Package

pybricks	**pybricks** is a python package which contains all the modules to interact with EV3 Brick and associated hardware

Module

ev3brick	**ev3brick** is a module which contains *mehods* and *class methods* to deal with EV3 brick
ev3devices	**ev3devices** is a module that deals with *Motor* and *Sensor* classes
parameters	Special kind of Module which deals with constants (Ex: color BLUE, Post.A etc). Also some static sound file names etc.,
tools	Used for dealing with Timing and Datalogging. Provides ability print output to terminal, wait and stop watch etc.,
robotics	Provides drivebase functionality. This is the module that makes your EV3 Move in different dicrection.

ev3Brick

buttons()	*Method* that get a **List** of buttons pressed
light (Color)	*Method* that sets a back color on the brick. Refer to *parameters* module for Color constants

ev3Brick (cont)

sound.beep(frequency,duration,volume)	*classmethod* using **sound** class. Play a beep/tone
sound.beeps (number)	*classmethod* using **sound** class. Play a number of default beeps with a brief pause in between.
sound.file (file_name,volume)	*classmethod* using **sound** class. Play a sound file.
display.clear()	*classmethod* of display on the brick. Clear everything on the display.
display.text(text,coordinate)	*classmethod* of display. Takes inputs and displays on the brick LCD panel
battery.voltage	*classmethod* of battery. Get the voltage of the battery
battery.current	*classmethod* of battery. Get the current supplied by the brick

ev3devices

Motor (port, direction=Direction.CLOCKWISE, gears=None)	Class that provides dealing with each Motor
TouchSensor(port)	Deals with Touch
ColorSensor(port)	Reads Color
InfraredSensor(port)	Gets the distance using infrared light
UltrasonicSensor (port)	Gets the distance using sound waves
GyroSensor(port, direction=Direction.CLOCKWISE)	Used for measuring the robot's rotational motion. Helps in creating balncing Robots

Motor Methods

angle()	Get the rotation angle of the motor.
reset_angle (angle)	Reset the angle of the motor
speed()	Get the speed (angular velocity) of the motor.
stop (stop_type=Stop.COAST)	Stop the motor. Refer to *Stop* in parameters to see available stop types
run(speed)	Keep the motor running at a constant speed (angular velocity).

Motor Methods (cont)

run_time (speed, time, stop_type=Stop.COAST, wait=True)	Run the motor at a constant speed (angular velocity) for a given amount of time
run_angle (speed, rotation_angle, stop_type=Stop.COAST, wait=True)	Run the motor at a constant speed (angular velocity) by a given angle.
run_target (speed, target_angle, stop_type=Stop.COAST, wait=True)	Run the motor at a constant speed (angular velocity) towards a given target angle.
track_target (target_angle)	Track a target angle that varies in time. This method is useful in fast loops where the motor target changes continuously.
stalled()	Check whether the motor is currently stalled.
run_until_stalled (speed, stop_type=Stop.COAST, duty_limit=default)	Run the motor at a constant speed (angular velocity) until it stalls. The motor is considered stalled when it cannot move even with the maximum torque.

Motor Methods ...

set_dc_settings (duty_limit, duty_offset)	Configure the settings to adjust the behavior of the dc() command. This also affects all of the run commands, which use the dc() method in the background.
set_run_settings (max_speed, acceleration)	Configure the maximum speed and acceleration/deceleration of the motor for all run commands. This applies to the **run**, **run_time**, **run_angle**, **run_target**, or **run_until_stalled** commands you give the motor. See also the default parameters for each motor.
set_pid_settings (kp, ki, kd, tight_loop_limit, angle_tolerance, speed_tolerance, stall_speed, stall_time)	Configure the settings of the position and speed controllers. See also pid and the default parameters for each motor.

TouchSensor

pressed	Returns **true** or **false**

ColorSensor

color()	Measures the color of surface
ambient()	Measures the ambient light intensity
reflection()	Measure the reflection of a surface using a red light.
rgb()	Measure the reflection of a surface using a red, green, and then a blue light.

InfraredSensor

distance()	Measure the relative distance between the sensor and an object using infrared light.
beacon (channel)	Measure the relative distance and angle between the remote and the infrared sensor.
buttons (channel)	Check which buttons on the infrared remote are pressed.

Ultrasonic Sensor

distance silent=False)	Measure the distance between the sensor and an object using ultrasonic sound waves.
presence()	Check for the presence of other ultrasonic sensors by detecting ultrasonic sounds.

By **FLL Tech Tacos** (flltech2019)

cheatography.com/flltech2019/

Published 6th October, 2019.
Last updated 6th October, 2019.

Gyroscopic Sensor

speed()	Get the speed (angular velocity) of the sensor.
angle()	Get the accumulated angle of the sensor.
reset_angle (angle)	Set the rotation angle of the sensor to a desired value.

parameters

Port	...
Direction	...
Stop	...
Color	...
Button	...
Align	...
ImageFile	Information, LEGO, Objects and Eyes
SoundFile	Expressions (CHEERING etc.,), Information, Communication, Movements, Color, Mechanical, Animals, Numbers and System

Tools

print (value, ..., sep, end, file, flush)	Print values on the terminal or a stream.
wait(time)	Pause the user program for a specified amount of time.

Tools (cont)

StopWatch Class	A stopwatch to measure time intervals. Similar to the stopwatch feature on your phone. **Supported Methods:** time, pause, resume, reset

robotics

DriveBase (left_motor, right_motor, wheel_diameter, axle_track)	Class representing a robotic vehicle with two powered wheels and optional wheel caster(s).

robotics Methods

drive (speed, steering)	Start driving at the specified speed and turnrate, both measured at the center point between the wheels of the robot.
drive_time (speed, steering, time)	Drive at the specified speed and turnrate for a given amount of time, and then stop.
stop (stop_type=Stop.COAST)	Stop the robot.