Fable

Programming Reference

Updated: 19-12-2017

Contents

Chapter 1: Interface Overview

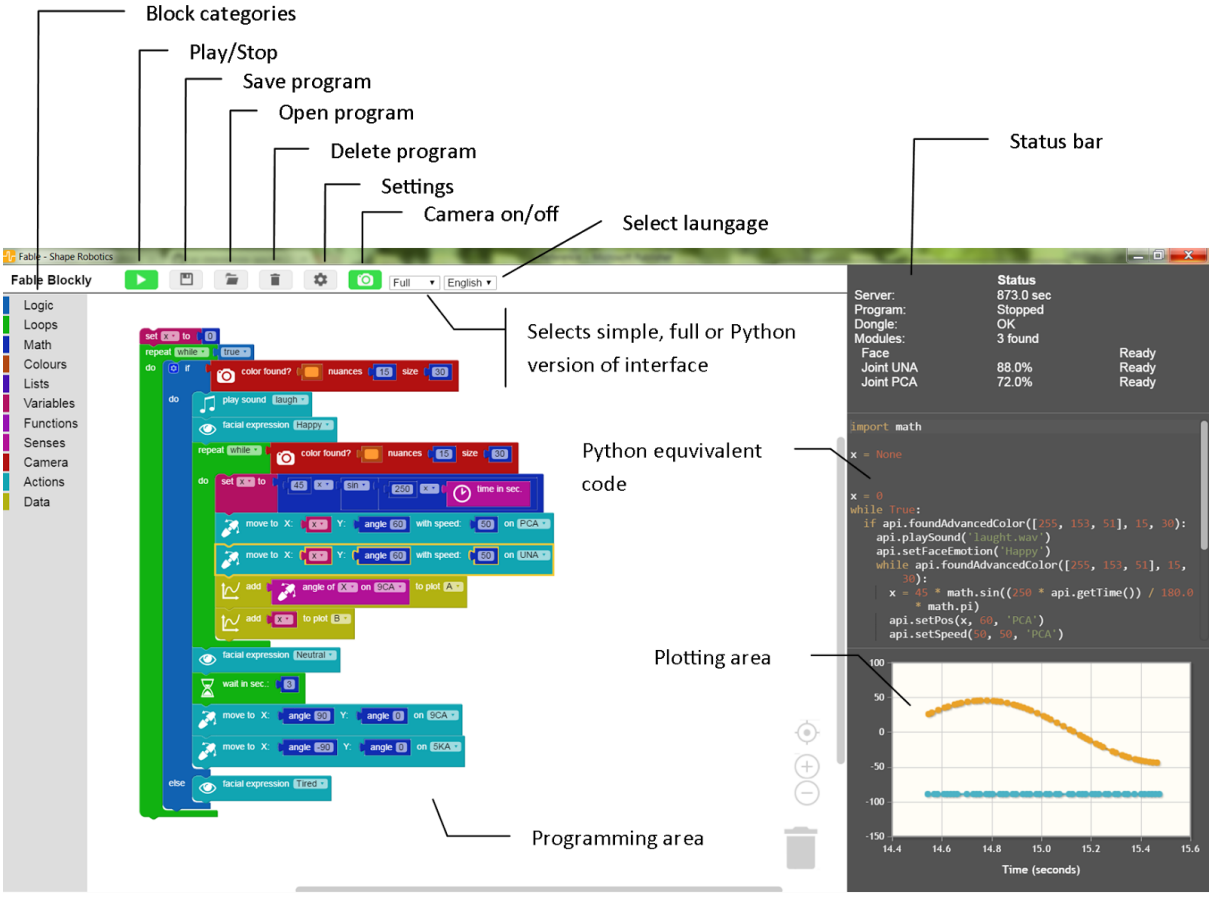
The Fable PC Interface - Blockly

Users can program their robots by using the Fable software. The interface supports both block-[1] and text-[2] based programming. The software supports three complexity levels:

Simple: Block based programming for beginner or younger children. A limited number of blocks makes the interface easily accessible.

Full: Blocks based programming for more experienced or older children. Support for more advanced features such as math, functions, logic and variables.

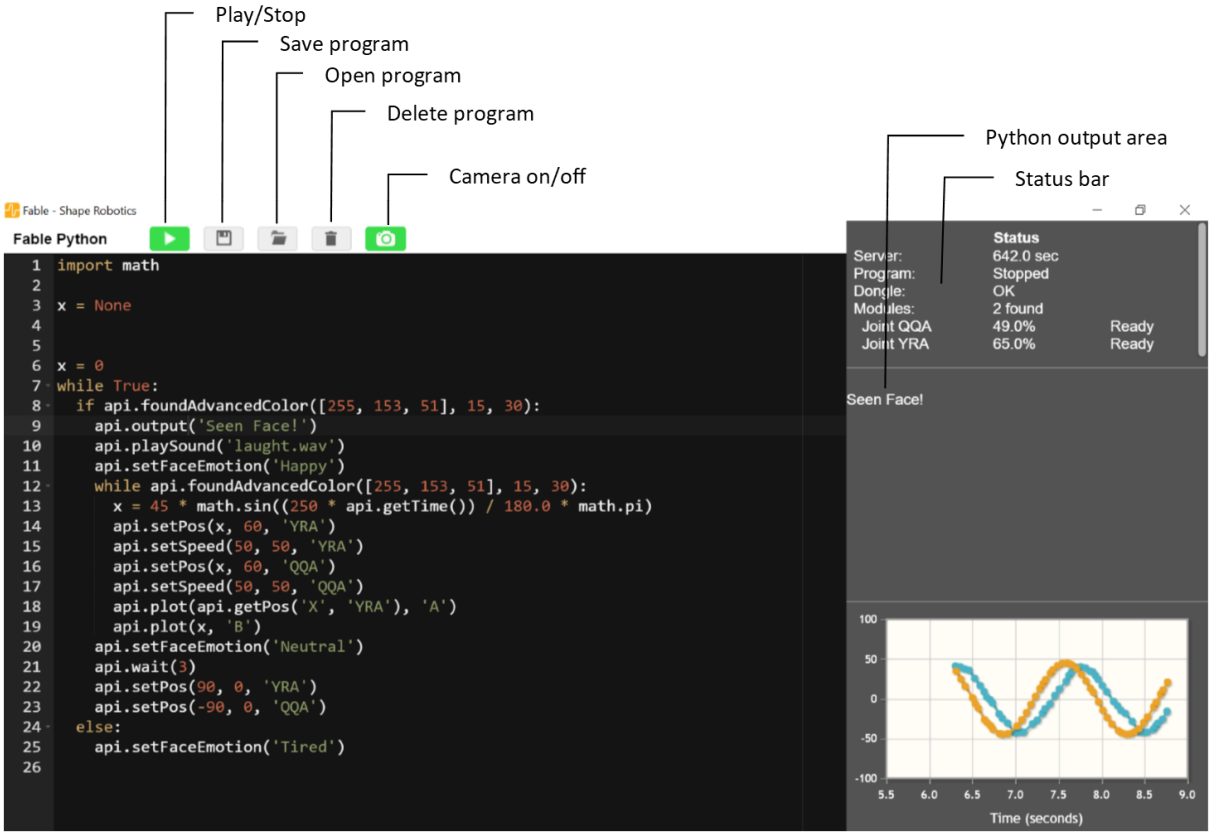
Python: Text based programming using the Python programming language.



[1] Based on Google Blockly
[2] Based on Python

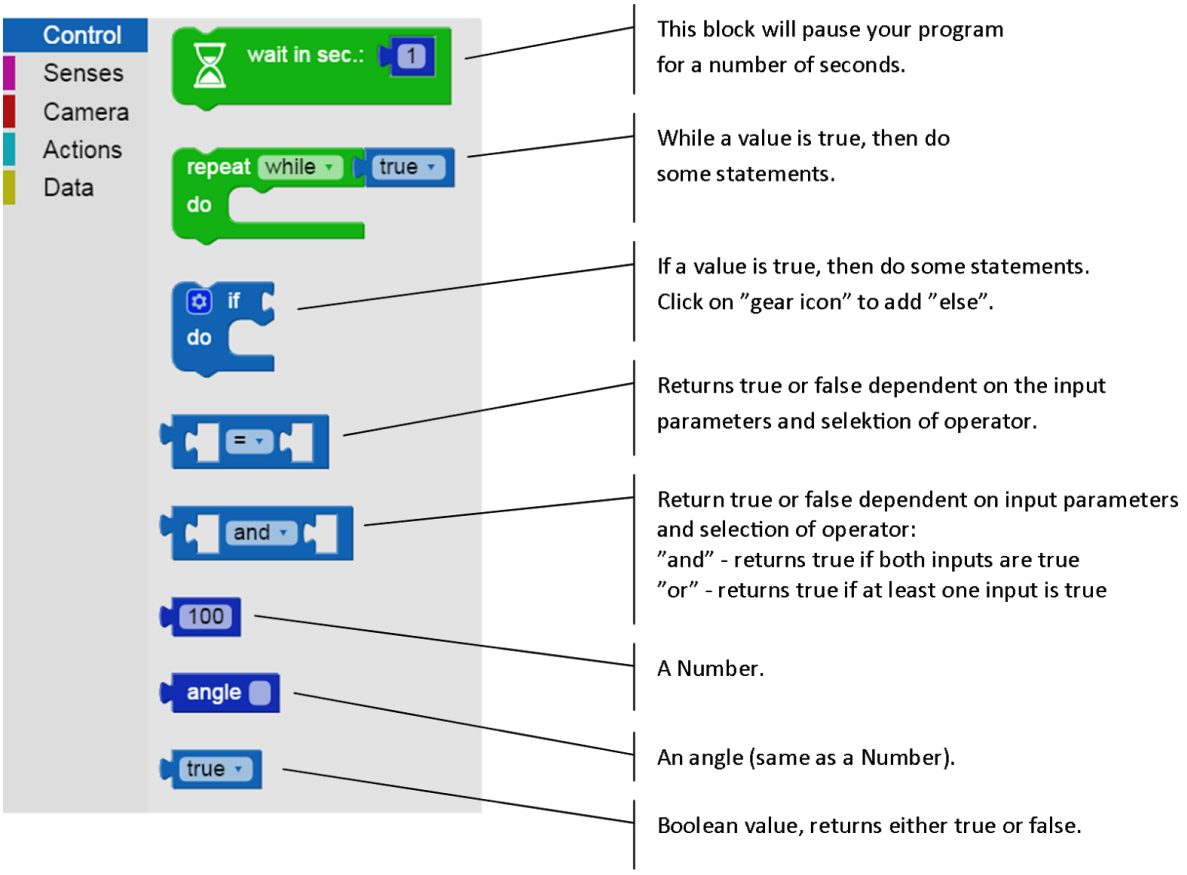
The Fable PC Interface - Python

The Fable robots can be programmed in Python. Any block program will automatically be translated into Python code which can then be modified and extended by the user.

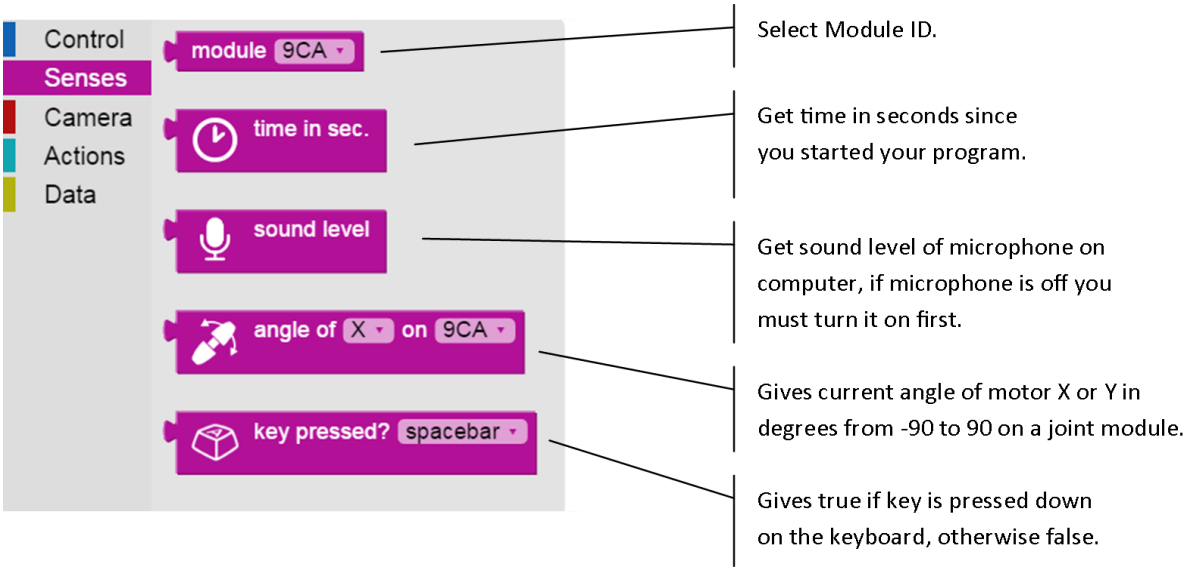


Chapter 2: Simple View Blocks

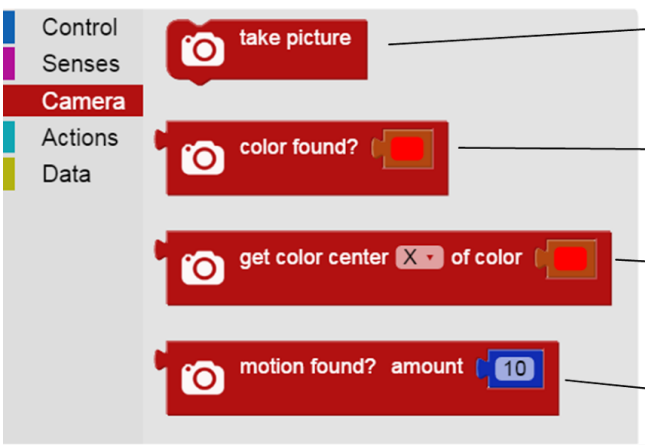
Control



Senses



Camera



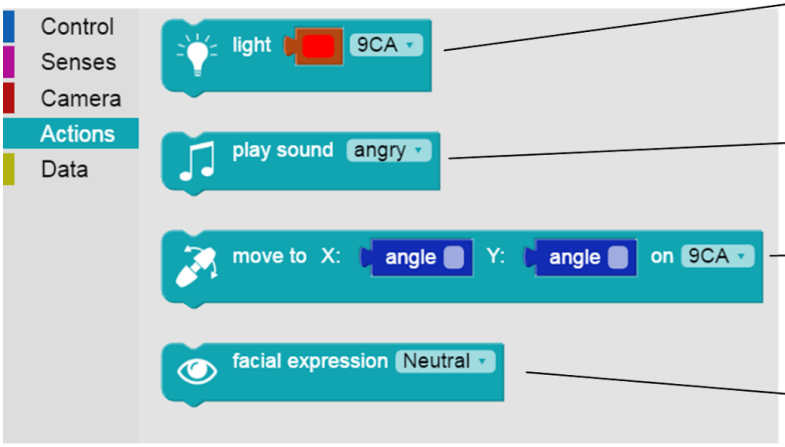
Save a picture from the camera, in folder "Fable pictures" on the desktop.

Returns true if the color is found in the image of the camera.

Gives X or Y coordinate of color center found in the image of the camera.

Returns true if motion is found. Amount is the minimum percentage of motion in the image required to trigger a detection.

Actions



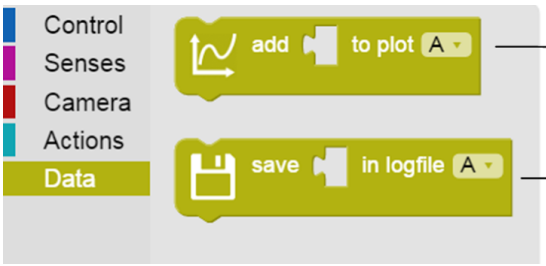
Change the light color on a module or dongle (will not change the communication channel).

Playback a standard audio sound on the computer.

Set the angle of motor X and Y that a joint module should move to, the angle is in degrees from -90 to 90.

Set the emotion of the robot's Face app (smartphone app).

Data

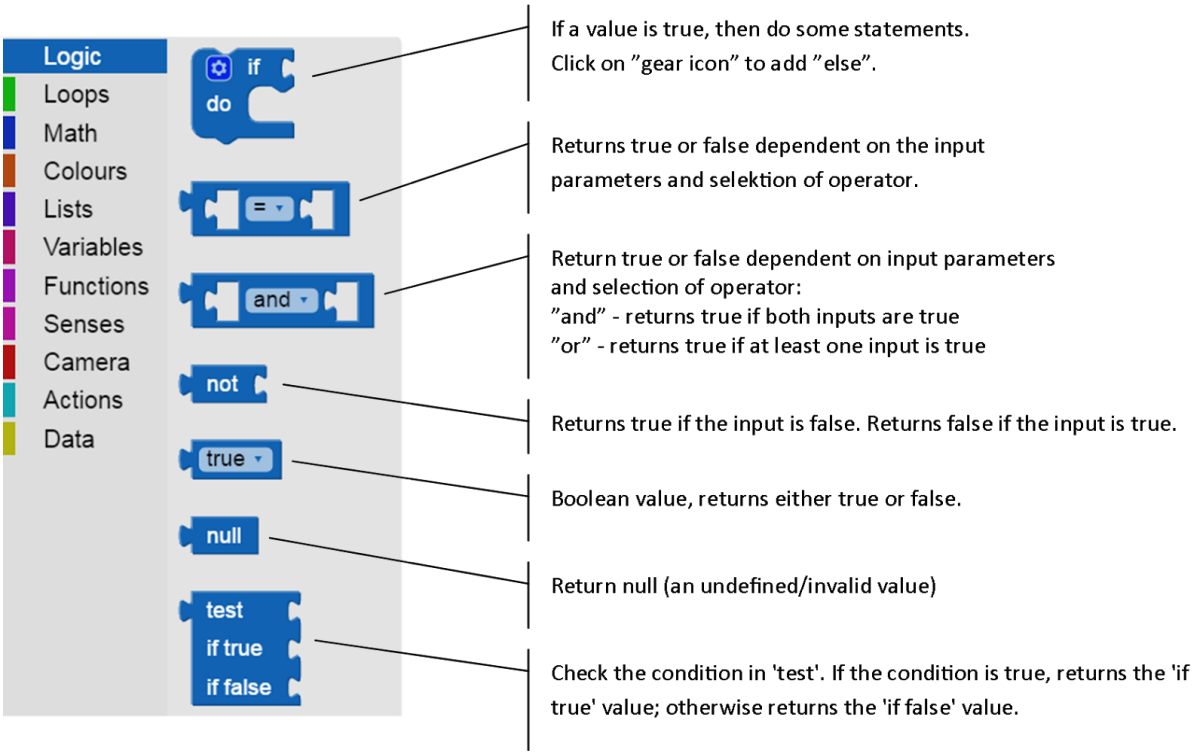


Add a data point to a time series plot which will be shown in the interface.

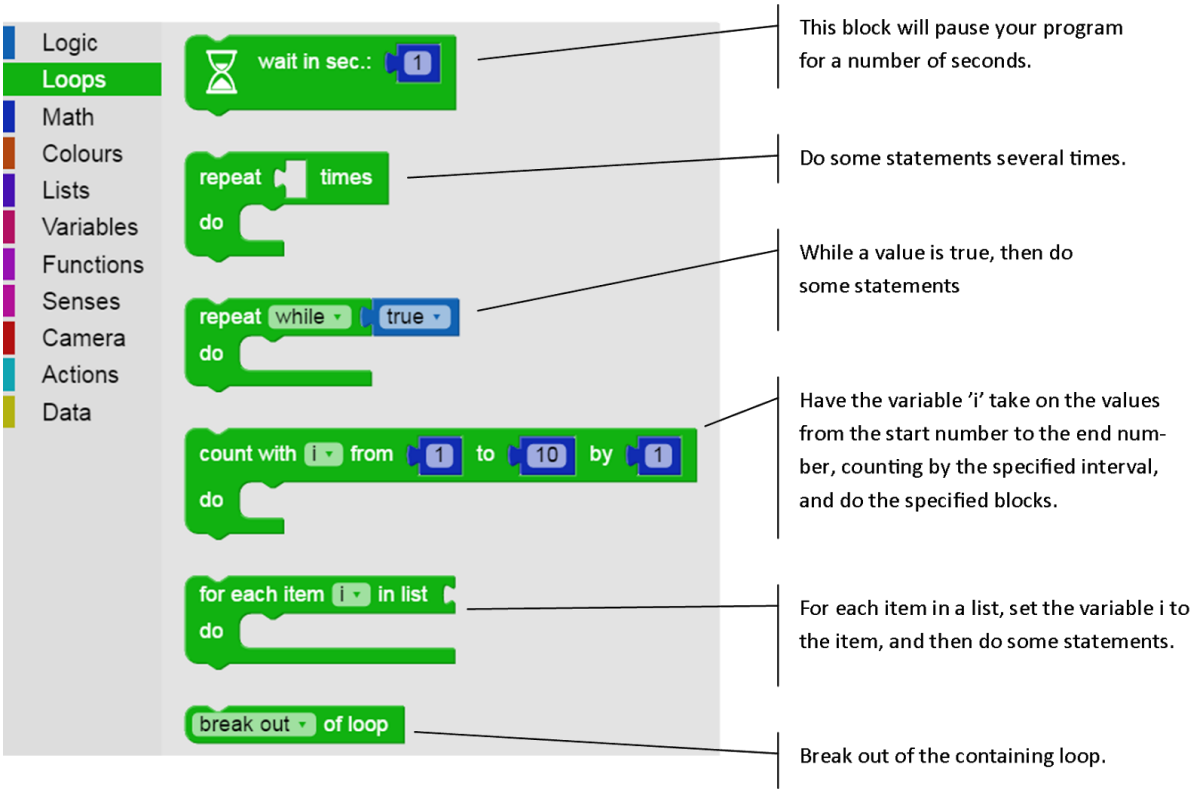
Add data to a log file which is created on your desktop, can be opened e.g. in MS Excel.

Chapter 3: Full View Blocks

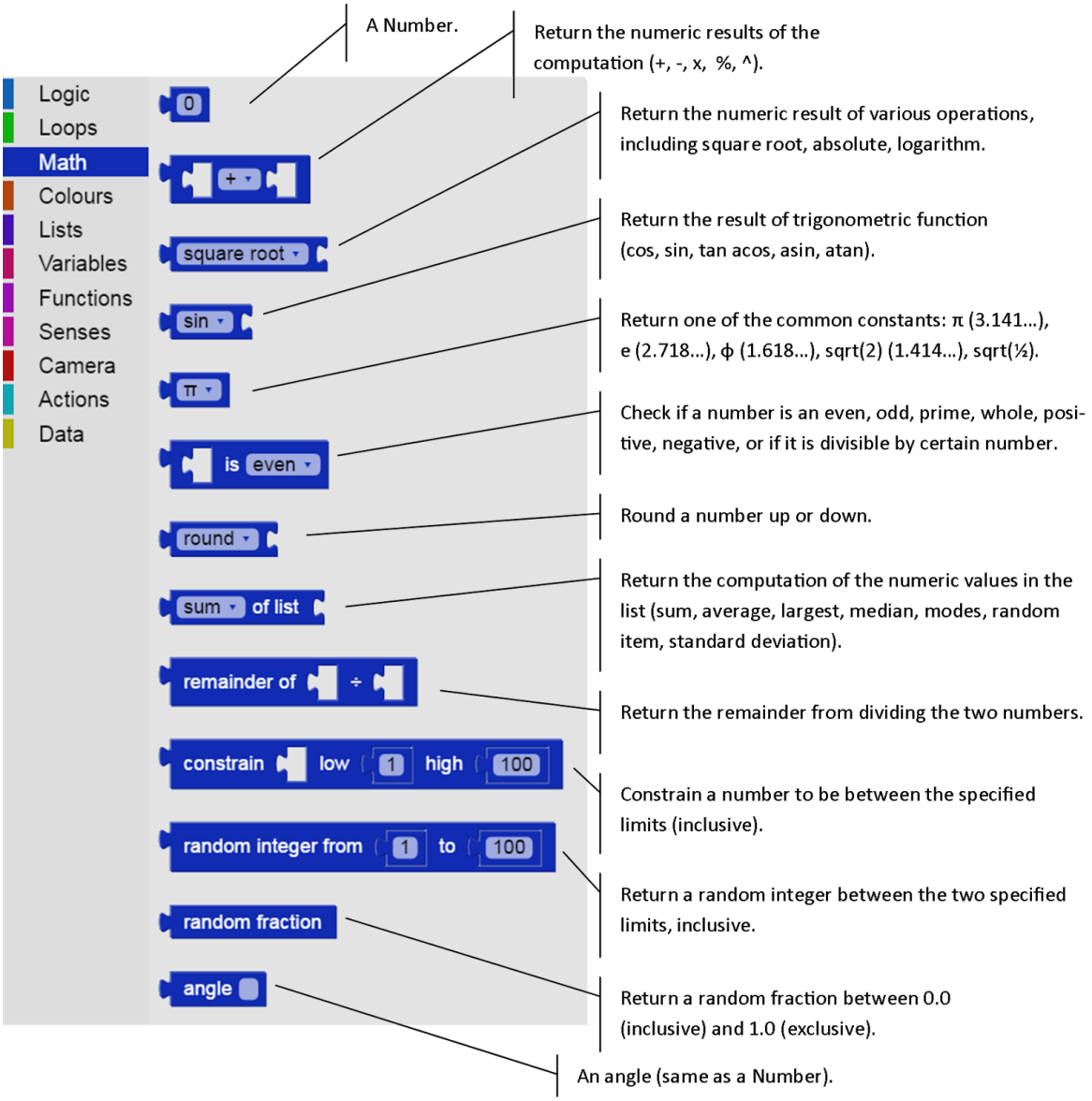
Logic



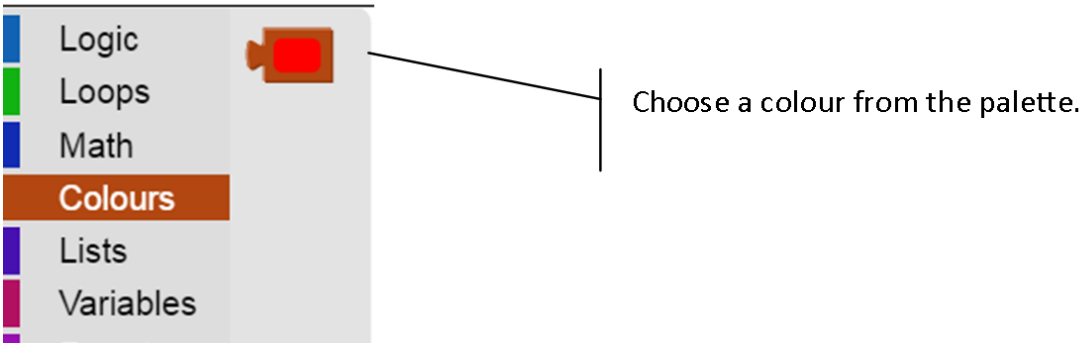
Loops


Logic
Loops
Math
Colours
Lists
Variables
Functions
Senses
Camera
Actions
Data
wait in sec.: 1
repeat times
do
repeat while true
do
count with i from 1 to 10 by 1
do
for each item i in list
do
break out of loop
This block will pause your program for a number of seconds.
Do some statements several times.
While a value is true, then do some statements
Have the variable 'i' take on the values from the start number to the end number, counting by the specified interval, and do the specified blocks.
For each item in a list, set the variable i to the item, and then do some statements.
Break out of the containing loop.

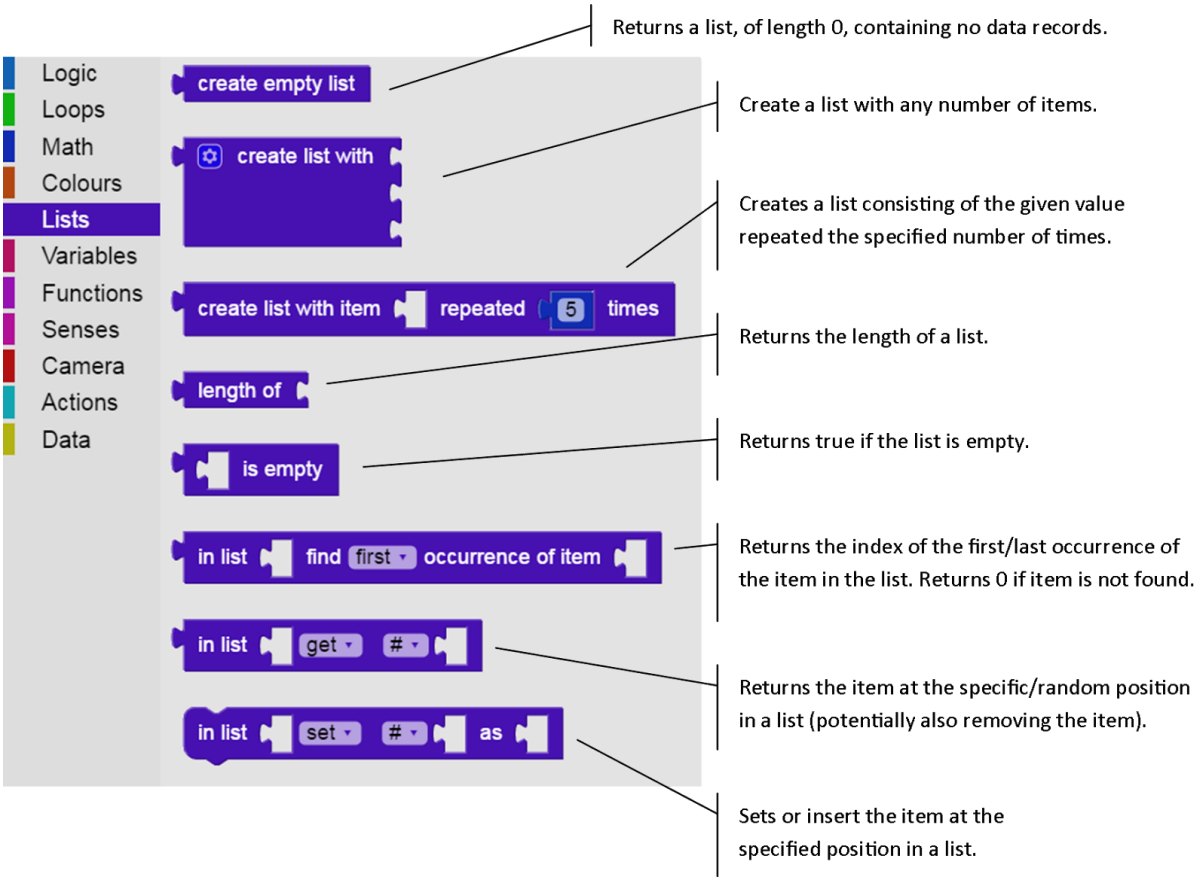
Math



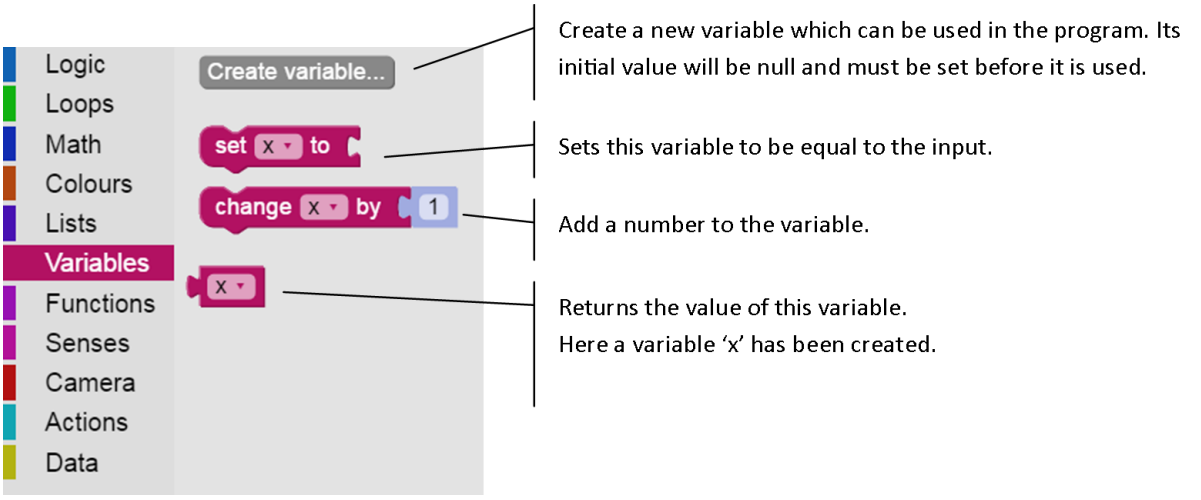
Colours



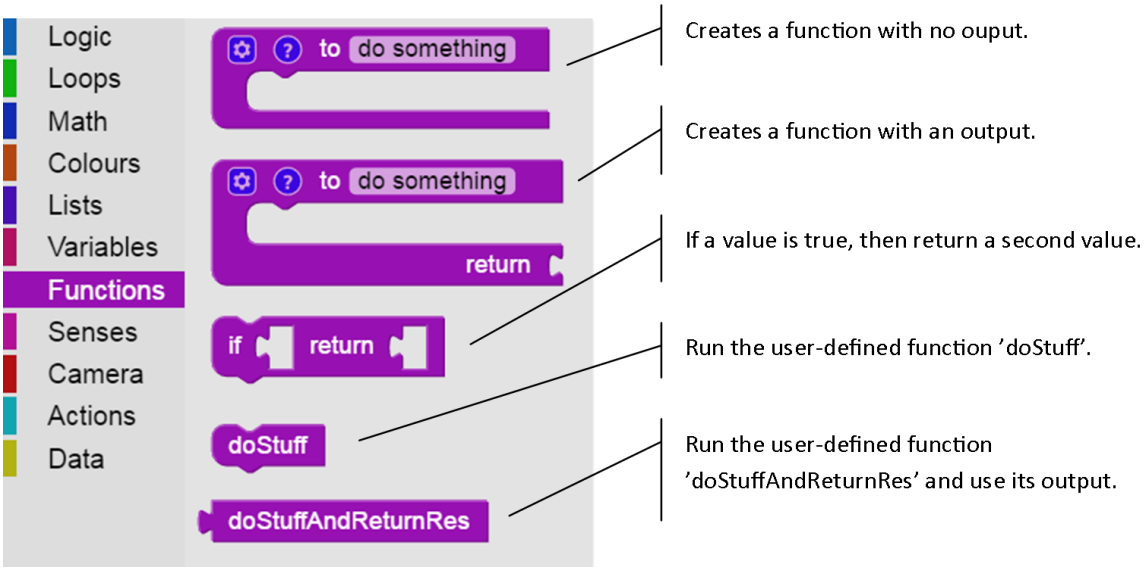
Lists



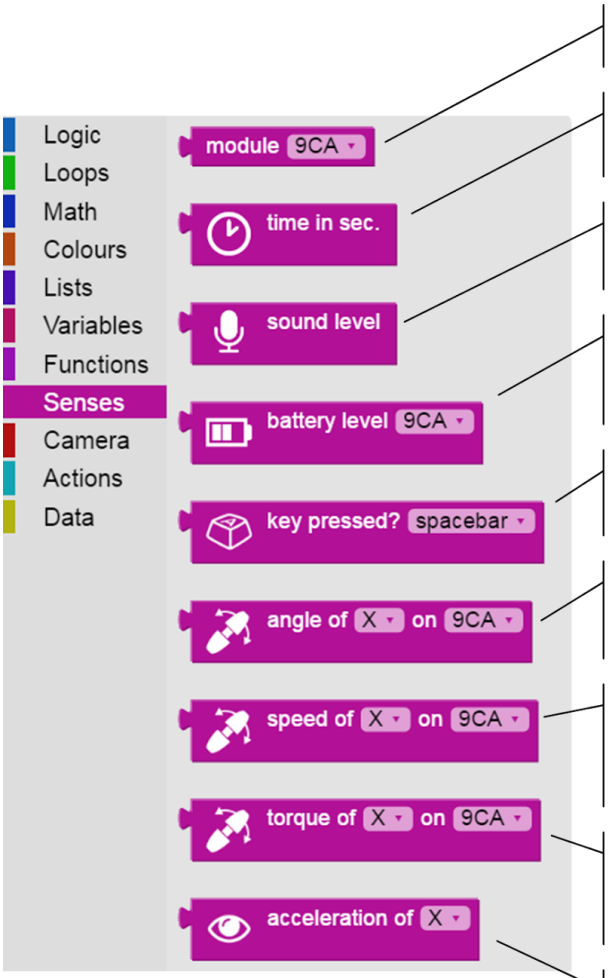
Variables



Functions


Logic
Loops
Math
Colours
Lists
Variables
Functions
Senses
Camera
Actions
Data
to do something
to do something
return
if return
doStuff
doStuffAndReturnRes
Creates a function with no ouput.
Creates a function with an output.
If a value is true, then return a second value.
Run the user-defined function 'doStuff'.
Run the user-defined function 'doStuffAndReturnRes' and use its output.

Senses



Select Module ID.

Get time in seconds since you started your program.

Get sound level of microphone on computer, if microphone is off you must turn it on first.

Read battery level of a module in percent from 0 to 100%.

Gives true if key is pressed down on the keyboard, otherwise false.

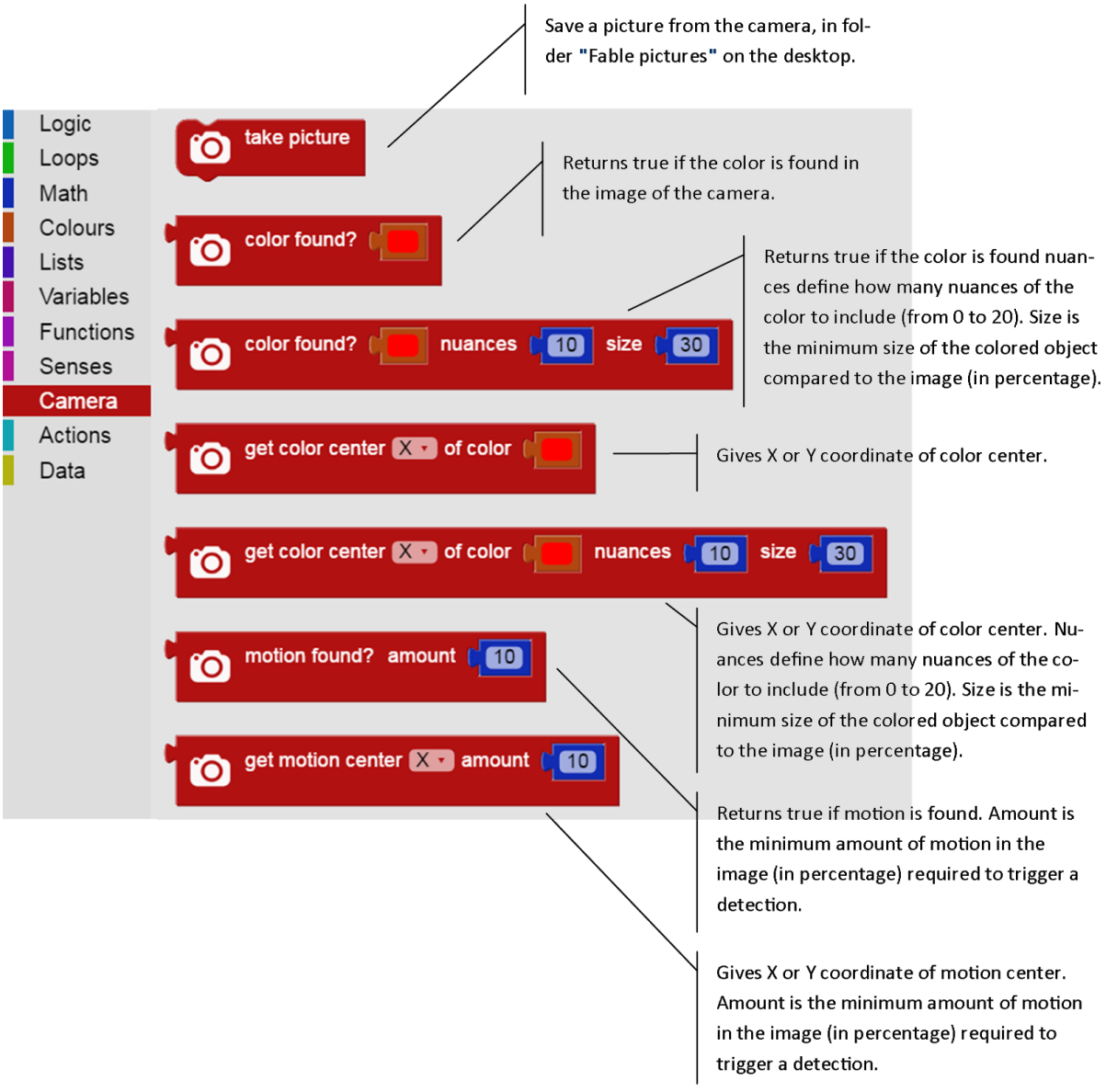
Gives current angle of motor X or Y in degrees from -90 to 90 on a joint module.

Give current angular velocity of motor X or Y in percent -100% to 100% on a joint module. The plus-minus sign indicates the direction of movement.

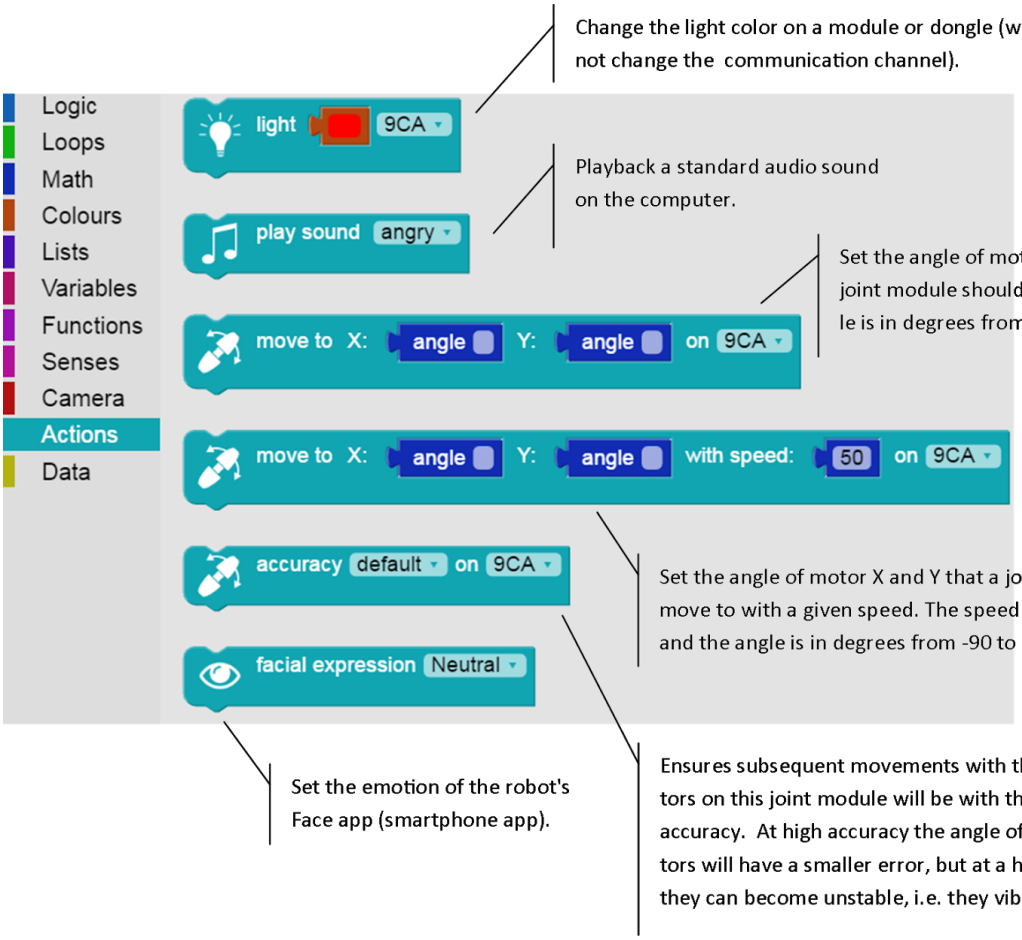
Give current torque of motor X or Y in percent -100% to 100% on a joint module. The plus-minus sign indicates the direction of the torque.

Give current acceleration in direction X, Y or Z of the smartphone with the Face App.

Camera


Logic
Loops
Math
Colours
Lists
Variables
Functions
Senses
Camera
Actions
Data
take picture
color found?
color found? nuances 10 size 30
get color center X of color
get color center X of color nuances 10 size 30
motion found? amount 10
get motion center X amount 10
Save a picture from the camera, in folder "Fable pictures" on the desktop.
Returns true if the color is found in the image of the camera.
Returns true if the color is found nuances define how many nuances of the color to include (from 0 to 20). Size is the minimum size of the colored object compared to the image (in percentage).
Gives X or Y coordinate of color center.
Gives X or Y coordinate of color center. Nuances define how many nuances of the color to include (from 0 to 20). Size is the minimum size of the colored object compared to the image (in percentage).
Returns true if motion is found. Amount is the minimum amount of motion in the image (in percentage) required to trigger a detection.
Gives X or Y coordinate of motion center. Amount is the minimum amount of motion in the image (in percentage) required to trigger a detection.

Actions



Change the light color on a module or dongle (will not change the communication channel).

Playback a standard audio sound on the computer.

Set the angle of motor X and Y that a joint module should move to, the angle is in degrees from -90 to 90.

Set the angle of motor X and Y that a joint module should move to with a given speed. The speed is from 0-100% and the angle is in degrees from -90 to 90.

Set the emotion of the robot's Face app (smartphone app).

Ensures subsequent movements with the motors on this joint module will be with the chosen accuracy. At high accuracy the angle of the motors will have a smaller error, but at a high load they can become unstable, i.e. they vibrate.

Data

Add a data point to a time series plot which will be shown in the interface.

Add data to a log file which is created on your desktop, can be opened e.g. in MS Excel.

Chapter 4: Python API

Quick overview of selected Python functions that can be used to program Fable robots.

Joint module

`def setPos(posX, posY, moduleID)`	Set the position of a specified joint from -90 to 90 degrees. E.g. `api.setPos(45, -60, 'PCA')`
`def setSpeed(speedX, speedY, moduleID)`	Sets speed of a specified joint -100% to 100% percent. Returns an instance of the joint with updated speed values. E.g. `api.setSpeed(50, 50, 'YRA')`
`def setTorque(torqueX, torqueY, moduleID)`	Sets the torque for a specified joint. Also sets the torque limit. This value ranges from -100% to 100% percent. E.g. `api.setSpeed(50, 50, 'YRA')`
`def setAccurate(accurateX, accurateY, moduleID)`	Modify the motor controller (compliance slopes, margins and punch) for the X and Y joint motors to make them move more accurate. E.g. `api.setAccurate('HIGH', 'DEFAULT', 'YRA')`
`def getPos(motorID, moduleID)`	Returns the position of one of the motors from a joint. The encoder has a resolution of 0.29 degrees. Motors can move from -90 to 90 degrees. E.g. `x = api.getPos('X', 'YRA')`
`def getSpeed(motorID, moduleID)`	Returns percentage representation from -100% to 100% of speed for one of the two motors from a specified joint. Note: is derived from the position sensor. E.g. `x = api.getSpeed('Y', 'YRA')`
`def getTorque(motorID, moduleID)`	Returns percentage representation from -100% to 100% of torque used by one of the two motors from a specified joint. Note: is derived from current flow in motors. `x = api.getTorque('Y', 'YRA')`

`def getMoving(motorID, moduleID)`	If a motor of a specified joint moves it returns True. E.g.
	`if api.getMoving('Y', 'YRA'):`
	`  api.output('It is moving!')`

`def getTemperature(motorID, moduleID)`	Returns the temperature of one of the two motors from a specified joint module (in degrees Celsius). E.g. `api.getTemperature('Y', 'QQA')`
`def getBattery(moduleID)`	Returns the battery level of a specified module. Returns number representation of battery level. E.g. `api.getBattery('QQA')`

Face app

`def setFaceEmotion(emotion)`	Set the emotional state in the Face. Valid arguments: 'Neutral':0, 'Happy': 1, 'Sad': 2, 'Angry': 3, 'Tired': 4 E.g. `api.setFaceEmotion('Sad')`
`def getFaceEmotion()`	Get the emotional state of the Face. Returns int as Emotional state, e.g. 1 for 'Happy'. E.g. `api.getFaceEmotion()`
`def getFaceAcceleration(direction)`	Get the acceleration of the smartphone running the face app. Valid directions are 'X', 'Y' or 'Z'. Returns acceleration of the selected axis in SI units e.g. `5.124` m/s^2. E.g. `api.getFaceAcceleration('X')`

Sounds and lights

`def getSoundLevel(milliseconds = 50)`	Returns the microphone sound level in RMS (default windows used is 50ms). E.g. `api.getSoundLevel()`
`def playBeep(frequency, duration, channel)`	Plays back a beep with a given frequency (Hz) for a given duration in seconds on a give channel (0-9). E.g. `api.setColor([255, 0, 0], 'QQA')`

`def playSound(filename)`	Play back a sound file from the installation folder: Fable\resources\app\app\sounds E.g. `api.playSound('angry.wav')`
`def setColor(color, moduleID)`	Sets light color to a specified module/dongle. The color is an RGB list of numbers scaled from 0-255. E.g. `api.setColor([255, 0, 0], 'QQA')`

Camera

`def takePicture()`	Save a picture from the camera, in folder "Fable pictures\" on the desktop E.g. `api.takePicture()`
`def foundSimpleColor(color)`	Looks in the image for a specific blob of color. Returns true if it finds it. E.g. `api.foundSimpleColor([102, 255, 51])`
`def foundAdvancedColor(color, nuances, size)`	Looks in the image for a specific blob. Returns true if the color is found. Nuances of the color to include (from 0 to 20). Size is the minimum size of the colored object compared to the image (in percentage). E.g. `api.foundAdvancedColor([153, 102, 255], 10, 30)`
`def getSimpleColorCenter(color)`	Returns an (x, y) list of the position of the colored blob in the image. (0, 0) is at the center. x and y scales from -100 to 100. E.g. `api.getSimpleColorCenter([255, 0, 255])[0]`
`def getAdvancedColorCenter(color, nuances, size)`	Returns an (x, y) list of the position of the colored blob in the image, with a minimum size and a range of nuances. (0, 0) is at the center. x and y scales from -100 to 100. E.g. `api.getAdvancedColorCenter([0, 255, 204], 10, 30)[1]`

`def detectedMotion(threshold)`	Returns True if motion is found. Threshold is the minimum amount of motion in the image (in percentage) required to trigger a detection. E.g. `api.detectedMotion(2)`
`def getMotionCenter(threshold)`	Returns an (x, y) list of the motion center. (0, 0) is at the center of the image. Threshold is the minimum amount of motion in the image (in percentage) required to trigger a detection. E.g. `api.getMotionCenter(2)[0]`

Setup and termination

`def setup(defaultPort=None, blocking=False)`	Setup the API and tries to connects to a dongle. If blocking=True it will not terminate before a dongle is connected. E.g. `api.setColor([255, 0, 0], 'QQA')` Note: This function should only be used when programming standalone, not in Fable interface.
`def terminate()`	Stops the dongle and sends a command to all modules to go to their default state. E.g. `api.terminate()` Note: This function should only be used when programming standalone, not in Fable interface.

Module handling

`def discoverModules()`	Checks for connected modules and populates a list with their details. Returns a list of found module IDs. E.g. `api.discoverModules()` Note: This function should only be used when programming standalone, not in Fable interface.
`def getModuleIDs()`	Returns a list of all module IDs seen. E.g. `api.getModuleIDs()`

Time and wait

`def getTime()`	Returns time elapsed since start of the program, in seconds. E.g. `api.getTime()`
`def wait(seconds)`	Creates a pause in the program for a given amount of seconds. E.g. `api.wait(0.1)`

Testing communication

`def getConnectionQuality(moduleID)`	Returns the connection quality of a specified module. E.g. `api.getConnectionQuality('QQA')`
`def checkDongleConnection()`	Checks dongle connection. Returns true if dongle is connected. E.g. `api.checkDongleConnection()`

Print, plot and data logs

`def output(*args)`	Print out data received as arguments in the output window. E.g. `api.output('Hello World', 5)` Note: This function should only be used when programming in the Fable interface, not standalone (use `print('Hello World', 5)` instead).
`def plot(data, label, windowSize = 50, windowTime = 2.5)`	Draws a time series plot. Adds a datapoint to the plot which will be shown in the interface. E.g. `api.plot(100, 'A')` Note: This function should only be used when programming in the Fable interface, not standalone (use `print('Hello World', 5)` instead).
`def log(data, filename = 'log.csv')`	Writes data to a log file which is placed on the desktop. E.g. `api.plot(100, 'A')`

Chapter 5: Support Contact

Contact

Support information are available online at http://shaperobotics.com
For personalized support contact Shape Robotics at: support@shaperobotics.com