Strings in Text Waves

A text wave contains an array of text strings. Each element of the wave can be treated using all of the available string manipulation techniques. In addition, text waves are commonly used to create category axes in bar charts. See **Text Waves** on page II-86 for further information.

String Properties

Strings in Igor can contain up to roughly two billion bytes.

Igor strings are usually used to store text data but you can also use them to store binary data.

When you treat a string as text data, for example if you print it to the history area of the command window or display its contents in an annotation or control, internal Igor routines treat a null byte as meaning "end-of-string". Consequently, if you treat a binary string as if it were text, you may get unexpected results.

When you treat a string as text, Igor assumes that the text is encoded using the UTF-8 text encoding. For further discussion, see **String Variable Text Encodings** on page III-478.

Unicode Literal Characters

A Unicode literal represents a character using its UTF-16 code value. It consists of "U+" followed by four hexadecimal digits. For example:

```
Print "This is a bullet character: " + U+2022
```

The list of Unicode character codes is maintained at http://www.unicode.org/charts.

Escape Sequences in Strings

Igor treats the backslash character in a special way when reading literal (quoted) strings in a command line. The backslash is used to introduce an "escape sequence". This just means that the backslash plus the next character or next few characters are treated like a different character — one you could not otherwise include in a quoted string. The escape sequences are:

`\t`	Represents a tab character
`\r`	Represents a return character (CR)
`\n`	Represents a linefeed character (LF)
`\'`	Represents a ' character (single-quote)
`\"`	Represents a " character (double-quote)
`\\`	Represents a \ character (backslash)
`\ddd`	Represents an arbitrary byte code ddd is a 3 digit octal number
`\xdd`	Represents an arbitrary byte code dd is a 2 digit hex number Requires Igor Pro 7.00 or later
`\udddd`	Represents a UTF-16 code point dddd is a 4 digit hex number Requires Igor Pro 7.00 or later
`\Udddddddd`	Represents a UTF-32 code point dddddddd is an 8 digit hex number Requires Igor Pro 7.00 or later

For example, if you have a string variable called "fileName", you could print it in the history area using:

```
fileName = "Test"
Printf "The file name is \"%s\"\r", fileName
```

which prints