Referee User Guide &
Language Reference

```
KICKOFF
WHILE 1
choice = 0
num1 = 0
num2 = 0
CHANT "What would you like to do?"
CHANT "(1) Addition, (2) Subtraction, (3) Multiplication, (4) Division, (5)
INPUTINT choice
IF choice == 1 PASS
CHANT "Input the two numbers"
INPUTINT num1
INPUTINT num2
result = num1 + num2
CHANT "Num1 + Num2 = " result
TRICK IF choice == 2 PASS
CHANT "Input the two numbers"
INPUTINT num1
INPUTINT num2
result = num1 - num2
CHANT "Num1 - Num2 = " result
TRICK IF choice == 3 PASS
CHANT "Input the two numbers"
INPUTINT num1
INPUTINT num2
result = num1 * num2
CHANT "Num1 * Num2 = " result
TRICK IF choice == 4 PASS
CHANT "Input the two numbers"
INPUTINT num1
INPUTINT num2
result = num1 / num2
CHANT "Num1 / Num2 = " result
TRICK IF choice == 5 PASS
CHANT "Input the number you want to sum up to"
INPUTINT num1
total = 0
FOR i = 1 TO num1
total = total + i
DRIBBLE i
CHANT "The sum of numbers up to " num1 " is " total
SHOOT!
MEXICANWAVE
FINALWHISTLE
~
~
~
```

Contents

Installation Instructions

Referee is designed for the Windows operating system. To use the GUI, you will also need Python 2.7 installed on your computer.

To run the compiler, you can double click the gui.py file, selecting it to open with the python interpreter (and not editing it in IDLE). Experienced users may want to call the referee.exe compiler on the command line instead.

Quick Start Guide

Once installed, the compiler is very easy to use by nature. Clicking the purple box at the top right corner of the GUI will allow you to navigate between the online tutorials and the compiler interface. These tutorials display information on how to compile basic programs.

To get started, on the main screen click on the Browse... button next to the Input File box. Referee comes preinstalled with some sample programs, under the Examples directory. Navigate to this directory and you will see the following text files:

You may open these files with a text editor of your choice to view the code. Select an example by pressing

OK, then do exactly the same on the second Browse... box below to specify a location to save the compiled program. The default .py extension is automatically added if you do not specify a file type. Finally, click Compile & Run and the program will shortly finish compiling, opening the program in IDLE. You may run this program by pressing F5.

Language Reference

The following pages specify the features available in the language. It is not meant as a tutorial, only as a reference book that you can use once more familiarised with the language and wish to know more detail on a specific language feature. If you want a tutorial, open Referee's GUI and navigate to a tutorial using the purple box in the top right hand corner.

Data type Reference

Integer

The integer datatype is used every time you want to use a number in your programs. To set a variable as an integer, simply write the number in an assignment statement like so:

```
count = 5
```

Since Referee uses Python as a base language, the integers have no maximum or minimum value limited by the language. In Python variables are dynamically allocated, so the only limitations are the hardware of your system.

String

The string datatype is used for non numerical characters that you do not wish to do calculations on. To set a variable as an integer, simply write the number in an assignment statement like so, using the “ speech marks to specify the start and end of the string:

message = “Hello World”

To write the “ character in a string, you may use the following escape sequence: \”. Other escape sequences are available:

\n	Newline character
\r	Carriage return
\t	Tab character

Command Reference

KICKOFF

KICKOFF specifies the start of a program. Simply place the line at the start of the program. Since later versions of Referee this line is no longer required to compile successfully.

See also: FINALWHISTLE

FINALWHISTLE

Finalwhistle works similarly to KICKOFF, specifying the end of the program. Place the line at the end of the program. Since later versions of Referee this line is no longer required to compile successfully.

See also: KICKOFF

CHANT

The CHANT command is used to output information to the screen. It is the first command you learn, and almost every program you write will use it to show the result of calculations, or inform the user what the program is doing. A basic chant statement is as follows:

CHANT “Hello World”

This outputs the string “Hello World” to the screen. When compiled, the Python equivalent is
print(“Hello World”) and the output looks like this: Hello World

You can output multiple things at the same time, using the same CHANT command. An unlimited number of values can be printed on the same line in this way. Simply separate the values to output by a space like this:

```
name = “Ben”
CHANT “Hello, my name is “ name
```

The CHANT command supports the output of Strings and Integers (See the Datatype reference manual for more information). It does not support any arithmetic operations before printing; the following statement is invalid:

```
CHANT  5 + 6
```

If you wish to do this, you must first perform the calculation and store in a variable like so:

```
result = 5 + 6
CHANT result
```

See also: SUBSTITUTE

Assignment

Assignment is not a command as such, but a useful technique to store data. The syntax of an assignment statement is as follows:

count = 5

This will set the variable, labelled 'count' to be equal to 5. Later in the program this can be changed again, or output to the user using a CHANT statement. To perform an assignment operation you need a variable name on the left, then an = sign, and then a value or arithmetic operation. The following is a valid assignment statement:

number = 5 * 4 * 3 + 100 / 10 – 50

number will be set to the result of the arithmetic operation (20).

IF

The IF statement is used to change the logic flow of the program depending on a condition. The basic syntax is as follows:

IF condition PASS
(Optional TRICK IF statements)
(Optional TRICK statement)
SHOOT!

On the first line the condition must be a resolvable statement using constants, variables and operators. A comparison operator must be used to test two sides of the statement, and you may use as many arithmetic operators to create a resolvable expression on either side of the comparison operator. The PASS command indicates that the condition has finished being defined, and from this point on the program will be looking for a SHOOT! Statement to signify the end of the IF block.

TRICK IF statements may be used as many times as desired, being of similar syntax to the IF statement. Only one TRICK statement may be used at the end of these TRICK IF statements.

The SHOOT! statement ends the IF block. If the initial IF condition resolves as false the program will jump to this point.

Sibling commands: PASS, TRICK, SHOOT!

FOR

A FOR loop is a type of iteration that will execute a block of code a set number of times. The basic syntax is as follows:

FOR assignment TO maximum
DRIBBLE (loop counter)

The assignment part of the statement sets a variable equal to a value. This is known as the loop counter variable. During every iteration of the FOR loop, the variable will be compared to maximum. If the variable equals maximum, execution will skip to the DRIBBLE statement, ending the iteration. Otherwise, the block will execute, and upon reaching the DRIBBLE statement the loop counter variable will be incremented. Execution will then jump to the initial FOR statement.

Sibling Commands: DRIBBLE
See also: WHILE

WHILE

A WHILE loop is a type of iteration that will execute a block of code a based upon a condition. The basic syntax is as follows:

WHILE condition
MEXICANWAVE

The condition must be a resolvable statement using constants, variables and operators. A comparison operator must be used to test two sides of the statement, and you may use as many arithmetic operators to create a resolvable expression on either side of the comparison operator.

The loop is organised into a block, executing all statements between the WHILE statement and the MEXICANWAVE statement. When the WHILE statement is reached, the program will resolve the condition. If the condition resolves as true, the block will be executed. If the condition resolves as false, execution will jump to the MEXICANWAVE statement.

A valid WHILE loop is shown below:

Sibling Commands: MEXICANWAVE
See also: FOR

SUBSTITUTE

The SUBSTITUTE command allows the program to receive user input to the program whilst it is running. The syntax is as follows:

variable = “”
SUBSTITUTE variable
The variable must first be defined before you may store data to it, which is why the first statement is an assignment statement simply setting the variables value to an empty string. The SUBSTITUTE requires an already defined variable, thus cannot take constants.

When run, the program will allow user input on a blank new line. You cannot add constants to the SUBSTITUTE command to prompt the user for input, thus most user input statements should be in this form:

CHANT “Please input the data”
variable1 = “”
SUBSTITUTE variable1

See also: SUBINT

SUBINT

The SUBINT command allows the program to receive user input to the program whilst it is running, in the form on an integer. User input is stored in a string by default, so if you specifically want an integer to be input use this command. The syntax and setup is identical to the SUBSTITUTE command.

See also: SUBSTITUTE

Operator Reference

The following comparison operators are supported by the language:

Operator Explanation	Code
Checks both operands for equality. If the operands are not equal, the condition is FALSE. If they are equal, the condition is TRUE.	==
Checks both operands for equality. If the operands are not equal, the condition is TRUE. If they are equal, the condition is FALSE.	!=
Checks each operand, determining which is larger. If the left operand is greater than the right, the condition is TRUE. Otherwise the condition is FALSE.	>
Checks each operand, determining which is larger. If the left operand is greater than or equal to the right, the condition is TRUE. Otherwise the condition is FALSE.	>=
Checks each operand, determining which is larger. If the left operand is less than the right, the condition is TRUE. Otherwise the condition is FALSE.	<
Checks each operand, determining which is larger. If the left operand is less than or equal to the right, the condition is TRUE. Otherwise the condition is FALSE.	<=

The following arithmetic operators are supported by the language:

Operator Explanation	Code
The subtraction operator will subtract the right operand from the left operand, returning the result.	-
The Addition operator will add the right operand to the left operand, returning the result.	+
The Division operator will divide the right operand by the left operand, returning the result.	/
The multiplication operator will multiply the right operand by the left operand, returning the result.	*

The = operator is also supported, assigning the right operand to the left operand (if it is a variable).

Advanced Features

Experienced users may wish to use the command line interface to the compiler instead of the graphical interface. To do this simply call the referee.exe file with the following syntax:

\pathToFile\referee.exe \pathToFile\input.txt \pathToFile\output.py

You may wish to include ‘ quote marks to ensure the entire argument is taken, otherwise the command line will use the space character in the file names as a delimiter. Error messages are sent to the standard output, and nothing is send to the standard error.