Display Variable Values

The value of variables can be displayed using the **fmt.Println( )** function that was used in Chapter 1 to display the "Hello World!" message. Alternatively, the desired format in which to display the variable value can be specified to a **fmt.Printf( )** function using a suitable "format specifier" and the variable name:

Specifier	Description	Example
%s	A string of characters	**"Go Fun!"**
%d	An integer -32768 to +32767	**100**
%f	A floating-point number	**0.123456**
%c	A single character	**'A'**
%t	A boolean value	**true**
%p	A machine memory address	**0x0022FF34**
%v	The value in a default format	*(any of the above)*
%T	The data type of the variable	**int**

Don't forget

The **%v** format specifier can be used to display any value, and the **%T** format specifier is useful to confirm the data type of any variable.

A format specifier can ensure that the output occupies a minimum number of spaces by stating the required number of spaces after the **%** character – for example, to ensure that an integer always fills at least seven spaces with the specifier **%7d**. If it is preferable for the blank spaces to be filled with zeros, just add a zero to make the specifier into **%07d**.

A precision specifier is a . full stop (period) followed by a number that can be used with the **%f** format specifier to determine how many decimal places to display – for example, to display two decimal places with **%.2f**. The precision specifier can be combined with the minimum space specifier to control the number of spaces and number of decimal places – for example, to display seven spaces including two decimal places and empty spaces filled by zeros with **%07.2f**. By default, empty spaces precede the number so it is right-aligned. They can also be added after the number to make it left-aligned by prefixing the minimum space specifier with a minus sign.

1. Create a directory named "vars" inside your "src" folder

src\vars\main.go

2. Begin a **main.go** program with package and import declarations

```
package main
import "fmt"
```

3. Add a main function that declares and initializes two variables

```
func main( ) {
    num := 100
    pi := 3.1415926536
    // Statements to be inserted here.
}
```

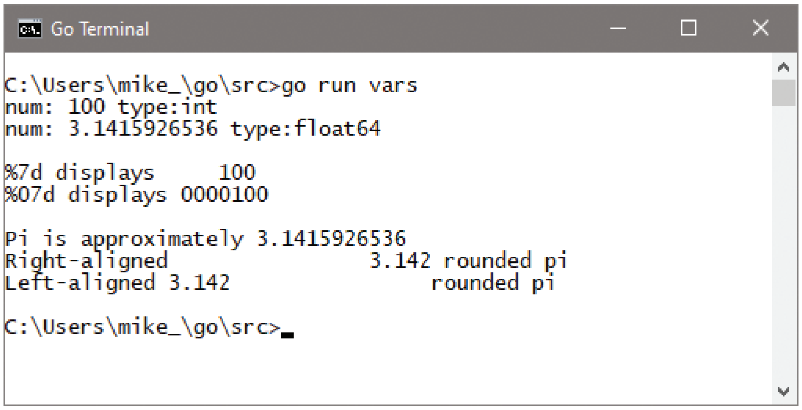
4. In the main function, insert statements to output the variable values in various formats

```
fmt.Printf( "num: %v type:%T \n", num, num )
fmt.Printf( "pi: %v type:%T \n\n", pi, pi )

fmt.Printf( "%%7d displays %7d \n", num )
fmt.Printf( "%%07d displays %07d \n\n", num )

fmt.Printf( "Pi is approximately %1.10f \n", pi )
fmt.Printf( "Right-aligned %20.3f rounded pi \n", pi )
fmt.Printf( "Left-aligned %-20.3f rounded pi \n", pi )
```

5. Save the program file in the "vars" directory, then run the program to see the variable values in the specified formats



Go Terminal

```
C:\Users\mike_\go\src>go run vars
num: 100 type:int
num: 3.1415926536 type:float64

%7d displays     100
%07d displays 0000100

Pi is approximately 3.1415926536
Right-aligned                3.142 rounded pi
Left-aligned 3.142                rounded pi

C:\Users\mike_\go\src>_
```

Hot tip

The **fmt.Printf( )** function does not add a new line after the output. You must manually include a **\n** newline escape sequence to move the printer head to the next line. To display a **%** character with the **fmt.Printf( )** function, prefix it with another **%** character as seen here.

Beware

Notice that the floating-point value is rounded when the format specifier allocates fewer decimal places – it is not simply truncated.