Natural Language Programming Language User Manual

Created by Jack Farrell and Conall Kavanagh

Table of Contents

Setting up the Enviroment

In order for the computer to recognise the language and be able to execute the code we need to download a few things.

if you are on a linux machine then all that is neede is to set up antlr4 on your machine, this can be done by following the guide here.

For windows users it is recommended to set up Windows Subsytem for Linux. By following the guide here it should be ready to go. Once it is up and running follow the instructions for linux machines to set up antlr4.

Setting up the language

Once you cloned the repo and have antlr4 installed move to the directory src/code. it is here where your files should be written. it is also possible to create a seperate directory here to store all the programms that you make.
`~/2023-ca326-jfarrell-ckavanagh$ cd src/code`

Now that we have the blueprints of the language we need to run the run.sh file found here. this can be done by typing into the console `./run.sh` you will know it has worked when the following appears

```
java org.antlr.v4.gui.TestRig GrammarName startRuleName
[-tokens] [-tree] [-gui] [-ps file.ps] [-encoding encodingname]
[-trace] [-diagnostics] [-SLL]
[input-filename(s)]
```

once you see that it is time to start making your own programms.

Commands

There are a lot of commands each with their own use and ways of being written. This guide will give you an overview on each of the commands and some examples to show how each one works.

Here is the order of commands and features that will be covered in the guide

1. Strings, Numbers and Bools
2. Say
3. Assigning Variables
4. Lists
5. Loops
6. If Statement
7. Functions
8. Comments

1 - Strings, Numbers and Bools

All of these are different types of variables
String variables hold sentences and can be used to improve the look of a programms output by making it a full sentence.
Number variables are assigned a value that can change during the programms execution and can be called in place of hard coded numbers.
Bool variables are simple True or False variables. They can have no other value besides True or False and act like a switch.

Strings

Strings are what the computer sees as words or sentences.
They are differentiated form other commands by the use of quotation marks ("")

```
say x;
say "x";
```

The first line would say the value of variable x while the second would simply say the letter x
the use of strings is important to differentiate what is meant to be executed by the computer and what is meant to be human readable.

Numbers

Number variables are used to make the programme more flexible instead of hardcoding the number value in

variables can be changed by using mathematical opperations.

```
x is 5;
```

This assigns the number 5 to the variable x. this means that when x is called, like in the string example, the number 5 will be said instead of the letter x

```
x is x + 1;
```

While this may look wrong in a mathematical sense, in coding this simply increments the value of x by one
say x was 5 after running that line the value of x is now 6.

Bools

As stateted above boolean variables (or bools) are simple True False variables that are commonly used as the conditions for other commands and influences how they work.

```
x is True
```

What this has done is create the boolean variable and assign it to True. The value of the bool can change by typing the same command but replaceing True with False.

> ***Note***
> It is important to know that both True and False need to have their first letter capitalised as the computer will not recognise it as a bool and return an error.

2 - Say

The say command is our first and most basic commands, having said that it is the one command that can appear in almost every programme as it is responsible for getting the computer to say the output of the programme.

Lets take the line `x is (5 + 2) * 2;` that the computer can easily calculate the answer but without the say command the user will never know what answer the computer got as nothing is telling it to say the value of x

```
x is (5 + 2) * 2;
say x;
```

```
14
```

By adding in the say command we can get the computer to say what the value of x is. The say command can output any type of variable and the assosiated value.

```
say "Hello World";
```

```
x is True;
say x;
```

The output for these would be `Hello World` and `true` respectivly.

3 - Assigning Variables

We would have covered this a bit in the earlier examples but we will go over it again here to make sure that you know what is happening.

Assigning values to variables is very simple first we have the name of our variable in this case we will use `x`. We then follow that up with `is` and then the value we are assigning to it. Dont worry about the variable type the computer will automatically determine if it is a string, number or bool based on the assigned value.

```
x is 5;
y is "hello";
z is True;
```

The computer has automatically assigned x as a number variable, y as a string variable, and z as a boolean variable.

4 - Lists

Lists are a way of storing multiple numbers or strings in a single variable.

```
x is [1, 2, 4, "Hello", 5, 6];
say x;
```

`[1.0, 2.0, 4.0, Hello, 5.0, 6.0]`

These items can be added and removed from the list by using the `+` or `-` operations similar to incrementing a variable.

```
x is [1, 2, 4, "Hello", 5, 6];
x is x + 3;
x is x + 7;
x is x - "Hello";
x is x - 3;
say x;
```

`[1.0, 2.0, 4.0, 5.0, 6.0, 7.0]`

As you can see from the above example the items in the list are not ordered and any new items added join on the right hand side

5 - Loops

Loops are used to execute a set of commands a certain number of times before continuing with the rest of the programme. Loops allow for code to be repeated without having to write it out multiple times and take up increased file space

```
loop 5 times {
        say "in the loop";
}
```

```
in the loop
in the loop
in the loop
in the loop
in the loop
```

As you can see with only one `say` command we were able to get the computer to say the string 5 times
Loops can also repeat based on a number variable

```
x is 5;
z is 1;
loop x times {
        say z;
        z is z + 1;
}
```

```
1
2
3
4
5
```

Not only did the loop execute correctly using a variable but the incrementation of the `z` variable highlights how loops reuse code.

6 - If Statement

there are 2 types of if statements

> If statements
> If else statements

If Statements

If statements work off of bools, where if the bool is true the main section of code is then ran if not then the code is skipped and we progress through the rest of the programme

```
y is True;
x is False;

if y {
        # Code here is then ran
}

if x {
        # Code here is skipped
}
```

```
x is True;

if x {
    say "x is true";
}
```

`x is true`

This is the simplest form of if statements as if the statement is false then nothing happens

If-Else Statements

If else statements, as you might be able to interpret, are an extension of the if statement. While if statements simply pass if the variable is false if-else statements execute a seperate block of code.

```
x is True;

if x {
    say "x is true";
} else {
        say " x is false";
}
```

from the example above if `x is False` then instead of nothing being said by the computer, the string `"x is false` is outputted instead.

7 - Functions

A function is a block of code which only runs when it is called. This allows for code to be written once in the programme but used multiple times in different places.

In order for functions to work you first need to make the function

Making the Function

Making the function is very simple. you start with "on" then the function name followed by brackets to indicate any inputs the function may need and the cureley brackets to indicate that the code within them is part of the function. It should look something like this

```
on FunctionName(x, y) {
say x;
say  y;
}
```

This function, called `FunctionName`, when called will look for 2 variables called x and y and then say the value of those variables

Now that we have this code created we need to call the function for it to be run.

Calling the Function

Calling the function only requires typing out the functions name and inputting any variables that are necessary for it

`FunctionName(1, 2);`

This calls the above function and will output

```
1
2
```

> It is not necessary for functions to require input variables, Some functions can work just fine without any.
> In these cases the making and calling of the function would look like this:
>
> ```
> on FunctionName() {
> # Code
> }
>
> FunctionName();
> ```

8 - Comments

Comments can be used to make the code more readable and easier to remember what ceratin parts of the programme do. Comments may also used to prevent execution when testing code as if you dont want a line to be read by the computer you can comment it out and it will skip the line.

The computer recognises comments by the use of the `#` symbol.
anything written after the # is considered to be a comment and will not be executed.

```
# This is a comment and wont be read

x is "Hello World"; # i can use comments to explain what the code does
say x;
# say y;
# i can comment out code that i do not want to be run without having to
delete it
```

```
Hello World
```

Comments are a useful tool and recommended to use especially in more complex programms.