THE COMMAND LINE

Objectives

- Define what the command line is
- Learn command line navigation and file structure
- Learn to manipulate files and folders via command line

What the Command Line Does

The command line (or terminal) is a faster and more powerful way to maneuver your operating system than by using a GUI (**g**raphical **u**ser **i**nterface), such as Windows Explorer or Mac Finder.

Use special keywords to do everything you can with a GUI *and more.*

```
sed 's/dude/Colt/g' report.txt > report_new.txt
```

How we'll use it

We will use the terminal to:

- navigate around
- create and remove directories and files
- move, copy, and paste things
- use version control (git) to keep track of changes
- Later, execute Python scripts

INSTALLATION TIME

It's actually not that bad!
(for once)

Option 1: Mac

There's a
video for that!

Option 2: PC

There's a
video for that!

Option 3: Cloud9

There's a
video for that!

MAC INSTALLATION

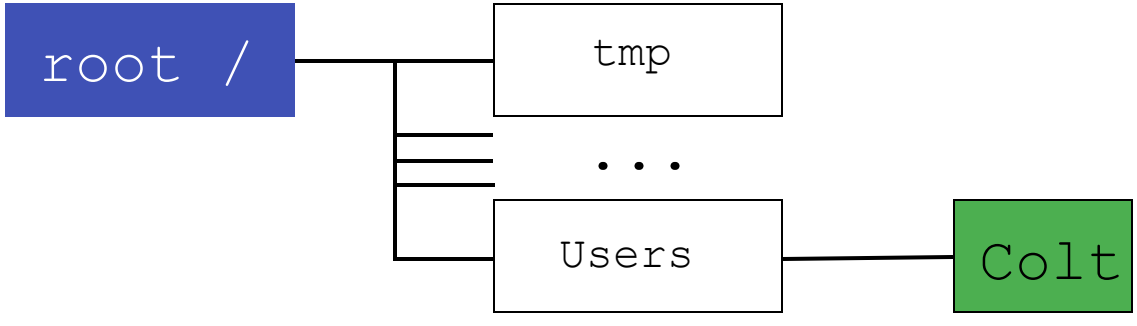
PC
INSTALLATION

Cloud9

INSTALLATION

OS File Structure

Operating Systems organize their folders in a hierarchy (a tree) with parents and children, all relative to a base **root** directory.

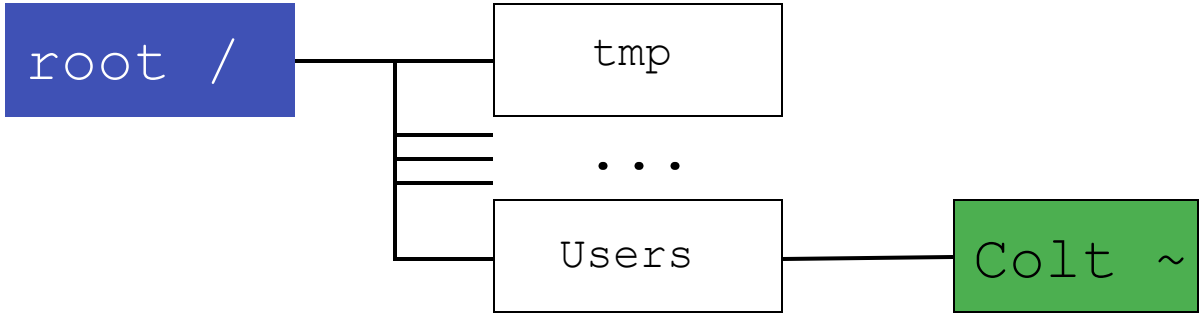


Files and directories have **absolute paths** based on the root, where each additional level down adds a " **/**".

The absolute path for "Colt" is: `/Users/Colt`

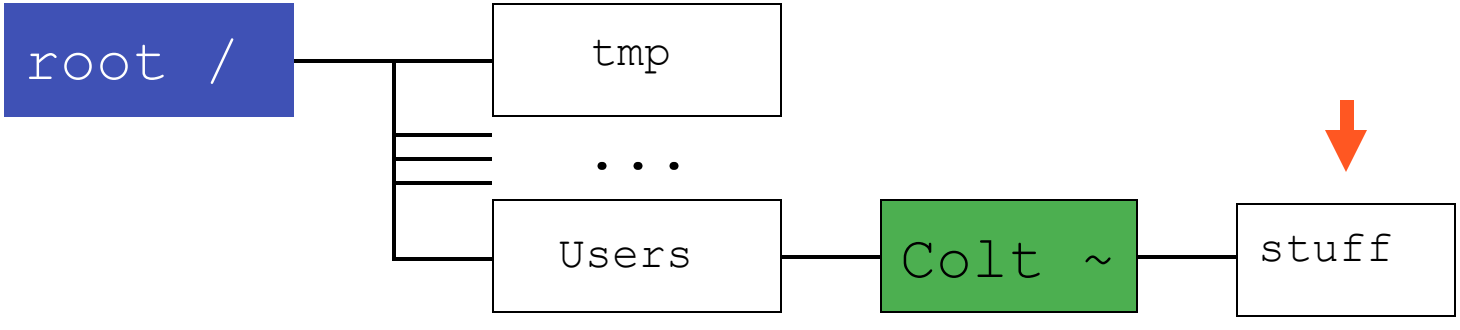
Where am I?

The **green** directory below is a special directory called "home", which is also known as "~". This is the default directory upon opening your terminal.



How do I find out where I am?

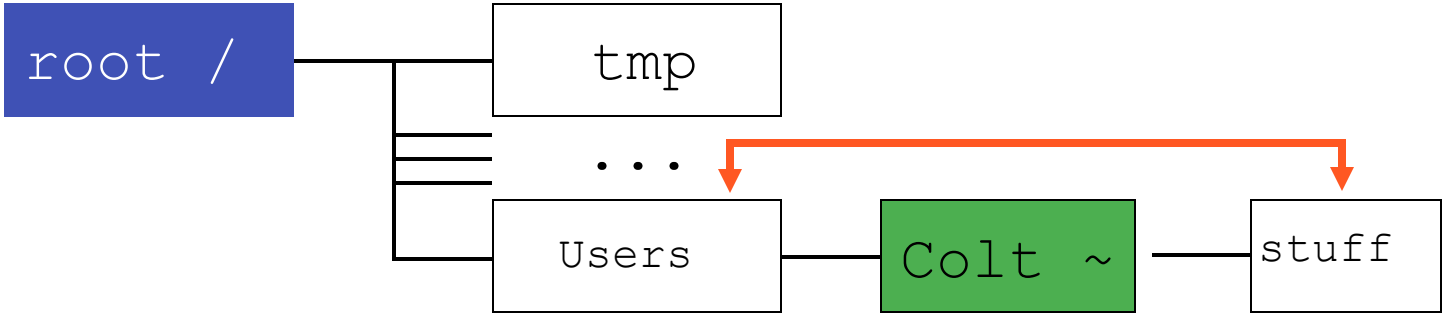
The command "**pwd**" (**p**rint **w**orking **d**irectory) will tell you the full *absolute path* of where you're at!




```
pwd
/Users/Colt/stuff
```

Navigating *Absolutely*

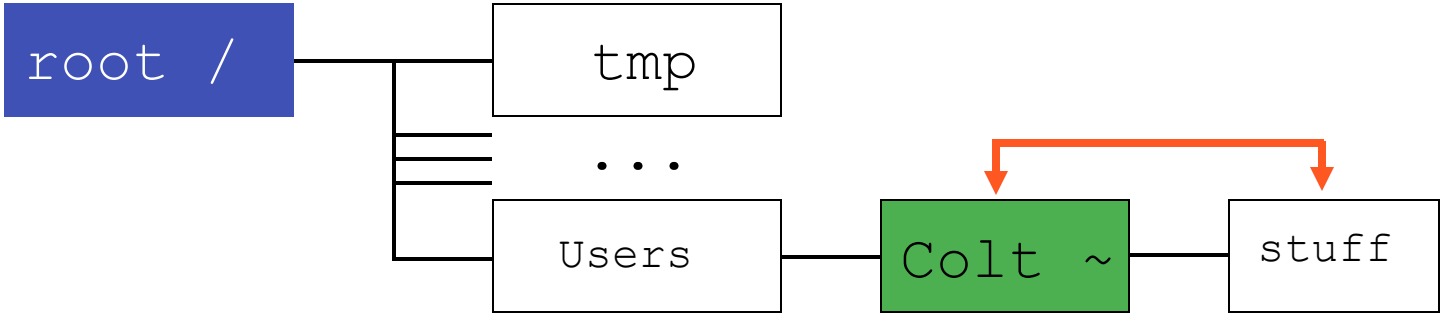
The command "**cd**" ("**c**hange **d**irectory") followed by the absolute path of the folder will navigate you directly there.




```
pwd
/Users/Colt/stuff
cd /Users
pwd
/Users
```

Navigating *Relatively*

The dot "**.**" stands for current directory, and dot-dot "**..**" stands for parent directory. This allows for relative navigation:




```
pwd
/Users/Colt/stuff
cd ..
pwd
/Users/Colt
```

What's Inside?

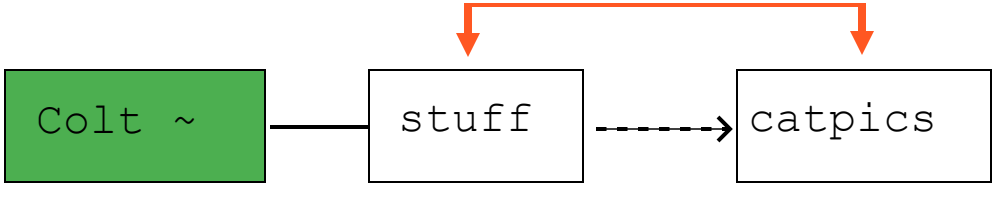
The keyword "**ls**" will "**l**i**s**t" the contents of a directory. You can supply options such as "-**a**" to list **a**ll files (including hidden ones), or "-**l**" for a

```
ls -la
total 136
drwxr-xr-x+   42 csteele  staff   1428 Oct  3 10:10 .
drwxr-xr-x     5 root     admin    170 Sep 20 13:10 ..
drwx------     5 csteele  staff    170 Oct  2 10:28 .Trash
-rw-------     1 csteele  staff   7538 Oct  2 10:29 .bash_history
-rw-r--r--     1 csteele  staff    769 Sep 24 17:06 .bash_profile
-rw-------     1 csteele  staff      7 Sep 20 13:51 .python_histo
drwx------@    4 csteele  staff    136 Sep 20 13:27 Applications
drwx------+   10 csteele  staff    340 Oct  2 10:37 Desktop
drwx------+    4 csteele  staff    136 Oct  1 10:41 Documents
drwx------+   20 csteele  staff    680 Oct  2 10:08 Downloads
drwx------@   59 csteele  staff   2006 Sep 20 19:44 Library
drwx------+    3 csteele  staff    102 Sep 20 13:10 Movies
drwx------+    3 csteele  staff    102 Sep 20 13:10 Music
drwx------+    3 csteele  staff    102 Sep 20 13:10 Pictures
drwxr-xr-x+    5 csteele  staff    170 Sep 20 13:10 Public
```

QUIZ TIME!

Creating Directories

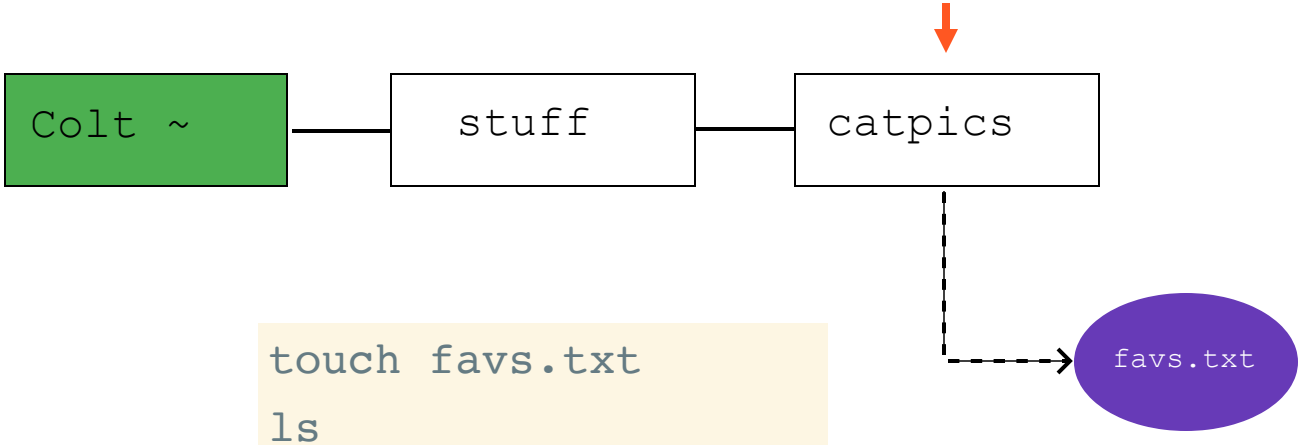
The command "**mkdir**" ("**m**a**k**e **dir**ectory") followed by the name of the new directory will create a new child directory inside the current directory.




```
mkdir catpics
ls
catpics
cd catpics
pwd
/Users/Colt/stuff/catpics
```

Creating Files

The command "**touch**" followed by the filename and file-type extension will create a new file of that type.




```
touch favs.txt
ls
favs.txt
```

SUPER QUICK ACTIVITY!

- Make a new "animals" directory
- Inside of "animals" create "salamanders" and "frogs" directories
- Inside of "salamanders" add a new file "axolotl.txt"
- Inside of "frogs" add a new file: PyxicephalusAdspersus.txt (pixieFrog.txt is fine)

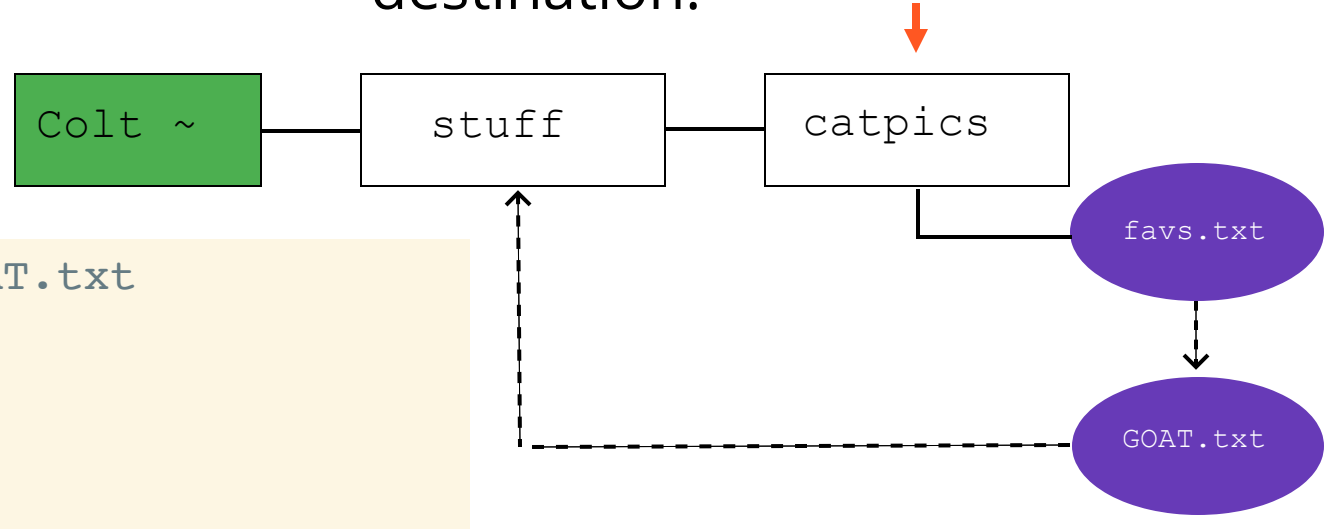
AXOLOTLS ARE AWESOME

- Really adorable smile
- Once worshipped by Aztecs
- Can regenerate limbs, skin, and spinal cord!
- 1000x more resistant to cancer than any other animal on earth!
- They glow in the dark

PIXIE

Moving / Renaming Things

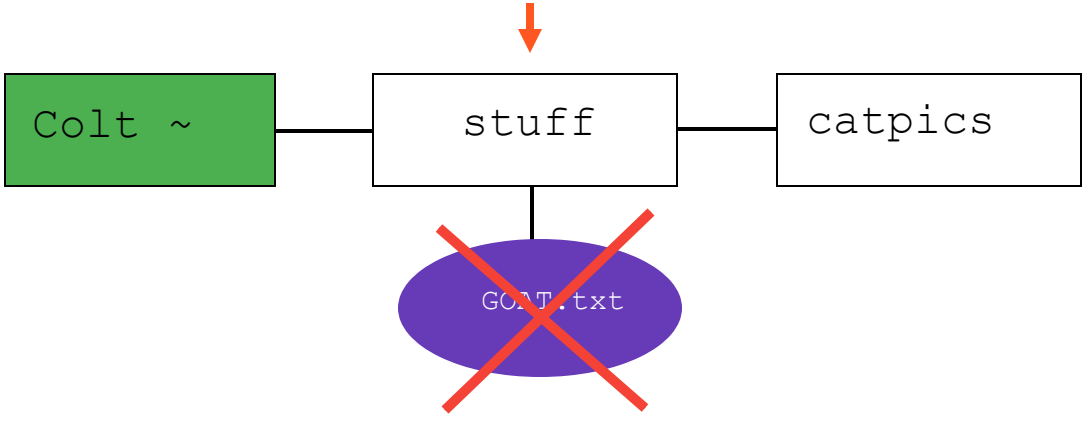
Files can be moved or renamed using the "**mv**" (" **m**o**v**e") keyword, which takes two arguments: the source and the destination.




```
mv favs.txt GOAT.txt
ls
GOAT.txt
mv GOAT.txt ../
ls
cd ..
ls
GOAT.txt catpics
```

Removing Files

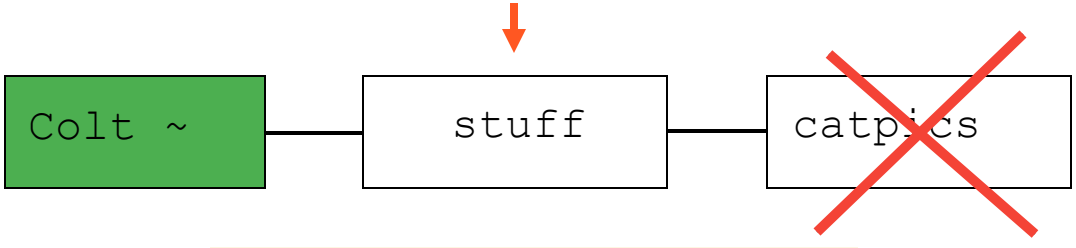
Files can be deleted using the "**rm**" ("re**m**ove") keyword.




```
ls
GOAT.txt catpics
rm GOAT.txt
ls
catpics
```

Removing Directories

Directories can also be deleted using the "**rm**" keyword, with the added option "**-r**" ("**r**ecursive"). You can also use "**-f**" ("**f**orce") to prevent warnings.




```
ls
catpics
rm -rf catpics
ls
```

Warning: "rm -rf" is a dangerous command! Be extremely careful what folder you pass to it because you will never get it back.

QUIZ TIME!

Recap

- OS file structure is hierarchical, tree-based
- Navigate using these commands:
  - cd "change directory"
  - pwd "print working directory"
  - ls "list contents"
- Remember these aliases:
  - / is root directory
  - ~ is home
  - . is current
  - .. is parent
- Manipulate files with:
  - "mkdir" create directories
  - "touch" create files
  - "mv" move and rename
  - "rm" to remove files, "-r" to remove directories

YOUR
TURN

GIT AND GITHUB

Objectives

- Learn the difference between Git and GitHub
- Download & Install Git
- Learn how to stage, commit, and push files to GitHub

Git

Git is the most popular "version control system" - a tool to keep track of file changes over time.

It allows you connect your code to online repositories to back up everything easily.

It also makes collaborating with other people on code projects more manageable through branches.

GitHub

GitHub is an online service that hosts git repositories for developers.

GitHub repositories keep track of entire code history online, so you get both the current state of files and also all previous versions.

It's also a major platform for collaboration on both private enterprise and open source projects

Git Workflow

1a. Initialize a local repository on your computer

1b. Clone (download) an online repository onto your computer

2. Change (create, edit, move, remove) files on your computer

3. Stage the files to be committed

4. Commit the files to a new version

5. Push changes to the online repository

Initializing a Git Repository

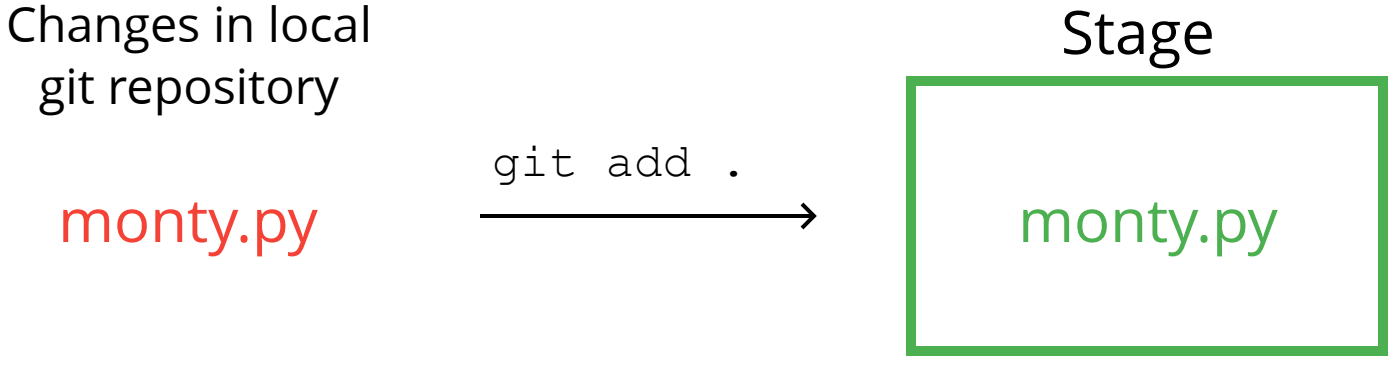
Once git is installed, any directory on your computer can be turned into a repository by typing "**git init**" inside the folder.

```
Initialized empty Git repository in /Users/Colt/test/.git
```

This command creates a "hidden" **.git** folder in the directory which is where your versions are stored.

Staging Files

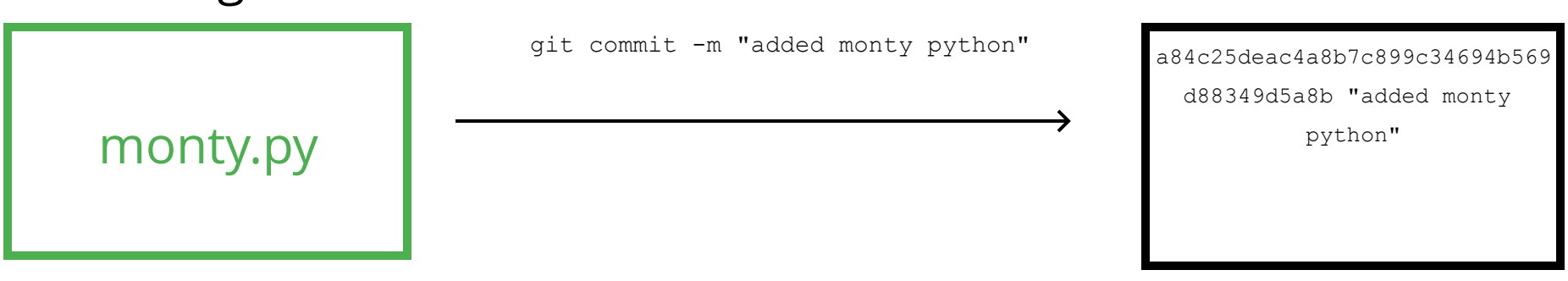
One you have created some files in your repository, you can add them to the stage by typing "**git add**" followed by the filename or parent directory of the files to add all of them.



Staging a file takes a snapshot of it at a point in time, which will prepare it for a new commit (the next version of the repo).

Committing

Use the command "**git commit**" to save a new version of the repository. Use the "**-m**" (**m**essage) argument followed by a short description to explain the new version.



The new commit stores the snapshot in the **.git folder**. Every commit gets **indexed** with an auto-generated unique **hash** so git can find it easily later.

A bit about Remotes

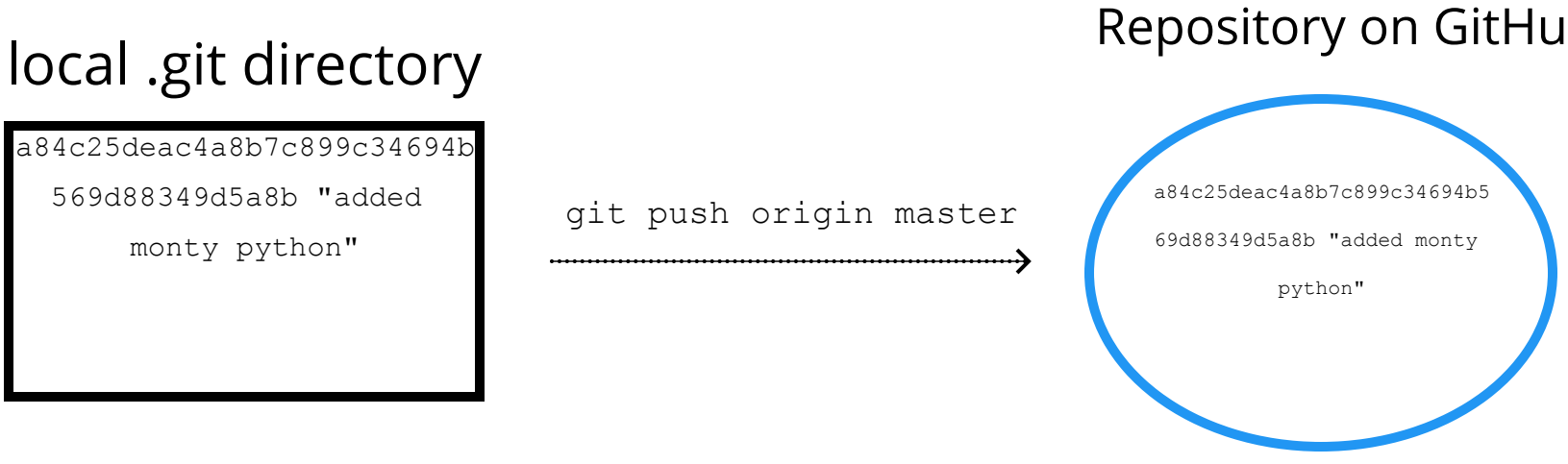
So far, all of the changes we're making are affecting our **local .git** folder only. Our changes are thus only on our machine.

We can add a remote origin by making a GitHub repository and typing the command "**git remote add origin**" like so:

```
git remote add origin https://github.com/<your_github_handle>/<your_repo_name>.git
```

Pushing

After you've set up a remote origin, use the command "**git push**" to publish your changes to your online GitHub repository.



Pushing synchronizes the local and remote repositories. Sometimes you will have to "**git pull**" first to bring your local repo up-to-date before pushing.

Recap

1a. Initialize a local repository on your computer

```
git init
```

or

1b. Clone (download) an online repository onto your computer

```
git clone https://github.com/<your_github_handle>/<your_repo>.git
```

2. Change (create, edit, move, remove) files on your computer

```
echo "just putting text in this file" >> README.md
```

3. Stage the files to be committed

```
git add .
```

4. Commit the files to a new version

```
git commit -m "my first commit"
```

5. Push changes to the online repository

```
git push origin master
```

YOUR TURN