Loops and conditionals

If you need help at any time, put your **red** sticky note on the back of your laptop. When you've finished the steps on the *front* of this page, put your **green** sticky note on the back of your laptop. Then, you can turn the page over and try out some of the more advanced tricks on the back while you wait for the rest of the group to be ready.

Loops!

Like most programming languages, Bash supports loops. Try the following:

```
for i in 1 2 3; do
  echo "$i"
done
```

Here `i` is the loop variable, and to access the value of the variable we put a dollar sign before it: `$i`. There's nothing special about `i`, we can name our variable whatever we want. For example:

```
for animal in "bird" "cat" "dog" "monkey"; do
  echo "I am a $animal" > "$animal".txt
done
```

You can also loop over a list of files. For example, to loop over all the "txt" files in your current directory, try

```
for file in *.txt; do
  echo "There is a file called $file"
done
```

Conditionals

You can also use "if" statements in Bash. For example, try

```
i=2
if [ $i -eq 0 ]
then
  echo "The value is zero!"
fi
```

Try setting

```
i=0
```

then run the "if" statement again.

More conditionals

Bash conditionals can be tricky. Use the Internet as a research aid, and try to write statements that perform different actions based on the value of a variable:

- Print a message only if a string variable starts with “h”
- Print a message only if the file named in the loop variable exists
- Print a message only if a string variable is equal to either “yes” or “no”
- Print a message only if the strings in two different variables are the same.
- Print a message if an integer value is greater than 10.