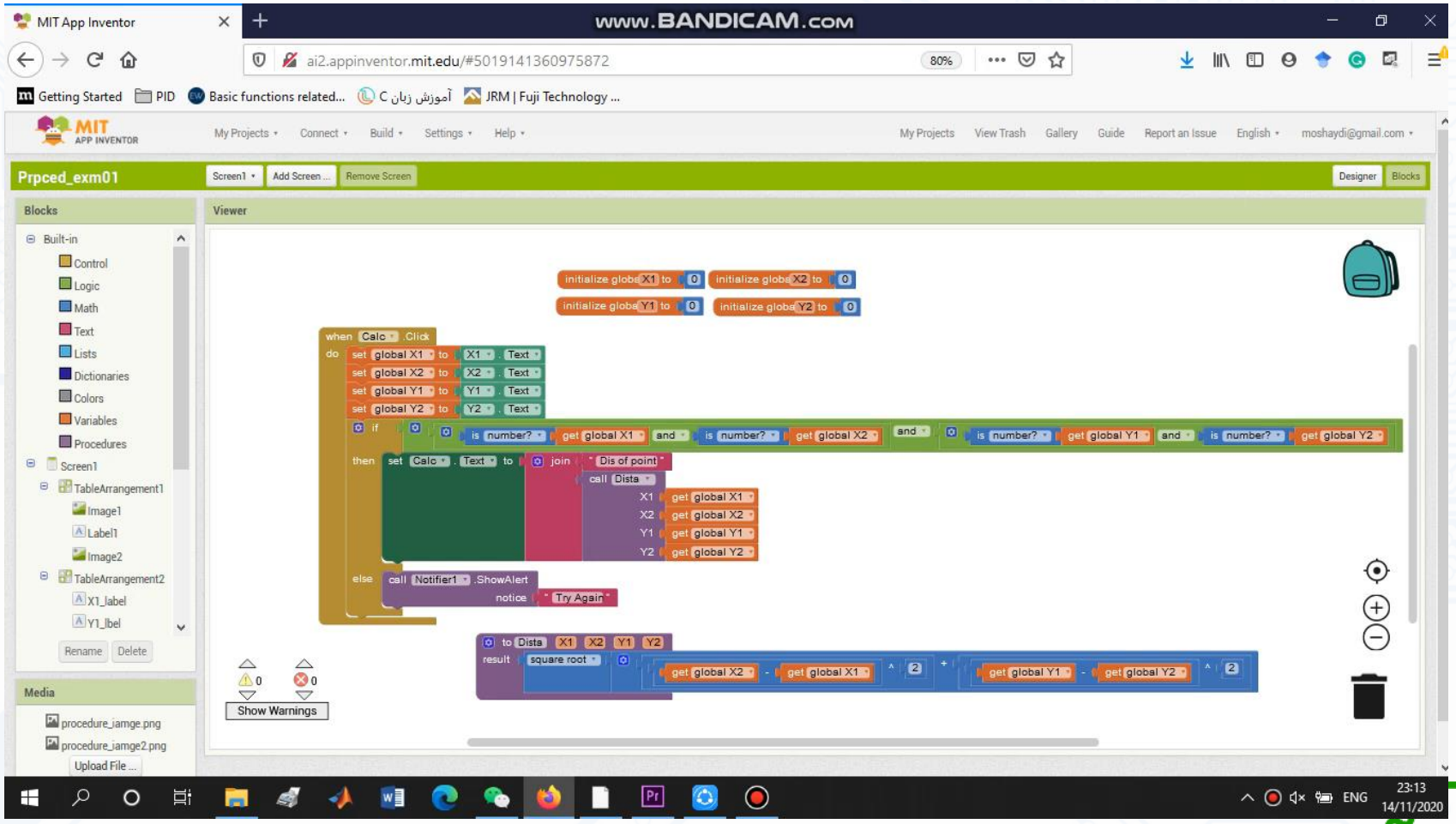
Distance calculator Designer section

Solve the problem

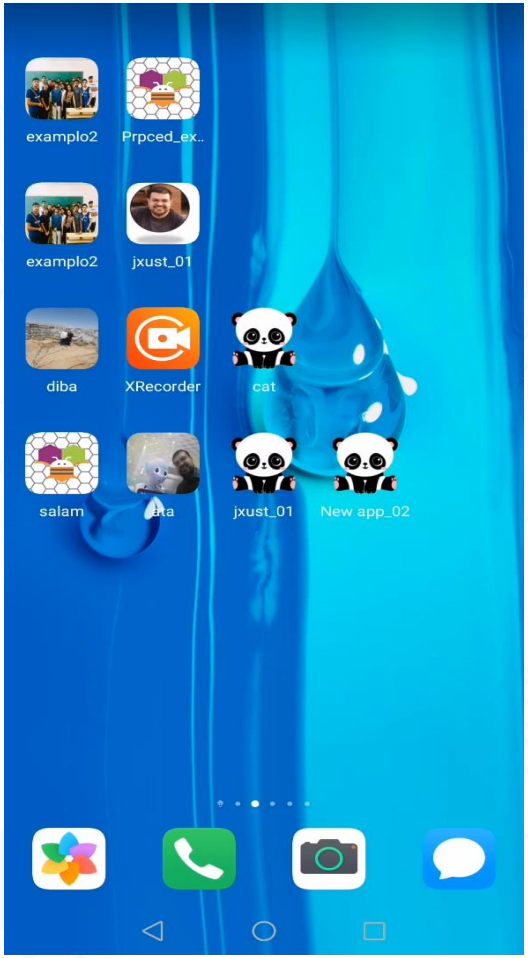


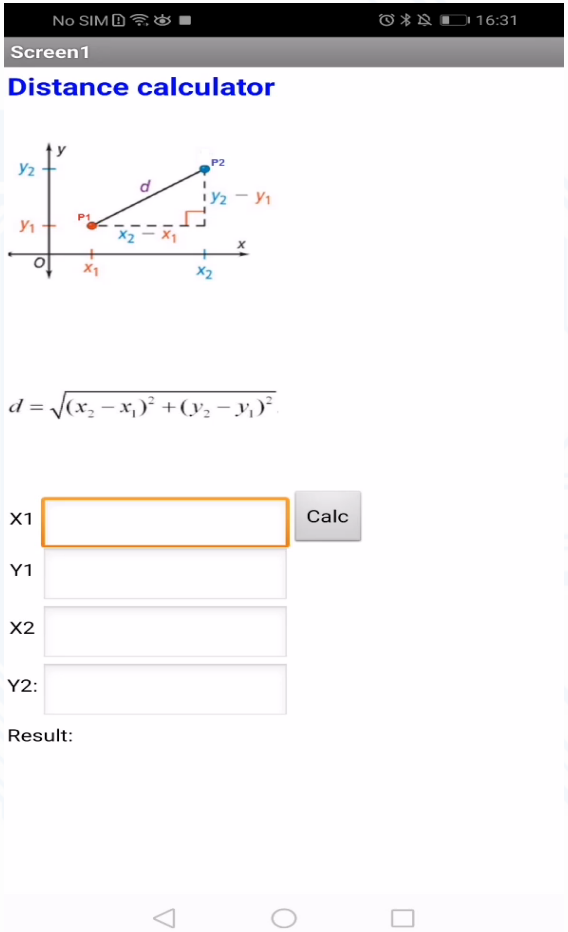
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

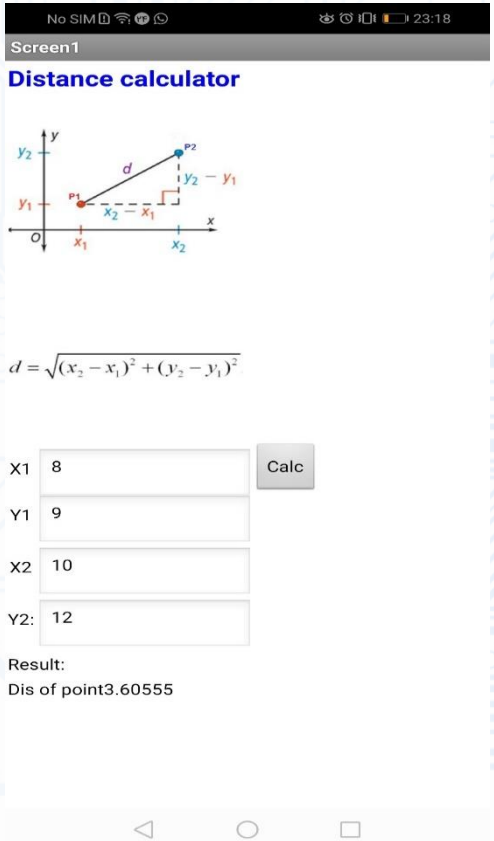
Solve the problem



Final App Demo



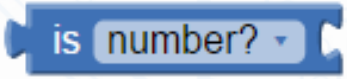




Some Notable point

- When calling distance, we send the calculation parameters. The Procedure will return the result of that calculation.
- We have created a function called distance, which parameters we supply and return a result.
- **If any of the data is entered, the application will fail for lack of data in solving the calculation.**

By Block "Is number?" that is in the part of Mathematics



- it has to check, before calling distance if all the variables introduced are numbers.
- If any non - number, a message is displayed warning with Notifier , and operations are not performed.

江西理工大學信息工程学院

JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY SCHOOL OF INFORMATION ENGINEERING

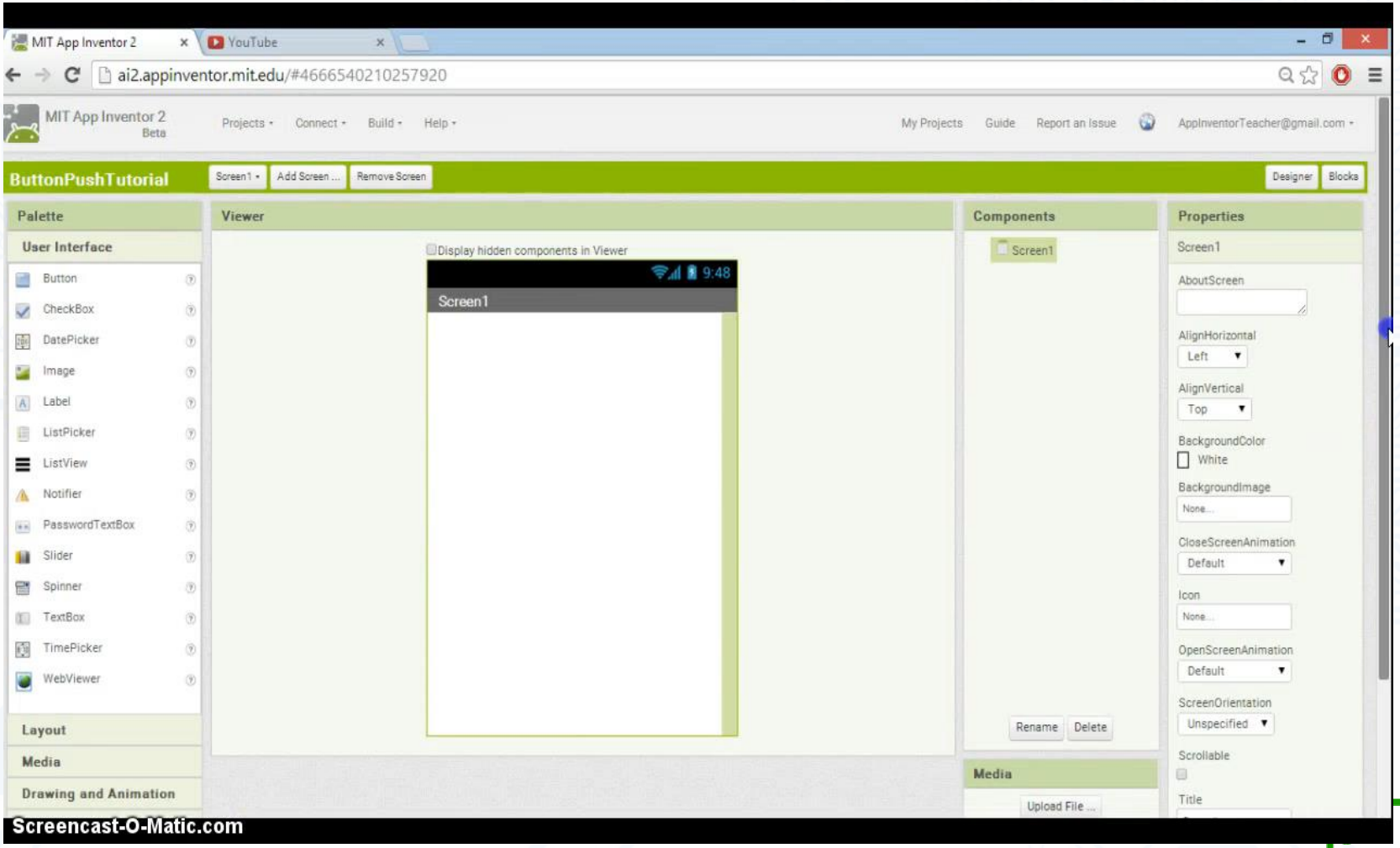
MOBILE APPLICATION DEVELOPMENT

APP Inventor _Example 02

Review the example and introduce some blocks

App Inventor

Animated Push Button example



江西理工大學信息工程学院

JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY SCHOOL OF INFORMATION ENGINEERING

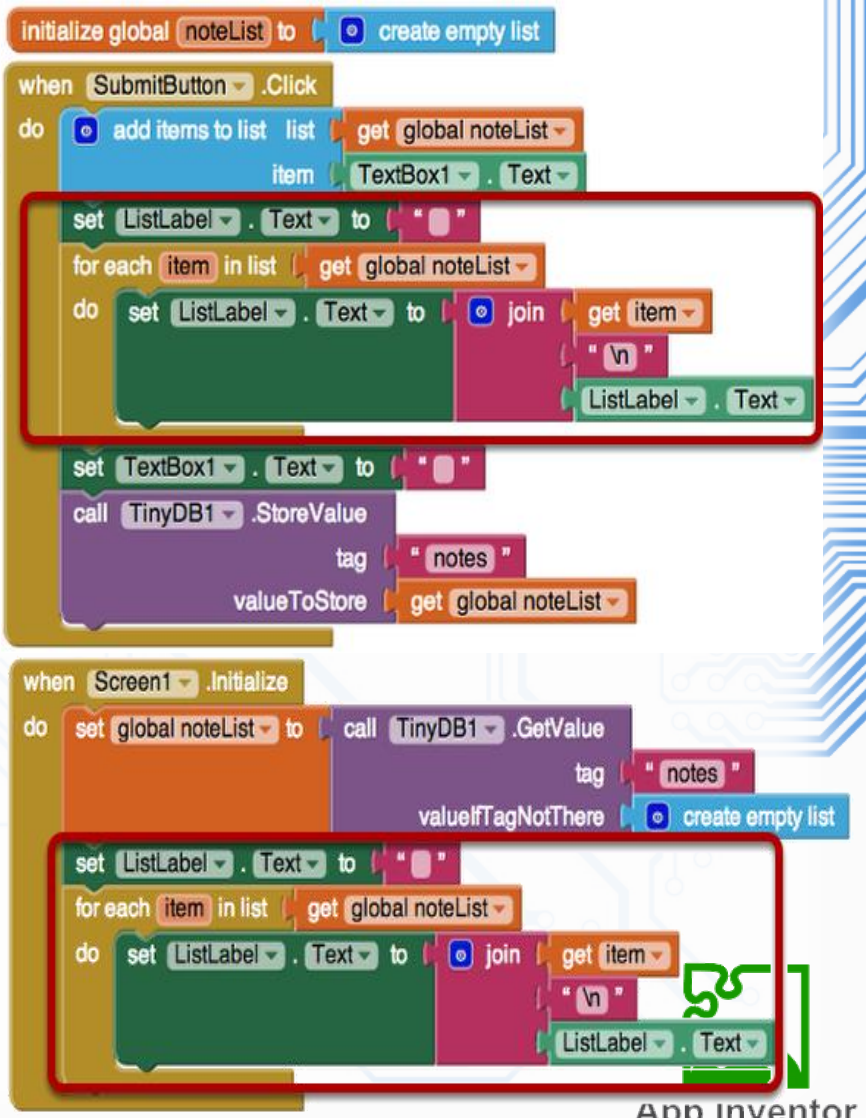
MOBILE APPLICATION DEVELOPMENT

More point about procedure

App Inventor

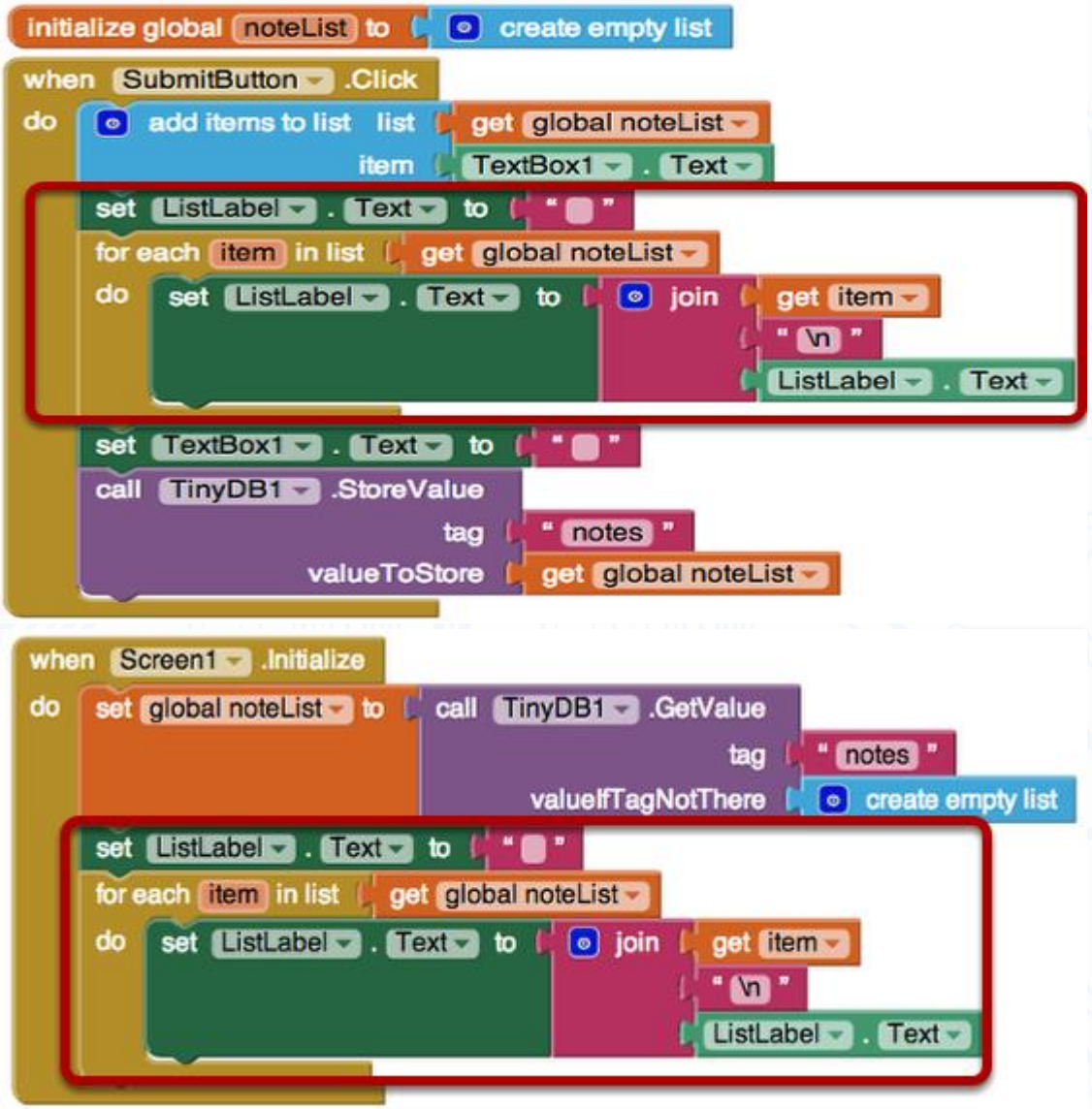
Example 1.
How do I define a procedure that displays the items of a list?

- A procedure is a named sequence of instructions (blocks).
- In real life, when I tell my son to “brush your teeth”, I really mean for him to perform a bunch of detailed steps like grabbing his toothbrush, opening the toothpaste, putting it on the toothbrush, etc. “brush your teeth” is a procedure, a name for a sequence of instructions.
- In App Inventor, you can define a procedure, place blocks into it, and then can call it from anywhere in the app.



Example 1.

How do I define a procedure that displays the items of a list?



Example 1.

How do I define a procedure that displays the items of a list?

- Consider the code above. Both event handlers have code to display a list. When a user submits a new entry (when SubmitButton.Click), the item is added to the list and the list is displayed. When the app is launched (when Screen.Initialize), the data is retrieved from the database into the list, and the list is displayed. The blocks to display a list are the same in the two event handlers.

- This code is ripe for refactoring. Refactoring means to modify the code so that it is more readable and maintainable from a programmer's perspective-- refactoring doesn't change the behavior of the app at all.

- Removing duplicate code is a common way to refactor. The basic idea is that software changes a lot: bugs are found that need to be fixed, specifications for how the software should behave change, and code, especially good code, is often repurposed. When you make changes to software, you don't want to have to find and also change a bunch of "dependencies", e.g., other code that does the same thing. It is better to have code that does a particular thing in one place, a procedure, and call that procedure from all the places that need it. Then if the procedure needs to be changed, it is changed in only one place.

Example 1.

How do I define a procedure that displays the items of a list?

- In the blocks above-left the code to display a list appears in two event handlers. We can refactor by defining a procedure displayList, moving the common blocks into it, then calling it from the two event handlers:

- The code to display the list is now in the procedure displayList, and both SubmitButton.Click and Screen1.Initialize call the procedure. The call block represents all the blocks within the procedure definition. Calling a procedure means to jump into the blocks within the procedure and execute them. Once all the blocks are executed, program control jumps back to the block below the call. When the SubmitButton is clicked, add items to list is called and then the call to displayList is made. Program control jumps down to the procedure where ListLabel.Text is set to the empty string and the for each block is executed. When the for each completes, program control jumps back up to below the displayList call within the when SubmitButton.Click event handler-- TextBox1.Text is set to the empty string and TinyDB.StoreValue is called.

Example 1.

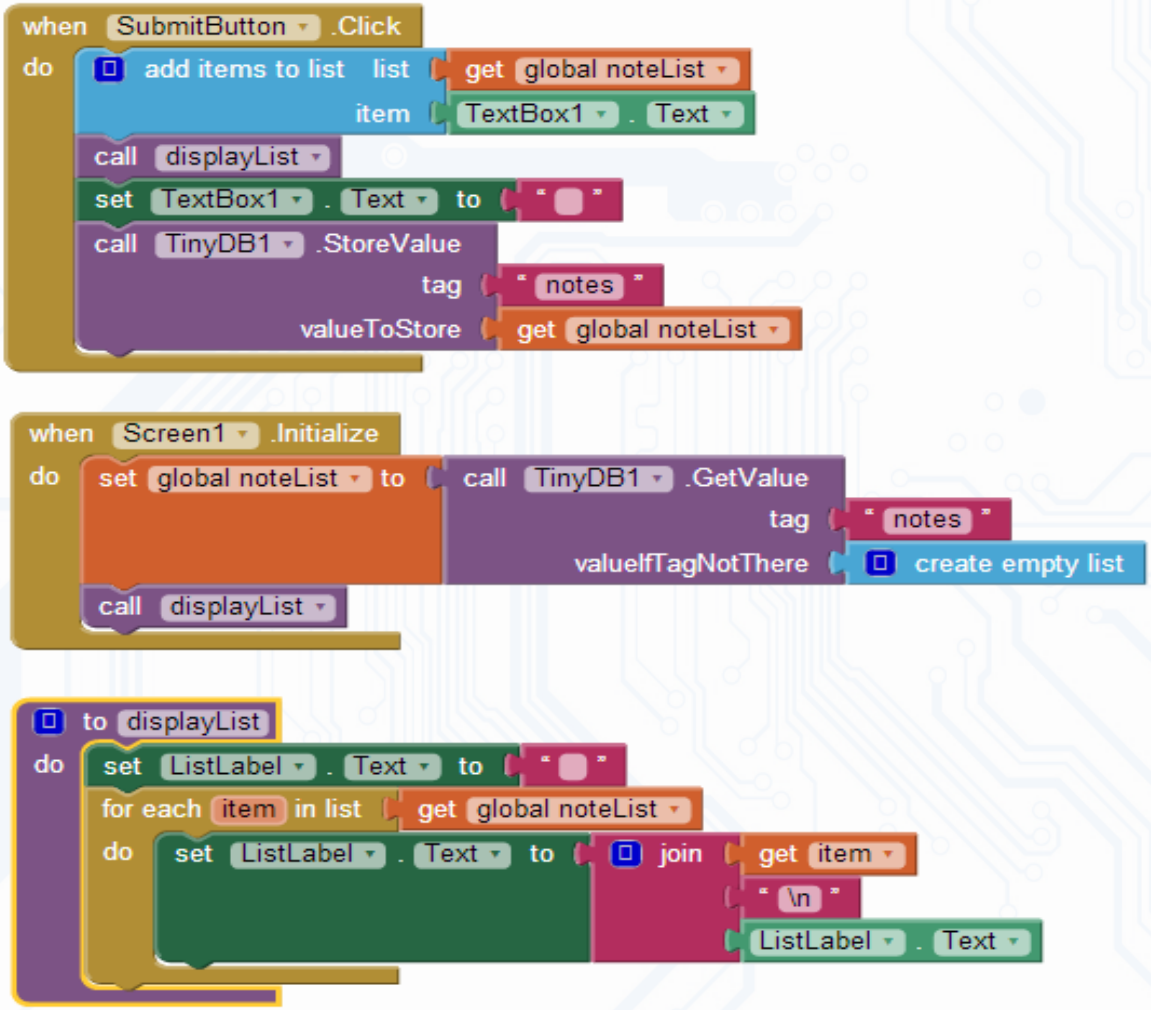
How do I define a procedure that displays the items of a list?

- The behavior, from a user's perspective is the same, but the code is better from a programmer's perspective in terms of maintenance. With this refactored code, if a bug is found in the list display code, or if it is decided the list should be displayed in a different way (e.g., commas in between items instead of new lines), the code would only have to be changed in one place.

- Designing your App with Procedures The example above refactors existing code. As you become familiar with procedures, you'll begin to include them from the beginning of your design process.You'll begin to think in terms of a larger software architecture consisting of event handlers, procedures, and the calls between them. As you're designing an app, before you drag in a single block, you'll think of the building blocks-- the procedures-- you need.

Example 1.

How do I define a procedure that displays the items of a list?



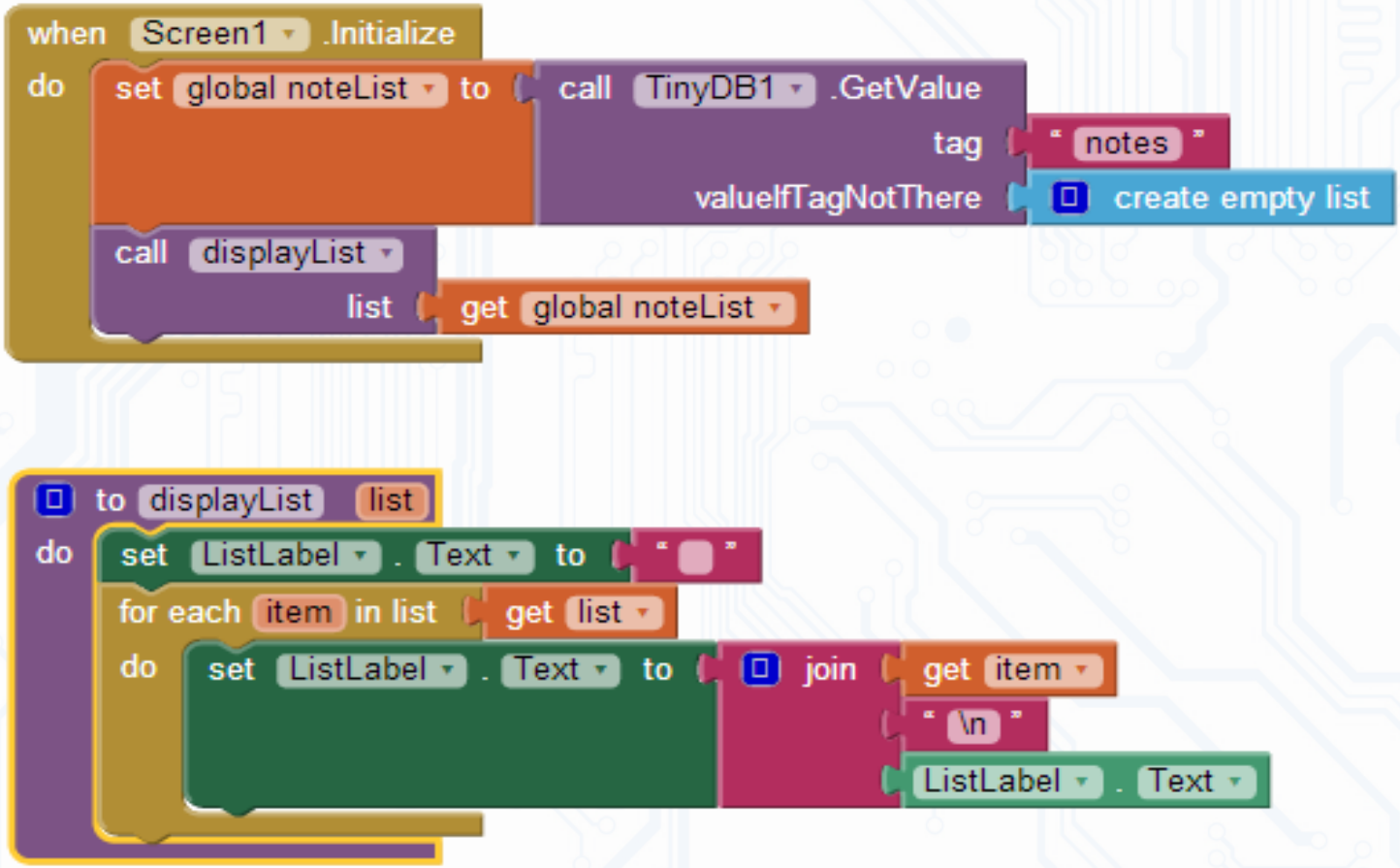
Example 2.

How do I define a procedure that can display any list?

- A parameter is information a procedure needs to do its job. You define parameters for a procedure so that it can be reused more generally.
- This version of displayList has been given a parameter list so that it can be used to display any list sent to it. In this case, the variable noteList is sent as the parameter, but you could call it from somewhere else in your app with a different list.
- Note that the original displayList procedure was specific to the variable noteList-- you couldn't use it to display other lists.
- There is one more issue with this procedure which makes it less than optimal for reuse. This procedure displays the list you sent it in a specific label, ListLabel.
- If your app had multiple lists and labels, you'd want your displayList procedure to work more generally. The next example will illustrate how to fix this.

Example 2.

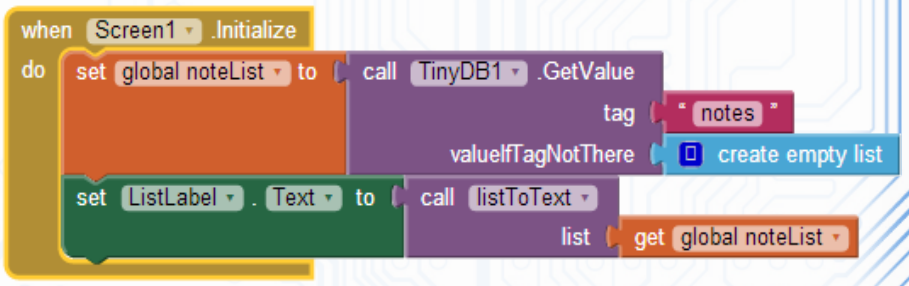
How do I define a procedure that can display any list?

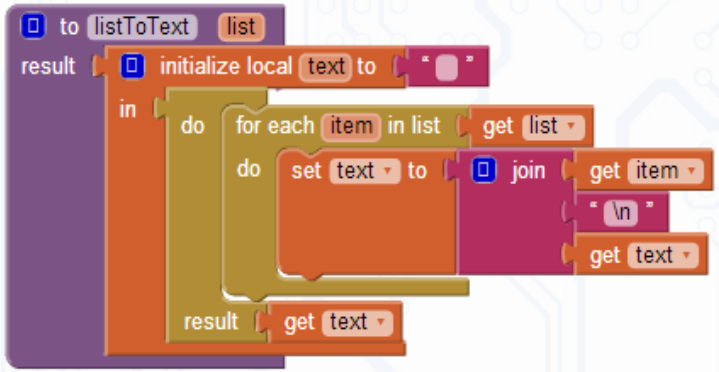


Example 2.

How do I define a procedure that can display any list in any label?

- Instead of actually displaying a list, the procedure does the work of converting a list variable into text which can be put into any label.
- The procedure is generic-- it takes any list and returns text, and it doesn't refer to any specific components.
- Any code that calls this procedure, e.g., Screen1.Initialize, will provide the specific list as a parameter to listToText, and will place the resulting text in a specific label.





Student Task_8

- **Distance calculator Designer section and repeat the example 1 and 2 about procedure and compare**

Repeat this examples and make based on our task format

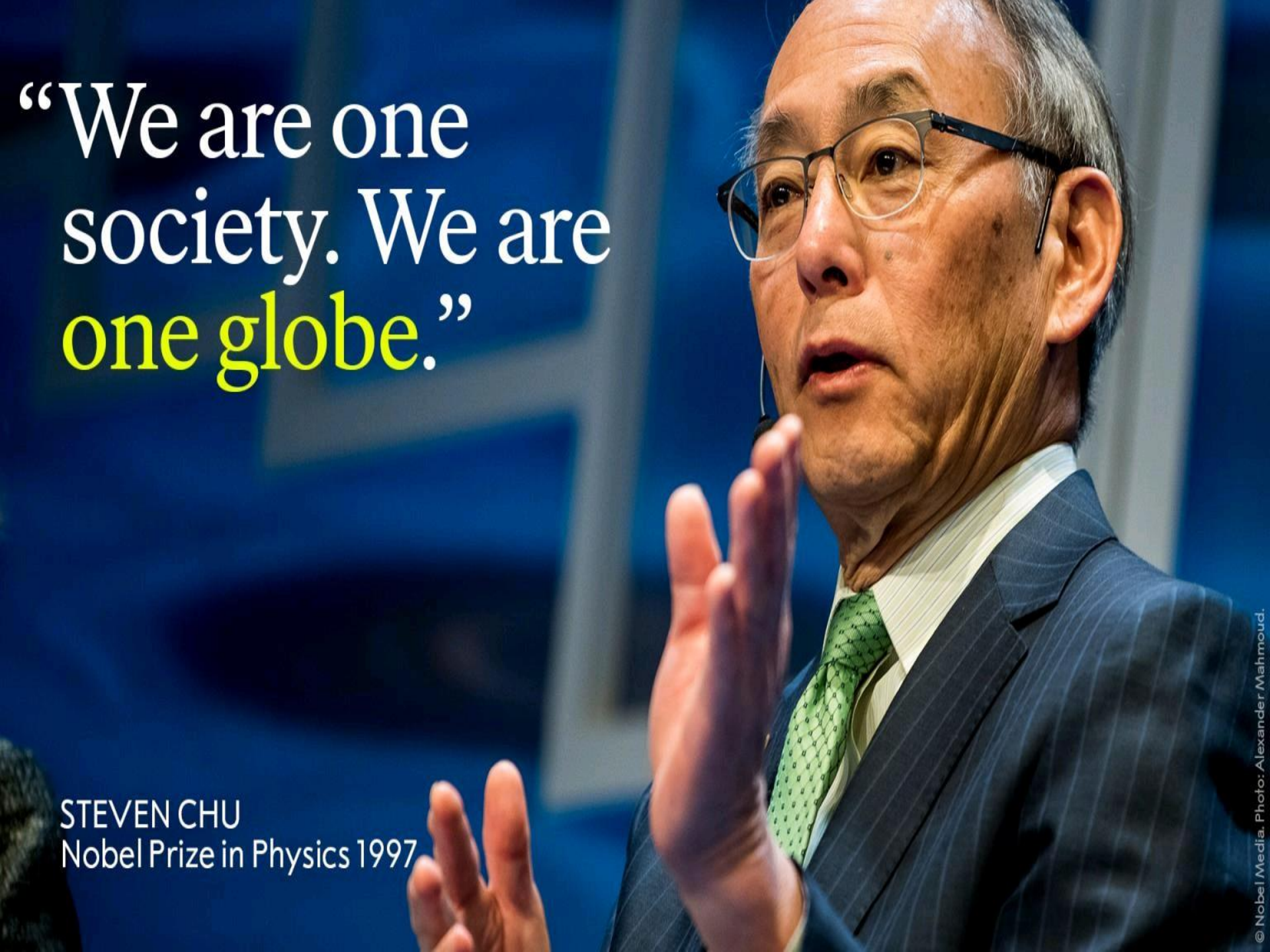
Next lecture

- **You have time to send your task**
- **Send the file in PPT(power point format) to JUST MOOC**
- **Your file should have this format of name**

<Task number><student name><Student ID>.ppt

Reference

- http://ai2.appinventor.mit.edu/reference/blocks/lists.html#selectlistitem
- **https://appinventor.mit.edu/explore/content/alertme.html**
- **Teaching with AppInventor** http://appinventor.mit.edu/explore/teach.html
AppInventor Tutorials:
http://appinventor.mit.edu/explore/ai2/tutorials.html
- **Sounds** http://www.soundbible.com
- **App Inventor:** http://appinventor.googlelabs.com/
- **Appinventor.org:** http://www.appinventor.org/
- **Wolber, Abelson et al. text:** http://www.appinventor.org/text2011
- **Group:** http://groups.google.com/group/app-inventor-instructors
- **Wolber course:** http://appinventor.org/course-in-a-box
- **Morelli course:** http://turing.cs.trincoll.edu/~ram/cpsc110/


"We are one society. We are one globe."
STEVEN CHU
Nobel Prize in Physics 1997
© Nobel Media. Photo: Alexander Mahmoud.

江西理工大学

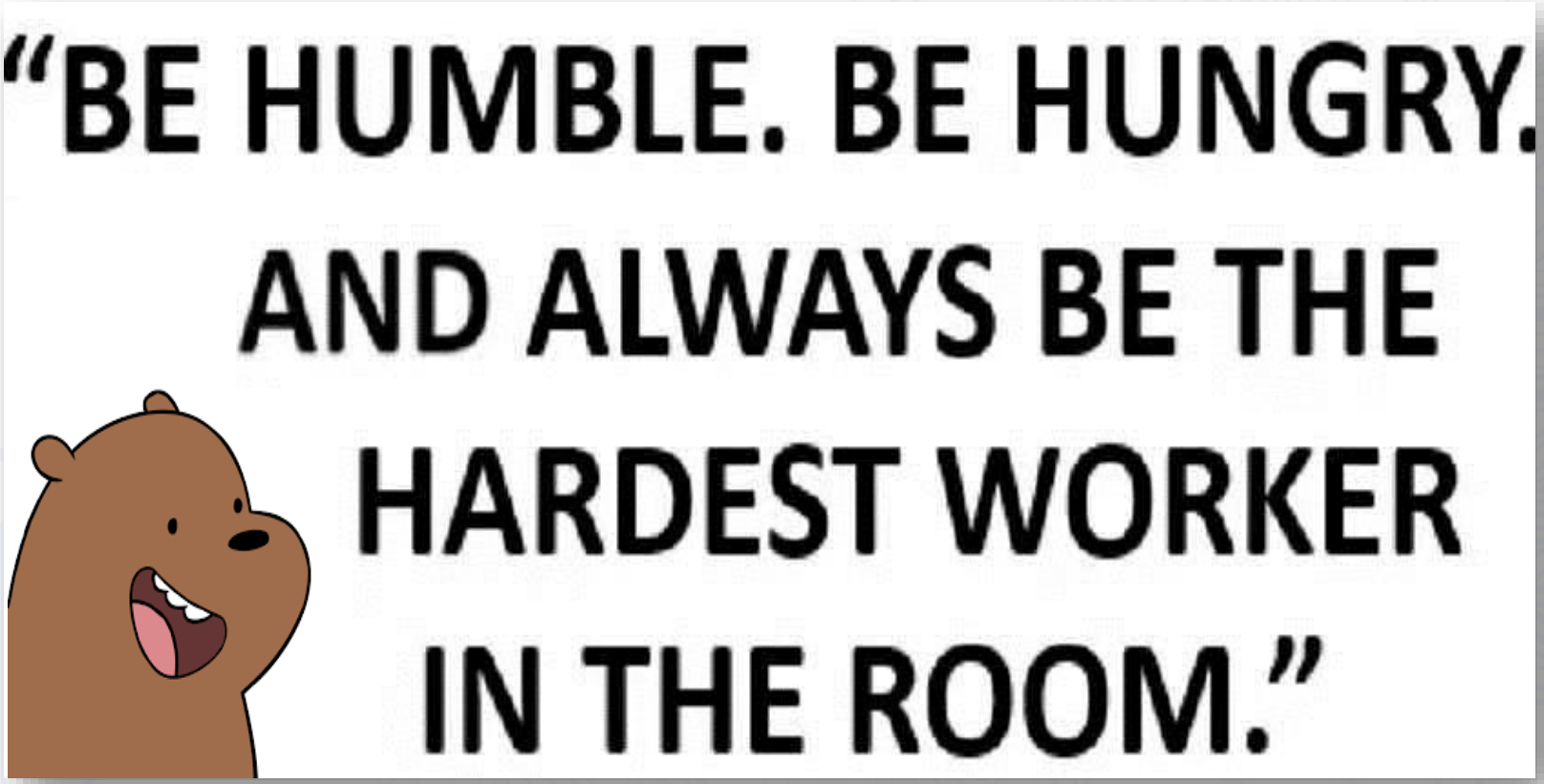
Jiangxi University of Science and Technology

信息工程学院

School of information engineering

Digital Image Processing

THANK YOU


"BE HUMBLE. BE HUNGRY.
AND ALWAYS BE THE
HARDEST WORKER
IN THE ROOM."