

Introduction to Design (Track 3)

5.1 List

Zilu Liang

www.zilu-liang.net/id3

Important!!

Please disinfectize your hands before entering the classroom!

入室前にアルコールを使用して手指消毒を行ってください。

Please disinfectize your chair and table!

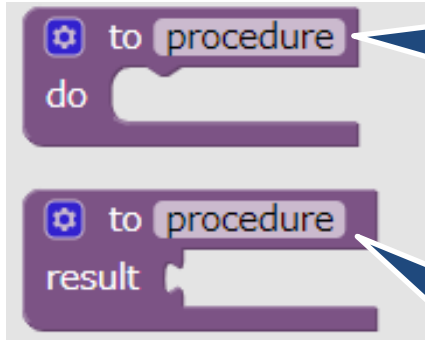
- ①ペーパーにアルコールを噴霧してください。
- ②アルコールが噴霧されたペーパーで、使用箇所（テーブル、椅子など）を拭き取ってください。
- ③使用済のペーパーは廊下のごみ箱に捨ててください。



1. Recap of Week 4

Procedure

A procedure is a **set of instructions** that is grouped together, given a name, and can perform specific tasks. It may also be called a function in programming.

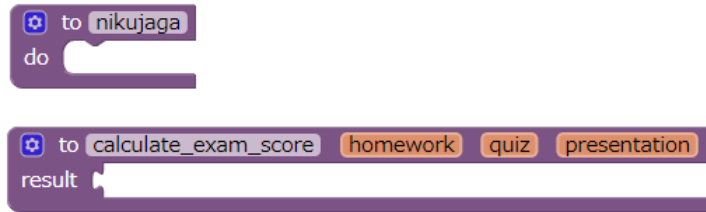


This type of procedure only do something; when this procedure is called, a set of actions will be performed.

This type of procedure returns some value. When this procedure is called, a set of actions will be performed, and a result will be generated and returned.

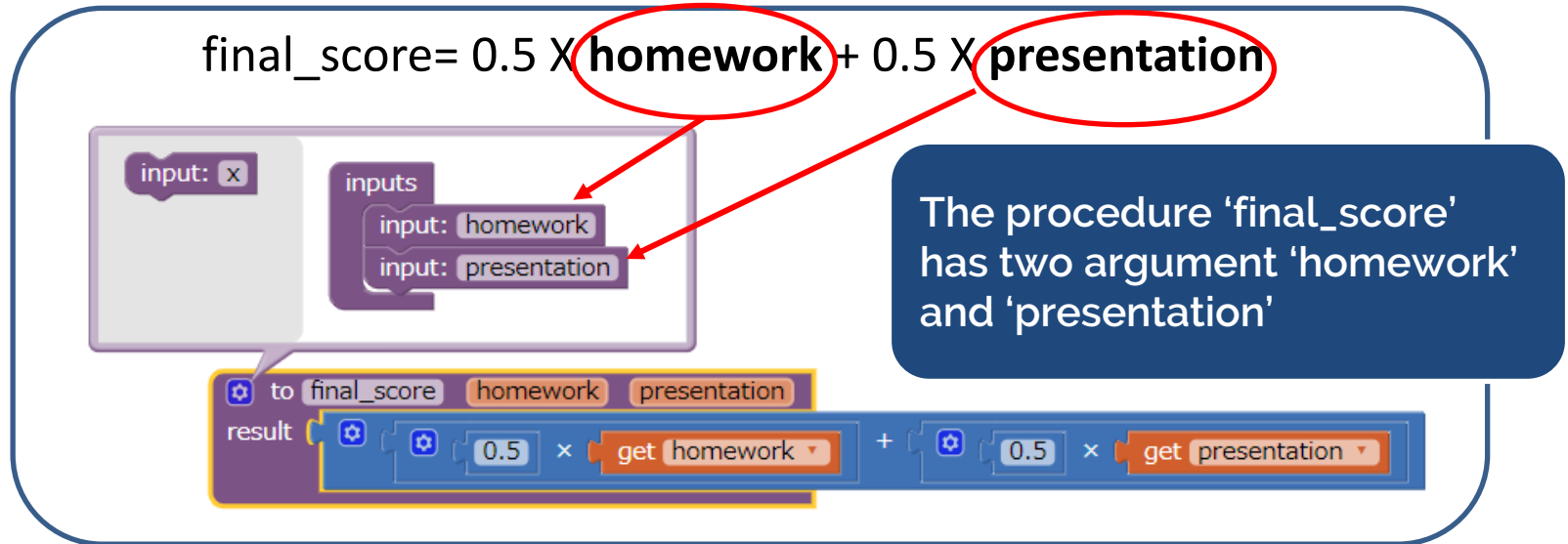
Procedure Name

- 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.



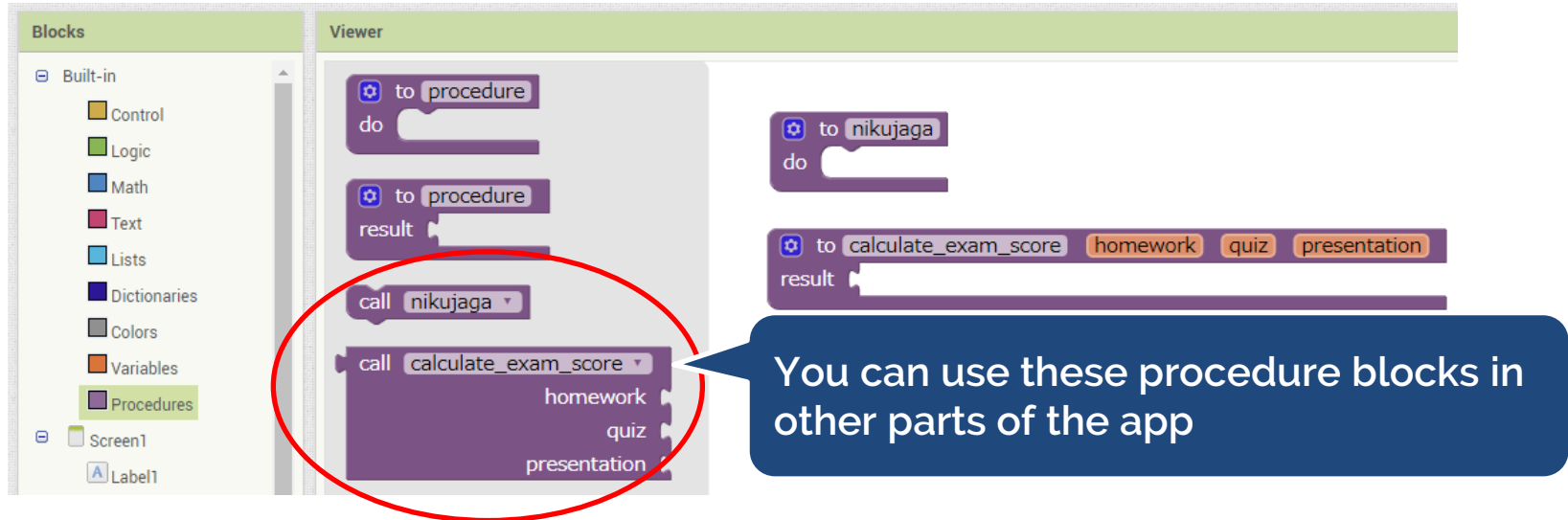
Procedure Argument

- An argument is an **input** to a procedure.
- You can use the mutator button to add arguments.



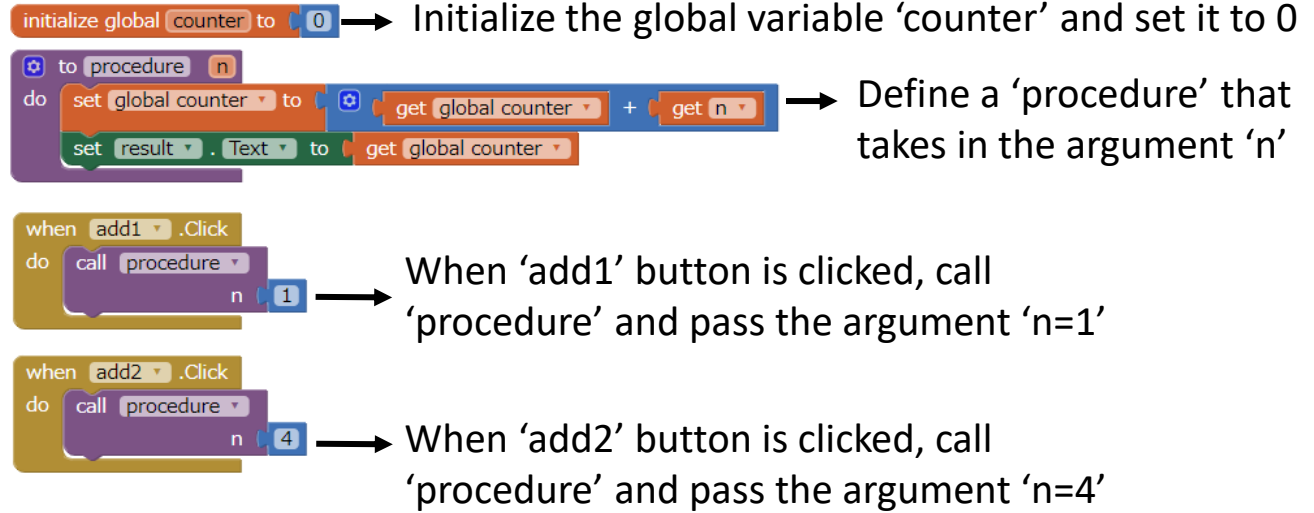
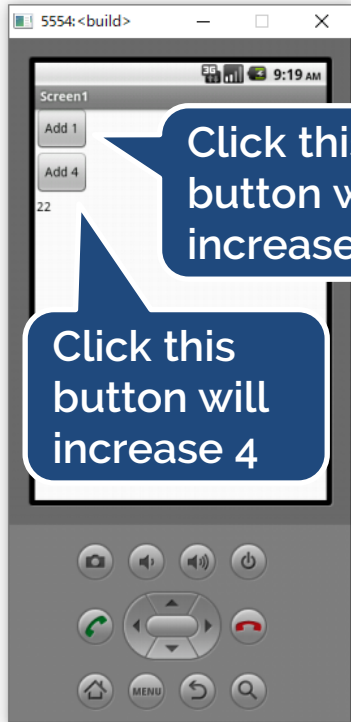
Procedure in App Inventor

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.



The screenshot displays the App Inventor interface. On the left, the 'Blocks' panel shows a list of built-in categories: Control, Logic, Math, Text, Lists, Dictionaries, Colors, Variables, and Procedures. The 'Procedures' category is highlighted. In the center, the 'Viewer' panel shows a sequence of blocks: two 'to procedure' blocks, a 'call nikujaga' block, and a 'call calculate_exam_score' block. The 'call calculate_exam_score' block is circled in red. To the right, a 'to nikujaga' block and a 'to calculate_exam_score' block are shown. A blue callout box points to the 'call calculate_exam_score' block with the text: 'You can use these procedure blocks in other parts of the app'.

Code Anatomy: Counter App



Code Anatomy: BMI Calculator App

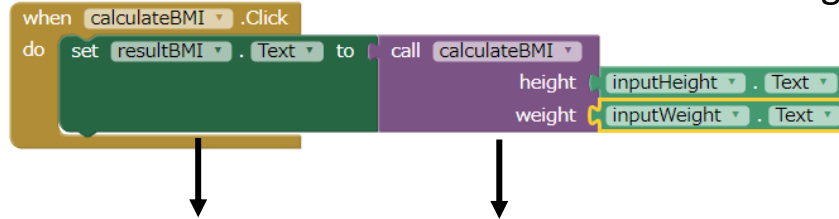


A user needs to input weight and height

Click this button will calculate BMI



Define a 'calculateBMI' procedure that takes in the argument 'height' and 'weight' to calculate BMI



3. Set the 'resultBMI' text to the result returned from the 'calculateBMI' procedure

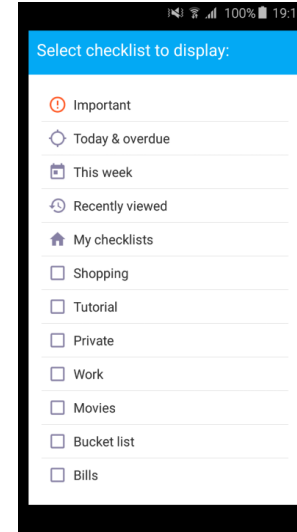
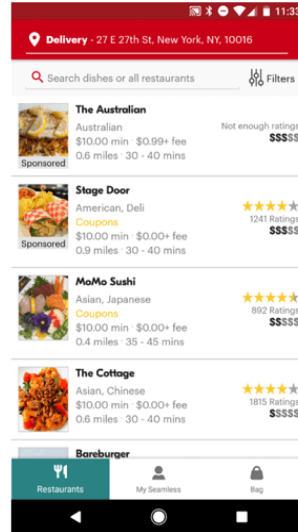
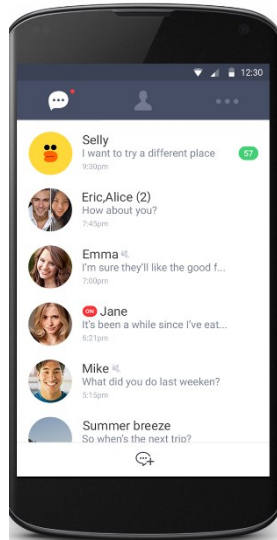
2. Call the 'calculateBMI' procedure, and return the result value to the block on the left

1. Set the 'weight' and 'height' arguments to the values that a user input in the textboxes

2. List


List

- A list is multiple pieces of data stuck together, one after another in fixed order.
- Lists are a type of data structure used in every programming language.



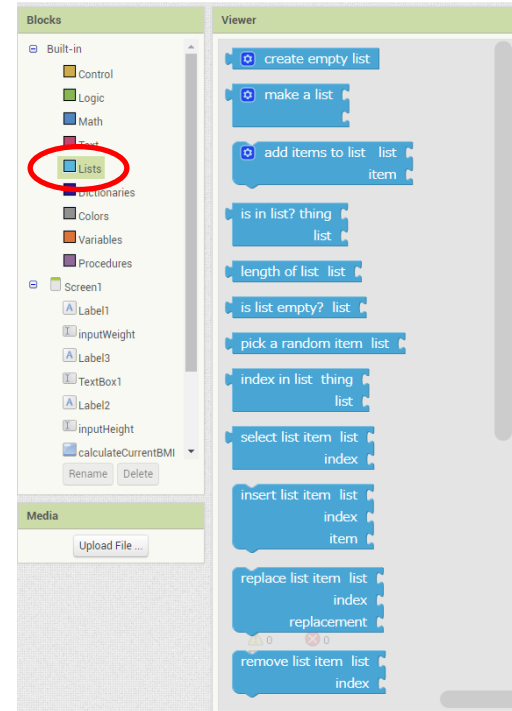
List

- The position of an element in a list is called its **index**. In App Inventor, the first element in a list is index 1. the index increase one by one.
- We can reference a specific element inside a list if we know its index and the list name.
- The data of each item can be different types.

						
Index	1	2	3	...	N-1	N
Data type	string	string	integer		double	Boolean

The List Block in App Inventor

- Create empty list
- Make a list with some items
- Add items to list
- Get the length of the list
- Check if the list is empty
- ...



Create Empty List

- The list itself is a container and can be treated as a single piece of data
- List can be stored in variables (they are just a type of data) and passed as inputs and outputs to blocks

initialize global `my_list` to 

Initialize a global variable with data type of list

get global `my_list` ▼

Using the getter block, we'll get all items in `my_list`

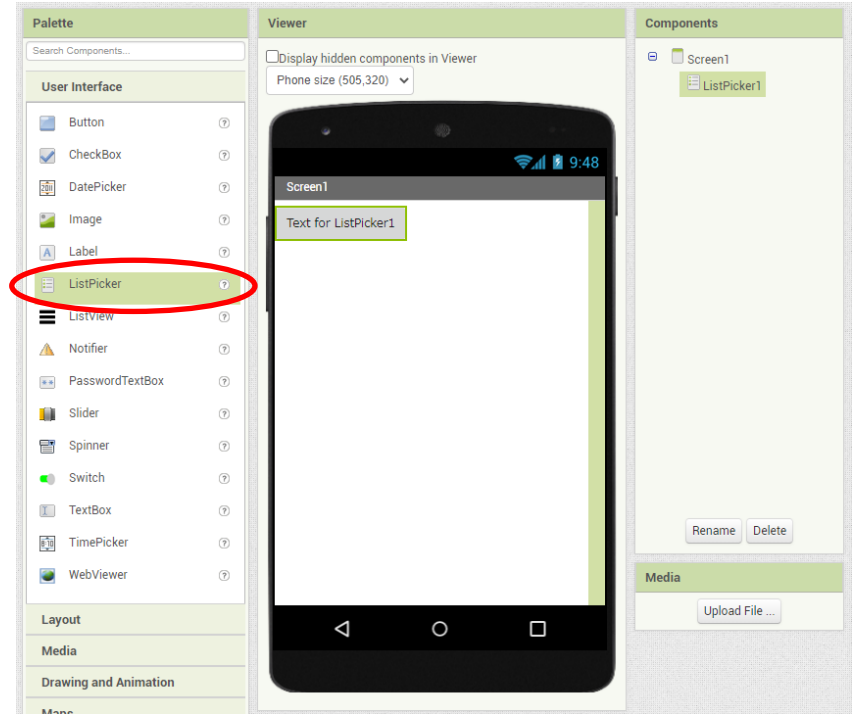
Create List with Some Stuff Already in It

- Initialize a global variable with the 'make a list' block
- Click on the mutator to add items to the list

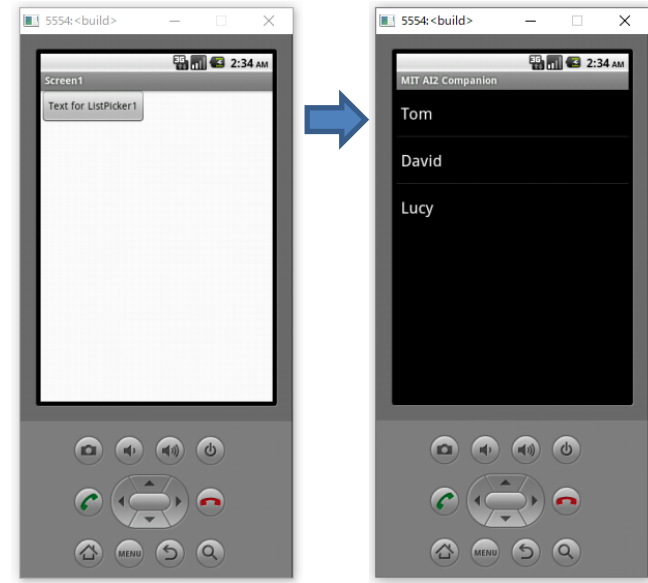
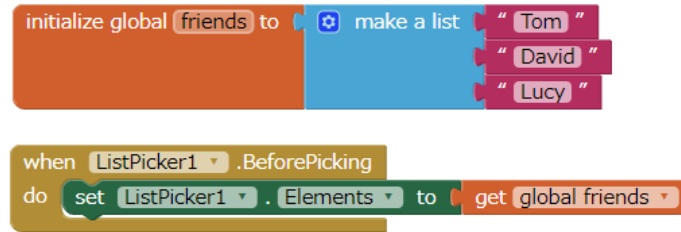


The ListPicker

- The ListPicker component in the User Interface Pane allows the user to select a list

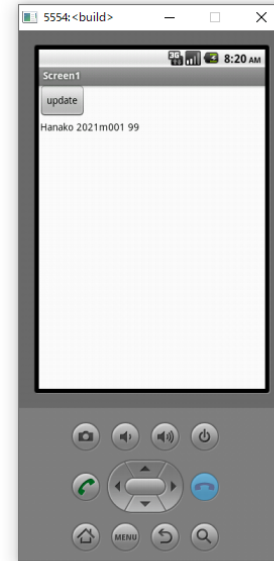
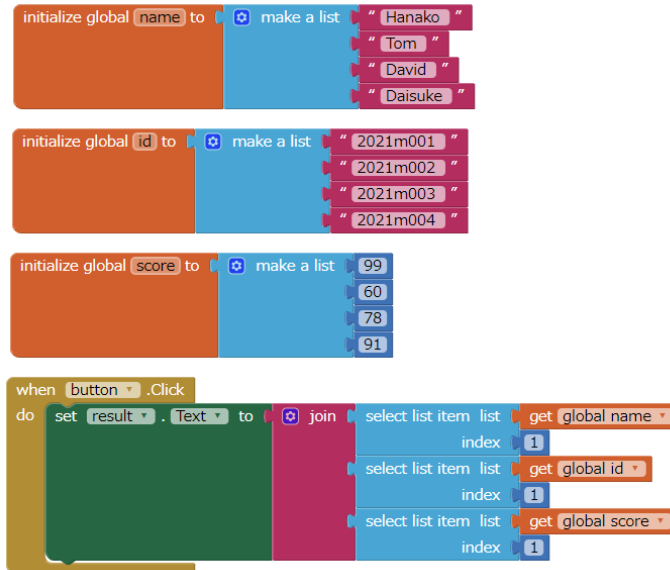


Use ListPicker to Show All Items



Using Multiple Lists Together

- For instance, if we want to keep a record of students' final score, we need to store data like 'name', 'id', 'score' for each student.
- We can create three lists to store 'name', 'id', 'score' separately.



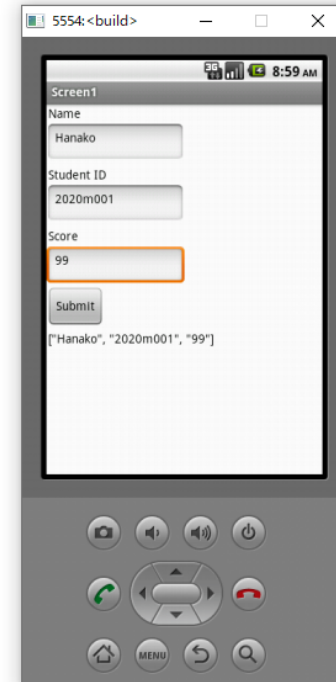
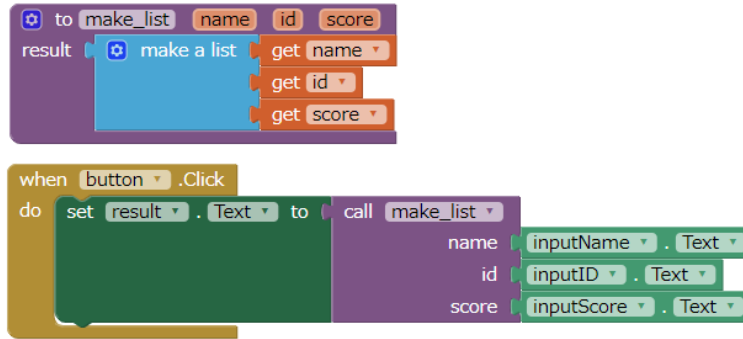
Using List as Data Structure

- A data structure is a way of organizing the data so that it can be used efficiently. We can use list as a data structure.
- Now we have a new solution to the example on the previous slide, we can use a list to store the 'name', 'id' and 'score' of each student, and combine the list of each student into a higher-level list.

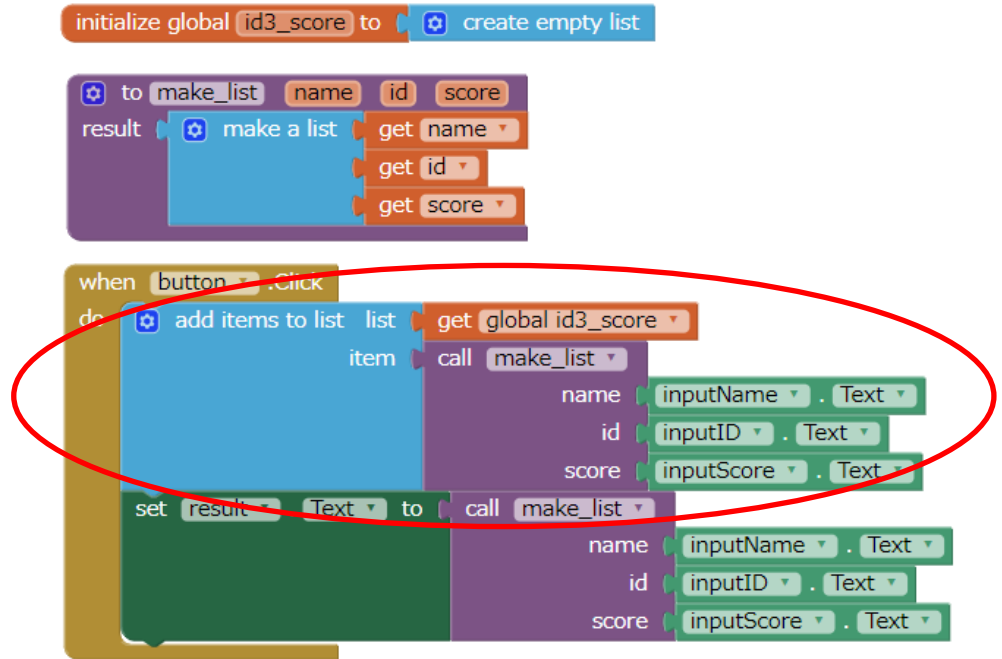
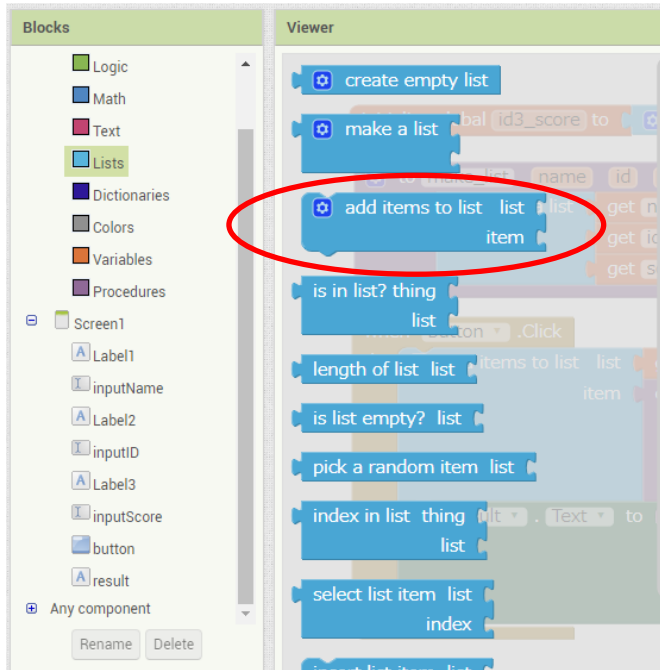


Use Procedure to Create List

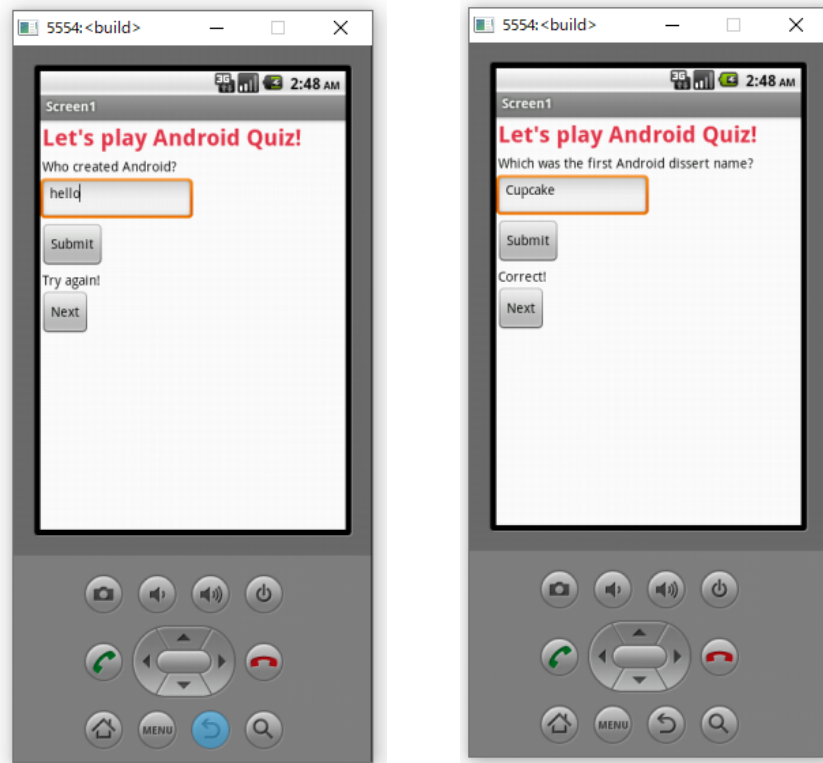
- Procedure is often used to manage the operation of a list, such as making a list from a user's inputs.



Add Elements to a List



Hands-on: Android Quiz App



Code Anatomy: Android Quiz App

```
initialize global QuestionList to make a list  
  "Who created Android?"  
  "Which was the first Android dessert name?"  
  "What is the name of the Android app marketplace?"  
  "What Google-branded phone did the company launch..."  
  "What candy bar is Android 4.4 named after?"
```

→ Initialize a global variable with five questions

```
initialize global AnswerList to make a list  
  "Andy Rubin"  
  "Cupcake"  
  "Google Play"  
  "Nexus 5"  
  "Kit Kat"
```

→ Initialize a global variable with the corresponding answers

```
when Screen1.Initialize  
do set showQuestionLabel.Text to select list item list get global QuestionList  
  index 1
```

→ Show the first question when app starts

```
initialize global currentQuestionIndex to 0
```

→ Initialize a global variable to store the index of the current question

```
when nextButton.Click  
do set global currentQuestionIndex to get global currentQuestionIndex + 1  
  if get global currentQuestionIndex > length of list list get global QuestionList  
  then set global currentQuestionIndex to 1  
  set showQuestionLabel.Text to select list item list get global QuestionList  
    index get global currentQuestionIndex
```

→ If 'Next' button is clicked, increase the index by 1

→ If it's larger than the length of the question list, reset it to 1.

```
when submitButton.Click  
do if compare texts inputAnswer.Text = select list item list get global AnswerList  
  index get global currentQuestionIndex  
  then set correctIncorrectLabel.Text to "Correct!"  
  else set correctIncorrectLabel.Text to "Try again!"
```

→ Get user's answer and compare with the correct answer

Any questions?



KYOTO UNIVERSITY of ADVANCED SCIENCE

京都先端科学大学