Minecraft Pi Python API

Introduction

This document details the commands you can use as part of the Raspberry Pi Minecraft python library mcpi.

Once you have gone through the lesson you can use and experiment with these to do even more cool things with code and minecraft.

The library can be found at https://github.com/martinohanlon/mcpi and original documentation at https://www.stuffaboutcode.com/p/minecraft-api-reference.html

To keep things simple, this document is easy to print and only covers commands that can be run on a stock raspberry Pi. The API can do more with additional software, just look at the links above to find out how.

Structure

- minecraft.py
    - Class Minecraft - main class for connecting and interacting with the game
        - Class camera - changing camera angle and position
        - Class player - getting and changing the players position and setting
        - Class entity - getting and changing entities position and setting
        - Class events - retrieving events which have occured in the game
- block.py
    - Class Block - definition of a block, specifically its type
- event.py
    - Class BlockEvent - definition of a block event, specifically what event, what block and what player
- vec3.py
    - Class Vec3 - generic class for managing a 3 dimension vector (i.e. x,y,z)
- connection.py - internal module used by the api
- util.py - internal module used by the api

Class: Minecraft	
Main class for interacting with the Minecraft world, includes functions for creating a connection, modifying players and blocks and capturing events	
`.create()`	Create connection to Minecraft (address, port) => Minecraft object

	.create(address = "localhost", port = 4711)
	"Create connection to Minecraft (address, port) => Minecraft object"
	`#use default address and port`
	`mc = minecraft.Minecraft.create()`
	`#specify ip address and port`
	`mc = minecraft.Minecraft.create("192.168.1.1", 4711)`

.getBlock(x,y,z)	"Get block (x,y,z) => id:int"
	`#retrieves the block type at 0,0,0` `blockType = mc.getBlock(0,0,0)`
.getBlockWithData(x,y,z)	"Get block with data (x,y,z) => Block"
	`#retrieves the block type for the block at 0,0,0` `blockType = mc.getBlock(0,0,0)`
.setBlock(x,y,z)	"Set block (x,y,z,id,[data])"

	`#sets a block at an x, y, z co-ordinate to a particular type`
	`mc.setBlock(0,0,0,block.DIRT.id)`
	`#sets a block to a particular type and 'subtype'`
	`mc.setblock(0,0,0,block.WOOD.id, 1)`

.setBlocks(x0,y0,z0,x1,y1,z1,blockType, blockData)	"Set a cuboid of blocks (x0,y0,z0,x1,y1,z1,id,[data])"
	`#sets many blocks at a time, filling the gap between 2 sets of x, y, z co-ordinates` `mc.setBlocks(-1, -1, -1, 1, 1, 1, block.STONE.id)`
.getHeight(x,z)	"Get the height of the world (x,z) => int"
	`#find the y (vertical) of an x, z co-ordinate which represents the 'highest' (non-air) block` `y = mc.getHeight(0,0)`
.getPlayerEntityIds()	Get the entity ids of the connected players => [id:int]

	`#get the entity id's of the players connected to the game`
	`entityIds = mc.getPlayerEntityIds()`
	`for entityId in entityIds:`
	`    print entityId`

.saveCheckpoint()	Save a checkpoint that can be used for restoring the world
	`mc.saveCheckpoint()`
.restoreCheckpoint()	Restore the world state to the checkpoint
	`mc.restoreCheckpoint()`
.postToChat(message)	Post a message to the game chat
	`#write 'Hello Minecraft World' to the chat window` `mc.postToChat("Hello Minecraft World")`
.setting(setting, status)	Set a world setting (setting, status). keys: world_immutable, nametags_visible

	`#change world immutable to True`
	`mc.setting("world_immutable", True)`
	`#change nametags_visible setting to False`

	`mc.setting("nametags_visible", False)`

Minecraft.player Use a specific player to get and set world params	
.getPos()	Gets the player's position in the world as a Vec3 of floats (decimal numbers), if the player is in the middle of a block x.5 is returned
	`#get players position as floats` `playerPos = mc.player.getPos()`
.setPos(x,y,z)	Moves the player to a position in the world by passing co-ordinates ([x,y,z])
	`#set the players position as floats` `mc.player.setPos(0.0,0.0,0.0)`
.getTilePos()	Gets the position of the 'tile' the player is currently on.
	`#get the position of the tile the players is on` `playerTile = mc.player.getTilePos()`
.setTilePos(x,y,z)	Move the player to a tile position in the world by passing co-ordinates ([x,y,z])
	`#set the position of the tile the player is on` `mc.player.setTilePos(0,0,0)`
.setting(setting, status)	Set a player setting (setting, status). keys: autojump
	`#change the autojump setting to True` `mc.player.setting("autojump", True)`

Minecraft.entity

The entity functions are used in conjunction with the .getPlayerEntityIds() function to interact with the entity (or players) in a game. Entity functions are useful for multiplayer games.

.getPlayerEntityIds()	interact with the entity (or players) in a game

	`#get the entity id's of the players connected to the game`
	`entityIds = mc.getPlayerEntityIds()`
	`1stEntityId = entityIds[0]`
	`2ndEntityId = entityIds[1]`
	`...`

.getPos(entityId)	Gets an entities position in the world as a Vec3 of floats (decimal numbers), if the entity is in the middle of a block x.5 is returned
	`#get first entity position as floats` `entityPos = mc.entity.getPos(entityId)`
.setPos(entityId,x,y,z)	Moves the entity to a position in the world by passing co-ordinates ([x,y,z])
	`#set the players position as floats` `mc.player.setPos(entityId,0.0,0.0,0.0)`
.getTilePos(entityId)	Gets the position of the 'tile' the entity is currently on.
	`#get the position of the tile the entity is on` `entityTile = mc.entity.getTilePos(entityId)`
.setTilePos(entityId, x,y,z)	Move the entity to a tile position in the world by passing co-ordinates ([x,y,z])
	`#set the position of the tile the entity is on` `mc.player.setTilePos(entityId,0,0,0)`

Minecraft.camera

.setNormal(entityId)	Set camera mode to normal Minecraft view ([entityId])
	`#set camera mode to normal for a specific player` `mc.camera.setNormal(entityId)`
.setFixed()	Set camera mode to fixed view
	`#set camera mode to fixed` `mc.camera.setFixed()`
.setFollow(entityId)	Set camera mode to follow an entity ([entityId])
	`#set camera mode to follow for a specific player` `mc.camera.setFollow(entityId)`
.setPos(x,y,z)	Set camera entity position (x,y,z)
	`#set camera position to a specific position of x, y, z` `mc.camera.setPos(0,0,0)`

Minecraft.events	
.pollBlockHits()	Block Hits (Only triggered by sword) => [BlockEvent]

	`#get block event hits that have occured since the last time the function was run`
	`blockEvents = mc.events.pollBlockHits()`
	`for blockEvent in blockEvents:`
	`  print blockEvent`

.clearAll()	Clear all old events
	`#clear all events that have happened since the events where last got` `mc.events.clearAll()`

Class: Block

The definition of a Block in Minecraft, used to describe a block type and (if applicable) its data; also contains constants for the blocks type id's, e.g. BLOCK.AIR.id

.Block()	create and store a block of a specific type, sometimes with a data value

```
#create block of a specific type
blockObj = block.Block(id)
#create a block of a specific type and apply a data value
blockObj = block.Block(id, data)
```

BLOCK IDs

Block	id	Block	id	Block	id
AIR	0	SANDSTONE	24	CRAFTING_TABLE	58
STONE	1	BED	26	FARMLAND	60
GRASS	2	COBWEB	30	FURNACE_INACTIVE	61
DIRT	3	GRASS_TALL	31	FURNACE_ACTIVE	62
COBBLESTONE	4	WOOL	35	DOOR_WOOD	64
WOOD_PLANKS	5	FLOWER_YELLOW	37	LADDER	65
SAPLING	6	FLOWER_CYAN	38	STAIRS_COBBLESTONE	67
BEDROCK	7	MUSHROOM_BOWN	39	DOOR_IRON	71
WATER_FLOWING	8	MUSHROOM_RED	40	REDSTONE_ORE	73
WATER	WATER_FLOWING	GOLD_BLOCK	41	SNOW	78
WATER_STATIONARY	9	IRON_BLOCK	42	ICE	79
LAVA_FLOWING	10	STONE_SLAB_DOUBLE	43	SNOW_BLOCK	80
LAVA	LAVA_FLOWING	STONE_SLAB	44	CACTUS	81
LAVA_STATIONARY	11	BRICK_BLOCK	45	CLAY	82

SAND	12	TNT	46	SUGAR_CANE	83
GRAVEL	13	BOOKSHELF	47	FENCE	85
GOLD_ORE	14	MOSS_STONE	48	GLOWSTONE_BLOCK	89
IRON_ORE	15	OBSIDIAN	49	BEDROCK_INVISIBLE	95
COAL_ORE	16	TORCH	50	STONE_BRICK	98
WOOD	17	FIRE	51	GLASS_PANE	102
LEAVES	18	STAIRS_WOOD	53	MELON	103
GLASS	20	CHEST	54	FENCE_GATE	107
LAPIS_LAZULI_ORE	21	DIAMOND_ORE	56	GLOWING_OBSIDIAN	246
LAPIS_LAZULI_BLOCK	22	DIAMOND_BLOCK	57	NETHER_REACTOR_CORE	247

BLOCK DATA subtype of a block			
Block	**id**	**data**	**description**
WOOL	35	0	White
		1	Orange
		2	Mageta
		3	Light Blue
		4	Yellow
		5	Lime
		6	Pink
		7	Grey
		8	Light Grey
		9	Cyan
		10	Purple
		11	Blue
		12	Brown
		13	Green
		14	Red
		15	Black
WOOD	17	0	Oak (up/down)
		1	Spruce (up/down)
		2	Birch (up/down)
SAPLING	6	1	Oak
		2	Spruce

		3	Birch
GRASS_TALL	31	0	Shrub
		1	Grass
		2	Fern
		3	Grass (color affected by biome) NOT ON PI
TORCH	50	1	Pointing east
		2	Pointing west
		3	Pointing south
		4	Pointing north
		5	Facing up
STONE_BRICK	98	0	Stone Brick
		1	Mossy stone brick
		2	Cracked stone brick
		3	Chiseled stone brick
STONE_SLAB / STONE_SLAB_ DOUBLE	44 / 43	0	Stone
		1	Sandstone
		2	Wooden
		3	Cobblestone
		4	Brick
		5	Stone Brick
		6	Nether Brick NOT ON PI
		7	Quartz NOT ON PI

SNOW_BLOCK	80	0 - 7	Height of snow, 0 being the lowest, 7 being the highest. NOT ON PI
TNT	46	0	Inactive
		1	Ready to explode
LEAVES	18	1	Oak leaves
		2	Spruce leaves
		3	Birch leaves
SANDSTONE	24	0	Sandstone
		1	Chiseled sandstone
		2	Smooth sandstone
STAIRS_COBBLESTONE / STAIRS_WOOD	67 / 53	0	Ascending east
		1	Ascending west
		2	Ascending south
		3	Ascending north
		4	Ascending east (upside down)
		5	Ascending west (upside down)
		6	Ascending south (upside down)
		7	Ascending north (upside down)
LADDERS / CHESTS / FURNACES / FENCE_GATE	65 / 54 / 61 / 107	2	Facing north
		3	Facing south
		4	Facing east
		5	Facing west
WATER_STATIONARY /	9 / 11	0-7	0-7: Level of the water, 0 being the highest, 7 the lowest

LAVA_STATIONARY			
NETHER_REACTOR_CORE	247	0	Unused
		1	Active
		2	Stopped / used up

BlockEvent The definition of a BlockEvent in Minecraft, used to describe an event in Minecraft affecting blocks; returned by the Minecraft.events.pollBlockHits() method.	
.pollBlockHits()	Describe an event in Minecraft through blocks being hit with an event watcher
	`blockEvent = mc.events.pollBlockHits()`
.type	Type of block event; there is only 1 event currently implemented BlockEvent.HIT
	`blockEventType = blockEvent.type` *BlockEvent types:* 0: BlockEvent.HIT
.pos	The position of the block where the event occured, i.e. the block which was hit. .pos returns a Vec3 object of x,y,z co-ordinates
	`blockEventPos = BlockEvent.pos`
.face	The face of the block where the event occured
	`blockEventFace = BlockEvent.face`

.entityId	entityId of the player who caused the block event, i.e. the player who hit the block
	`blockEventPlayer - BlockEvent.entityId`
.entityId	entityId of the player who caused the block event, i.e. the player who hit the block
	`blockEventPlayer - BlockEvent.entityId`

Vec3 The definition of a 3 part vector in Minecraft, i.e. a set of x, y, z co-ordinates; x and z are the horizontal positions, y the vertical	
.Vec(x,y,z)	set a position with x, y and z coordinates
	`position = vec3.Vec(0,0,0)`
.x	get x position
	`xPos = position.x`
.y	get y position
	`yPos = position.y`
.z	get z position
	`zPos = position.z`

The legal bit