Examine input/output in the Linux shell

In this example, I'll use the echo command to examine how input is received and how output is returned in the shell. Next, I'll use the expr command to further explore input and output while performing some basic calculations in the shell.This is derived from the class, Tools of the Trade, Linux and SQL Module 2 in the Google Cybersecurity Certification Specialty. This is done in an emulation of Bash on cloudskillboost.google

Let's start with outputting text in bash.

```
analyst@96c859d715f2:~$ echo hello
hello
analyst@96c859d715f2:~$ echo "hello"
hello
```

Both with and without "" in the input display the same output in this scenario, however it is important to use quotation marks as they declare the text as a string and remove chance of the bash interpreter misunderstanding special characters, as seen below. This is true in most programming languages.

```
analyst@96c859d715f2:~$ echo ^&*#)(
-bash: syntax error near unexpected token `)'
analyst@96c859d715f2:~$ echo "^&*#)("
^&*#)(
```

Next, let's do some simple calculations.

```
analyst@96c859d715f2:~$ expr 32 - 8
24
analyst@96c859d715f2:~$ expr 3500 * 12
42000
```

expr can be used for adding, subtracting, dividing, and multiplying (+, -, /,*)

```
analyst@96c859d715f2:~$ expr 40 - 4000
-3960
analyst@96c859d715f2:~$ expr 40 / 2
20
```

Improper spacing in bash will not do calculations.

```
analyst@96c859d715f2:~$ expr 40/0
40/0
analyst@96c859d715f2:~$ expr 5+2
5+2
```

Notice, just like a calculator, dividing by zero will result in error.

```
analyst@96c859d715f2:~$ expr 40 / 0
expr: division by zero
```