codecademy

Bash Scripting

Bash Script Arguments

Arguments can be added to a bash script after the script's name. Once provided they can be accessed by using `$(position in the argument list)`. For example, the first argument can be accessed with `$1`, the second with `$2`, the third with `$3`, etc.

```
#!/bin/bash
# For a script invoked by saycolors red green
blue
# echoes red
echo $1

# echoes green
echo $2

# echoes blue
echo $3
```

Bash Script Variables

Variables in a bash script are set using the `=` sign and accessed using `$`.

```
greeting="Hello"

echo $greeting
```

`read` Keyword

The `read` command can be used to prompt the user for input. It will continue to read user input until the Enter key is pressed.
Some prompt text can also be specified using `-p` with the `read` command.

```
#!bin/bash
echo "Press Enter to continue"
read
read -p "Enter your name: " name
```

Bash Shebang

Bash script files start with `#!/bin/bash`. This special line tells the computer to use *bash* as the intepreter.

Bash Aliases

Aliases can be created using the keyword `alias`. They are used to create shorter commands for calling bash scripts. They can also be used to call bash scripts with certain arguments.

```
# For example, to create an alias that invokes
the saycolor
# script with the argument "green", the
following syntax is used:
alias saygreen='./saycolors.sh "green"'
```

Bash Scripts

Reusuable sets of *bash* terminal commands can be created using *bash scripts*. *Bash scripts* can run any command that can be run in a terminal.

Bash script comparison operators

In bash scripting, strings are compared using the `==` (Equal) and `!=` (Not equal) operators.

```
#!bin/bash
word1="Hello"
word2="Hello"
word3="hello"

if [ $word1 == $word2 ]
then
  echo "Strings are equal"
fi

if [ $word1 != $word3 ]
then
  echo "Strings are not equal"
fi
```