We will receive the formatted output:

```
No      Name        Mark
1       Sarath      80.35
2       James       91.00
3       Jeff        77.56
```

How it works...

`%s`, `%c`, `%d`, and `%f` are format substitution characters for which an argument can be placed after the quoted format string.

`%-5s` can be described as a string substitution with left alignment (`-` represents left alignment) with width equal to `5`. If `-` was not specified, the string would have been aligned to the right. The width specifies the number of characters reserved for that variable. For `Name`, the width reserved is `10`. Hence, any name will reside within the 10-character width reserved for it and the rest of the characters will be filled with space up to 10 characters in total.

For floating point numbers, we can pass additional parameters to round off the decimal places.

For marks, we have formatted the string as `%-4.2f`, where `.2` specifies rounding off to two decimal places. Note that for every line of the format string a newline (`\n`) is issued.

There's more...

While using flags for `echo` and `printf`, always make sure that the flags appear before any strings in the command, otherwise Bash will consider the flags as another string.

Escaping newline in echo

By default, `echo` has a newline appended at the end of its output text. This can be avoided by using the `-n` flag. `echo` can also accept escape sequences in double-quoted strings as an argument. When using escape sequences, use `echo` as `echo -e "string containing escape sequences"`. For example:

```
echo -e "1\t2\t3"
1   2   3
```

Printing a colored output

Producing a colored output on the terminal is very interesting and is achieved by using escape sequences.

Colors are represented by color codes, some examples being, reset = 0, black = 30, red = 31, green = 32, yellow = 33, blue = 34, magenta = 35, cyan = 36, and white = 37.

To print a colored text, enter the following command:

```
echo -e "\e[1;31m This is red text \e[0m"
```

Here, `\e[1;31m` is the escape string that sets the color to red and `\e[0m` resets the color back. Replace `31` with the required color code.

For a colored background, reset = 0, black = 40, red = 41, green = 42, yellow = 43, blue = 44, magenta = 45, cyan = 46, and white=47, are the color codes that are commonly used.

To print a colored background, enter the following command:

```
echo -e "\e[1;42m Green Background \e[0m"
```

Playing with variables and environment variables

Variables are essential components of every programming language and are used to hold varying data. Scripting languages usually do not require variable type declaration before its use as they can be assigned directly. In Bash, the value for every variable is string, regardless of whether we assign variables with quotes or without quotes. Furthermore, there are variables used by the shell environment and the operating environment to store special values, which are called **environment variables**. Let us look at how to play with some of these variables in this recipe.

Getting ready

Variables are named with the usual naming constructs. When an application is executing, it will be passed a set of variables called environment variables. To view all the environment variables related to a terminal, issue the `env` command. For every process, environment variables in its runtime can be viewed by:

```
cat /proc/$PID/environ
```

Set `PID` with a process ID of the process (`PID` always takes an integer value).

For example, assume that an application called **gedit** is running. We can obtain the process ID of gedit with the `pgrep` command as follows:

```
$ pgrep gedit
12501
```