1 What is RASI?

RASI, which stands for *Robocracy Autonomous Scripting Interface*, is a scripted language developed by Robocracy engineers to allow for faster programming and debugging times. Its principle usage was to allow for the developer to write code which could be run without needing to rebuild the entire application, and wait sometimes minutes for it to be loaded onto the android device. RASI has a simple syntax which is easy to understand and use, but it can be extended to do anything that could be done in Java.

2 RASI Language Design

Rasi consists of three main parts, the interpreter, the lexer, and `RasiCommands`.

- `RasiCommands` is the heart of what makes Rasi so useful, as it is where the programmer can write java functions that can be called from a rasi file, and in doing so, allow RASI to do anything that Java could.
- **The Rasi Interpreter** takes in commands from the lexer and then actually calls the functions that are in `RasiCommands`. This is what actually runs the code from your java program.
- **The Rasi Lexer** is what takes in a RASI file and translates it into commands and arguments for the interpreter to execute.[More stuff about how it is great]

3 RASI Syntax

RASI is based on calling java functions with `commands`, followed by a list of arguments with an optional TAG. The syntax for calling a command without a tag is as follows: `command argument1 argument2` This calls the `command` function defined in `RasiCommands` with arguments `argument1` and `argument2`. Tags are used to control whether a command should execute. If you would like the program to execute a tag, call `addTag TAG_TO_BE_ADDED` before you use it. If you do not, any commands prefixed with that tag will not be executed.

```
addTag BLUE

BLUE: sayHello Hi
RED:  sayGoodbye Bye!

end
```

In this example, the program first adds the tag `BLUE`, then calls `sayHello("Hi")`, skips execution of sayGoodbye, and ends the program.

4 Supported Types

RASI currently only supports the Java basic types and strings. So, you can not pass in your MagicUnicorn class as an argument. Full supported types are:

1. Integers
2. Doubles
3. Characters
4. Float
5. Boolean
6. String