

## APPL 1.0 Documentation

**APPL (Avasaram Platform Programming Language)** is a powerful language used to extend the built in functionalities of the Avasaram platform. It could be used to create **advanced filters** and **advanced display columns**. Language syntax for **APPL** is very similar to the popular language **JAVA**.

APPL scripts must start with a curly brace "{" and end with a curly brace "}". All the statements in the script must end with a semicolon ";".

A sample script for **APPL filters** to find stocks with last traded price greater than 10.

```
{  
stock.getLast() > 10;  
}
```

A sample script to add an **APPL display column** of maximum return probability.

```
{  
strategy.getMaxReturnProbability();  
}
```

### APPL Core Objects and Classes

Object/Class	Description
strategy	The strategy object which contains information about the strategy being accessed.  <b>Example</b> to get the ticker symbol of the underlying stock. <code>strategy.getTicker()</code>
stock	The underlying object for the strategy. <b>Example</b> to get the last traded price. <code>stock.getLast()</code>
longLegOne	The first long leg associated with the strategy. In case of a married put strategy this is the put option purchased to protect the stock. Presence of this object depends on the strategy that is being accessed.  <b>Example</b> to get the option symbol. <code>longLegOne.getOptionSymbol()</code>
shortLegOne	The first short leg associated with the strategy. In case of a covered call strategy this is the call option sold against the stock. Presence of this object depends on the strategy that is being accessed.  <b>Example</b> to get the option symbol. <code>shortLegOne.getOptionSymbol()</code>

longLegTwo	The second long leg associated with the strategy. Presence of this object depends on the strategy that is being accessed. This is usually present in a multi leg strategy like butterflies and condors.  <b>Example</b> to get the option symbol. <code>shortLegOne.getOptionSymbol()</code>
shortLegTwo	The second short leg associated with the strategy. Presence of this object depends on the strategy that is being accessed. This is usually present in a multi leg strategy like butterflies and condors.  <b>Example</b> to get the option symbol. <code>shortLegOne.getOptionSymbol()</code>
Util	This is a utility class containing commonly used methods.
Math	Utility class containing mathematical functions.  <b>Example</b> to get the square root. <code>Math.sqrt(4)</code>
TA	Technical Analysis class containing TA specific functions.  <b>Example</b> to check for existence of Bullish DOJI Pattern in last 20 days <code>TA.hasPattern(CandleStickPattern.DOJI, Bias.BULLISH,20);</code>

### Filters using APPL script

Advanced filters could be made using APPL scripts. The APPL expression used for building the filter must evaluate to a Boolean value. To add a new APPL filter click on the "Add APPL Filter" button from the custom screener. See the figure below.

The screenshot shows a 'Screener Details' interface. At the top, there are fields for 'Screener Name' (Covered Call Screener) and 'Strategy Name' (Covered Call). Below this is a 'Filters' section with a dropdown menu set to 'Data Source' and an 'Add Filter' button. A list of filters is displayed: 'Option Volume' (GREATER THAN 10), 'Probability of Max Return' (GREATER THAN 40%), and 'Stock Price' (BETWEEN 10 and 50). Each filter has 'Edit' and 'Remove' links. Below the list is an 'Advanced APPL Filters' section with an 'Add APPL Filter' button, which is highlighted by a red arrow. Below this button, it says 'No APPL filters are configured!'. At the bottom, there is a 'Display Column Configuration' section with 'My APPL Columns' set to 'Ann return', and buttons for 'Edit APPL Column' and 'Add APPL Column'.

Figure below shows an example APPL filter which returns strategies with Annual Return > 10

After entering the script click the validate button to check for any error. After validation click save to add the filter to your screener.

## Advanced APPL Filter

Name:

Expression

```
{
maxRetPcentage = strategy.getMaxReturnPercentage();
returnPeriod = shortLegOne.getOptionExpiryInDays();
annualReturn = (maxRetPcentage/returnPeriod)*365;
//Return Covered calls with annual return greater than 10
annualReturn > 10;
}
```

## Examples

1. Filter to retrieve strategy data which has its stock with last traded price between 10 and 50.

```
{
stock.getLast() > 10 && stock.getLast() < 50;
}
```

2. Filter to retrieve a stock which has an occurrence of Candle stick pattern "Two Crows" in last 20 days.

```
{
TA.hasPattern(CandleStickPattern.TWO_CROWS,Bias.BEARISH,20);
}
```

3. Filter to find a Delta Neutral position for calendar spread.

```
{
shortDelta = shortLegOne.getDelta();
longDelta = longLegOne.getDelta();
netDelta = longDelta - shortDelta;
netDelta < 0.02;
}
```

## Display Column using APPL script

Display columns could be added to the screener results easily by creating an APPL script. Click on the **"Add APPL Column"** to create a new display column. See the figure below.

The screenshot displays the 'Screener Details' interface for a 'Covered Call Screener'. It shows various filters and an advanced APPL filter. The 'Display Column Configuration' section is highlighted, showing a list of available values and selected values. A red arrow points to the 'Add APPL Column' button.

**Screener Details**

Screener Name: Covered Call Screener Strategy Name: Covered Call

Filters: Data Source Add Filter

Option Volume: GREATER THAN 10 Edit Remove

Probability of Max Return: GREATER THAN 40% Edit Remove

Stock Price: BETWEEN 10 and 50 Edit Remove

Advanced APPL Filters: Add APPL Filter

Annual Return > 10 { maxRetPcentage = strategy.getMaxReturnP ... Edit Remove

Display Column Configuration: My APPL Columns: Test Edit APPL Column Add APPL Column

Available Values: Option Ask

Selected Values: Stock Ticker, Last Traded Price, Option Name, Option Expiry, Option Strike, Option Bid, Option Volume, Open Interest, Time Value(%), Return on Exercise, Probability of Max Return, Action Column: Analyzer

Reset to Default Columns

After entering the script click the **"validate and preview"** button to check for any errors and preview results. After validation click **"save"** to add the display column to your screener. Unlike the filters the script for the display column does not have to evaluate to boolean.

Figure below shows the APPL script for display column which shows a simple annualized return.

### Advanced APPL Display Column

Column Title:

Column Desc:

Expression:

```
{
maxRetPcentage = strategy.getMaxReturnPercentage();
returnPeriod = shortLegOne.getOptionExpiryInDays();
annualReturn = (maxRetPcentage/returnPeriod)*365;
Math.convertToTwoDigitPrecision(annualReturn);
}
```

Preview	
Ticker	AnnRet%
MVIS	945.78
AONE	346.44
GERN	305.68
AGQ	278.72

### Examples

1. APPL Script to create a display column to create a simple annualized rate of return for a covered call strategy.

```
{
maxRetPcentage = strategy.getMaxReturnPercentage();
returnPeriod = shortLegOne.getOptionExpiryInDays();
annualReturn = (maxRetPcentage/returnPeriod)*365;
Math.convertToTwoDigitPrecision(annualReturn);
}
```

2. APPL Script to create a display column showing Net Delta for calendar spread.

```
{
shortDelta = shortLegOne.getDelta();
longDelta = longLegOne.getDelta();
netDelta = longDelta - shortDelta;
Math.convertToTwoDigitPrecision(netDelta);
}
```

Available methods for Object : strategy

Function Name	Return Type	Description
getTicker()	String	Gets the ticker symbol associated with the stock of this strategy.  <b>Example Usage:</b> <code>strategy.getTicker();</code>
getStrategyShortName()	String	Gets the short name for the strategy.  <b>Example Usage:</b> <code>strategy.getStrategyShortName();</code>
getMaxReturnPercentage()	double	Gets the maximum return in percentage
getMaxReturn()	double	Gets the maximum return
getMaxRisk()	double	Gets the maximum risk for the strategy
getMaxReturnProbability()	double	Gets the probability of maximum return
getAnyReturnProbability()	double	Gets the probability of any return

Available methods for Object : stock		
Function Name	Return Type	Description
getLast()	double	Gets the last traded price  <b>Example Usage:</b> <code>stock.getLast();</code>
getOpen()	double	Gets the opening price  <b>Example Usage:</b> <code>stock.getOpen();</code>
getHigh()	double	Gets the high price
getLow()	double	Gets the low price
getBid()	double	Gets the bid price
getBidSize()	double	Gets the bid size
getAsk()	double	Gets the ask price
getAskSize()	double	Gets the ask size
getWkLow52()	double	Gets the 52 Week low

getWkHigh52()	double	Gets the 52 Week High
getAvgVolume()	long	Gets the average volume
getStockName()	String	Gets the Name of the stock
getVolumeTraded()	long	Gets the volume traded
getAtmCallVolatility()	double	Get the At the Money Volatility of the Call Options
getAtmPutVolatility()	double	Get the At the Money Volatility of the Put Options

Available methods for Objects : longLegOne/longLegTwo/shortLegOne/shortLegTwo		
Function Name	Return Type	Description
getExpiry()	String	Gets the expiry of the option as a String  <b>Example Usage:</b> longLegOne.getExpiry(); OR longLegTwo.getExpiry(); OR shortLegOne.getExpiry(); OR shortLegTwo.getExpiry();
getOptionExpiryInDays()	long	Gets the expiry of the option in days
getOptionSymbol()	String	Gets the option symbol
getOptionType()	String	Gets the type of Option. "C" for Call and "P" for Put
isITM()	boolean	Returns true if the option is in the money
getImpliedVolatility()	double	Gets the implied volatility for the option
getStrike()	double	Gets the Strike of this option
getBid()	double	Gets the bid price for the option.
getAsk()	double	Gets the ask price for the option
getLast()	double	Gets the last traded price for the option.
getVolume()	long	Gets the trade volume for the option.
getOpenInterest()	long	Gets the open interest for the option.
getDelta()	double	Gets the delta for the option.

getGamma()	double	Gets the gamma for the option
getTheta()	double	Gets the theta for the option.
getVega()	double	Gets the vega for the option.
getRho()	double	Gets the rho for the option.

Available methods for Class : Math		
Function Name	Return Type	Description
abs(double a)	double	Returns the absolute value of a double value. <b>Example Usage:</b> <code>Math.abs(12);</code>
acos(double a)	double	Returns the arc cosine of a value; the returned angle is in the range 0.0 through $\pi$ <b>Example Usage:</b> <code>Math.acos(12);</code>
asin(double a)	double	Returns the arc sine of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$ .
atan(double a)	double	Returns the arc tangent of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$ .
cbrt(double a)	double	Returns the cube root of a double value
ceil(double a)	double	Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
cos(double a)	double	returns the trigonometric cosine of an angle.
cosh(double a)	double	Returns the hyperbolic cosine of a double value.
exp(double a)	double	Returns Euler's number $e$ raised to the power of a double value.
log(double a)	double	Returns the natural logarithm (base $e$ ) of a double value.
log10(double a)	double	Returns the base 10 logarithm of a double value
pow(double a)	double	Returns the value of the first argument raised to the power of the second argument.

sin(double a)	double	Returns the trigonometric sine of an angle.
sqrt(double a)	double	Returns the correctly rounded positive square root of a double value
tan(double a)	double	Returns the trigonometric tangent of an angle.
convertToTwoDigitPrecision(double a)	double	Converts the number to have a two digit precision after the decimal
convertToThreeDigitPrecision(double a)	double	Converts the number to have a three digit precision after the decimal

Available methods for Class : TA		
Function Name	Return Type	Description
hasPattern( <b>CandleStickPattern</b> .type, <b>Bias</b> .type, <b>int</b> noOfDays )	boolean	Check for the existence of the CandleStick pattern with the Bias (Bullish,Bearish) in the specified days.  <b>Example Usage:</b> <b>TA.hasPattern(CandleStickPattern.DOJI, Bias.BULLISH, 20) ;</b>  See below for the complete list of CandleStick Pattern types
hasPositiveSMACrossover( <b>int</b> lineOnePeriod, <b>int</b> lineTwoPeriod, <b>int</b> days )	boolean	Checks if SMA line-1 Crossed SMA line-2 towards positive (Upward) direction.
hasNegativeSMACrossover( <b>int</b> lineOnePeriod, <b>int</b> lineTwoPeriod, <b>int</b> days )	boolean	Checks if SMA line-1 Crossed SMA line-2 towards negative (Downward) direction.
hasPositiveEMACrossover( <b>int</b> lineOnePeriod, <b>int</b> lineTwoPeriod, <b>int</b> days )	boolean	Checks if EMA line-1 Crossed EMA line-2 towards positive (Upward) direction.
hasNegativeEMACrossover( <b>int</b> lineOnePeriod, <b>int</b> lineTwoPeriod, <b>int</b> days )	boolean	Checks if EMA line-1 Crossed EMA line-2 towards negative (Downward) direction.
hasWilliamsRCrossAbove( <b>int</b> period, )	boolean	Checks if WilliamsR line with period "period" crossed above the value in the specified days.

<b>double</b> value, <b>int</b> lastNoOfDays )		
hasWillamsRCrossBelow( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if WillamsR line with period "period" crossed below the value in the specified days.
hasWillamsRAbove( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if WillamsR line with period "period" is above the value in the specified days.
hasWillamsRBelow( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if WillamsR line with period "period" is below the value in the specified days.
hasRSICrossAbove( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if RSI line with period "period" crossed above the value in the specified days.
hasRSICrossBelow( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if RSI line with period "period" crossed below the value in the specified days.
hasRSIAbove( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if RSI line with period "period" is above the value in the specified days.
hasRSIBelow( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if RSI line with period "period" is below the value in the specified days.
hasMFICrossAbove( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if MFI line with period "period" crossed above the value in the specified days.
hasMFICrossBelow( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if MFI line with period "period" crossed below the value in the specified days.

hasMFIAbove( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if MFI line with period "period" is above the value in the specified days.
hasMFIBelow( <b>int</b> period, <b>double</b> value, <b>int</b> lastNoOfDays )	boolean	Checks if MFI line with period "period" is below the value in the specified days.
hasMACDAboveSignal( <b>int</b> fastPeriod, <b>int</b> slowPeriod, <b>int</b> signalPeriod, <b>int</b> lastNoOfDays )	boolean	Checks if MACD Crossed above the signal line in the specified days.
hasMACDBelowSignal( <b>int</b> fastPeriod, <b>int</b> slowPeriod, <b>int</b> signalPeriod, <b>int</b> lastNoOfDays)	boolean	Checks if MACD Crossed below the signal line in the specified days.
hasTouchedUpperBBand( <b>int</b> period, <b>int</b> deviation, <b>int</b> lastNoOfDays )	boolean	Checks if the underlying's Closing price touched the Upper Bollinger Band in the specified days.
hasTouchedLowerBBand( <b>int</b> period, <b>int</b> deviation, <b>int</b> lastNoOfDays )	boolean	Checks if the underlying's Closing price touched the Lower Bollinger Band in the specified days.
probabilityStockCrossAbove( <b>double</b> price, <b>int</b> nofDays )	double	Returns the probability of stock crossing above the price at the specified number of days.
probabilityStockCrossBelow( <b>double</b> price, <b>int</b> nofDays )	double	Returns the probability of stock crossing below the price at the specified number of days.
probabilityStockTouch( <b>double</b> priceOne, <b>int</b> nofDays )	double	Returns the probability of stock touching the price in specified number of days.
probabilityStockBetween( <b>double</b> priceOne, <b>double</b> priceTwo, <b>int</b> nofDays )	double	Returns the probability of stock being in the range at the specified number of days.

probabilityStockCrossAbove( <b>double</b> price, <b>double</b> volatility, <b>int</b> nofDays )	double	Returns the probability of stock crossing above the price at the specified number of days.
probabilityStockCrossBelow( <b>double</b> price, <b>double</b> volatility, <b>int</b> nofDays )	double	Returns the probability of stock crossing below the price at the specified number of days.
probabilityStockTouch( <b>double</b> priceOne, <b>double</b> volatility, <b>int</b> nofDays )	double	Returns the probability of stock touching the price in specified number of days.
probabilityStockBetween( <b>double</b> priceOne, <b>double</b> priceTwo, <b>double</b> volatility, <b>int</b> nofDays )	double	Returns the probability of stock being in the range at the specified number of days.

CandleStick Pattern Types	
Type	Description
TWO_CROWS	Two Crows Pattern. <b>Example Usage:</b> <code>CandleStickPattern.TWO_CROWS</code>
THREE_BLACK_CROWS	Three Black Crows Pattern
THREE_INSIDE_UP_DOWN	Three Inside Up/Down Pattern
THREE_LINE_STRIKE	Three Line Strike Pattern
THREE_OUTSIDE	Three Outside Up/Down Pattern
THREE_STARS_IN_SOUTH	Three Stars In The South Pattern
THREE_WHITE_SOLDIERS	Three Advancing White Soldiers Pattern
ABANDONED_BABY	Abandoned Baby Pattern
ADVANCE_BLOCK	Advance Block Pattern
BELT_HOLD	Belt Hold Pattern
BREAKAWAY	Breakaway Pattern
CLOSING_MARUBOZU	Closing Marubozu Pattern
CONCEAL_BABY_SWALL	Concealing Baby Swallow Pattern

<b>COUNTERATTACK</b>	Counterattack Pattern
<b>DARK_CLOUD_COVER</b>	Dark Cloud Cover Pattern
<b>DOJI</b>	Doji Pattern
<b>DOJI_STAR</b>	Doji Star Pattern
<b>DRAGONFLY_DOJI</b>	Dragon Fly Doji Pattern
<b>ENGULFING</b>	Engulfing Pattern
<b>EVENING_DOJI_STAR</b>	Evening Doji Star Pattern
<b>EVENING_STAR</b>	Evening Star Pattern
<b>GAP_SIDE_SIDE_WHITE</b>	Up/Down Gap Side-By-Side White Lines
<b>GRAVE_STONE_DOJI</b>	Gravestone Doji
<b>HAMMER</b>	Hammer Pattern
<b>HANGINGMAN</b>	Hanging Man Pattern
<b>HARAMI</b>	Harami Pattern
<b>HARAMICROSS</b>	Harami Cross Pattern
<b>HIGHWAVE</b>	High Wave Pattern
<b>HIKKAKE</b>	Hikkake Pattern
<b>HIKKAKEMOD</b>	Hikkake Modified Pattern
<b>HOMING_PIGEON</b>	Homing Pigeon Pattern
<b>IDENTICAL_THREE_CROWS</b>	Identical Three Crows Pattern
<b>INNECK</b>	In Neck Pattern
<b>INVERTED_HAMMER</b>	Inverted Hammer Pattern
<b>KICKING</b>	Kicking Pattern
<b>KICKING_BY_LENGTH</b>	Kicking - bull/bear determined by the longer marubozu Pattern
<b>LADDERBOTTOM</b>	Ladder Bottom Pattern
<b>LONGLEGGED_DOJI</b>	Long Legged Doji Pattern
<b>LONGLINE</b>	Long line Pattern
<b>MARUBOZU</b>	Marubozu Pattern

<b>MATCHING_LOW</b>	Matching Low Pattern
<b>MATHOLD</b>	Mat Hold Pattern
<b>MORNING_DOJI_STAR</b>	Morning Doji Star Pattern
<b>MORNINGSTAR</b>	Morning Star Pattern
<b>ONNECK</b>	On Neck Pattern
<b>PIERCING</b>	Piercing Pattern
<b>RICKSHAW_MAN</b>	Rickshaw Man Pattern
<b>RISEFALL_THREE_METHODS</b>	Rising/Falling Three Methods Pattern
<b>SEPARATING_LINES</b>	Separating Lines Pattern
<b>SHOOTING_STAR</b>	Shooting Star Pattern
<b>SHORTLINE</b>	Short Line Pattern
<b>SPINNING_TOP</b>	Spinning Top Pattern
<b>STALLED_PATTERN</b>	Stalled Pattern Pattern
<b>STICKSANDWICH</b>	Stick Sandwich Pattern
<b>TAKURI</b>	Takuri Pattern
<b>TASUKIGAP</b>	Tasuki Gap Pattern
<b>THRUSTING</b>	Thrusting Pattern
<b>TRISTAR</b>	Tristar Pattern
<b>UNIQUE_THREE_RIVER</b>	Unique Three River Pattern
<b>UPSIDEGAP_TWO_CROWS</b>	Upside Gap Two Crows Pattern
<b>XSIDEGAP3METHODS</b>	Upside/Downside Gap Three Methods

<b>Bias Types</b>	
<b>BULLISH</b>	<p>Bullish Direction Usage: <b>Bias.BULLISH</b></p> <p><b>Example</b> <code>hasPattern(CandleStickPattern.DOJI, Bias.BULLISH,20);</code></p>

BEARISH	Bearish Direction
NEUTRAL	Neutral Direction

## APL Language Elements

Item	Description
Comments	<p>Also specified using //, e.g.  <b>// This is a comment</b></p> <p>Multiple lines comments are specified using /*...*/, e.g.  <b>/* This is a  multi-line comment */</b></p>
Identifiers / variables	<p>Must start with a-z, A-Z. Can then be followed by 0-9, a-z, A-Z, _ or \$. e.g.</p> <ul style="list-style-type: none"> <li>Valid: <b>var1,a99</b></li> <li>Invalid: <b>9v,!a99,1\$</b></li> </ul> <p>Variable names are <b>case-sensitive</b>, e.g. var1 and Var1 are different variables.</p> <p><b>N.B.</b> the following keywords are reserved, and cannot be used as a variable name or property when using the dot operator:  <b>or, and, eq, ne, lt, gt, le, ge, div, mod, not, null, true, false, new.</b></p>
Scripts	A script in <b>APL</b> is made up of zero or more statements.
Statements	A statement can be the empty statement, the semicolon (;) , block, assignment or an expression. Statements are optionally terminated with a semicolon.
Block	A block is simply multiple statements inside curly braces ({, }).
Assignment	Assigns the value of a variable ( <b>var = 'a value'</b> )
Method calls	<p>Calls a method of an object, e.g.  <b>Math.sqrt(4)</b>  will call the <b>sqrt</b> method from <b>Math</b> object.</p>

## Literals

Item	Description
Integer Literals	1 or more digits from 0 to 9
Floating point Literals	1 or more digits from 0 to 9, followed by a decimal point and then one or more digits from 0 to 9.
String literals	Can start and end with " delimiters, e.g. "Hello world" The escape character is \; it only escapes the string delimiter
Boolean literals	The literals true and false can be used, e.g. <b>val1 == true</b>
Null literal	The null value is represented as in java using the literal null, e.g. <b>val1 == null</b>
Array literal	A [ followed by one or more expressions separated by , and ending with ], e.g. [ 1, 2, "three" ]  This syntax creates an Object[].  APPL will attempt to strongly type the array; if all entries are of the same class or if all entries are Number instance, the array literal will be an MyClass[] in the former case, a Number[] in the latter case.  Furthermore, if all entries in the array literal are of the same class and that class has an equivalent primitive type, the array returned will be a primitive array. e.g. [1, 2, 3] will be interpreted as int[].

## Operators

Operator	Description
Boolean and	The usual && operator can be used, e.g. <b>cond1 &amp;&amp; cond2</b>
Boolean or	The usual    operator can be used <b>cond1    cond2</b>
Boolean not	The usual ! operator can be used e.g. <b>!cond1</b>
Bitwise and	The usual & operator is used, e.g. <b>33 &amp; 4</b> <b>, 0010 0001 &amp; 0000 0100 = 0.</b>
Bitwise or	The usual   operator is used, e.g. <b>33   4</b> <b>, 0010 0001   0000 0100 = 0010 0101 = 37.</b>

Bitwise xor	<p>The usual ^ operator is used, e.g.</p> <p><b>33 ^ 4</b>  <b>, 0010 0001 ^ 0000 0100 = 0010 0100 = 37.</b></p>
Bitwise complement	<p>The usual ~ operator is used, e.g.</p> <p><b>~33</b>  <b>, ~0010 0001 = 1101 1110 = -34.</b></p>
Ternary conditional ?:	<p>The usual ternary conditional operator condition ? if_true : if_false operator can be used ,e.g.</p> <p><b>val1 ? val1 : val2</b></p>
Equality	<p>The usual == operator can be used. For example</p> <p><b>val1 == val2</b></p> <ol style="list-style-type: none"> <li>1. null is only ever equal to null, that is if you compare null to any non-null value, the result is false.</li> <li>2. Equality uses the java equals method</li> </ol>
Inequality	<p>The usual != operator can be. For example</p> <p><b>val1 != val2.</b></p>
Less Than	<p>The usual &lt; operator can be used .For example</p> <p><b>val1 &lt; val2</b></p>
Less Than Or Equal To	<p>The usual &lt;= operator can be used. For example</p> <p><b>val1 &lt;= val2</b></p>
Greater Than	<p>The usual &gt; operator can be used as well as the abbreviation gt. For example</p> <p><b>val1 &gt; val2</b></p>
Greater Than Or Equal To	<p>The usual &gt;= operator can be used. For example</p> <p><b>val1 &gt;= val2</b></p>
Addition	<p>The usual + operator is used. For example</p> <p><b>val1 + val2</b></p>
Subtraction	<p>The usual - operator is used. For example</p> <p><b>val1 - val2</b></p>
Multiplication	<p>The usual * operator is used. For example</p> <p><b>val1 * val2</b></p>
Division	<p>The usual / operator is used. For example</p>

	val1 / val2
Modulus (or remainder)	The % operator is used <b>5 % 2</b>
Negation	The unary - operator is used. For example  -12
Array access	Array elements may be accessed using square brackets e.g.  <b>arr1[0]</b>

## Conditionals

Operator	Description
if	Classic, if/else statement, e.g.  <pre> if ((x * 2) == 5) {     y = 1; } else {     y = 2; } </pre>