CUBE IMPLEMENTATION DOCUMENTATION

Practical Work Intelligent Systems

**Team Members: Repository name:** A1-05

* Bernalte Sánchez, Raúl raulbs7
* Rodríguez-Barbero González, Aharon1377

Aarón

* Rodríguez Villafranca, Enrique ERV501

Index

[1. Language chosen 3](#_Toc23019989)

[2. Structure of the Cube 3](#_Toc23019990)

[3. Code 3](#_Toc23019991)

[3.1. Class *Cube* 3](#_Toc23019992)

[3.1.1. Method *generateMoves* 3](#_Toc23019993)

[3.1.2. Method *move* 3](#_Toc23019994)

[3.1.3. Method *printState* 4](#_Toc23019995)

[3.1.4. Method *turnRight* 4](#_Toc23019996)

[3.1.5. Method *cubeString* 4](#_Toc23019997)

[3.1.6. Method *cubeMD5* 4](#_Toc23019998)

[3.1.7. Method *json2cube* 4](#_Toc23019999)

[3.1.8. Method *cube2Json* 4](#_Toc23020000)

[3.2. Class *Frontier* 5](#_Toc23020001)

[3.3. Class *TreeNode* 5](#_Toc23020002)

[3.4. Class *Sucessors* 5](#_Toc23020003)

# Language chosen

We have decided to use Python as the programming language for our project due to its versatility and simplicity, just as the used libraries have proven to be

# Structure of the Cube

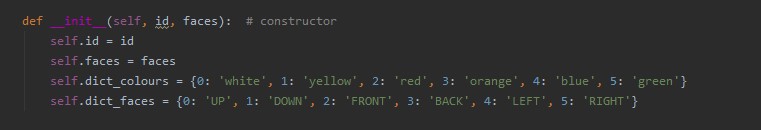
The structure of the cube consists of an MD5 identifier (as required), whose structure is a String, and the representation of the cube’s faces which has been implemented as a dictionary of lists, each list containing a face of the cube.

The reasoning why we implemented these specific structures was in hopes of the higher efficiency and comprehension level as well as the lowest memory waste.

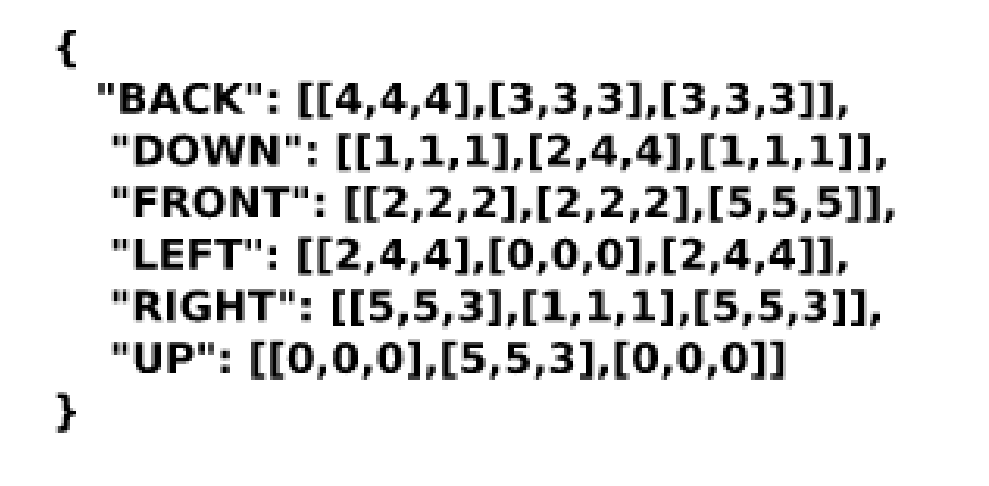
# Code

In this section, we are going to explain the main functionalities of the most important parts of the code:

## Class *Cube*



Our Cube is defined by:

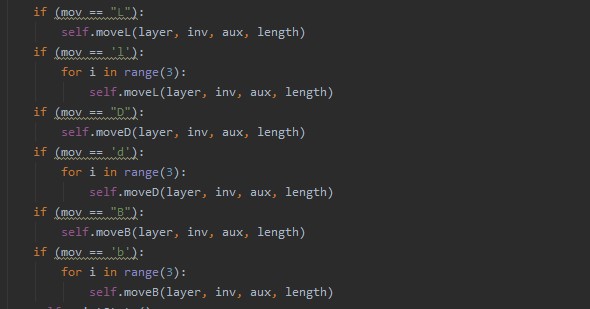
* id: it’s the md5 code that contains the current cube’s faces codified.
* faces: position of each face in the following format (see image below).
* dict\_colours: python dictionary used in the graphical representation of the cube, it contains the color-number relation.
* dict\_faces: python dictionary used in the movement of the faces.

### Method *generateMoves*

Here we compute the valid methods for the cube, depending on its dimensions (NxNxN).

### Method *move*

Performs the select move:

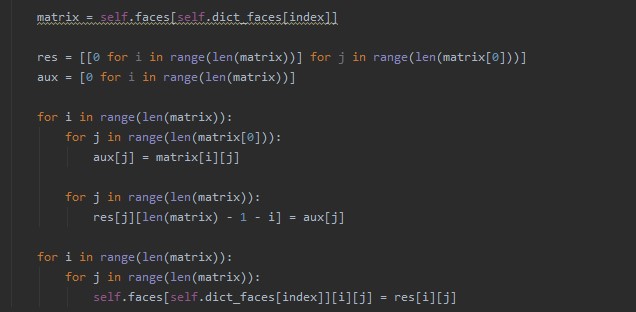


### Method *printState*

Draws and prints the current state of the cube (using Turtle library) taking into account the positions and the colors we specified in the dictionary “dict\_colours”.

### Method *turnRight*

Rotates a face of the cube 90º to the right.



### Method *cubeString*

Creates the string with the current cube state.

### Method *cubeMD5*

Codifies the previously created String into md5.

### Method *json2cube*

Opens the json file we are provided and transforms it into an object cube.

### Method *cube2Json*

Writes into a new json file the state of the cube.

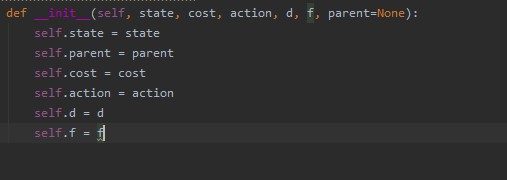
## Class *Frontier*

Defines our frontier. We decide to create a class, in which we have an array representing the frontier. The principal porpose to have a class with only this attribute, is to have two methods to manage this special array.

It contains the insertNode method, used to include nodes with an increasing order of the f value in the frontier, and the removeNode method in order to pop the node in the first position.

## Class *TreeNode*

Definition of the cube’s node.



## Class *Sucessors*

Generates the sucessors of the cube state.

