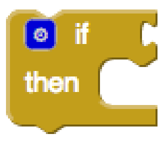
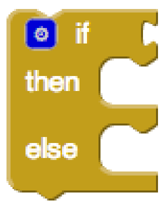
# MIT App Inventor Control Blocks

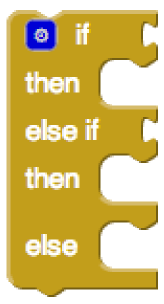
## if & else if



Tests a given condition. If the condition is true, performs the actions in a given sequence of blocks; otherwise, the blocks are ignored.

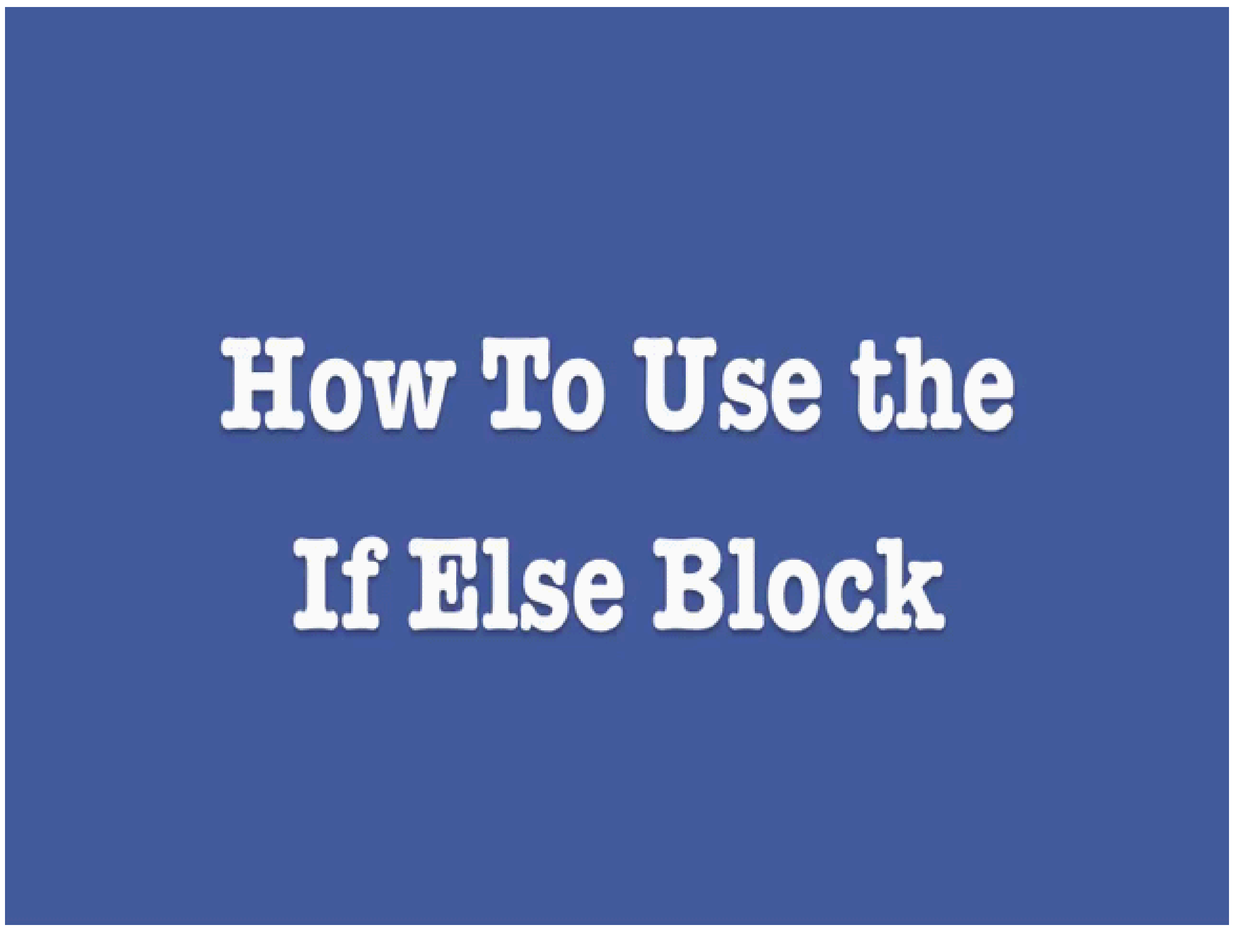


Tests a given condition. If the condition is true, performs the actions in the -then sequence of blocks; otherwise, performs the actions in the -else equence of blocks.

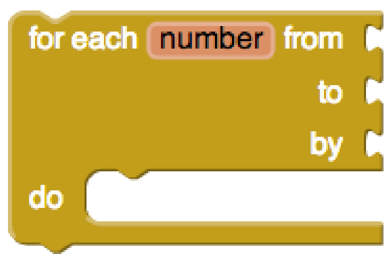


Tests a given condition. If the result is true, performs the actions in the -then sequence of blocks; otherwise tests the statement in the -else if section. If the result is true, performs the actions in the -then sequence of blocks; otherwise, performs the actions in the -else sequence of blocks.

The animation below shows how to use the if else mutator block.

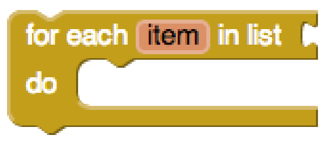


## for each number from to



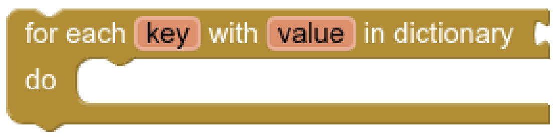
Runs the blocks in the do section for each numeric value in the range starting from *from* and ending at *to*, incrementing number by the value of *by* each time. Use the given variable name, number, to refer to the current value. You can change the name number to something else if you wish.

## for each item in list



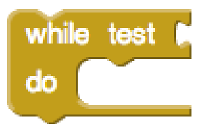
Runs the blocks in the do section for each item in the list. Use the given variable name, item, to refer to the current list item. You can change the name item to something else if you wish.

## for each key with value in dictionary



Runs the blocks in the do section for each key-value entry in the dictionary. Use the given variables, key and value, to refer to the key and value of the current dictionary entry. You can change the names key and value to something else if you wish.

## while

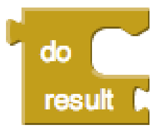


Tests the -test condition. If true, performs the action given in -do , then tests again. When test is false, the block ends and the action given in -do is no longer performed.

## if then else

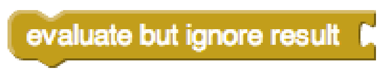
Tests a given condition. If the statement is true, performs the actions in the then-return sequence of blocks and returns the then-return value; otherwise, performs the actions in the else-return sequence of blocks and returns the else-return value. This block is similar to the ternary operator (?:) found in some languages.

## do with result



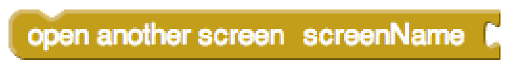
Sometimes in a procedure or another block of code, you may need to do something and return something, but for various reasons you may choose to use this block instead of creating a new procedure.

## evaluate but ignore result



Provides a “dummy socket” for fitting a block that has a plug on its left into a place where there is no socket, such as one of the sequence of blocks in the do part of a procedure or an if block. The block you fit in will be run, but its returned result will be ignored. This can be useful if you define a procedure that returns a result, but want to call it in a context that does not accept a result.

## open another screen



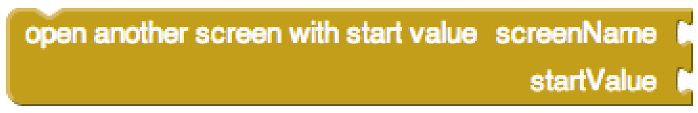
Opens the screen with the provided name.

The screenName must be one of the Screens created using the Designer. The screenName should be selected from the connected screen name dropdown block.

If you do open another screen, you should close it when returning to your main screen to free system memory. Failure to close a screen upon leaving it will eventually lead to memory problems.

App developers should never close Screen1 or use this block to return to Screen1. Use the `close screen` block instead.

### open another screen with start value



Opens another screen and passes a value to it.

### get plain start text

Returns the plain text that was passed to this screen when it was started by another app. If no value was passed, it returns the empty text. For multiple screen apps, use `get start value` rather than `get plain start text`.

### get start value

Returns the start value given to the current screen.

This value is given from using `open another screen with start value` or `close screen with value`.

### close screen

Closes the current screen.

### close screen with plain text

Closes the current screen and passes text to the app that opened this one. This command is for returning text to non-App Inventor activities, not to App Inventor screens. For App Inventor Screens, as in multiple screen apps, use `close screen with value`, not `close screen with plain text`.

### close screen with value

Closes the current screen and returns a value to the screen that opened this one.

### close application

Closes the application.

### break

When looping using the for range, for each, or while blocks it is sometimes useful to be able to exit the loop early. The `break` allows you to escape the loop. When executed, this will exit the loop and continue the app with the statements that occur after the loop in the blocks.