Highly Organized & Optimized Trading In Exchanges (HOOTIE)

Language Reference Manual

HOUTAN M. FANISALEK
COLUMBIA UNIVERSITY

INTRODUCTION

Most traders have little or no technical knowledge and rely on software and human interaction to perform complex trading routines. The primary objective of this language is to provide a way for traders to create an automated black box system that is simple enough to understand and use yet robust enough for high volume trading. The language and compiler will handle all of the details leaving the user free from worrying about actual code.

The HOOTIE language is very simple and those that have the basic knowledge of trading stocks can pick it up very quickly. Each command is written on a single line with no semicolon and each item is separated by spaces. The symbol is in reference to a stock symbol.

LEXICAL CONVENTIONS

There are five kinds of tokens in this language: identifiers, stock symbols, keywords, strings, and operators. Tabs, blanks, and comments are ignored except when acting as separators. Newlines are not ignored and used to terminate a statement.

COMMENTS

A comment or note is determined by beginning and ending with a double backslash and does not interfere with the execution of the program.

```
// This is a comment //
```

IDENTIFIERS

An identifier is a sequence of letters and digits with the first character being alphabetic including the underscore "_". Upper and lower case letters are NOT considered different.

STOCK SYMBOLS

Stock symbols are always capitalized and start with a dollar sign.

For Example: $MGM

KEYWORDS

The following capitalized keywords are reserved and may not be used as identifiers:

- LOOKUP
- BUY

- SELL
- WHEN
- OTHERWISE
- LOOP
- WAIT
- WAITEXE
- BREAKOUT
- AND
- OR
- NOT
- IS
- PRINT
- SET
- UNSET
- EXEUTE
- STOP
- EXIT
- TRUE
- FALSE
- MARKET
- LIMIT
- STOPLIMIT
- TRADESTOP%
- TRADESTOP$
- NOW
- TODAY
- DAYS

NUMBERS

Numbers can be either integer or floating point values with the following restrictions

- Numbers are a sequence of digits from 0 through 9.
    - Example: 3000000
- Numbers can end with a M (million) or B (billion) or T(Trillion)
    - Example: 3M
- Floating point values must have two and only two decimal places.
    - Example: 3000000.00

STRINGS

A string is a sequence of characters starting and ending with double quotes (“ “). For Example:

“THIS IS A STRING”

OPERATORS

An operator is a specific token which specifies an operation on one or more operand and may be an expression or constant. Arithmetic and comparison operators are left associative, Assignment is right associative, and logical operators have no associativity.

COMPARISON OPERATORS

Operator	Definition	Number Example	Stock Example
<	Less than	a > b	$NTFL IS > 400.00
>	Greater than	a < b	$NTFL IS > 400.00
<=	Less than or equal to	a <= b	$NTFL IS > 400.00
>=	Greater than or equal to	a >=b	$NTFL IS > 400.00
EQUALS	Equal to	a EQUALS b	$NTFL IS > 400.00

ARITHMETIC OPERATORS

Operator	Definition	Number Example	Stock Example
+	Addition	a + b	$NTFL + 400.00
-	Subtraction	a - b	$NTFL - 400.00
*	Multiplication	a * b	$NTFL * 400.00
/	Division	a / b	$NTFL / 400.00
%	Modulus	a % b	$NTFL % 400.00

ASSIGNMENT OPERATORS

Operator	Definition	Number Example	Stock Example
=	Assignment	@var = 5	@var = $NTFL

LOGICAL OPERATORS

Operator	Definition	Example	Stock Example
AND	Logical AND	@var1 AND @var2	N/A
OR	Logical OR	@var1 OR @var2	N/A
NOT	Logical NOT	NOT @var2	N/A

DECLARATIONS

There are two specific types of declarations. One type is the buying and selling of a stock and the other is the variable declarations.

BUY AND SELL

Buying or selling a stock begins with the BUY or SELL keyword followed by the required stock symbol, and optional conditions, and optional term. For example:

SELL *Symbol Quantity* [Conditions] [Term]
Example: SELL $AIG 2000 [LIMIT 34.00] [60DAYS]

BUY *Symbol Quantity* [Conditions] [Term]
Example: BUY $MGM 2000 [LIMIT 15.00] [30DAYS]

CONDITIONS

The following conditions can be used with the BUY or SELL statement

MARKET (this is the default)
LIMIT *value*
STOP *value*
STOPLIMIT *value*
TRADESTOP$ *value*
TRADESTOP% *value*

TERM

The following terms can be used with the BUY or SELL statement

NOW (default)
TODAY
value DAYS

VARIABLES

To declare a variable the SET keyword is issued with the variable identifier.

```
SET @variablename
```

To remove a variable the UNSET keyword is issued with the variable name that has been previously set.

```
UNSET @variablename
```

The assignment operator can assign values to each variable. For example:

```
SET @number_of_trades
@number_of_trades = 50
UNSET @number_of_trades
```

STATEMENTS

Statements are only one per line and executed in order however statements instead the EXECUTE blocks are executed concurrently. Whitespace has no effect.

WHEN STATEMENT

The WHEN statement is in the follow two form

```
WHEN ( condition ) {
   statement
}

WHEN ( condition ) {
   statement
} OTHERWISE {
   statement
}
```

In both cases if the condition is true the first statement after the WHEN is executed. If the statement is false and there is an OTHERWISE clause then that statement will be executed.

Example:

```
WHEN ($NFLX IS > 210.00) {
   PRINT “NETFLIX IS TOO EXPENSIVE!”
```

```
    EXIT
} OTHERWISE {
    EXIT
}
```

LOOP STATEMENT

The LOOP statement is in the form

LOOP (*condition*) {
    statement
}

The statement is executed repeatedly until the value of the condition becomes false.

BREAKOUT STATEMENT

The BREAKOUT statement is put within a WHEN or LOOP statement and causes the termination of that statement.

EXIT STATEMENT

The EXIT statement terminates the execution of the program completely.

PRINT STATEMENT

The print statement can be used to output strings or variables to the terminal. For example:

PRINT “HELLO STOCK MARKET WORLD”

EXECUTE STATEMENT

The execute statement begins with the EXECUTE keyword and the followed by an identifier.

EXECUTE *identifier* {

}

Each execute block is done concurrently and each program needs at least one execute block. For example:

```
EXECUTE a_merger {
        WHEN (BLOAQ IS >= .09) {
```

```
            SELL NFLX 2000
            EXIT
        }
}
```

WAITEXE STATEMENT

The WAITEXE is always followed after a BUY or SELL and halts the execution until the BUY or SELL finishes. For Example:

```
SELL NFLX 2000
WAITEXE
```

This will halt execution until 2000 shares of NFLX are sold.

WAIT STATEMENT

The WAIT followed by an integer will halt the execution for a specific number of milliseconds. For example:

```
WAIT 1000
```

This will halt execution of the program for 1000 milliseconds.

LOOKUP STATEMENT

The LOOKUP statement begins with the LOOKUP keyword followed by the stock symbol and a certain characteristic.

LOOKUP *Symbol [Information]*

INFORMATION

VOLUME
BETA
MCAP
RANGE

CODE SAMPLE

```
// THIS IS A SAMPLE PROGRAM //

PRINT LOOKUP NFLX
PRINT LOOKUP BLOAQ

EXECUTE wait_for_range {
        WHEN (NFLX IS > 210.0) {
                PRINT “NETFLIX IS TOO EXPENSIVE!”
                EXIT
        }
}
EXECUTE wait_for_volume {
        WHEN (NFLX IS VOLUME > 10MIL) {
                SELL NFLX 2000
                WAITEXE
                EXIT
        }
}
EXECUTE a_merger {
        WHEN (BLOAQ IS >= .009) {
                SELL NFLX 2000
                EXIT
        }
}
```