# Required Modules

iota.Trigger, iota.Device

# Constants

COMM\_ELSE

One of the valid commandTypes for the FunctionCode Class.

COMM\_ELSE FunctionCodes act as both ELSE statements and ELSE IF statements.

COMM\_ELSE FunctionCodes should have linkedCommands[] initialised with the IF statement that the ELSE statement is linked to, and any related COMM\_ELSE commands that are acting as ELSE IF statements.

COMM\_ELSE statements can have either 0 or (3+4x) values stored in params[]. COMM\_ELSE FunctionCodes with 0 params will be treated as an ELSE statement, COMM\_ELSE FunctionCodes with 3+ will be treated as an ELSE IF statement, where params[] is the expression the ELSE IF statement will check. When params[] has values, it must be formatted as:

[value, operator, value, logical expression, value, operator, value, logical expression…]. For example:

['x', '=', '1', 'or', 'y', '>=', '2']

value = "OTHERWISE"

## COMM\_END

One of the valid commandTypes for the FunctionCode.

COMM\_END FunctionCodes act as indicators for the end of a conditional statement, such as COMM\_IF or COMM\_WHILE FunctionCodes.

COMM\_END FunctionCodes should have linkedCommands[] initialised with the conditional FunctionCode that they are linked to.

COMM\_END FunctionCodes have nothing needed to be stored in params[].

value = "END"

COMM\_FOR

One of the valid commandTypes for the FunctionCode.

COMM\_FOR FunctionCodes act as FOR loops.

COMM\_FOR FunctionCodes have nothing needed to be stored in linkedCommands[].

COMM\_FOR FunctionCodes take 1 value in params[], the number of times that it should repeat the code attached to it. params[] must be formatted as:

[value]. For example:

['60']

value = "FOR"

COMM\_IF

One of the valid commandTypes for the FunctionCode.

COMM\_IF FunctionCodes act as IF statements.

COMM\_IF FunctionCodes have nothing needed to be stored in linkedCommands[].

COMM\_IF statements should have (3+4x) values stored in params[], where params[] is the expression the IF statement will check.. params[] must be formatted as:

[value, operator, value, logical expression, value, operator, value, logical expression…]. For example:

['x', '=', '1', 'or', 'y', '>=', '2']

value = "IF"

COMM\_SET

One of the valid commandTypes for the FunctionCode.

COMM\_SET FunctionCodes allow the user to set values stored throughout the schedule, or the values that devices have.

COMM\_SET FunctionCodes have nothing needed to be stored in linkedCommands[].

COMM\_SET FunctionCodes take 2 values in params[], the value that it needs to change, and the value it will be changed to. params[] must be formatted as:

[value, value]. For example:

['smartLight.brightness', '80']

value = "SET"

COMM\_WHILE

One of the valid commandTypes for the FunctionCode.

COMM\_WHILE FunctionCodes act as WHILE loops.

COMM\_WHILE FunctionCodes have nothing needed to be stored in linkedCommands[].

COMM\_WHILE statements should have (3+4x) values stored in params[], where params[] is the expression the WHILE statement will continually check until it is false. params[] must be formatted as:

[value, operator, value, logical expression, value, operator, value, logical expression…]. For example:

['x', '=', '1', 'or', 'y', '>=', '2']

value = "WHILE"

# Class FunctionCode

Description

This class holds the information for a single operation for the iota CodeBlocks language, allowing the CodeBlocks to be interpreted to python.

Values

* (str) commandType - The type of command the code is, for translating to python. Can be FOR, WHILE, IF, OTHERWISE, SET, GET or END.
* (int) number - Which FunctionCode of commandType the code is.
* (str) name - The unique name of the command, comprised of the commandType & number.
* ([FunctionCode]) linkedCommands - The FunctionCodes this FunctionCode is associated with.
* ([str]) params - The params that the command takes.
* (bool) hasRun - The number of times a FunctionCode has run

## Constructor

### Required Parameters

* self
* (str) commandType - The value the object’s commandType is set to.
* (int) number - The value the object’s number is set to.

### Optional Parameters

* ([FunctionCode]) linkedCommands - The value the object’s linkedCommands variable is set to. Default value: []
* ([str]) params - The value the object’s params variable is set to.  
  Default value: []

# Class Schedule

## Description

Class to run the hold and run user-defined code when a trigger is activated.

## Values

* (str) id - The unique ID for the Schedule.
* (str) name - The name that the user sets for the Schedule.
* (bool) isPublic - Whether the Schedule can be seen by other users.
* (int) rating - The sum of positive and negative ratings given to the Schedule.
* ([Trigger]) Triggers - The triggers that activate the schedule’s code.
* ([FunctionCode]) code - The code that is run when a trigger is activated.
* ([Device]) Devices - The devices that the schedule needs to connect to in order to run.
* ({str: str}) Variables – Holds variables for the user to set and retrieve
* (bool) isRunnning - Whether the Schedule is currently running.
* (bool) isActive - Whether the User wants the Schedule to run when the trigger activates.

## Constructor

### Required Parameters

* self
* (str) id - the value the object’s id is set to.
* (str) name - the value the object’s name is set to.

### Optional Parameters

* (bool) isPublic - the value the object’s isPublic variable is set to.  
  Default Value: False
* (int) rating - the value the object’s rating is set to  
  Default Value: 1
* ([Trigger]) triggers - the value the object’s triggers variable is set to  
  Default Value: []
* ([FunctionCode]) code - the value the object’s code variable is set to  
  Default Value: []
* (bool) isActive - the value the object’s isActive variable is set to  
  Default Value: False

## Public Methods

### RunCode(self)

#### Description

Runs the code stored in code, by calling the \_\_translateSchedule function repeatedly and then resetting the values

#### Required Parameters

* self

### findDevices(self)

#### Description

Gets the list of all devices by analysing the code value

#### Required Parameters

* self

#### Return Value

([Devices]) Returns the list of devices the Schedule uses.

### initDevices(self)

#### Description

Checks if all the devices connected to the schedule can be accessed and calls the functions to initiate connections.

#### Required Parameters

* self

#### Return Value

({Device: bool}) Returns a Dictionary of all the devices and whether or not the connection was successful.

## Private Methods

### \_\_translateSchedule(self, index)

#### Description

A recursive function called by runCode that interprets self.code to a python script that can then be run.

#### Required Parameters

* self

#### Optional Parameters

* (int) index - the index of the FunctionCode in self.code that is being translated.  
  Default Value: 0

#### Return Value

(int) Returns the next position of index so that the code can be continually called until the end of the program

### \_\_runConditional(self, index)

Description

A supporting function called by \_\_translateSchedule that Translates all the FunctionCodes in a conditional statement or Loop.

Required Parameters

* self
* (int) index - the index of the FunctionCode in self.code that the function starts its loop at.

### \_\_FindEnd(self, index, statement)

Description

A supporting function called by \_\_translateSchedule that searches for where the end of a specific conditional or loop statement ends. Analogous to finding the matching } to a {.

Required Parameters

* self
* (int) index - the index of the FunctionCode in self.code that the function starts looking for the end from.
* (FunctionCode) statement - the statement that \_\_FindEnd is looking for the end to.

#### Return Value

(int) Returns the index that the end of the conditional statement/loop is at.

### \_\_resetCounts(self)

Description

A supporting function called by runCode that resets how many times a FunctionCode has run.

Required Parameters

* Self

### \_\_resolveDevice(self, location)

Description

A supporting function called by findDevices that resolves an instance of a device being accessed to an evaluable string

Required Parameters

* self
* (str) location – the location of the device that is being resolved

#### Return Value

(str) the evaluable string that is created in the function

### \_\_addToErrorLog(self, exception)

Description

If the user has email updates on, this function emails the user, warning them if a schedule is unable to run.

Required Parameters

* self
* (str) errorMessage - the message that is sent to the user, informing them of the error.