Debugging

- The process of finding errors in your script or fixing unexpected behaviors is called debugging.

The bash shell provides some options that can help you in debugging your scripts. You can use these options by updating the first line in your script to include one or more of these options.

-x Option

The -x option prints commands and their arguments as they are executed. This menas that instead of variables being displayed, the values of those values are displayed. The same thing goes for expansions. Wildcards aren't displayed, but what the expand to is displayed. You-ll sometimes hear this type of debugging called 'print debugging', 'tracing' or 'x-trace'.

```
#!/bin/bash -x
```

On the command line

```
set -x # Starts debugging behavior
set +x # Stops  debuggin behavior

# These options can also used for a portion of the script
set -x
    # Debugging Behavior Section
set +x
```

-e Option

It causes your script to exit immediately if a command exits with a non-zero status.

```
#!/bin/bash -xe
set -xe
set +xe
```

-v Option

Prints the shell commands just like they are read from the script. Prints everything before any subtitutions and expansions are applied.

The combination -xv to see what line looks like before and after substitutions and expansions occur

```
#!/bin/vash -v
TEST_VAR="test"
```

```
echo "$TEST_VAR"
```

Output

```
#!/bin/bash -vx
TEST_VAR="test"
+ TEST_VAR=test
echo "$TEST_VAR"
+ echo test
>> test
```

set Command

From command line you can run `help set`.

Many times using -x, -e and/or -v is sufficient. But if you want a bit more control over debugging you can create your own codee to do it. One method is to create a variable called DEBUG, set to true if currently debugging or false if not.

Booleans

Bash built-in booleans are true and false. To use a boolean do not quote them.

Syntax Highlighting