

# Myro

## Overview

Below is a chapter by chapter summary of all the Myro features introduced in this text. For a more comprehensive listing of all the Myro features you should consult the Myro Reference Manual.

### Chapter 1

```
from myro import *
```

This command imports all the robot commands available in the Myro library. We will use this whenever we intend to write programs that use the robot.

```
initialize(<PORT NAME>)
```

```
init(<PORT NAME>)
```

This command establishes a wireless communication connection with the robot. <PORT NAME> is determined at the time you configured your software during installation. It is typically the word `com` followed by a number. For example, "`com5`". The double quotes (") are essential and required.

```
beep(<TIME>, <FREQUENCY>)
```

Makes the robot beep for <TIME> seconds at frequency specified by <FREQUENCY>.

```
getName()
```

Returns the name of the robot.

`setName (<NEW NAME>)`

Sets the name of the robot to `<NEW NAME>`. The new name should be enclosed in double quotes, no spaces, and not more than 16 characters long. For example: `setName ("Bender")`.

`gamepad ()`

Enables manual control of several robot functions and can be used to move the robot around.

## Chapter 2

`backward (SPEED)`

Move backwards at `SPEED` (value in the range -1.0...1.0).

`backward (SPEED, SECONDS)`

Move backwards at `SPEED` (value in the range -1.0...1.0) for a time given in `SECONDS`, then stop.

`forward (SPEED)`

Move forward at `SPEED` (value in the range -1.0..1.0).

`forward (SPEED, TIME)`

Move forward at `SPEED` (value in the range -1.0...1.0) for a time given in seconds, then stop.

`motors (LEFT, RIGHT)`

Turn the left motor at `LEFT` speed and right motor at `RIGHT` speed (value in the range -1.0...1.0).

`move (TRANSLATE, ROTATE)`

Move at the `TRANSLATE` and `ROTATE` speeds (value in the range -1.0...1.0).

`rotate (SPEED)`

Rotates at `SPEED` (value in the range -1.0...1.0). Negative values rotate right (clockwise) and positive values rotate left (counter-clockwise).

`stop()`

Stops the robot.

`translate(SPEED)`

Move in a straight line at `SPEED` (value in the range -1.0...1.0). Negative values specify backward movement and positive values specify forward movement.

`turnLeft(SPEED)`

Turn left at `SPEED` (value in the range -1.0...1.0)

`turnLeft(SPEED, SECONDS)`

Turn left at `SPEED` (value in the range -1.0..1.0) for a time given in seconds, then stops.

`turnRight(SPEED)`

Turn right at `SPEED` (value in the range -1.0..1.0)

`turnRight(SPEED, SECONDS)`

Turn right at `SPEED` (value in the range -1.0..1.0) for a time given in seconds, then stops.

`wait(TIME)`

Pause for the given amount of `TIME` seconds. `TIME` can be a decimal number.

### Chapter 3

`speak(<something>)`

The computer converts the text in `<something>` to speech and speaks it out. `<something>` is also simultaneously printed on the screen. Speech generation is done synchronously. That is, anything following the `speak` command is done only after the entire thing is spoken.

`speak(<something>, 0)`

The computer converts the text in `<something>` to speech and speaks it out. `<something>` is also simultaneously printed on the screen. Speech generation

is done asynchronously. That is, execution of subsequent commands can be done prior to the text being spoken.

`timeRemaining(<seconds>)`

This is used to specify timed repetitions in a while-loop (see below).

## Chapter 4

`randomNumber()`

Returns a random number in the range 0.0 and 1.0. This is an alternative Myro function that works just like the `random` function from the Python `random` library (see below).

`askQuestion(MESSAGE-STRING)`

A dialog window with `MESSAGE-STRING` is displayed with choices: 'Yes' and 'No'. Returns 'Yes' or 'No' depending on what the user selects.

`askQuestion(MESSAGE-STRING, LIST-OF-OPTIONS)`

A dialog window with `MESSAGE-STRING` is displayed with choices indicated in `LIST-OF-OPTIONS`. Returns option string depending on what the user selects.

`currentTime()`

The current time, in seconds from an arbitrary starting point in time, many years ago.

`getStall()`

Returns `True` if the robot is stalled when trying to move, `False` otherwise.

`getBattery()`

Returns the current battery power level (in volts). It can be a number between 0 and 9 with 0 indication no power and 9 being the highest. There are also LED power indicators present on the robot. The robot behavior becomes erratic when batteries run low. It is then time to replace all batteries.

## Chapter 5

`getBright()`

Returns a list containing the three values of all light sensors.

`getBright(<POSITION>)`

Returns the current value in the <POSITION> light sensor. <POSITION> can either be one of 'left', 'center', 'right' or one of the numbers 0, 1, 2.

`getGamepad(<device>)`

`getGamepadNow(<device>)`

Returns the values indicating the status of the specified <device>. <device> can be "axis" or "button". The `getGamepad` function waits for an event before returning values. `getGamepadNow` immediately returns the current status of the device.

`getIR()`

Returns a list containing the two values of all IR sensors.

`getIR(<POSITION>)`

Returns the current value in the <POSITION> IR sensor. <POSITION> can either be one of 'left' or 'right' or one of the numbers 0, 1.

`getLight()`

Returns a list containing the three values of all light sensors.

`getLight(<POSITION>)`

Returns the current value in the <POSITION> light sensor. <POSITION> can either be one of 'left', 'center', 'right' or one of the numbers 0, 1, 2. The positions 0, 1, and 2 correspond to the left, center, and right sensors.

`getObstacle()`

Returns a list containing the two values of all IR sensors.

`getObstacle(<POSITION>)`

Returns the current value in the <POSITION> IR sensor. <POSITION> can either be one of 'left', 'center', or 'right' or one of the numbers 0, 1, or 2.

`savePicture(<picture>, <file>)`

`savePicture([<picture1>, <picture2>, ...], <file>)`

Saves the picture in the file specified. The extension of the file should be ".gif" or ".jpg". If the first parameter is a list of pictures, the file name should have an extension ".gif" and an animated GIF file is created using the pictures provided.

`senses()`

Displays Scribbler's sensor values in a window. The display is updated every second.

`show(<picture>)`

Displays the picture in a window. You can click the left mouse anywhere in the window to display the (x, y) and (r, g, b) values of the point in the window's status bar.

`takePicture()`

`takePicture("color")`

`TakePicture("gray")`

Takes a picture and returns a picture object. When no parameters are specified, the picture is in color.

## Chapter 6 & 7

No new Myro features were introduced in these chapters.

## Chapter 8

`GraphWin()`

`GraphWin(<title>, <width>, <height>)`

Returns a graphics window object. It creates a graphics window with title, <title> and dimensions <width> x <height>. If no parameters are specified, the window created is 200x200 pixels.

`<window>.close()`

Closes the displayed graphics window <window>.

`<window>.setBackground(<color>)`

Sets the background color of the window to be the specified color. `<color>` can be a named color (Google: color names list), or a new color created using the `color_rgb` command (see below)

`color_rgb(<red>, <green>, <blue>)`

Creates a new color using the specified `<red>`, `<green>`, and `<blue>` values. The values can be in the range 0..255.

`Point(<x>, <y>)`

Creates a point object at (`<x>`, `<y>`) location in the window.

`<point>.getX()`

`<point>.getY()`

Returns the x and y coordinates of the point object `<point>`.

`Line(<start point>, <end point>)`

Creates a line object starting at `<start point>` and ending at `<end point>`.

`Circle(<center point>, <radius>)`

Creates a circle object centered at `<center point>` with radius `<radius>` pixels.

`Rectangle(<point1>, <point2>)`

Creates a rectangle object with opposite corners located at `<point1>` and `<point2>`.

`Oval(<point1>, <point2>)`

Creates an oval object in the bounding box defined by the corner points `<point1>` and `<point2>`.

`Polygon(<point1>, <point2>, <point3>, ...)`

`Polygon([<point1>, <point2>, ...])`

Creates a polygon with the given points as vertices.

`Text(<anchor point>, <string>)`

Creates a text anchored (bottom-left corner of text) at `<anchor point>`. The text itself is defined by `<string>`.

`Image(<centerPoint>, <file name>)`

Creates an image centered at `<center point>` from the image file `<file name>`. The image can be in GIF, JPEG, or PNG format.

All of the graphics objects respond to the following commands:

`<object>.draw(<window>)`

Draws the `<object>` in the specified graphics window `<window>`.

`<object>.undraw()`

Undraws `<object>`.

`<object>.getCenter()`

Returns the center point of the `<object>`.

`<object>.setOutline(<color>)`

`<object>.setFill(<color>)`

Sets the outline and the fill color of the `<object>` to the specified `<color>`.

`<object>.setWidth(<pixels>)`

Sets the thickness of the outline of the `<object>` to `<pixels>`.

`<object>.move(<dx>, <dy>)`

Moves the object `<dx>`, `<dy>` from its current position.

The following sound-related functions were presented in this chapter.

`beep(<seconds>, <frequency>)`

`beep(<seconds>, <f1>, <f2>)`

Makes the robot beep for `<seconds>` time at frequency specified. You can either specify a single frequency `<frequency>` or a mix of two: `<f1>` and `<f2>`.



```
<robot/computer object>.beep(<seconds>, <frequency>)  
<robot/computer object>.beep(<seconds>, <f1>, <f2>)
```

Makes the robot or computer beep for <seconds> time at frequency specified. You can either specify a single frequency <frequency> or a mix of two: <f1> and <f2>.

```
robot.playSong(<song>)
```

Plays the <song> on the robot.

```
readSong(<filename>)
```

Reads a song file from <filename>.

```
song2text(song)
```

Converts a <song> to text format.

```
makeSong(<text>)
```

```
text2song(<text>)
```

Converts <text> to a song format.

## Chapter 9

```
getHeight(<picture>)
```

```
getWidth(<picture>)
```

Returns the height and width of the <picture> object (in pixels).

```
getPixel(<picture>, x, y)
```

Returns the pixel object at x,y in the <picture>.

```
getPixels(<picture>)
```

When used in a loop, returns one pixel at a time from <picture>.

```
getRGB(pixel)
```

```
getRed(<pixel>)
```

```
getGreen(<pixel>)
```

```
getBlue(<pixel>)
```

Returns the RGB values of the <pixel>.

`makeColor(<red>, <green>, <blue>)`

Creates a color object with the given <red>, <green>, and <blue> values (all of which are in the range [0..255]).

`makePicture(<file>)`

`makePicture(<width>, <height>)`

`makePicture(<width>, <height>, <color>)`

Creates a picture object either by reading a picture from a <file>, or of the given <width> and <height>. If <color> is not specified, the picture created has a white background.

`pickAColor()`

Creates an interactive dialog window to select a color visually. Returns the color object corresponding to the selected color.

`pickAFile()`

Creates an interactive dialog window that allows user to navigate to a folder and select a file to open. Note: it cannot be used to create new files.

`repaint()`

`repaint(<picture>)`

Refreshes the displayed <picture>.

`savePicture(<picture>, <file>)`

`savePicture(<picture list>, <gif file>)`

Saves the <picture> in the specified file (a GIF or JPEG as determined by the extension of the <file>: .gif or .jpg). <picture list> is saved as an animated GIF file.

`setColor(<pixel>, <color>)`

`setRed(<pixel>, <value>)`

`setGreen(<pixel>, <value>)`

`setBlue(<Pixel>, <value>)`

Sets the color of <pixel> to specified <color> or <value>.

`show(<picture>)`

`show(<picture>, <name>)`

Displays the <picture> on the screen in a window named <name> (string).

```
takePicture()  
takePicture("gray")  
takePicture("blob")
```

Takes a picture from the Scribbler camera. It is a color picture by default, or grayscale (“gray”), or a filtered image based on the defined blob (“blob”). See chapter text for examples.

## **Chapter 10**

There were no new Myro features introduced in this chapter. Actually, when the chapter is complete it will have Myro primitives for neural nets/conx described here.

## **Chapter 11 & 12**

No new Myro features were introduced in this chapter.

# Index

## A

AAAI 18  
ABC song 68  
abstraction 36  
acos **183**, 326  
Aesop 43  
Aggressive, Vehicle#2 135-6  
Aibo 4  
algorithm 281, **292**, 302, 305  
Al Khwarizmi 283  
Alive, vehicle#1 131-5  
Anananova 317  
and 85, 86, **92**, 324  
Animated GIF 220  
AntiqueWhite 189  
append 111, **125**, 325  
Apple Strudel 280  
Artificial Intelligence 4, 88, **245-271**  
    movie 3, 245  
Asimov's Laws 17  
asin **183**, 326  
askQuestion 78, 79, **91**, 137, 332  
assignment 50, **66**, 323  
atan **183**, 326  
Avoiding obstacles 148-151

## B

backward 24, **38**, 330  
battery disposal 89  
beep 12, **16**, 201-205, **211**, 329, 336  
Behavior-based control 158  
Blobs 236  
blob filtering 236

Bluetooth 6, 8  
Bluetooth logo 18, 40  
blur image 229  
Boolean 85  
boundary conditions 296  
Braitenberg, Valentino 130, 153  
Braitenberg Vehicles 130-  
Breazeal, Cynthia 18, 317  
Brooks, Rodney 309-10, 317  
Burglar Alarm 145

## C

C 6  
C++ 293  
CamelCase 68  
case sensitive 48  
ceil 164, **182**, 326  
Celsius 67  
characters 52  
Chazelle, Brian 307  
Chess 250  
Chicken Kabobs 276-7  
Chinese Room 270  
Ching-Chong-Cha 177  
choice 257, **269**  
Circle 192, **210**, 335  
Clash, The 129  
close 189, **209**, 334  
Cockroach 153  
color\_rgb 194, **210**, 335  
CommonLisp 293  
comments 32  
computable 304

Computational Linguistics 248  
Computer Science 273, 275  
conditions 84, 87  
constant time algorithms 302  
Corral Exiting 151  
cos **183**, 326  
Coward, Vehicle#2 135-6  
currentTime 81-83, **91**, 332

**D**

Dar es Salaam 110, 112  
data 302  
Data Fountain 318  
dead reckoning 73  
decision making 176  
Deep Blue 250  
def 27, **39**, 118, 321  
degrees **183**, 326  
Delaware River 5  
Dijkstra, Edsger W. 273, 275  
Direct Control 159  
draw 190, **211**, 336

**E**

e **183**, 326  
edge detection 237  
Elvis 85  
emboss image 230  
Energy Problem 55  
enlarge image 227-8  
exp 139, 164, **182**, 326  
Explorer 43  
Explorer, Vehicle#3 138  
expression 50  
equal temperament 204  
Ericsson 18

Euclid v  
Error Checking 296-9  
Euro 67

**F**

Face Bank 312  
False 63, 84, 86, **91**, 324  
Fahrenheit 67  
Firefox 44  
floating point numbers 51  
floor 164, **182**, 326  
Flip Flap 312  
Flops 304  
Fluke Dongle 8, 9, 125  
Flying Circus 29  
Flynn, Anita M. 309-10  
Follower 146  
for 60, **66**, 83, 323  
formal languages 14  
forward 24, **38**, 330  
fractals 213  
Franke, Uli 313  
Franklin Institute 117  
from 34, 166, 322  
function 27, 34

**G**

Gamepad controller 14, 114-16  
gamepad 14, **16**, 108, 330  
Game playing 251-267  
Geier, Sven 187  
getBattery 88, 89, **91**, 332  
getBlob 237  
getBlue **238**, 337  
getBright 104-5, **122**, 332  
getGamepad 115-16, **122**, 333

getGamepadNow 115-16, **122**, 333  
getCenter 192, **211**, 336  
getGreen **238**, 337  
getHeight 218, **238**, 337  
getIR 106-7, **122**, 333  
getLight 103, **122**, 333  
getName 13, **16**, 329  
getObstacle 107-8, **123**, 333  
getP1 191  
getP2 192  
getPixel 222, **238**, 337  
getPixels 223, **238**, 337  
getRed **238**, 337  
getRGB 223, **238**, 337  
getStall 87, **91**, 332  
getX 190, **210**  
getY 190, **210**  
getWidth 218, **238**, 337  
GIF 217  
global name 196-7, 328  
Google 3  
Gore, Al 68  
Gormson, Harald B. 18  
GPS 314-15  
Grade A eggs 281  
GraphWin 188, **209**, 334  
Gray Poupon 47, 113  
grayscale images 102  
Gregorian Calendar 291

## H

Hallway Cruiser 145  
Hektor robot 314  
Hertz (Hz) 201  
hi-fidelity 202  
HiLo game 184

Hoare, C. A. R. 273  
Hogg, David 153  
Hugs & Kisses 251  
Human-robot interaction 315-16

## I

iCat robot 316  
IDLE 10, 27, 29, 36, 62  
Idle, Eric 29  
if-statement 120, **123**, 142, **154**. 324, 326  
image 216  
Image 201, **210**, 336  
image processing 226  
image understanding 232  
Imitation Game 246  
import 166, 322  
in 110, **124**, 325  
Indecisive 141  
init **16**, 329  
initialize 11, 12, **16**, 329  
input 56, **65**, 113, 322  
integers 51  
internet 5  
interoceptors 72  
invocation, function 28  
iPhone 85  
iRobot 2, 59

## J

jalapeno 72  
Jankenpon 177  
Java 6  
Joe, Gigolo 245  
Joel, Billy 187  
Jones, Crispin 311

Jones, Mick 129  
JPEG 217  
Julia Sets 213

**K**

Kasparov, Gary 250  
Kismet robot 317  
Kitaoka, Akiyoshi 215  
Koch Snowflakes 213  
Konane 250

**L**

Ladybug 129  
Larson, Doug 273  
LavenderBlush 189  
Law, Jude 245  
Leap Frog 313  
leap year 291-5  
Learning 267  
LED 89  
LEGO Mindstorms 3  
len 110, **124**, 325  
Lenhi, Jurg 313  
Light following 146-8  
Line 191, **210**, 335  
linear time algorithms 302  
List comprehensions 255  
lists 60, 109-13, 325  
Loan calculator 167-76  
local name 196-7, 328  
localtime **94**  
log 165, **183**, 326  
log10 165, **183**, 326  
logarithmic time algorithms 302  
logical operations 85  
loop 60

loop index variable 60  
Love, Vehicle#3 138  
Lousanne 314

**M**

main 65, 322  
makeColor 222, 224, **239**, 338  
makePicture 219, 221-, **239**, 338  
Mandelbrot Sets 213  
Mars Rover 1, 2, 4  
Martin, Fred 153  
math library 139-40, 164-6, 326  
Maze solver 151  
meaning of life 280  
Measuring Device 145-6  
Media Player 43  
Megapixel 217  
Mignot, Charles 318  
Minimax algorithm 262  
Minsvoort, Koert van 318  
MIT Media Lab 153, 317  
mixed case 68  
module 31, 33, 34, **299**  
Monty Python 6, 29  
Morris, Errol 310  
Moscow 110, 112  
motors 23, **38**, 330  
move (robot) 25, **38**, 330  
move (graphics object) 198, **211**, 336  
musical scale 204  
Myers, Mike 157  
Myro 6, 8, 13, 329  
myro Song Format 206

**N**

Names 48, 49, **65**, 163, 195-8, 322

NASA JPL 1, 3, 310  
Nash, Johnny 97  
Natural Language Understanding 248  
natural languages 14, 247  
Naughts & Crosses 251  
negative image 230  
New York 85, 110, 112  
Nexi robot 317  
Nike viii  
not 85, 86, **92**, 324  
notes 204  
numbers 50

## O

octave 204  
Ok Corral 151  
OLPC Project 22, 273-4  
Opportunity robot 1, 2, 17, 310  
or 85, 86, **92**, 324  
Orb Swarms 309, 319  
Osment, Haley Joel 71, 245-6  
Oval **210**, 335

## P

Papal Bull 290-1  
Paper Scissors Rock 177-82, 264-7  
parameters 30, 195-7  
Paranoid 141  
Paris 85  
Paro robot 23  
Pathfinder 17  
PB&J 265  
Pennsylvania 5  
Philadelphia 117  
pi **183**, 326  
pickAColor 222, **239**, 338

pickAFile **239**, 338  
pixels 102, 216-7  
playSong 208, **211**, 337  
Pleo 4, 20-22  
PNG 217  
Point 190, **210**, 335  
Polar coordinates 241  
Polka 206  
Polygon **210**, 335  
polynomial time algorithms 303  
Pope Gregory XIII 291  
pow 165, **183**, 326  
Powers, Austin 157  
print 46, **65**, 322  
Programming 6, 274  
Programming Language 6, 44, 279, **292**  
Proprioception 72  
proximity sensor 73  
Python 6, 44, 47  
Python Shell 10, 11

## Q

quadratic algorithms 303

## R

radians **183**, 326  
random 77, **92**, 93, 178, 324  
randomNumber **91**, 93, 332  
randint 77, **92**, 324  
range 60, **66**, 112, **124-5**, 325  
Reactive behaviors 143-152  
Reactive control 159  
read 117, **124**, 325  
readSong 208, **211**, 337  
Rectangle **210**, 335  
Refrigerator Detective 145



repaint 222, **239**, 338  
repetition 60, **66**, 176  
Resnick, Mitchel 153  
return 119, **124**, 324  
return values 163-4  
reverse 111, **125**, 325  
RGB 102, 193, 216, 223  
RoboCup 315  
Robot, definition 3, 4  
Robot Hall of Fame 18  
Robot Vision 232  
Rochambeau 177  
Rock Paper Scissors 177-82, 264-7  
Roomba 2, 3, 22, 59  
rotate 25, **38**, 330  
Rotating Snakes 214  
Royal Mail 5  
Run Module 46  
runic alphabet 18

**S**

Saab 167  
savePicture 100, **123**, 218-, **239**, 334, 338  
scale 204  
Scassellati, Brian 316  
Science Magazine 153  
scope 195  
Scribbler 6, 22  
Scribbler 208  
Scribbler drawings 313-14  
Scribbler sensors 99  
Sear, Cole 71, 72  
Searle, John 270  
sequential execution 176  
senses 99, 108, **123**, 334

Sensor Fusion 159  
setBackground 189, **209**, 335  
setBlue **239**, 338  
setColor 223, **239**, 338  
setFill 193, **211**, 336  
setGreen **239**, 338  
setName 13, **16**, 330  
setOutline 193, **211**, 336  
setPixel 225  
setRed **239**, 338  
setRGB 223  
setWidth 193, **211**, 336  
sharpen image 229  
shrink image 227-8  
shrinking factor 228-9  
show 100, **123**, 216, 218, **240**, 334, 338  
Shyamalan, M. Night 71, 72  
Sierpinski Triangles 213  
Simpson, Homer 42  
sin **183**, 326  
Sixth Sense movie 71, 72  
Snicker's moment 72  
Sojourner 17, 310  
solvable 304  
song2text **212**, 337  
SONY 4  
sort 111, **125**, 325  
Social Robotics 316  
Soviet Union 283  
Space Complexity 301-5  
speak 47, **64**, 125, 331  
Spitler, Phil 309  
split 113, **124**, 324  
Spirit robot 1, 2, 17, 310  
Spielberg, Steven 3, 245  
sqrt 164, **183**, 326

Squyres, Steve 1  
Start Python 9, **16**, 321  
strings 50, 52  
stop 25, **38**, 331  
Subsumption Architecture 160  
Sullivan, Jon 71  
syntax error 35  
urlopen 117, **124**, 325  
USDA 281-2

## T

Tag reading pen 313  
takePicture 100-2, **123**, 216-, **240**,  
334, 339  
Takada 312  
tan **183**, 326  
Tengu 311-12  
Testing 296-9  
text 200, **210**, 336  
Terra flops 304  
Thingamapoops 313  
Tic Tac Toe 250-264  
time **94**  
Space Complexity 301-5  
timeRemaining 62, **64**, 332  
Timid 141  
TOMY Company 312  
Toyota Prius 110  
Traffic Lights 157  
translate 25, **38**, 331  
True 63, 84, 86, **91**, 324  
Tumbleweed robot 2  
Turing, Alan 246, 270  
Turing Test 246  
turnLeft 24, **38-39**, 331  
turnRight 24, **39**, 331

## U

UGOBE Inc. 21  
uncomputable 304  
undraw **211**, 336  
Unicode 85  
Unimation 4  
unsolvable 304  
urllib 117

## V

Values 50, **65**  
variable 50  
Victoria Crater, Mars 3

## W

wait 29, **39**, 322, 331  
Wales 58  
Wall Detector 145  
Washington DC 301  
Washington state 301  
while 62, **66**, 83, 87, 323  
WhiteSmoke 189  
Wikipedia 18, 29  
Wong, Yingshun 157  
world population 53, 64, 89  
world wide web 5

## X

XO Laptop 273-4

## Y

Y2K Problem 307

## Z

Zamboni 58, 92  
Zefrank 313-14



## Scribblr: Myro Reference

