---

title: Git FUN!damentals
subtitle: History and reverting changes
minutes:

Git history

Git takes care of all these things for you behind the scenes (even branching and merging, which we'll talk about later), but sometimes you need to know what changed and when

```
git log
```

This should give you an output that looks something like this:

```
commit 0a6b46759cfe19b0e9dc450c4c498e7bfe1dd2b7
Author: Example Person <example.person@gmail.com>
Date:   Sun Aug 21 09:42:31 2016 -0700

    adding new files

commit 0514d532e21189099e93d203c3d34b974951ecab
Author: Example Person <example.person@gmail.com>
Date:   Sun Aug 21 09:41:31 2016 -0700

    fruit_list updated

commit a5e55672ccd14860fb9a3602ca2a1cf515bcd878
Author: Example Person <example.person@gmail.com>
Date:   Sun Aug 21 09:41:01 2016 -0700

    fruit_list added
```

This shows the entire history of this repository in reverse chronological order

Finding bugs

Now imagine that someone has issued a bug report for your README file being inaccurate

> You could rewrite it, but that is *WET* and *WET* is *BAD*

Or

> You could find when the accurate README was overwritten and reverting

Have you ever accidentally destroyed something good? Git is the fix for that!

To find when the change occurred, we'll look at the file changes within `log`

```
git log -p
```

```
commit 0a6b46759cfe19b0e9dc450c4c498e7bfe1dd2b7
Author: Example Person <example.person@gmail.com>
Date:   Sun Aug 21 09:42:31 2016 -0700

    adding new files

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..b75f824
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,2 @@
+# Files to ignore
+*.log
diff --git a/CITATION b/CITATION
new file mode 100644
index 0000000..e69de29
diff --git a/LICENSE b/LICENSE
new file mode 100644
index 0000000..e69de29
diff --git a/grapher.R b/grapher.R
new file mode 100644
index 0000000..e69de29
diff --git a/sorter.py b/sorter.py
new file mode 100644
index 0000000..e69de29

commit 0514d532e21189099e93d203c3d34b974951ecab
Author: Example Person <example.person@gmail.com>
Date:   Sun Aug 21 09:41:31 2016 -0700

    fruit_list updated

diff --git a/fruit_list.txt b/fruit_list.txt
index b3231cb..2c3ec01 100644
--- a/fruit_list.txt
+++ b/fruit_list.txt
@@ -1,3 +1,3 @@
 banana
-apple
+kiwi
 peach

commit a5e55672ccd14860fb9a3602ca2a1cf515bcd878
Author: Example Person <example.person@gmail.com>
Date:   Sun Aug 21 09:41:01 2016 -0700

    fruit_list added

diff --git a/fruit_list.txt b/fruit_list.txt
```

```
new file mode 100644
index 0000000..b3231cb
--- /dev/null
+++ b/fruit_list.txt
@@ -0,0 +1,3 @@
+banana
+apple
+peach
(END)
```

So we can see that the file was fine before commit 0514d5, and that it was modified by someone named `Example Person`

Recovering old files

To get a file back to its old state, call checkout with the commit hash and the file name

When you do this on your own computers, keep in mind the hash will be different

```
git checkout a5e556 fruit_list.txt
```

and now if you look at the fruit_list.txt file, you'll see it (and it alone!) has gone back to the way it was before the bug:

```
$ cat fruit_list.txt
```

```
banana
apple
peach
```

and we haven't lost any of the other files:

```
$ ls -a
```

```
CITATION    fruit_list.txt  passwords.log
LICENSE     grapher.R       sorter.py
```

If we decided we actually do like kiwis, to return to the master branch, we can type:

```
$ git checkout master fruit_list.txt
```

And now:

```
$ cat fruit_list.txt
```

will give us our kiwi back.