

Seminar 3

Python for Data Analysis

Agenda



Standard Library



Python Modules



Import Statements



Install python Packages



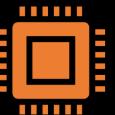
Data Analysis Packages

Pandas
Matplotlib

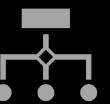
Standard Library

- Built-in Python
 - Every Python session has access to everything we've learned
 - The built-ins are general purpose building blocks: "primitive" data, control flow statements, basic operators, etc
- every Python installation also comes with special functions, methods, data types, operators to address specific types of problems
- Not imported by default

Python Modules



any .py file can also be referred to as a "Python module"



modules can be imported using one of four styles of import syntax. here's one of them



the list of modules already accessible to any vanilla Python installation because they are in the stdlib are listed online at <https://docs.python.org/3/library/> ...»



importing a module makes its code definitions accessible in whatever environment they are being imported to

Anatomy of import syntax 1



if import syntax is:

import module_name



then call syntax is:

module_name.member_name

Anatomy of import syntax 2



if import syntax is:

`import module_name as alias`



then call syntax is:

`alias.member_name`

Anatomy of import syntax 3



if import syntax is:

`from module_name import
member_name`



then call syntax is:

`member_name`

stdlib greatest hits



`datetime`



`random.seed, random.random`



`os.path.exists, os.path.join, os.path.abspath`



`csv.reader, csv.DictReader`



`csv.writer`



`json.loads, json.dumps`

Exercise

Get your feet wet

- In the Python interpreter, try using the 3 different styles of import syntax to import the following **functions**, and call them properly based on the type of import syntax you used.
 - `random.random`
 - `os.getcwd`

You will need to exit and re-enter your python session to clear your prior import syntax each time.

Introducing our data analysis packages

Pandas

- used for processing tabular data
- core data type is the DataFrame
- port of R's DataFrame paradigm

Matplotlib

- used to generate charts such as histograms or box plots from Python data structures
- port of MATLAB's charting functionality

Pandas



DataFrame



Series



Python attributes



DataFrame indexing



Query DataFrames with Boolean series

Create a pandas dataframe



```
var_name = pd.convenience_method(path_as_string)  
read_csv  
read_table  
read_json  
read_excel  
read_html
```



a two dimensional data structure
representing tabular data
has *columns* and *rows*
each column's data is of the same *data type*

use a
convenience
function
against a file
on disk

[pd.read_csv](#), for CSV data

[pd.read_table](#), for general reading of
tabular data, including .tsv files

[pd.read_json](#) for JSON data

[pd.read_excel](#) for Excel files, particularly
useful for excel files with many sheets

[pd.read_html](#) for reading HTML <table>s

Creating a pandas dataframe inline

- `var_name = pd.DataFrame(data=data_castable_to_dataframe)`
- `var_name = pd.DataFrame(data=data_castable_to_dataframe,
columns=list_of_column_names)`

Attributes

Python attributes

- instances of more complex data types have **attributes** associated with them
- they are accessible using the dot notation like `variable_name.attribute_name`
- these are not callable - in practical terms to us at this point, this means they don't need the parentheses () after them - and simply return the static data that attribute refers to

DataFrame attributes

- DataFrames are one case of a data type that has attributes associated with them
- three interesting ones for us are
 - `DataFrame.columns`
 - `DataFrame.shape`
 - `DataFrame.values`

Series



Attributes:

`series.name`
`series.dtype`
`series.shape`



is effectively the one-dimensional representation of a DataFrame axis - for example one row, or one column.

Exercise

Get your feet wet

- Load the data as a pandas DataFrame.
 - HINT: Use a convenience method to pull the data into a DataFrame from a file path!
- Describe the data in the DataFrame using the `describe()` method.
- Select just row 5 from the DataFrame. Now how about the value from row 5, column 2. How about selecting a whole column by its label?
- Use the `groupby()` method against a categorial column in your data.

Applying custom logic cellwise

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”

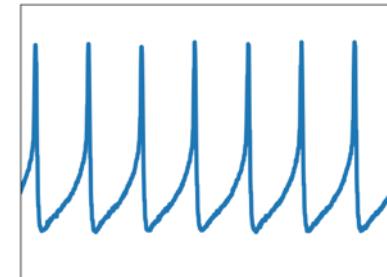
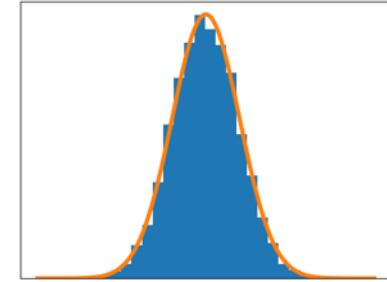
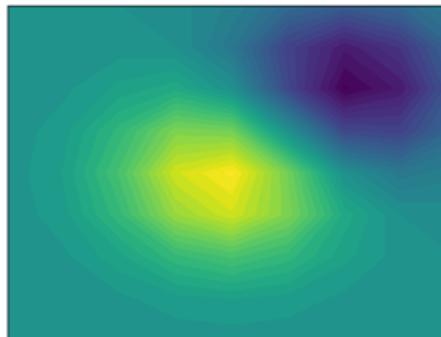
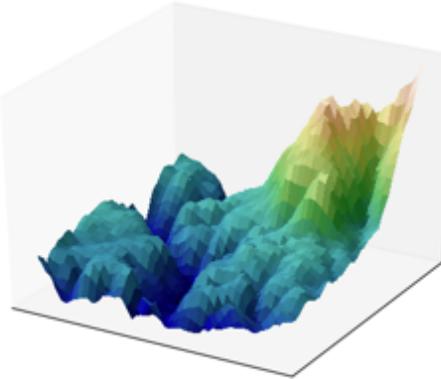
Exercise

Get your feet wet

- Using the chipotle.tsv file from Canvas, answer the following questions.
(HINT: What convenience method works on .tsvs?)
- What is the number of observations in this dataset?
 - HINT: (1) and (2) can be answered with the same DataFrame attribute!
- What is the number of columns in the dataset?
- What are the names of all the columns of this dataset?
- What was the most ordered item?
 - HINT: Consider a groupby with an aggregation!
 - HINT: You will need to add up the quantity field across items of the same item_name and look at the results. There is an aggregation method called sum().
- How many times was a Veggie Salad Bowl ordered?

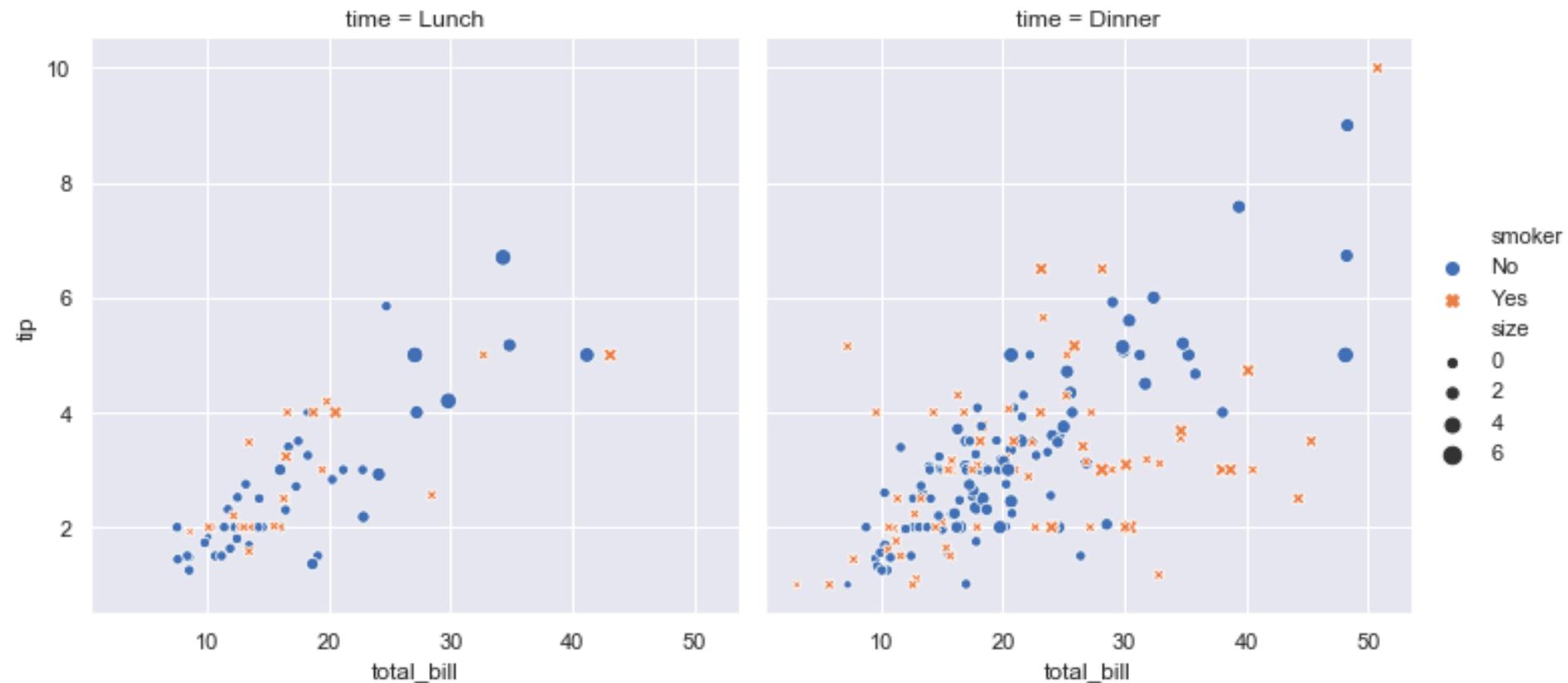
Matplotlib: Visualization with Python

- a comprehensive library for creating static, animated, and interactive visualizations in Python.



Seaborn: statistical data visualization

A Python data visualization library based on [matplotlib](#). It provides a high-level interface for drawing attractive and informative statistical graphics



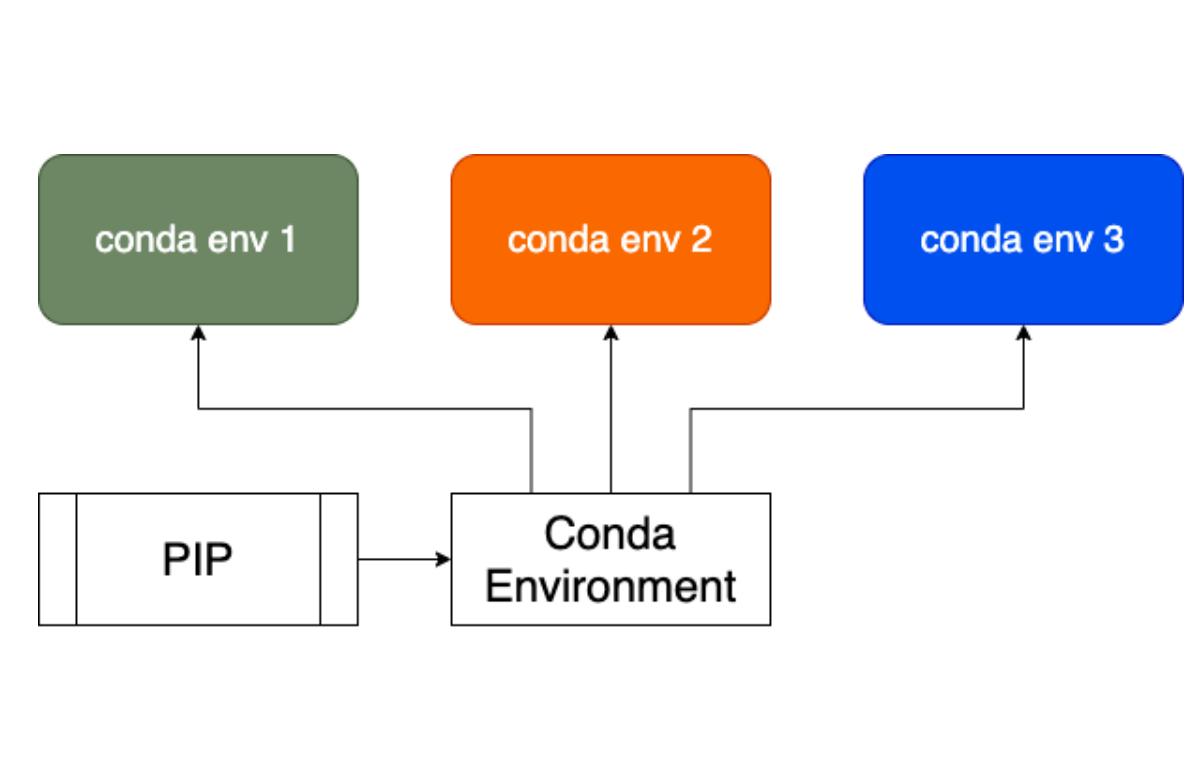
Virtual Environments

Computer environment

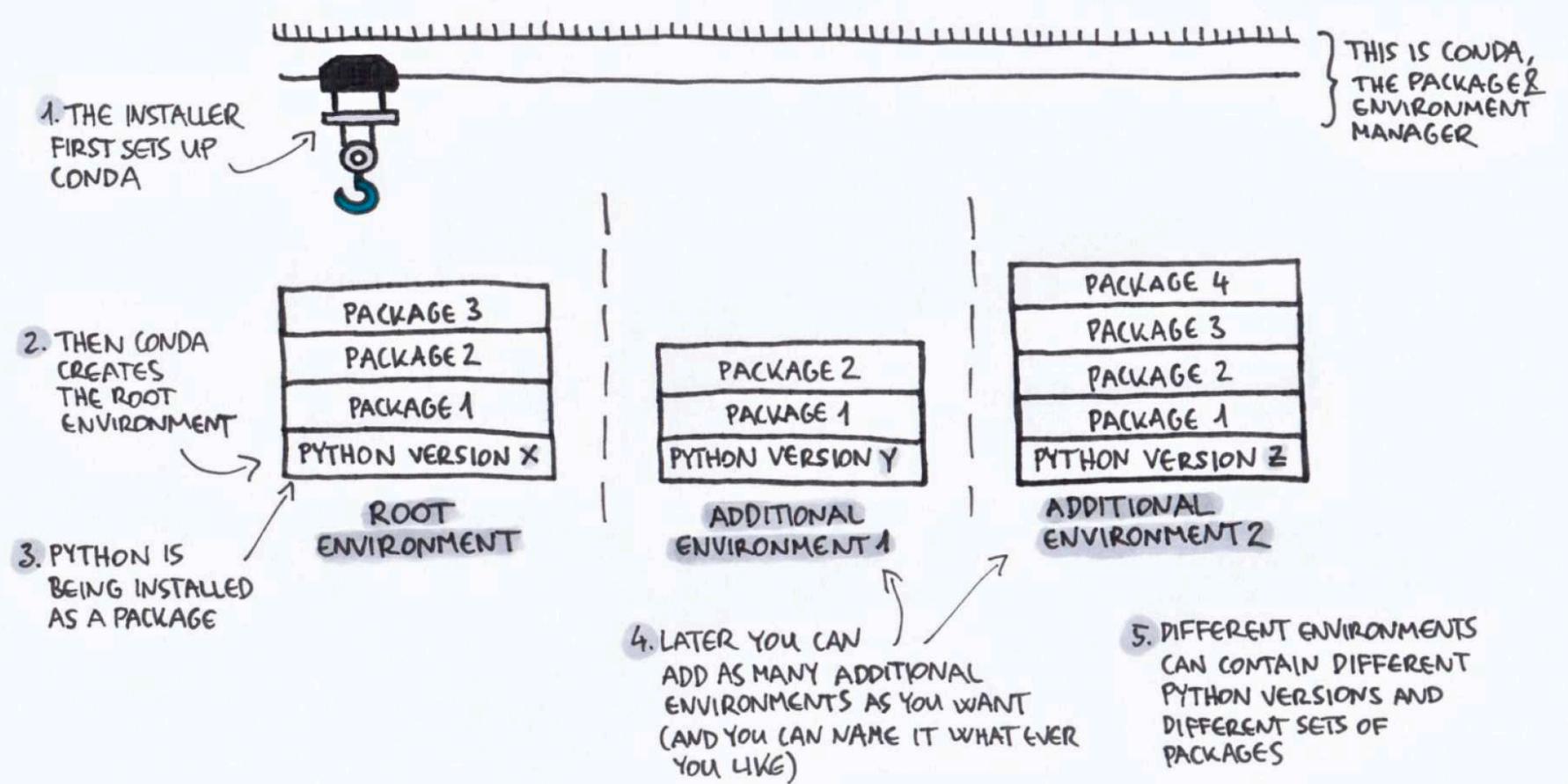
Python (v2.7)
Yellowbrick (v1.0)
Matplotlib (v3.0.0)
scikit-learn (v0.24)
numpy (v1.10.0)

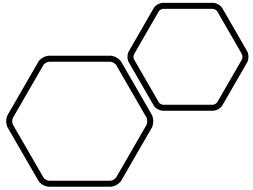
CONDA environment

Python (v2.7)
Yellowbrick (v1.1)
Matplotlib (v3.4.1)
scikit-learn (v0.24)
numpy (v1.13.0)



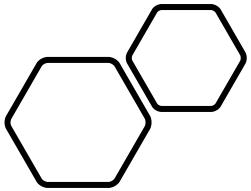
Virtual Environments





Collaboration

Why do we need
Version Control
Systems?



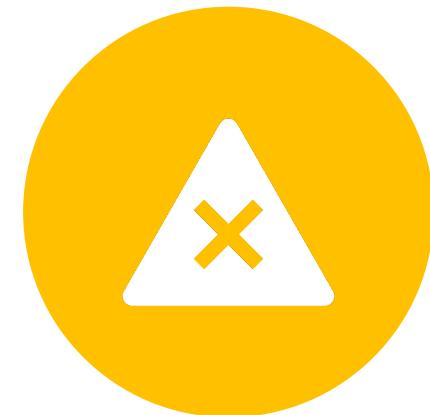
Version Control System



WHAT IS IT?



WHY DO WE NEED IT?

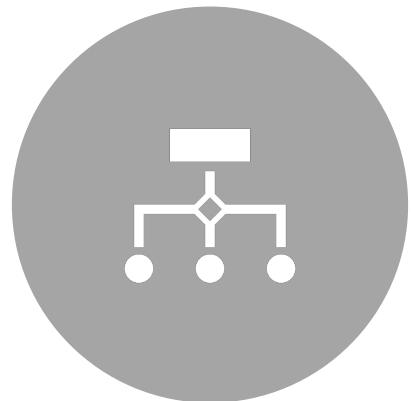


HOW DO WE USE IT?

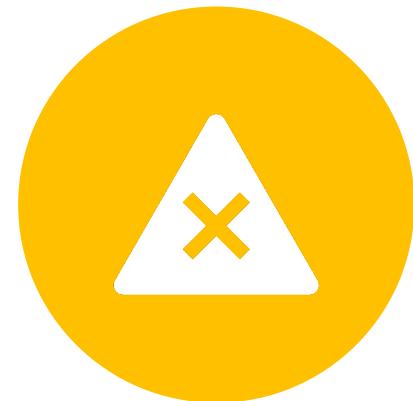
What is a Version Control System?



IT IS A CATEGORY OF SOFTWARE THAT
HELPS YOU MANAGE FILES OVER TIME.



IT TRACKS EVERY MODIFICATION TO
THE FILES



IF MISTAKES ARE MADE OR IF FILES ARE
DELETED, THEY CAN BE RESTORED TO
THEIR PREVIOUS STATE.

GIT – How do we use Version Control?

Git is a Popular Version Control System

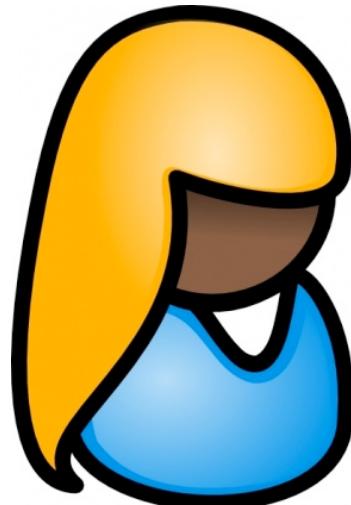
- Repository - A location where files are stored

Git Commands

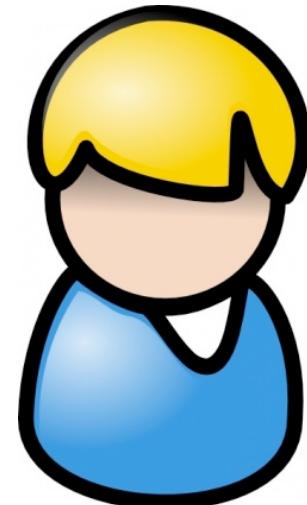
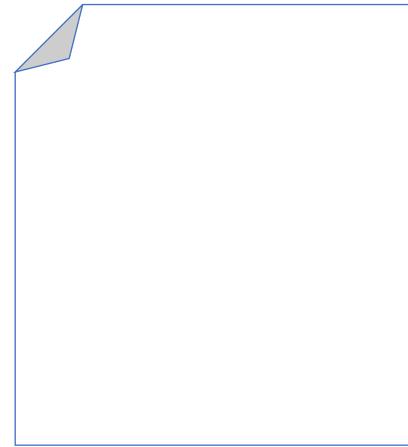
- git init – Initialize a git repository (repo)
- git clone – Copy a remote repo
- git status – Show the current state of the repository
- git add – Add changes to a staging area
- git commit – Save changes
- git branch – Create a copy of a repo
- git checkout – Switch between different copies of a repo
- git merge – Combine changes from different copies of a repo
- git push – Send file changes to another remote repo (GITHUB)
- git pull – Retrieve file changes from a remote repo (GITHUB)

Git and Github

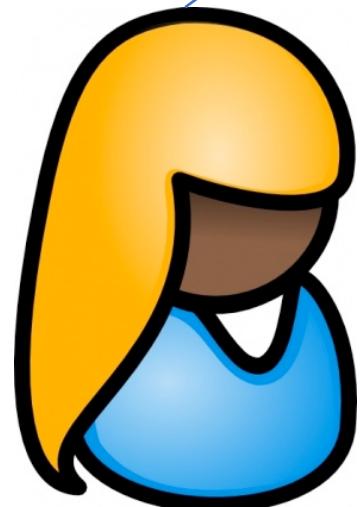
You must work with people



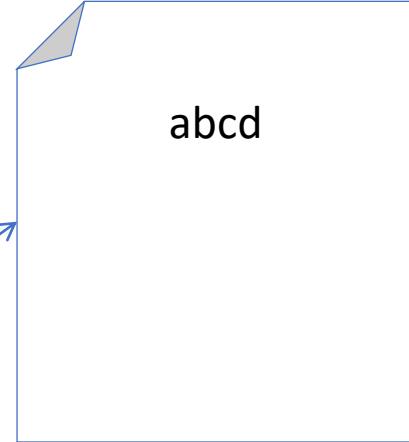
Hello, my name is Sally



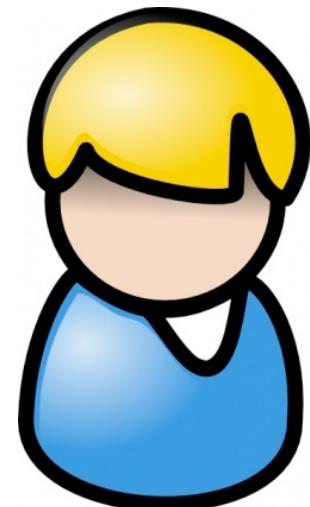
Hello, my name is Bob



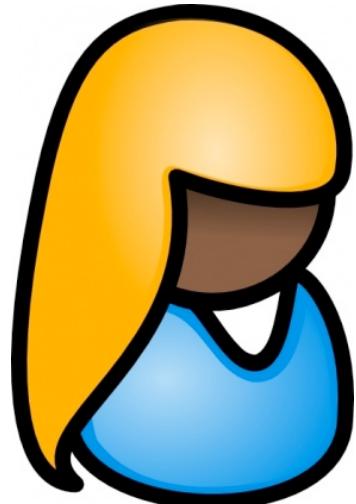
Hello, my name is Sally



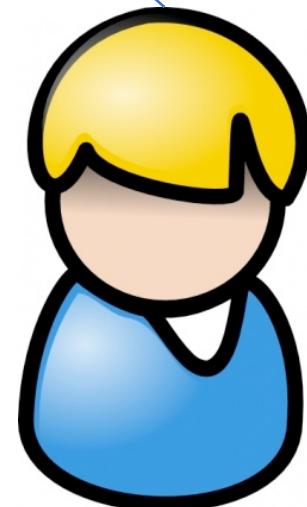
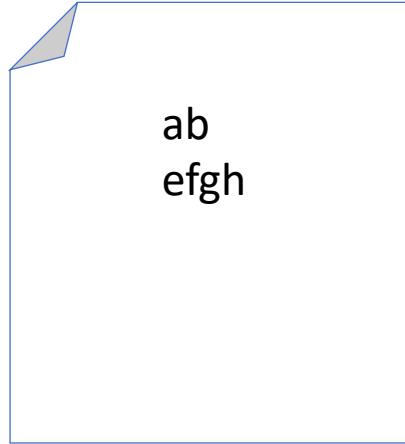
my_file.py



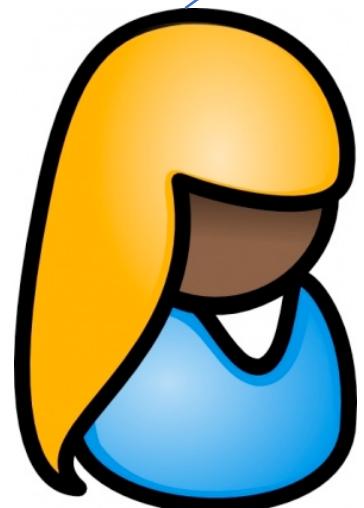
Hello, my name is Bob



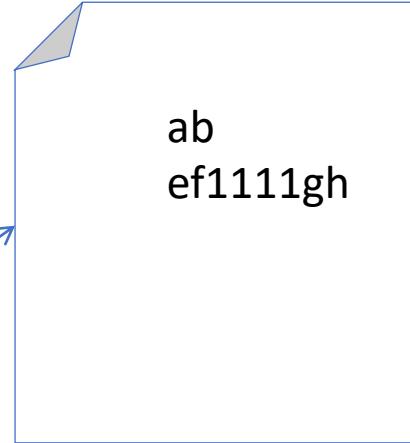
Hello, my name is Sally



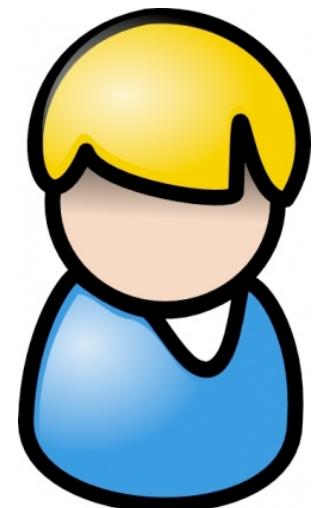
Hello, my name is Bob



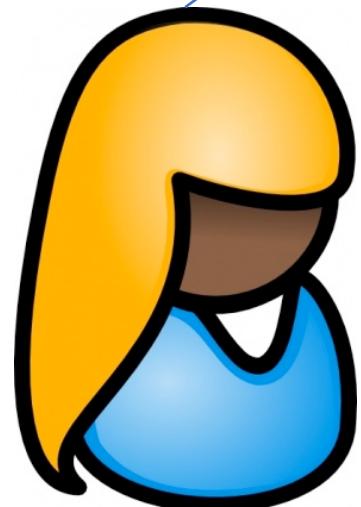
Hello, my name is Sally



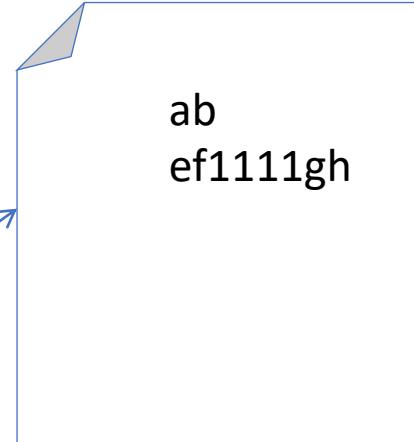
my_file_bobsally.py



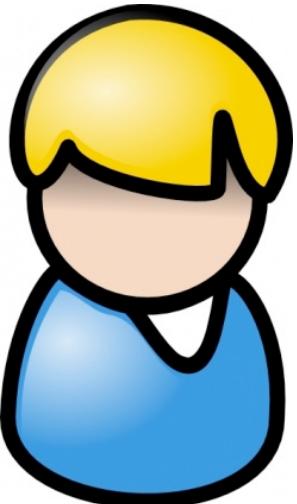
Hello, my name is Bob



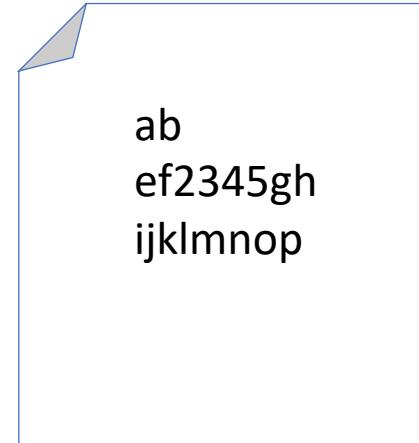
Hello, my name is Sally



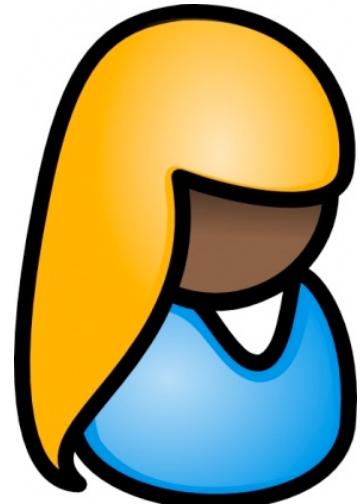
my_file_bobsally.py



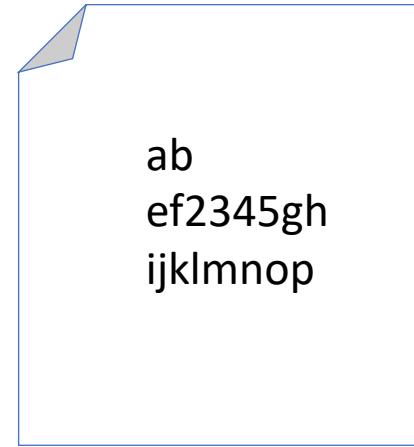
Hello, my name is Bob



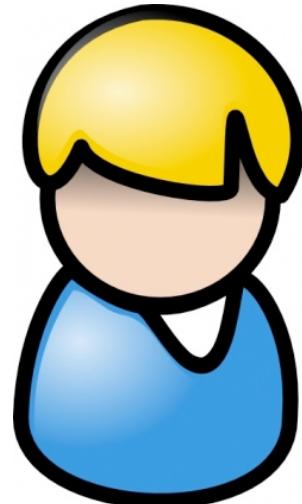
my_file_bob_2.py



Hello, my name is Sally

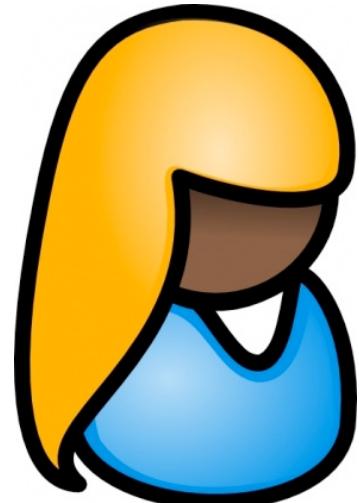


my_file_bob2_FINAL.py

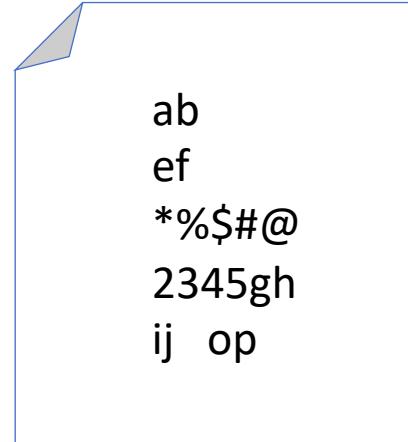


Hello, my name is Bob

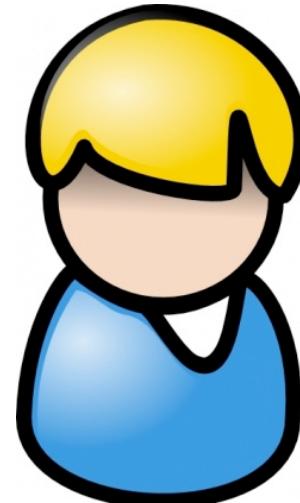
CHAOS



Hello, my name is Sally

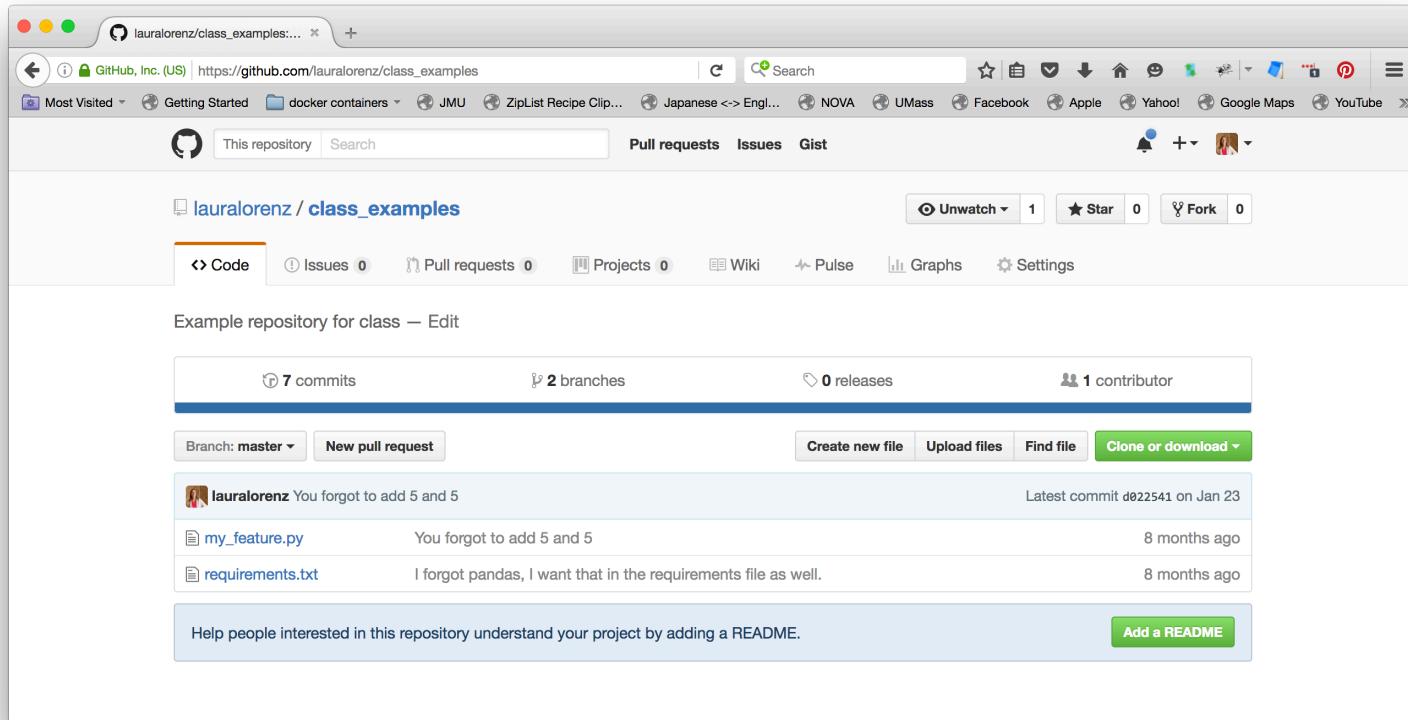


my_file_bob2_FINALFINAL.py

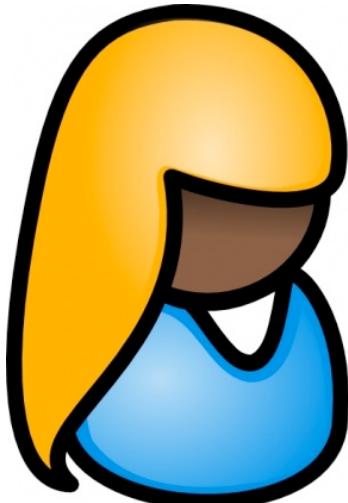


Hello, my name is Bob

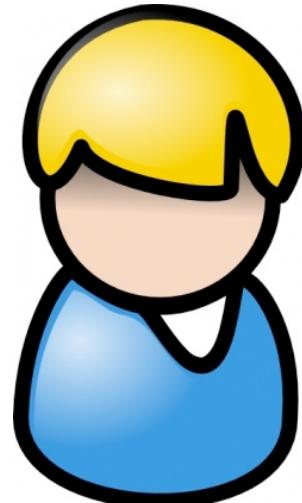
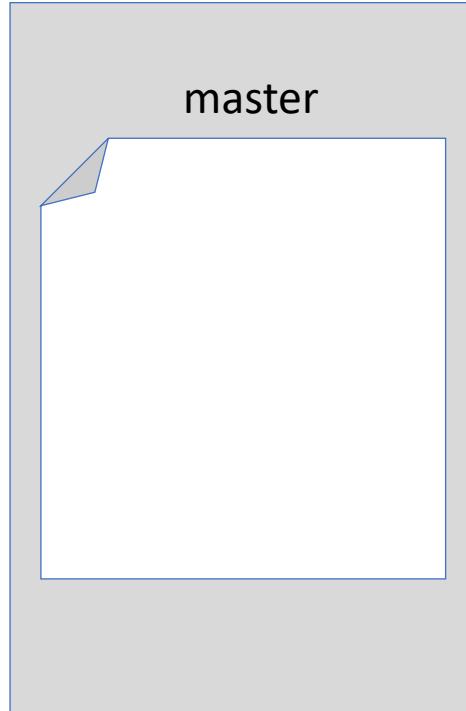
Git and Github



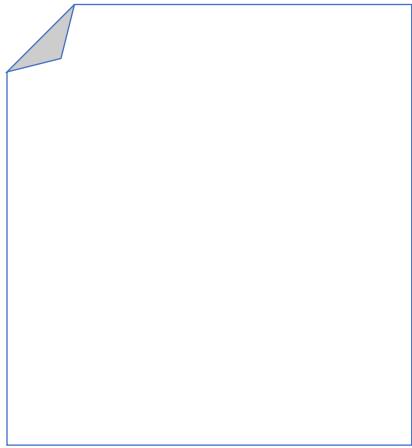
It can be better



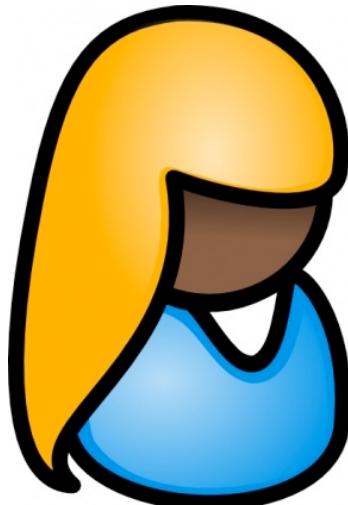
Hello, my name is Sally



Hello, my name is Bob

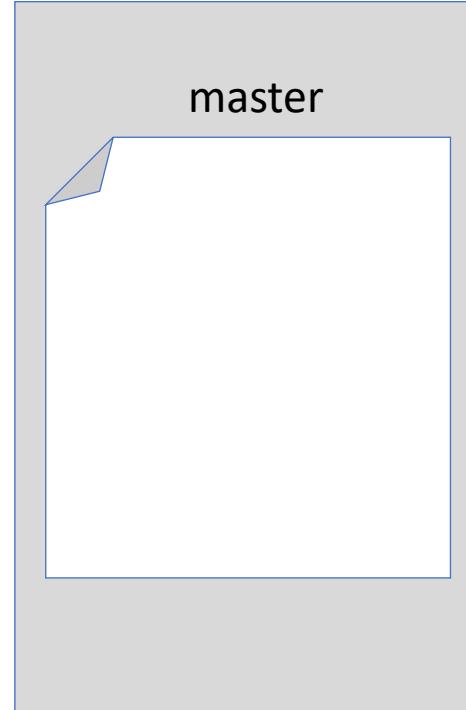


my_file.py

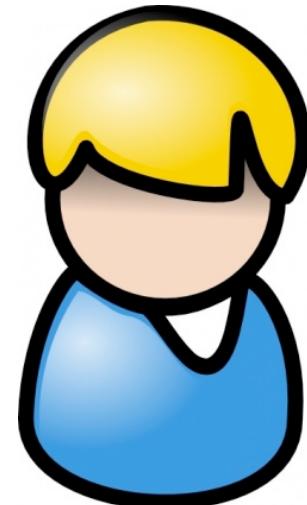


Hello, my name is Sally

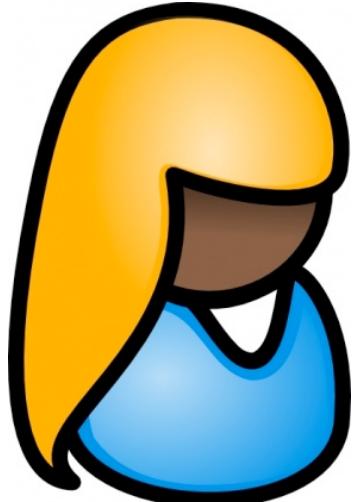
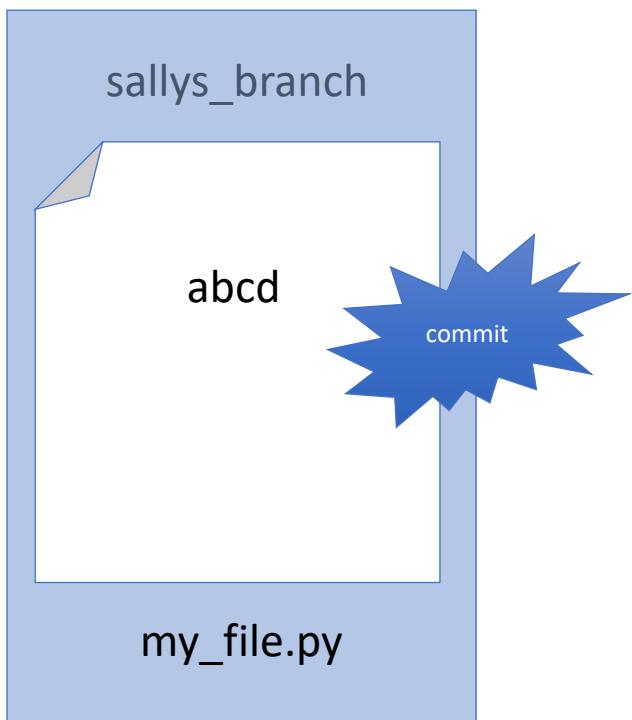
sallys_branch
branch



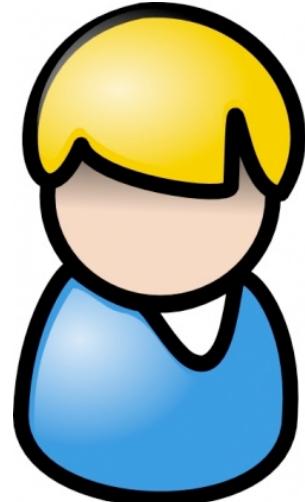
master



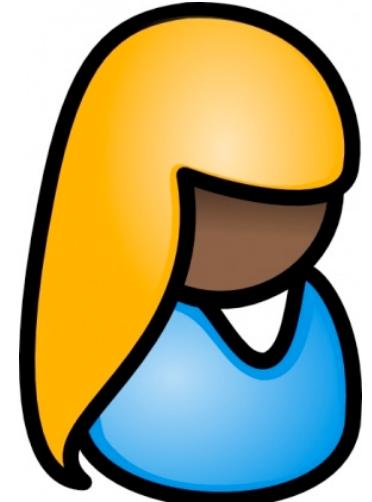
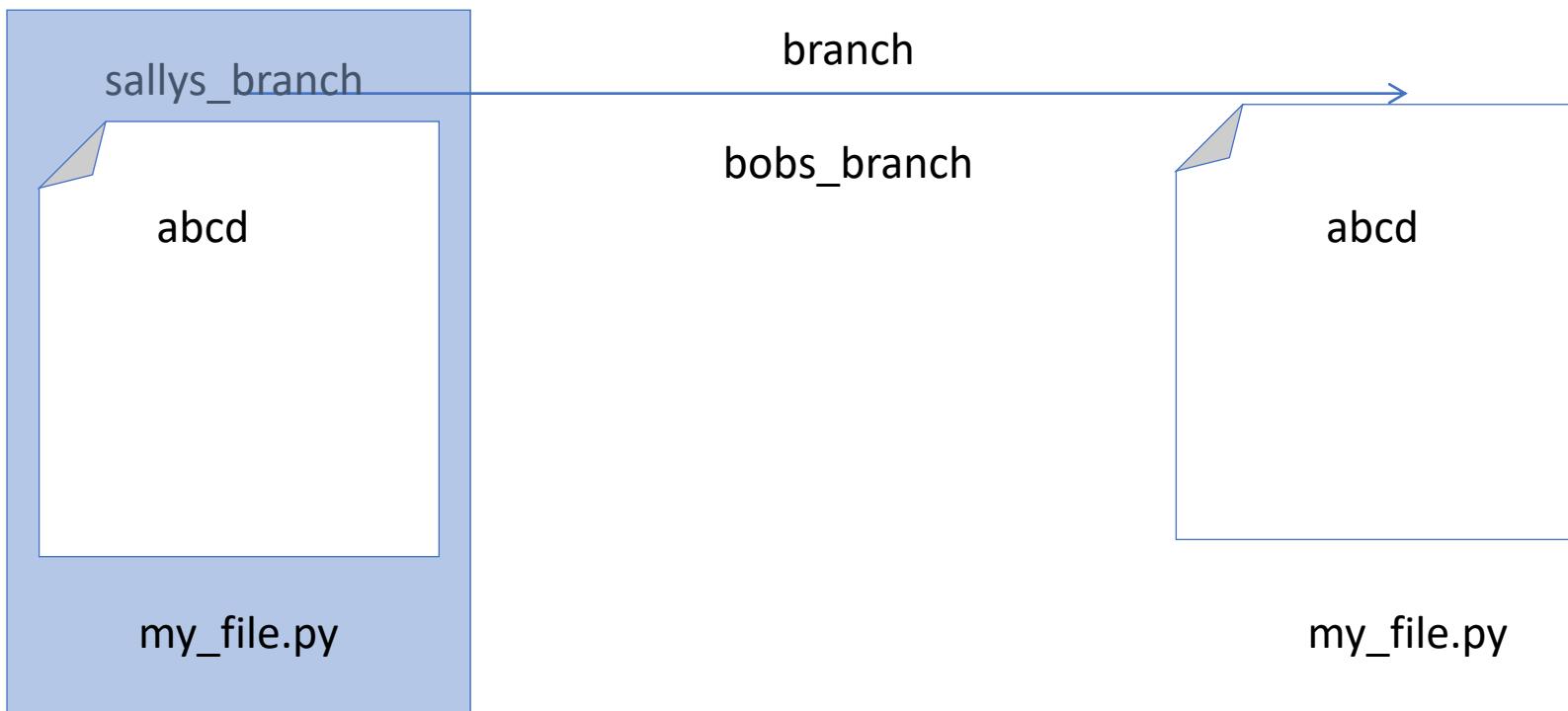
Hello, my name is Bob



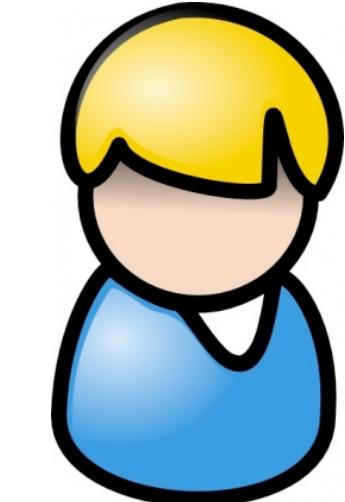
Hello, my name is Sally



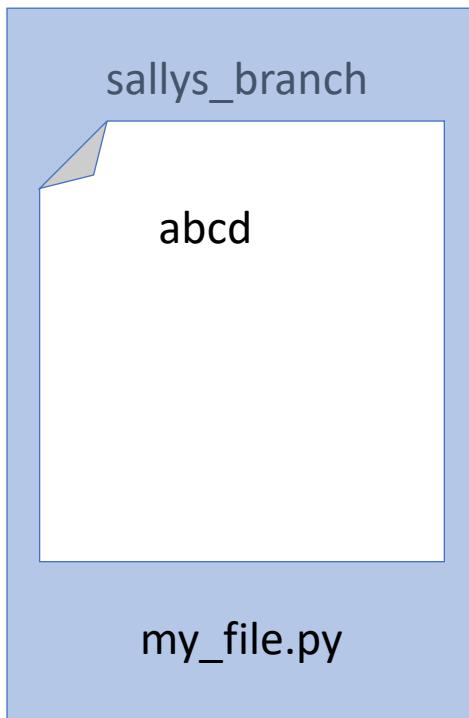
Hello, my name is Bob



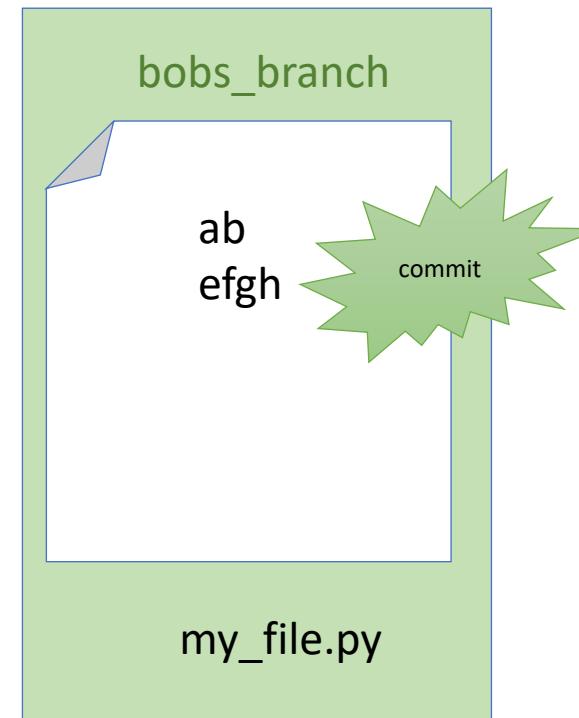
Hello, my name is Sally



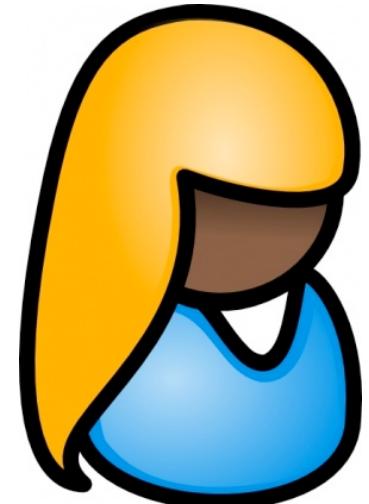
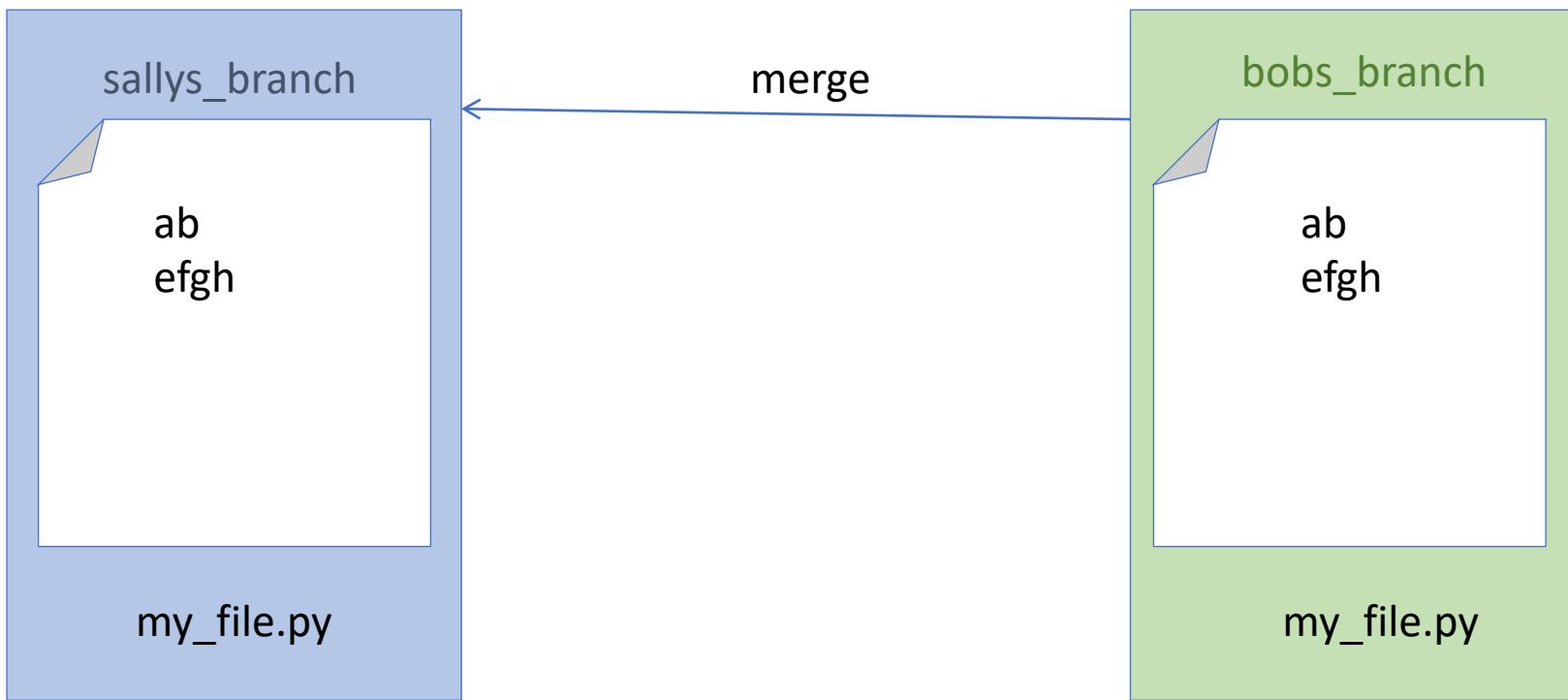
Hello, my name is Bob



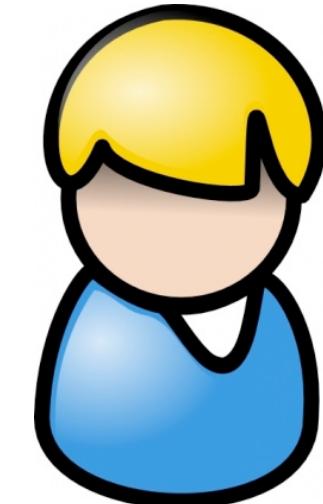
Hello, my name is Sally



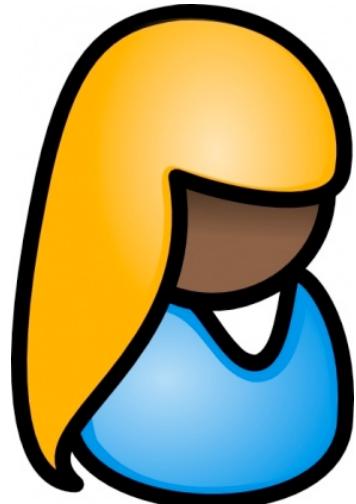
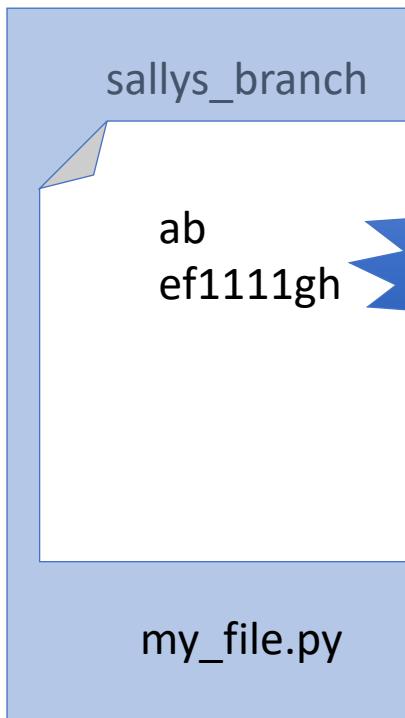
Hello, my name is Bob



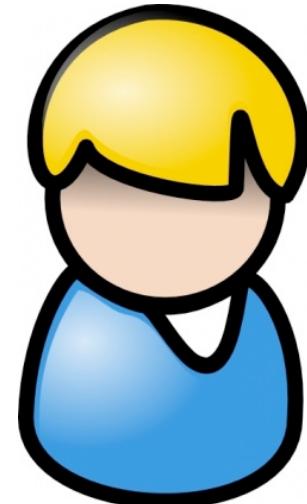
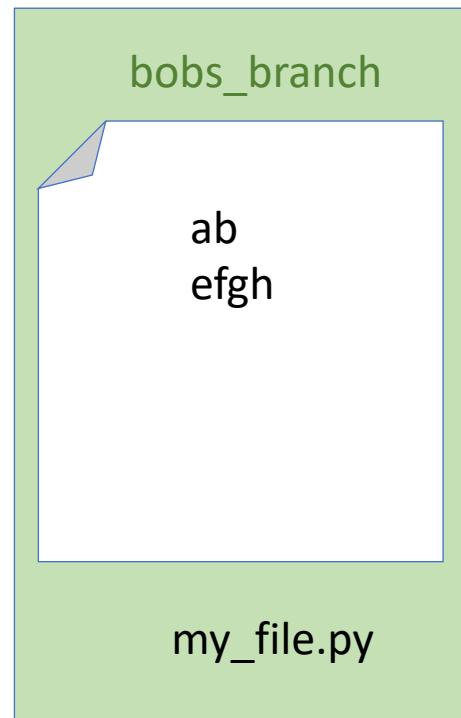
Hello, my name is Sally



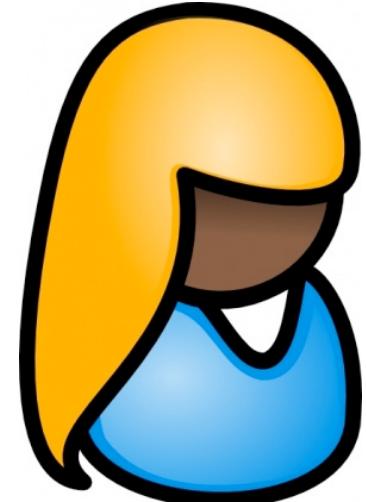
Hello, my name is Bob



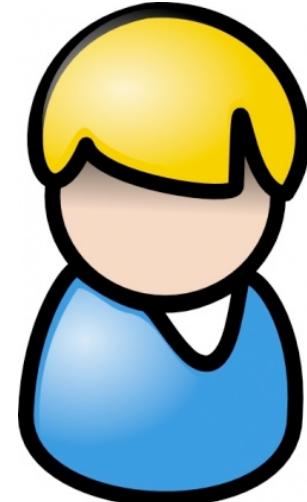
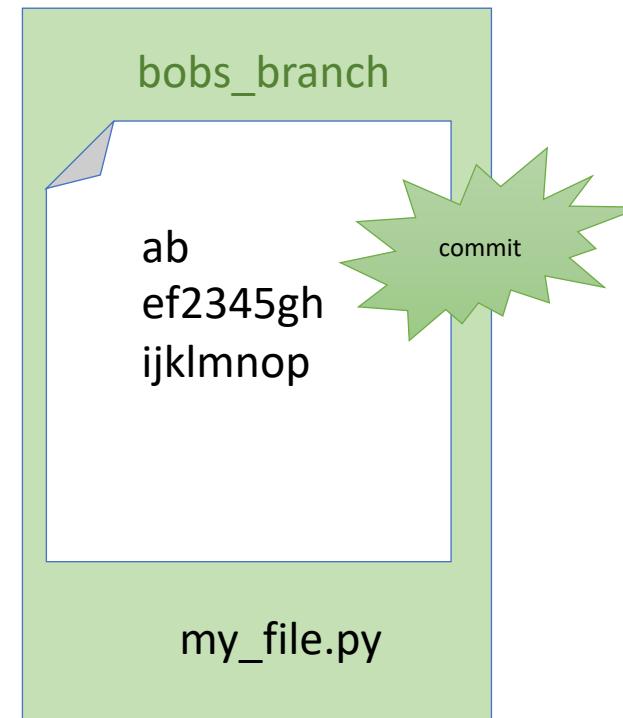
Hello, my name is Sally



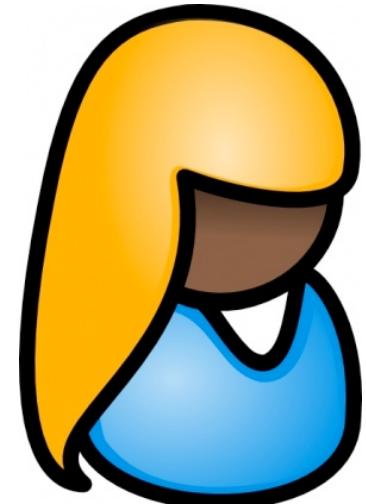
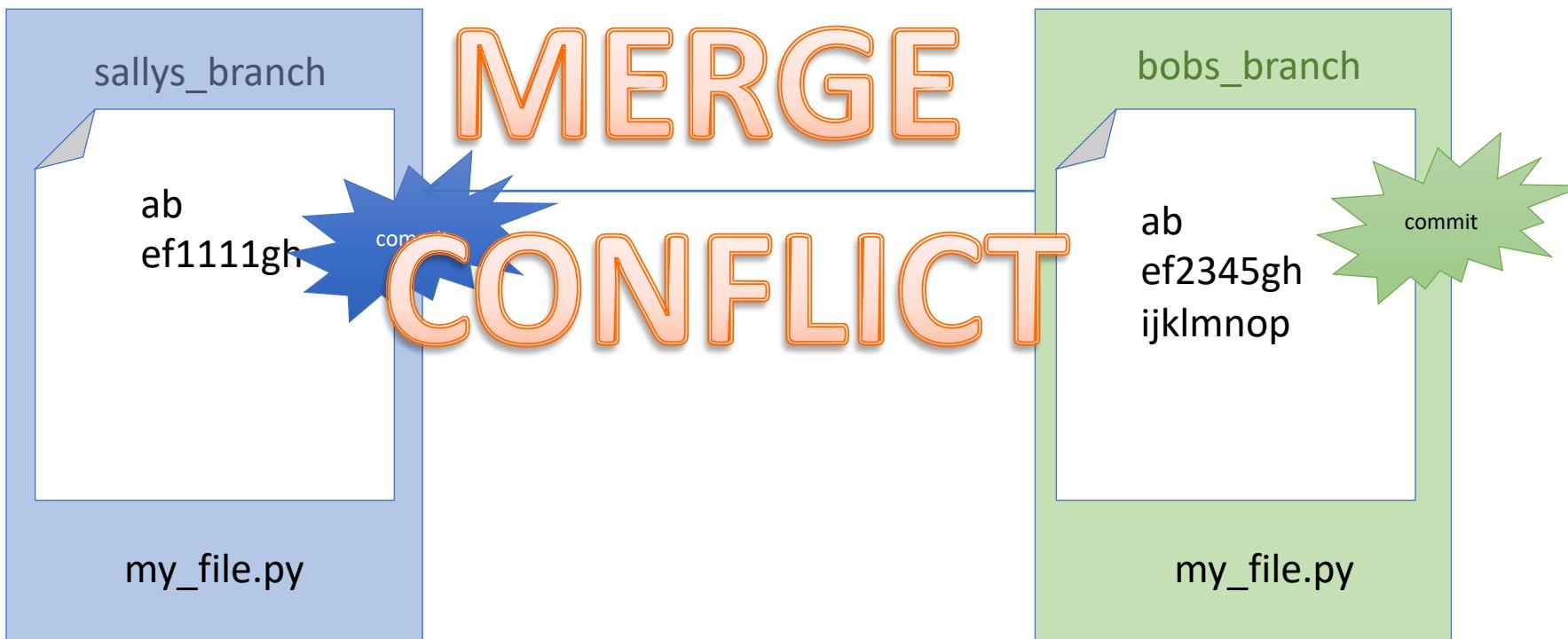
Hello, my name is Bob



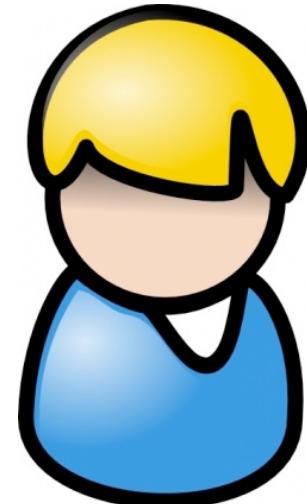
Hello, my name is Sally



Hello, my name is Bob

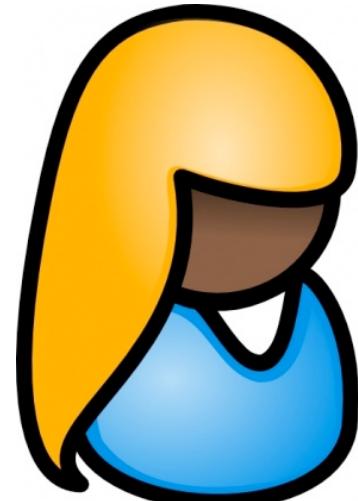
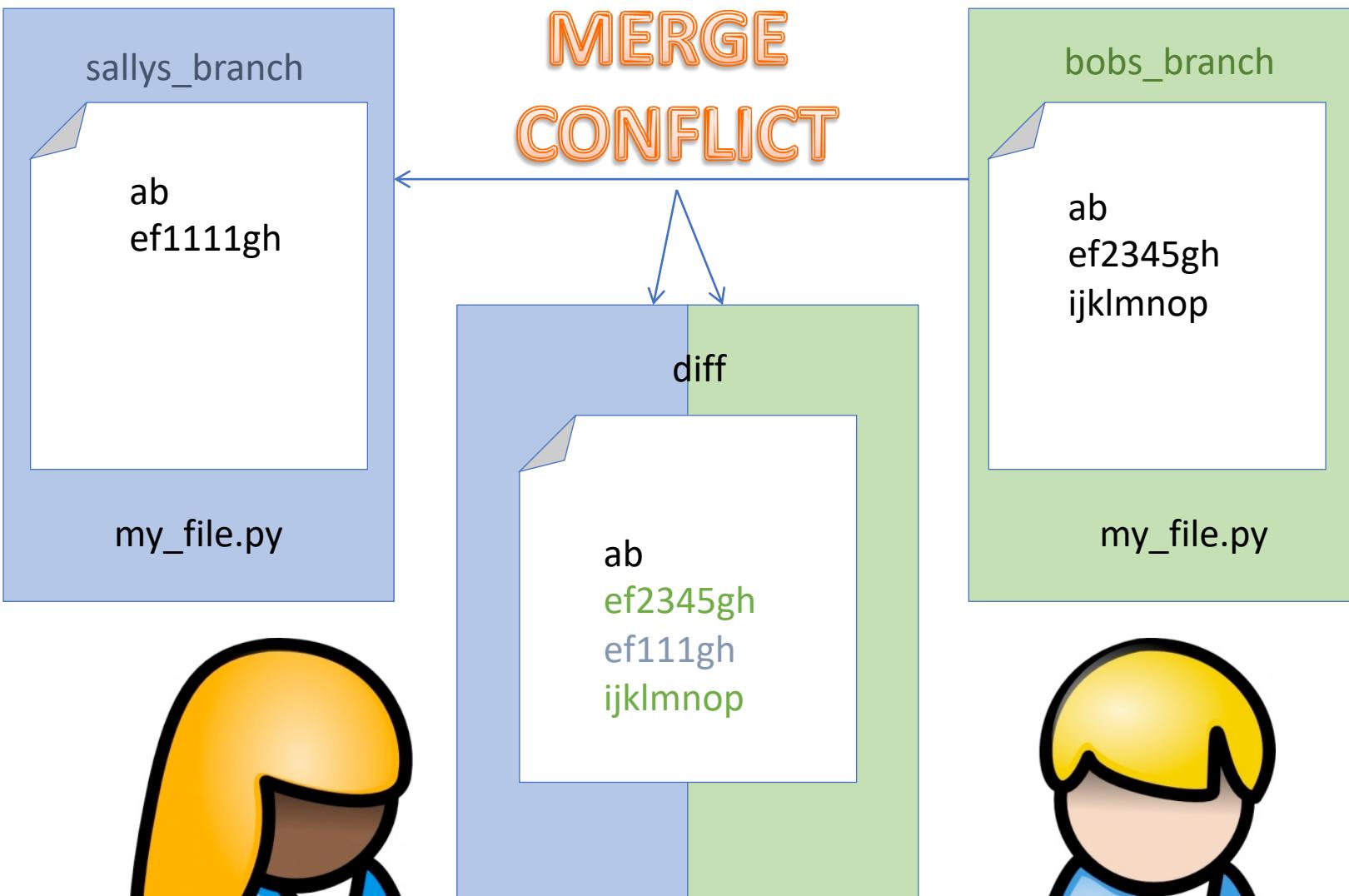


Hello, my name is Sally

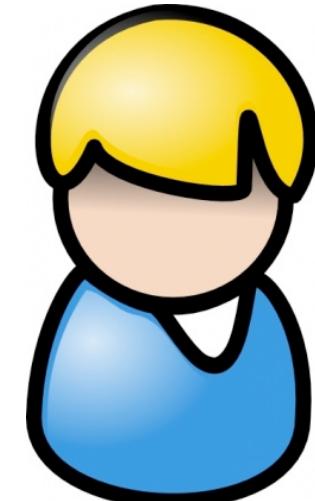


Hello, my name is Bob

MERGE CONFLICT



Hello, my name is Sally



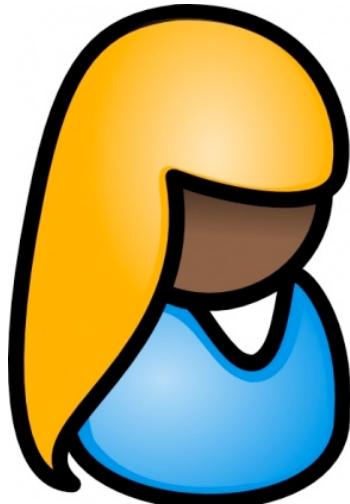
Hello, my name is Bob

sallys_branch

```
ab  
ef1111gh  
ijklmnop
```

commit

my_file.py

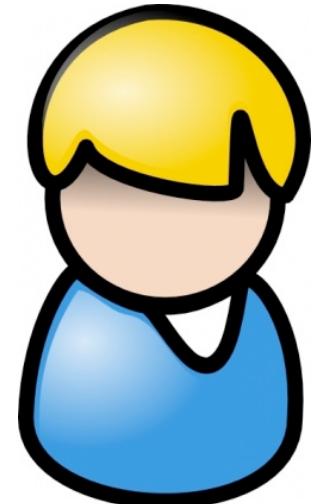


Hello, my name is Sally

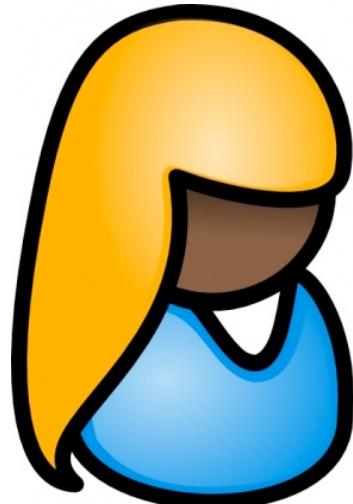
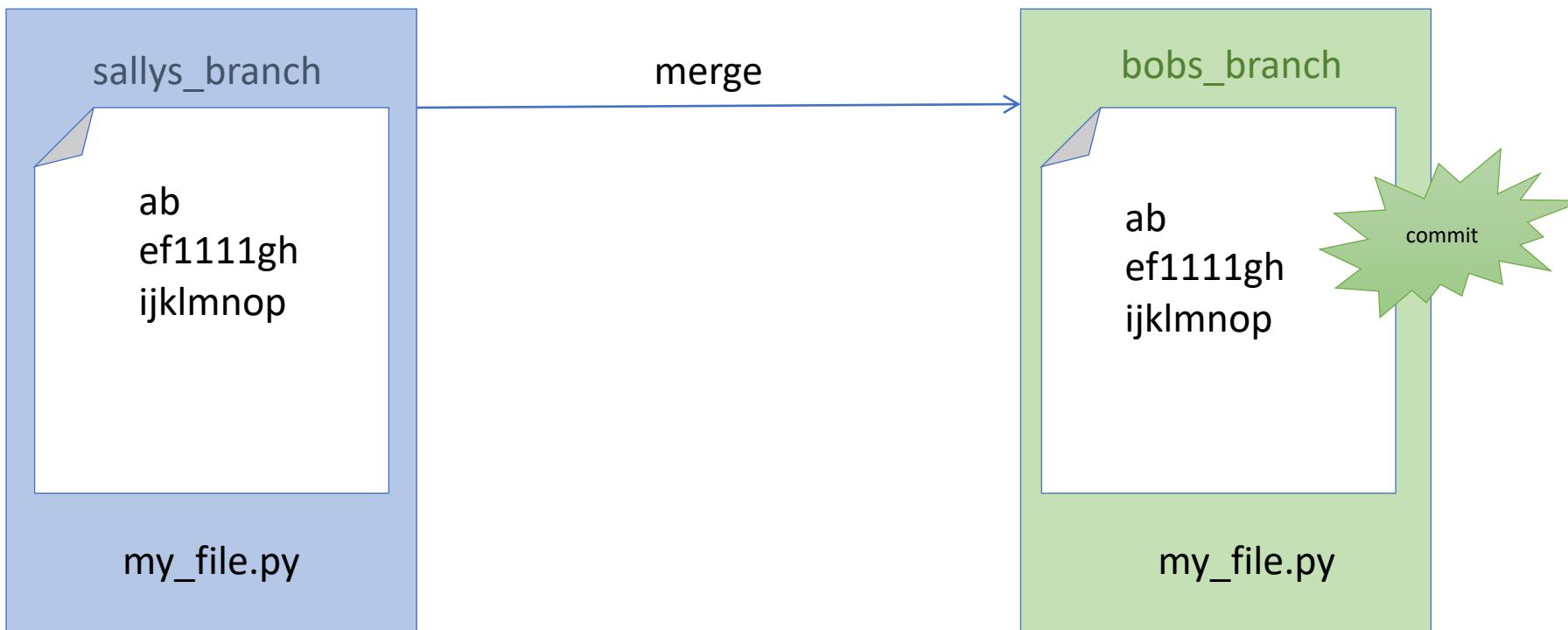
bobs_branch

```
ab  
ef2345gh  
ijklmnop
```

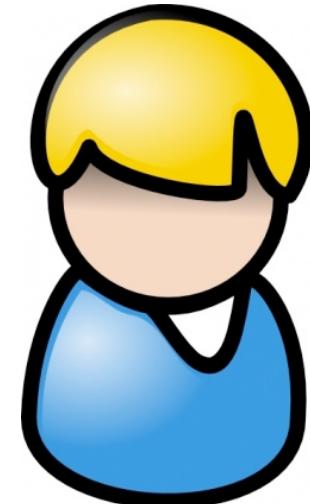
my_file.py



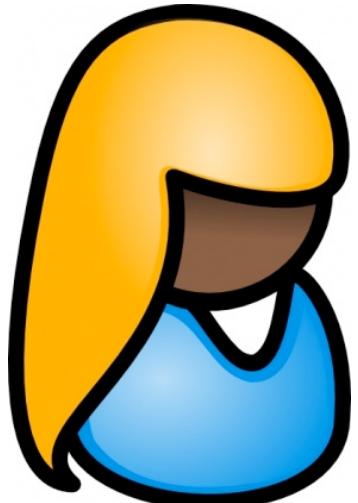
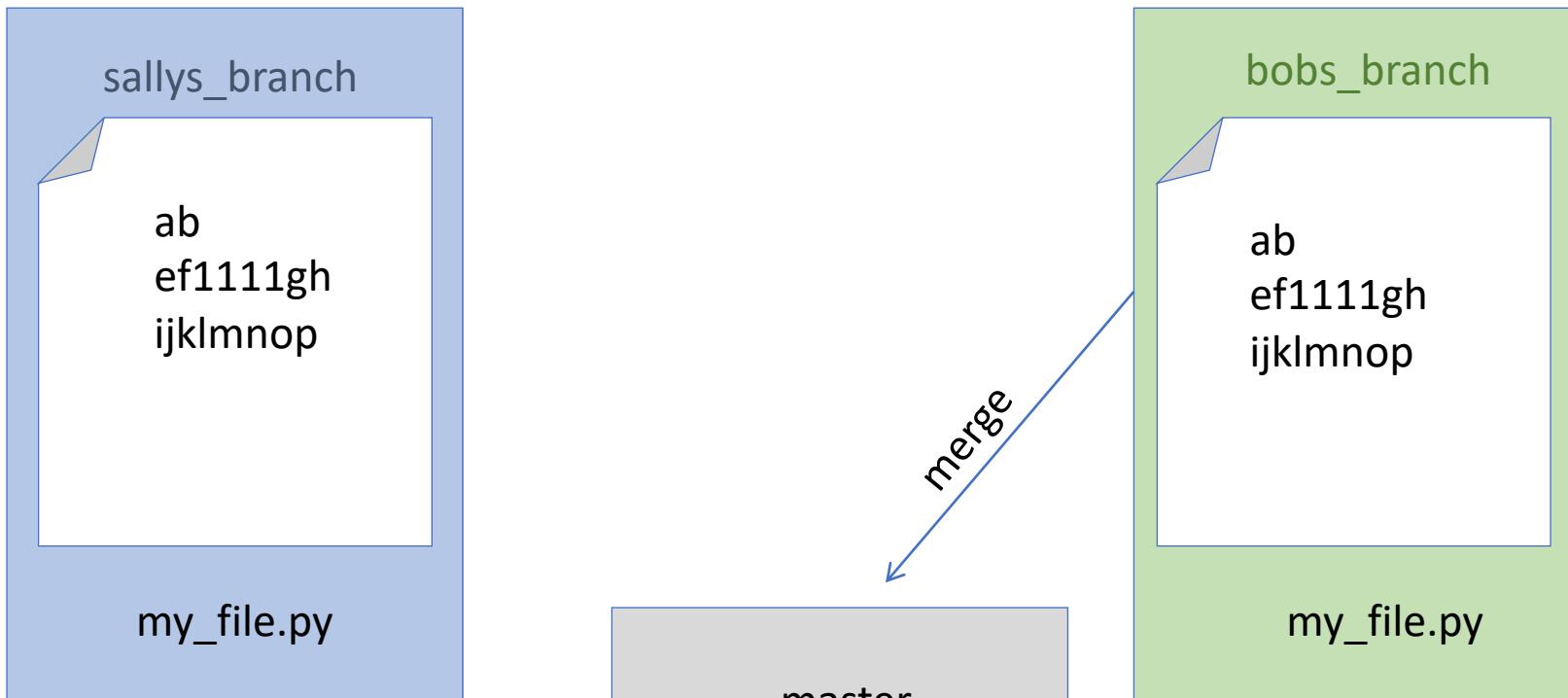
Hello, my name is Bob



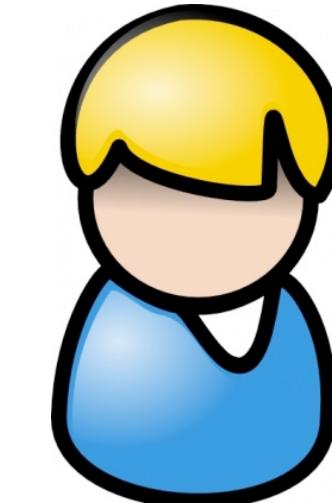
Hello, my name is Sally



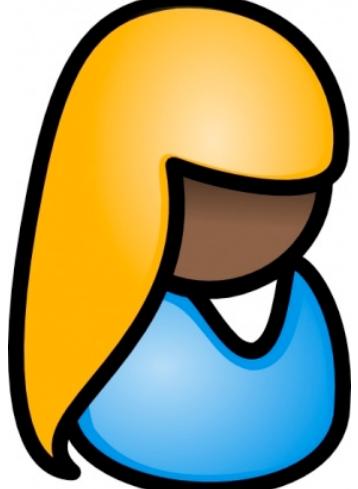
Hello, my name is Bob



Hello, my name is Sally



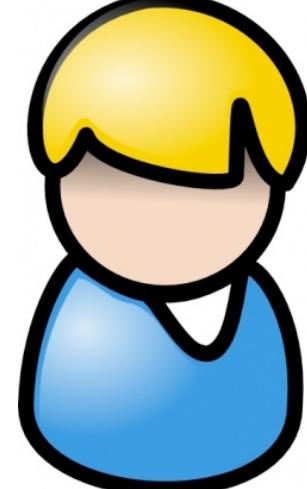
Hello, my name is Bob



sallys_branch

```
ab  
ef1111gh  
ijklmnop
```

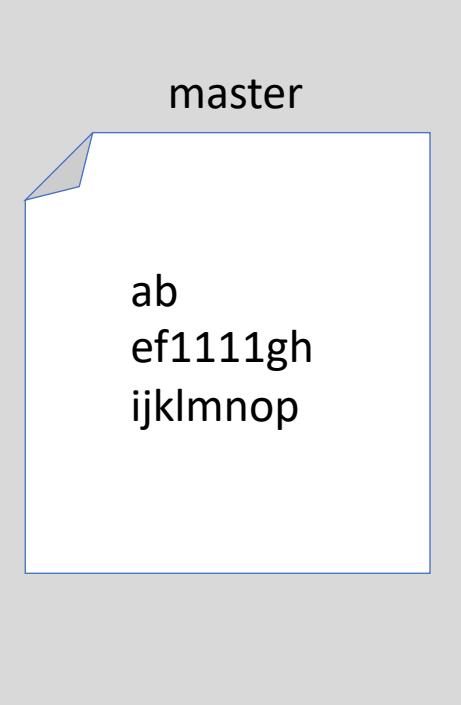
my_file.py



bobs_branch

```
ab  
ef1111gh  
ijklmnop
```

my_file.py

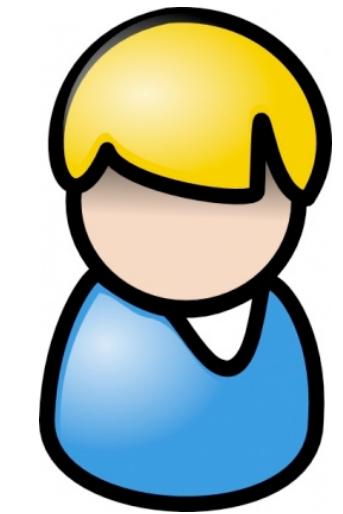
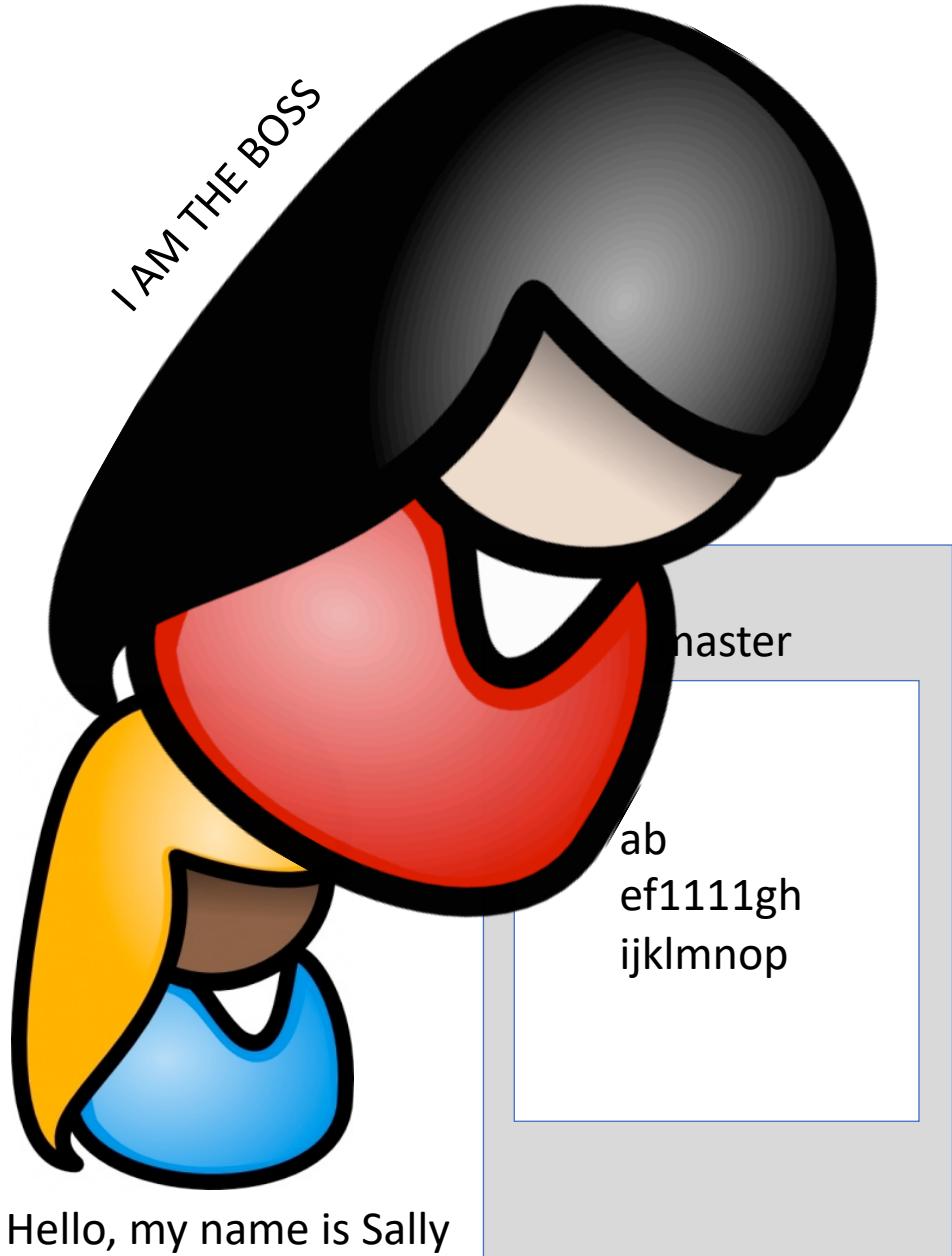


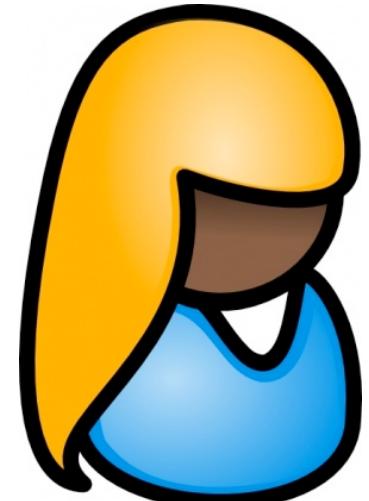
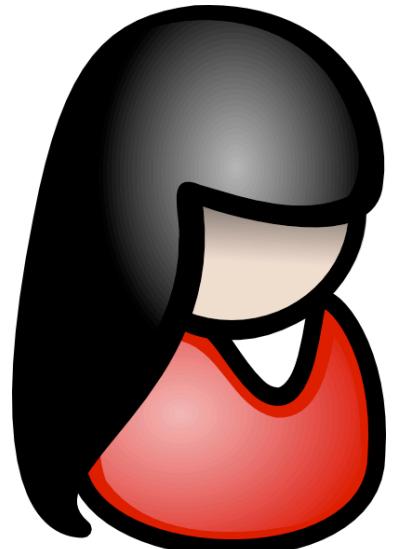
master

```
ab  
ef1111gh  
ijklmnop
```

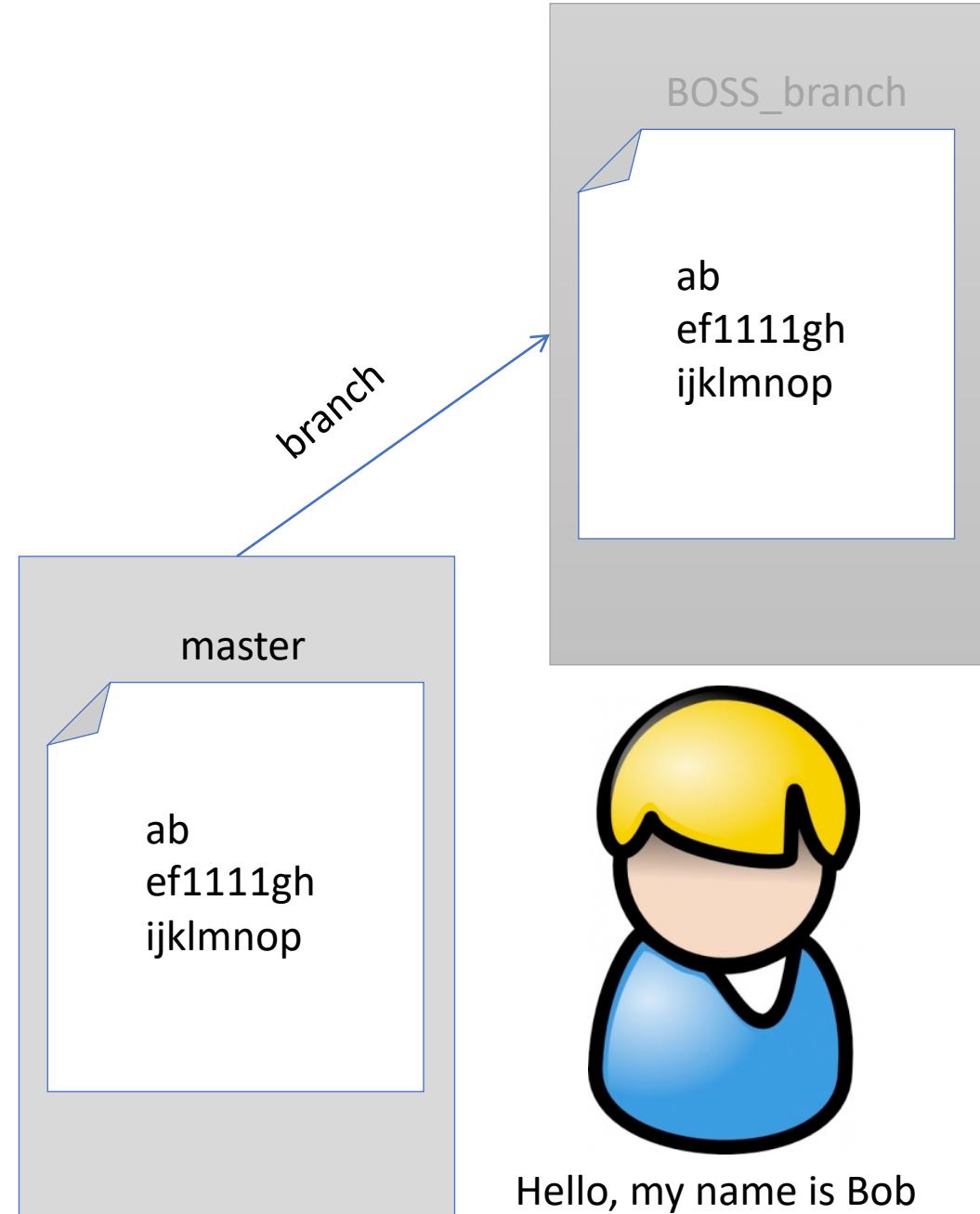
Hello, my name is Sally

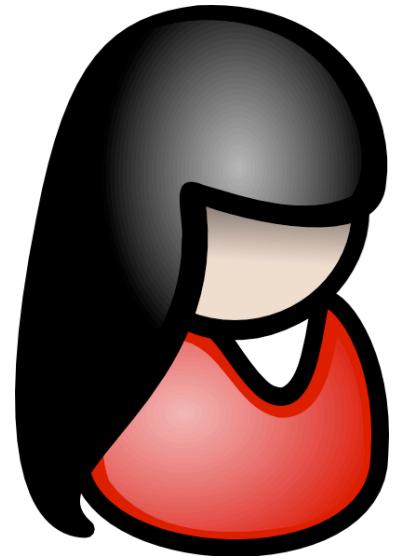
Hello, my name is Bob



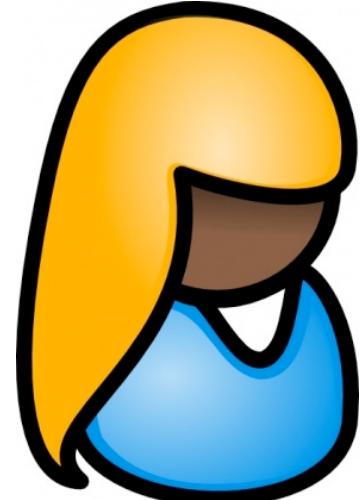


Hello, my name is Sally

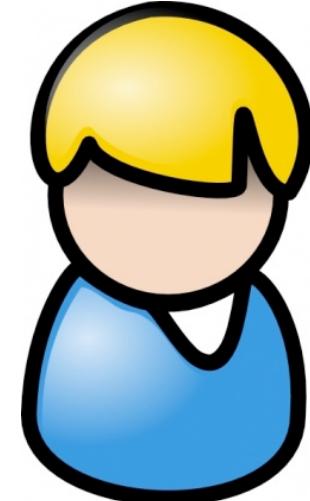
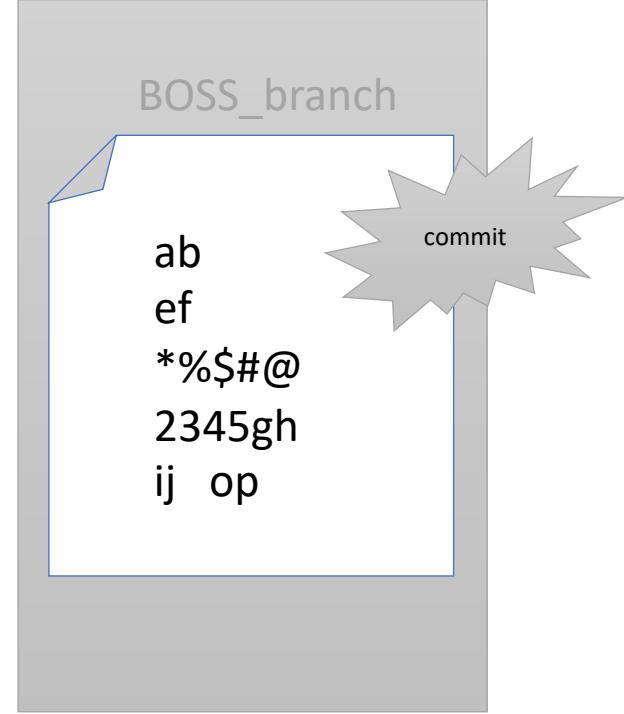




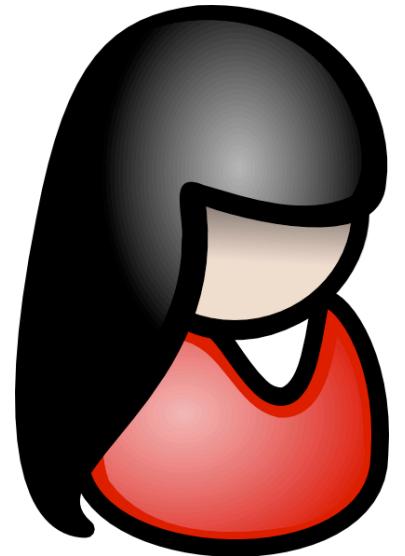
I AM THE BOSS



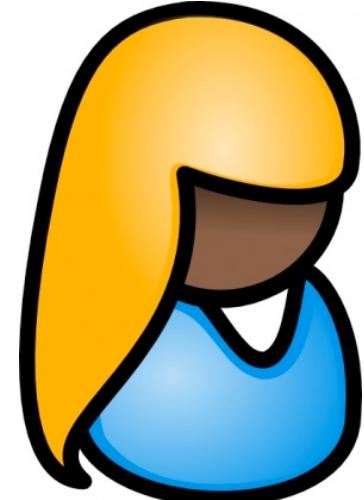
Hello, my name is Sally



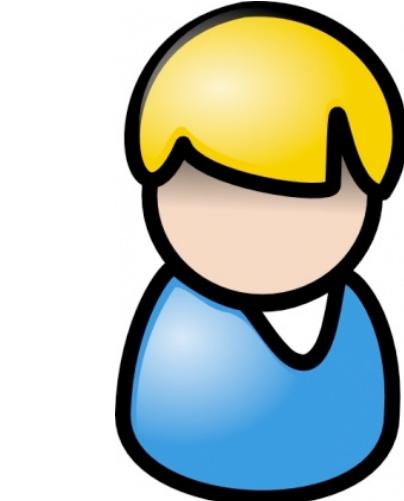
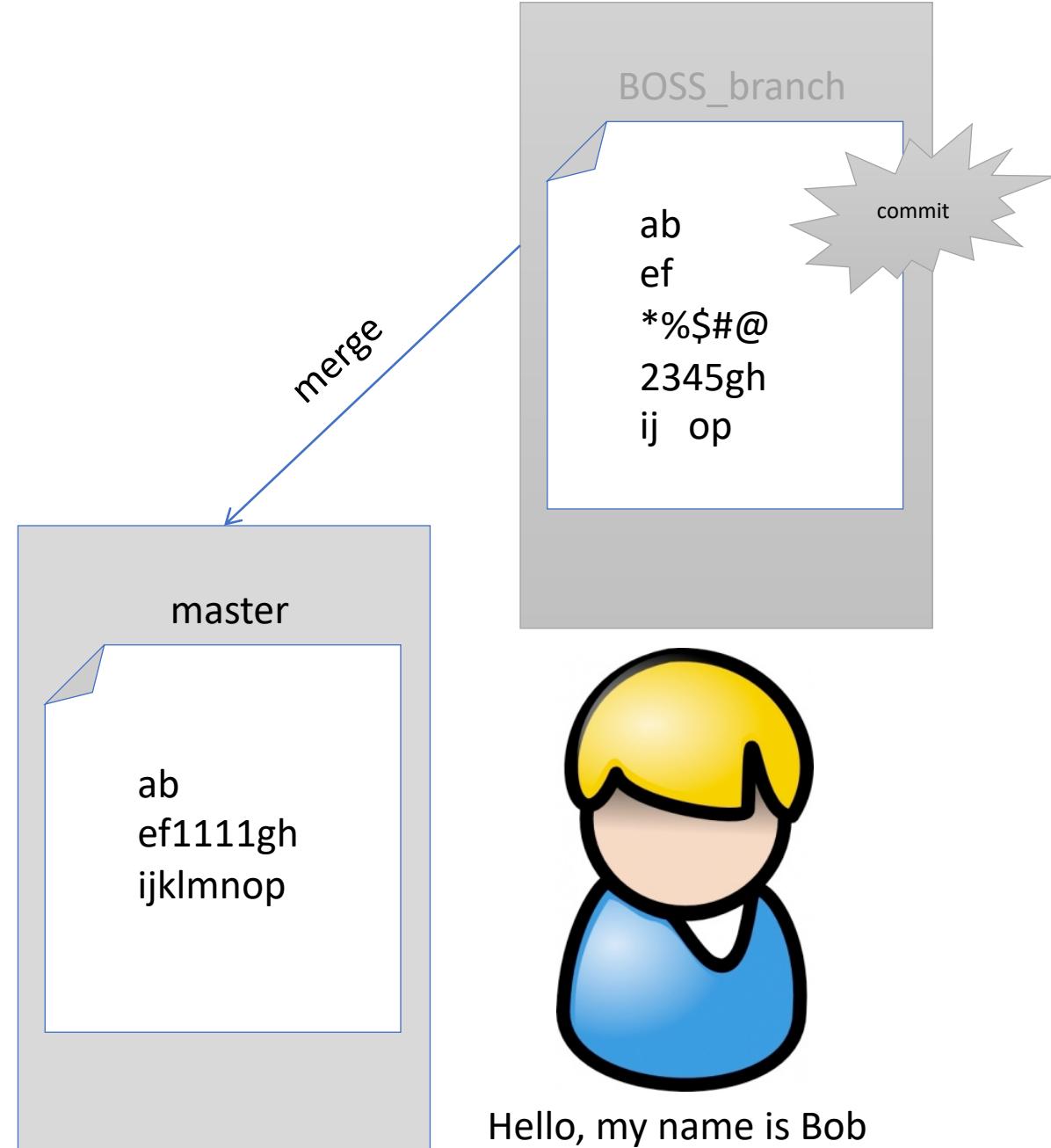
Hello, my name is Bob



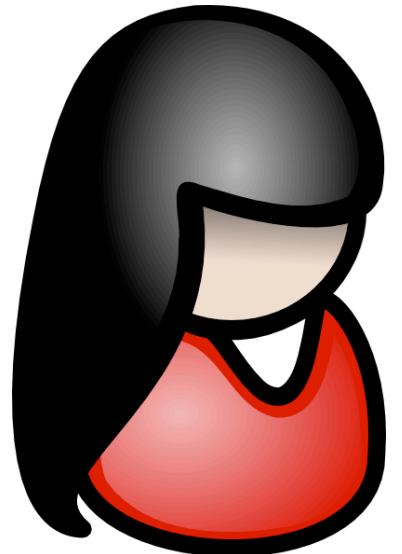
I AM THE BOSS



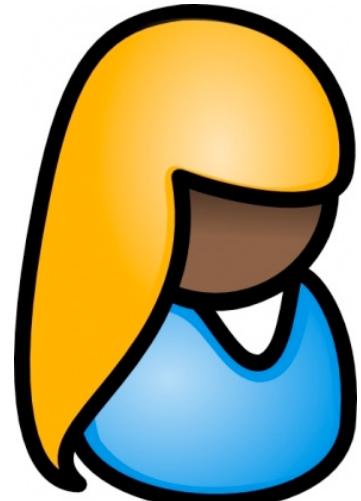
Hello, my name is Sally



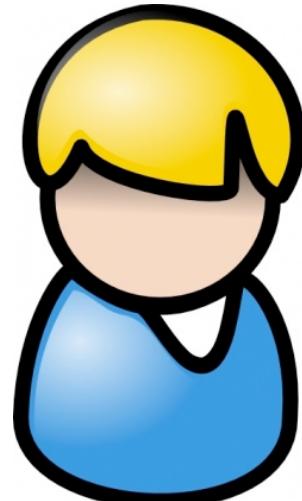
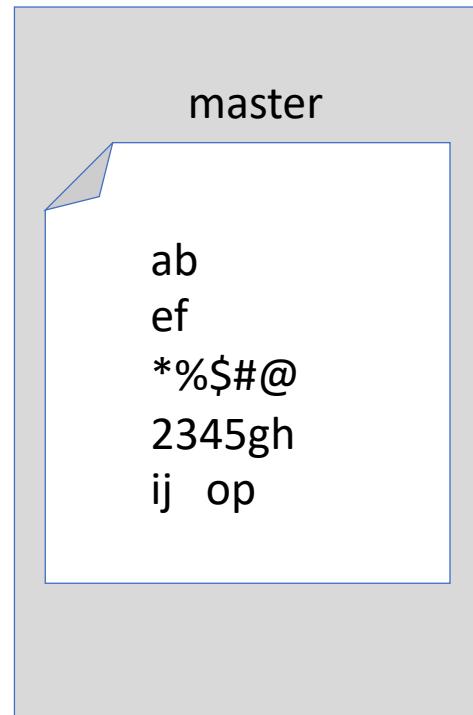
Hello, my name is Bob



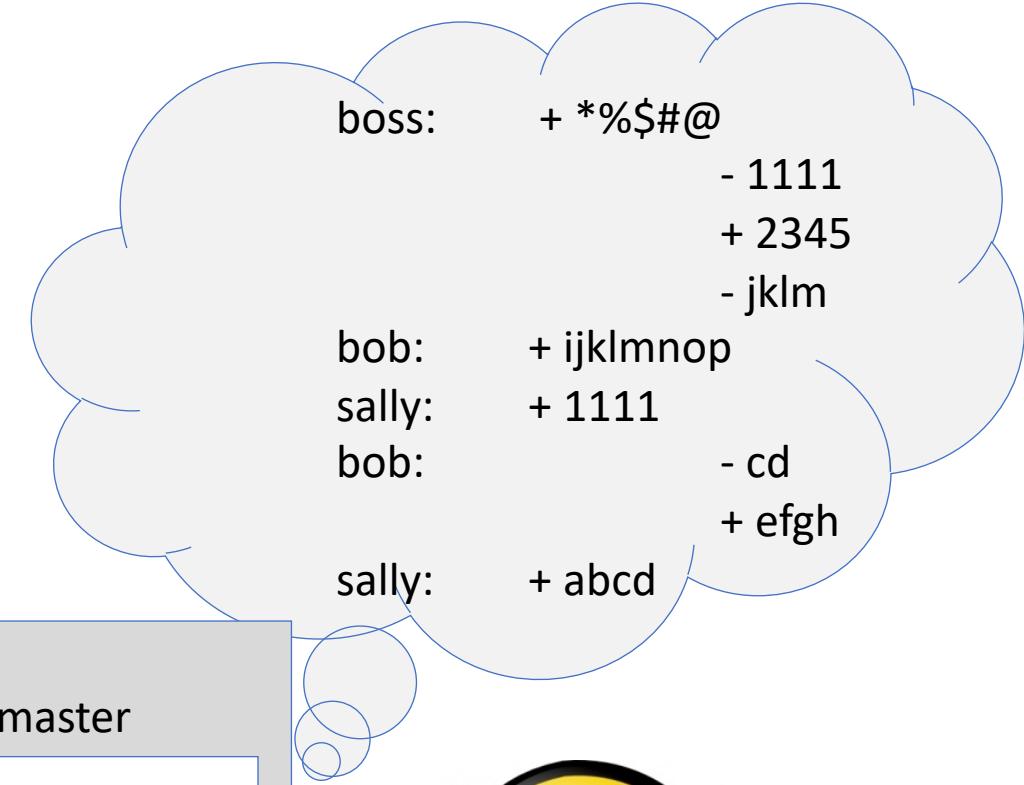
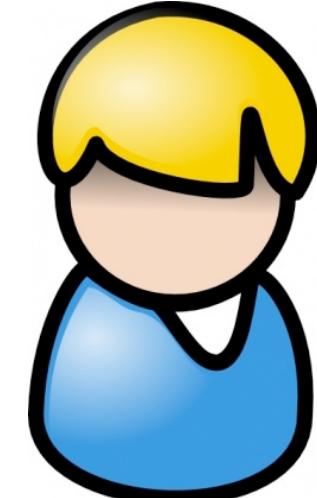
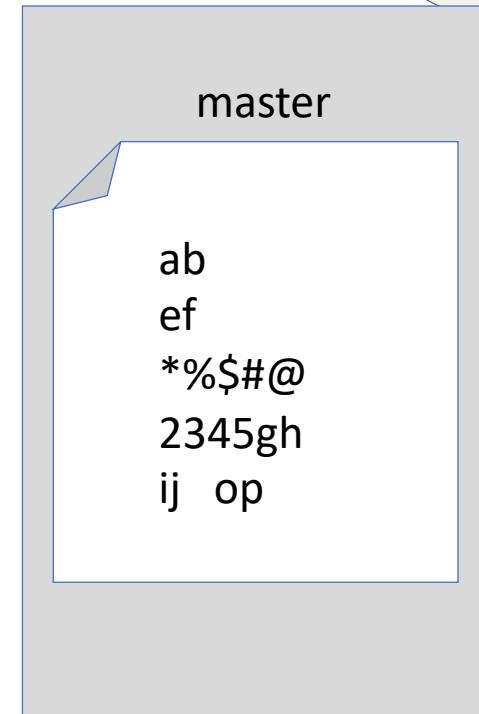
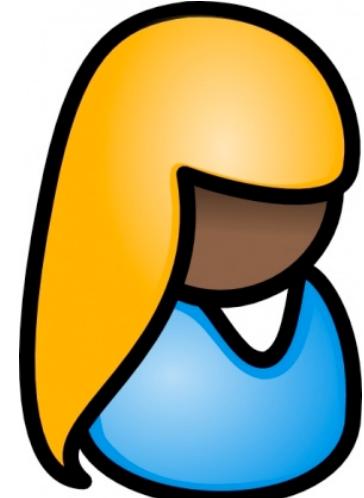
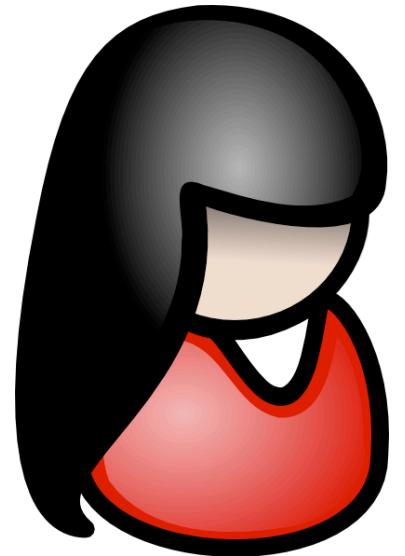
I AM THE BOSS



Hello, my name is Sally



Hello, my name is Bob



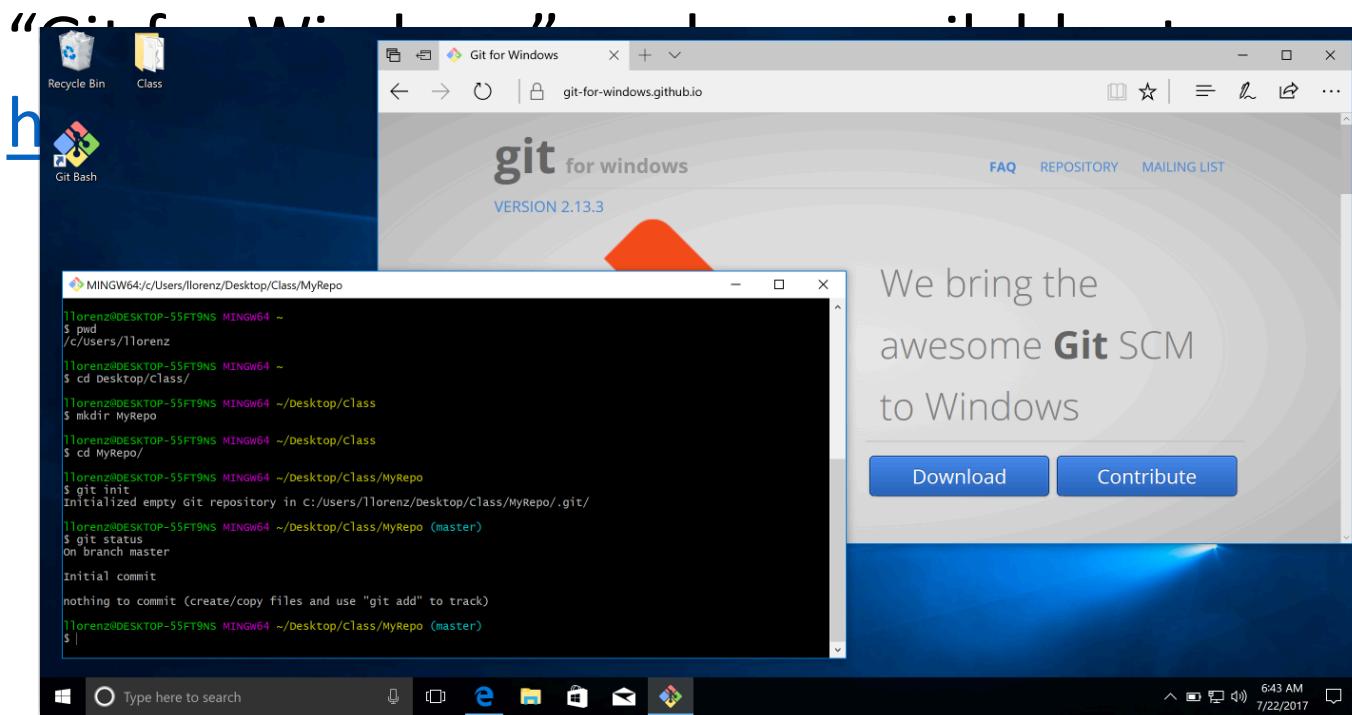
Using Git

Installation: Mac OS

- git is already preinstalled on your system, so you can use it from your normal command line application, Terminal, as depicted in the following slides.

Installation: Windows

- The easiest way to install a bash emulator that has git installed like Git BASH (part of the



There are also plugins for individual terminal applications (Powershell, Command Prompt) to configure using git from them directly. There is also a desktop application available called Github Desktop that will install git, but we will exclusively be learning how to use the command line, not the desktop app.

GIT

git init

or,

“let’s get started!”



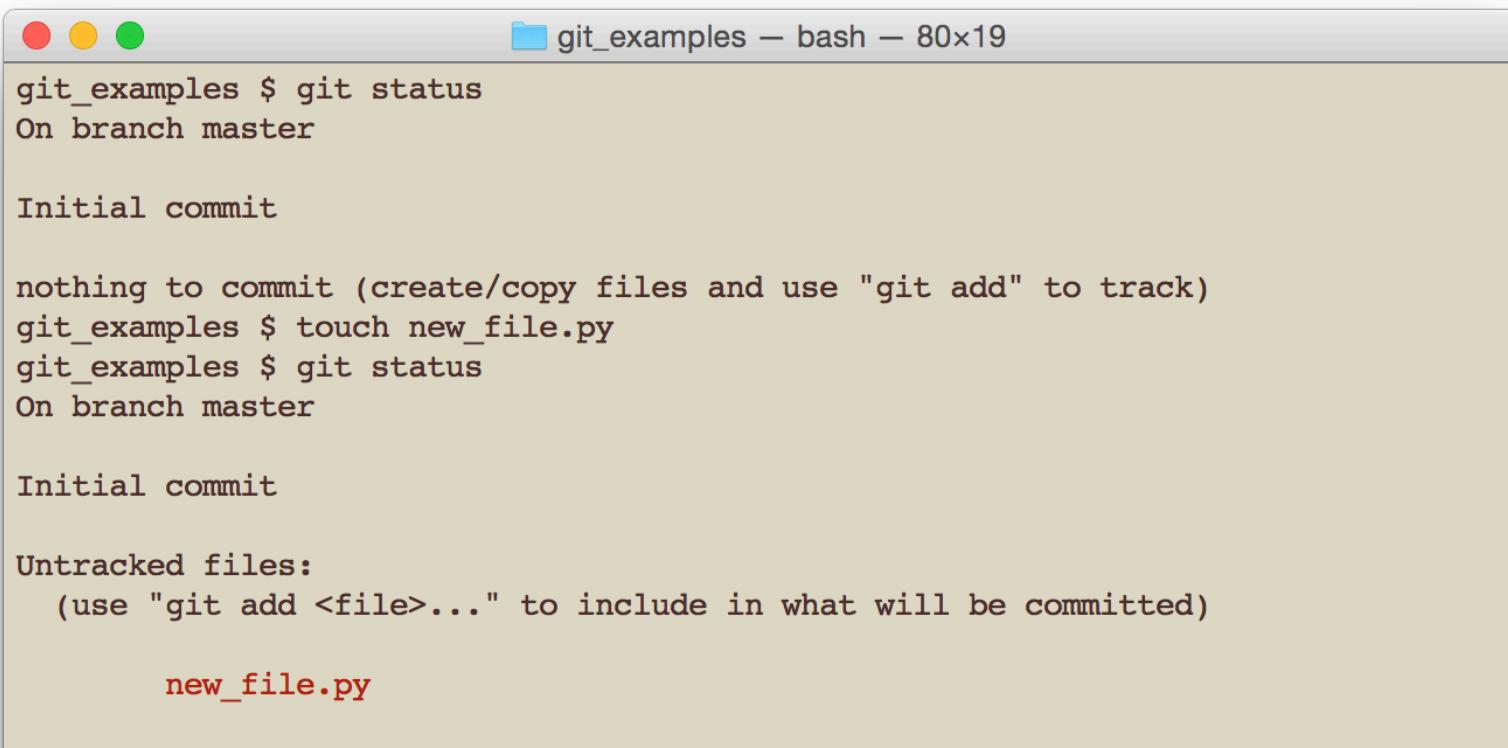
```
Fall 2016 $ mkdir git_examples
Fall 2016 $ cd git_examples/
git_examples $ git init .
Initialized empty Git repository in /Users/llorenz/Documents/Georgetown/Python Basics for Data Analysis/Fall 2016/git_examples/.git/
git_examples $ ls
git_examples $ ls -a
.  ..  .git
git_examples $ 
git_examples $ 
```

GIT

git status

or,

“what does git know about right now?”



```
git_examples $ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
git_examples $ touch new_file.py
git_examples $ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

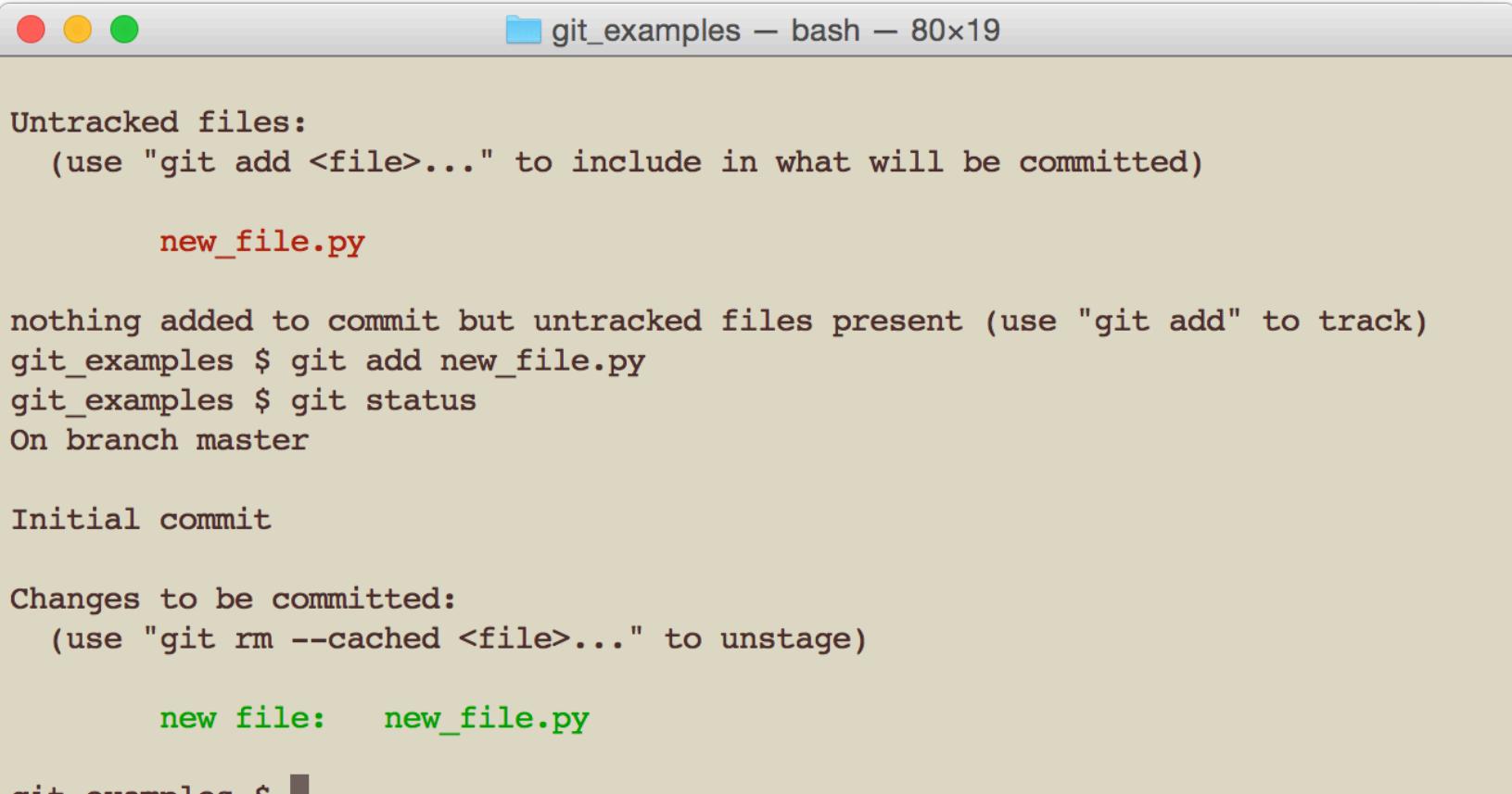
    new_file.py
```

GIT

git add

or,

“git, start tracking this file (or these new file changes)!!”



```
git_examples $ git add new_file.py
git_examples $ git status
On branch master
Initial commit

Changes to be committed:
(use "git rm --cached <file>..." to unstage)

    new file:   new_file.py

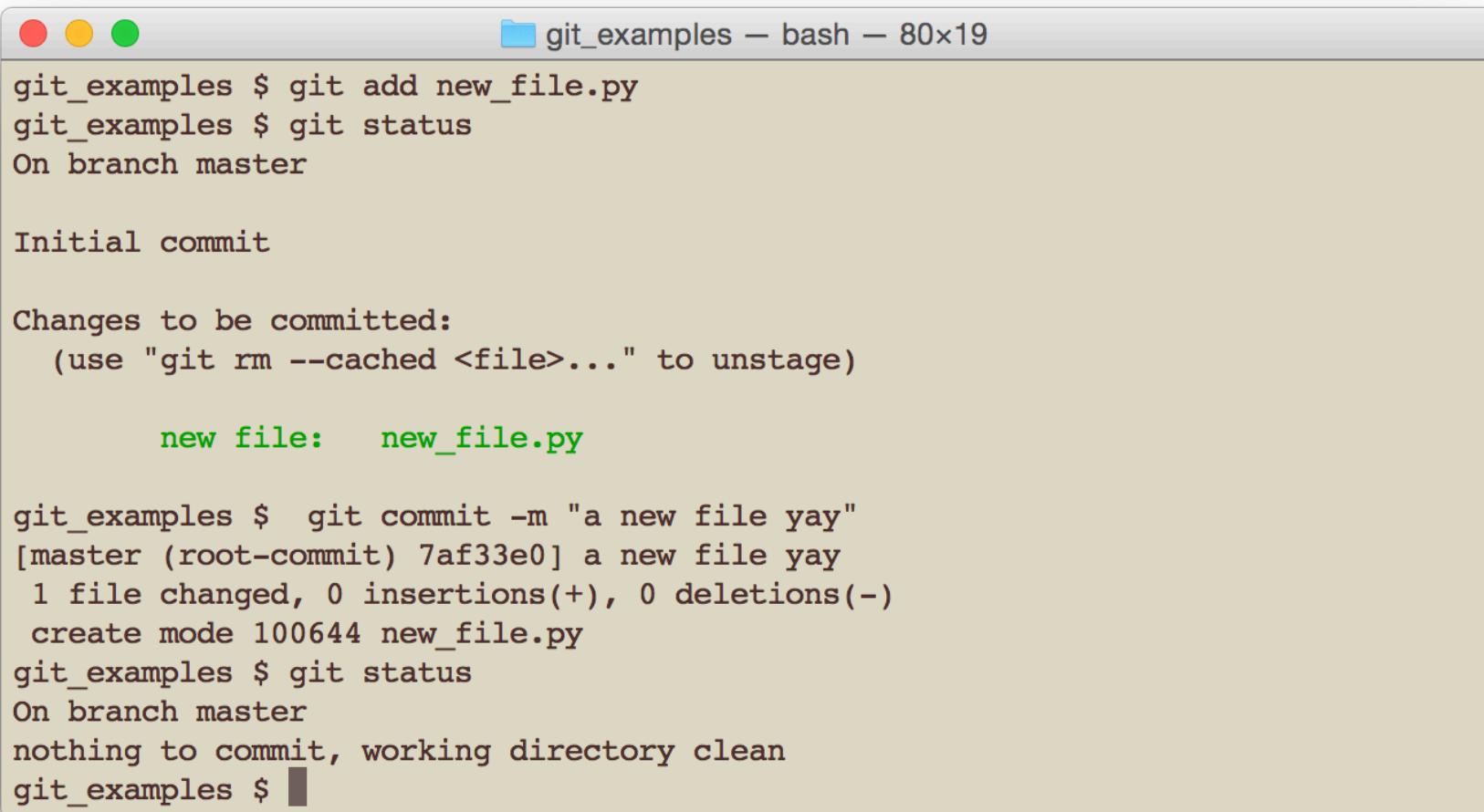
git_examples $
```

GIT

git commit

or,

“git, checkpoint everything I’ve added to the stage”



The screenshot shows a terminal window with the title bar "git_examples — bash — 80x19". The terminal content is as follows:

```
git_examples $ git add new_file.py
git_examples $ git status
On branch master

Initial commit

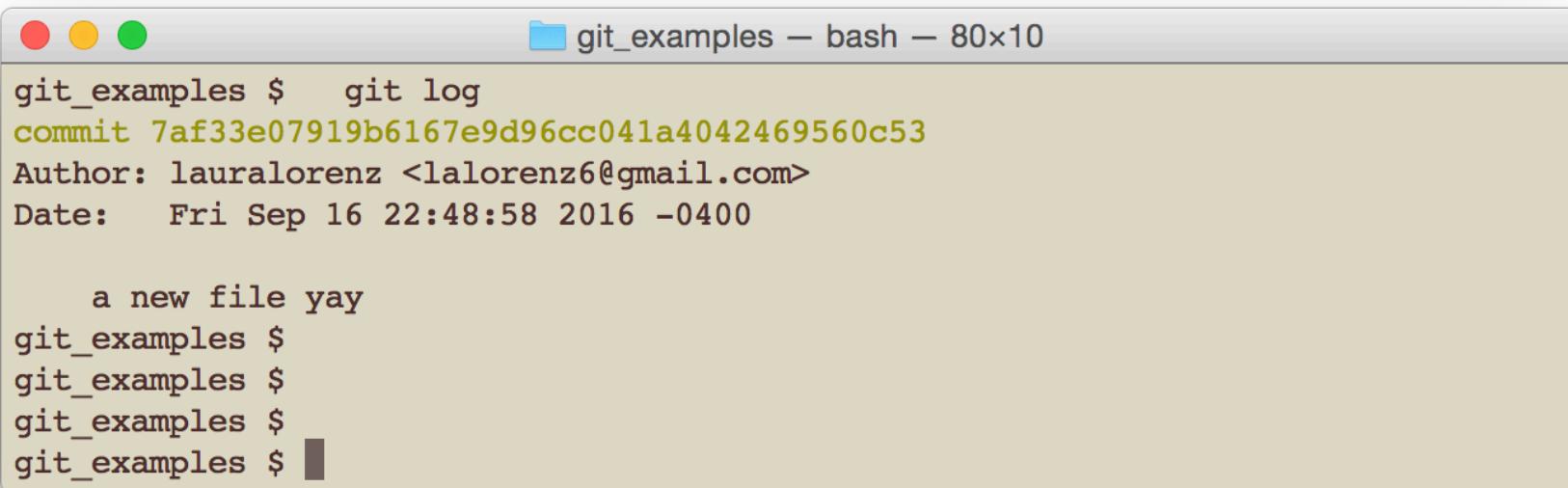
Changes to be committed:
(use "git rm --cached <file>..." to unstage)

    new file:   new_file.py

git_examples $ git commit -m "a new file yay"
[master (root-commit) 7af33e0] a new file yay
 1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 new_file.py
git_examples $ git status
On branch master
nothing to commit, working directory clean
git_examples $ █
```

GIT

git log
or,
“what did I do before again?”



The screenshot shows a macOS terminal window with a light gray background and a dark gray header bar. The title bar reads "git_examples — bash — 80x10". The window contains the following text:

```
git_examples $ git log
commit 7af33e07919b6167e9d96cc041a4042469560c53
Author: lauralorenz <lalorenz6@gmail.com>
Date:   Fri Sep 16 22:48:58 2016 -0400

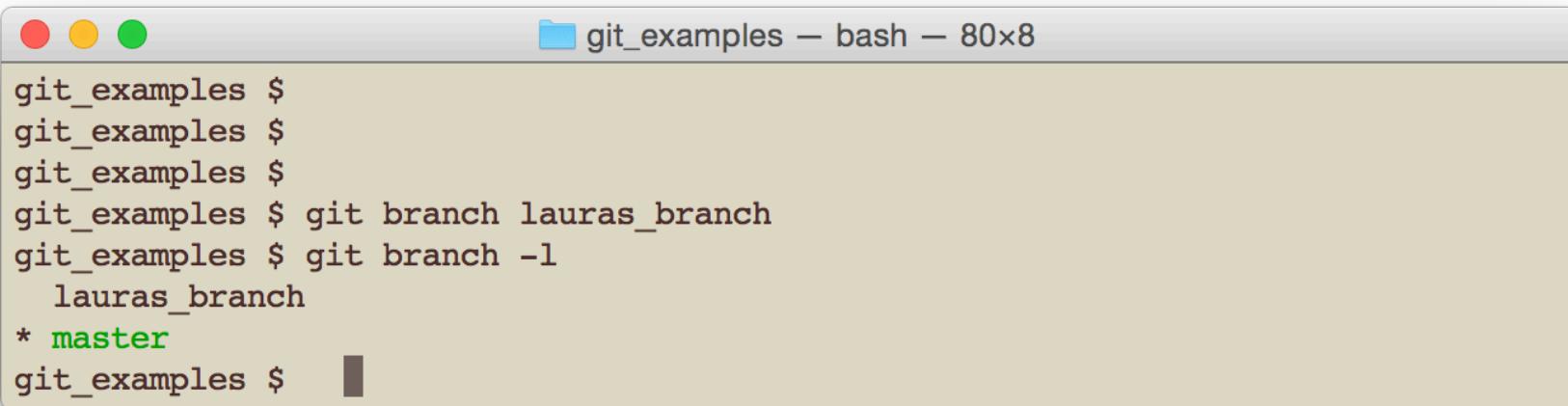
    a new file yay
git_examples $
```

GIT

git branch

or,

“start a new branch with space for me to work on stuff
separate from everyone else”



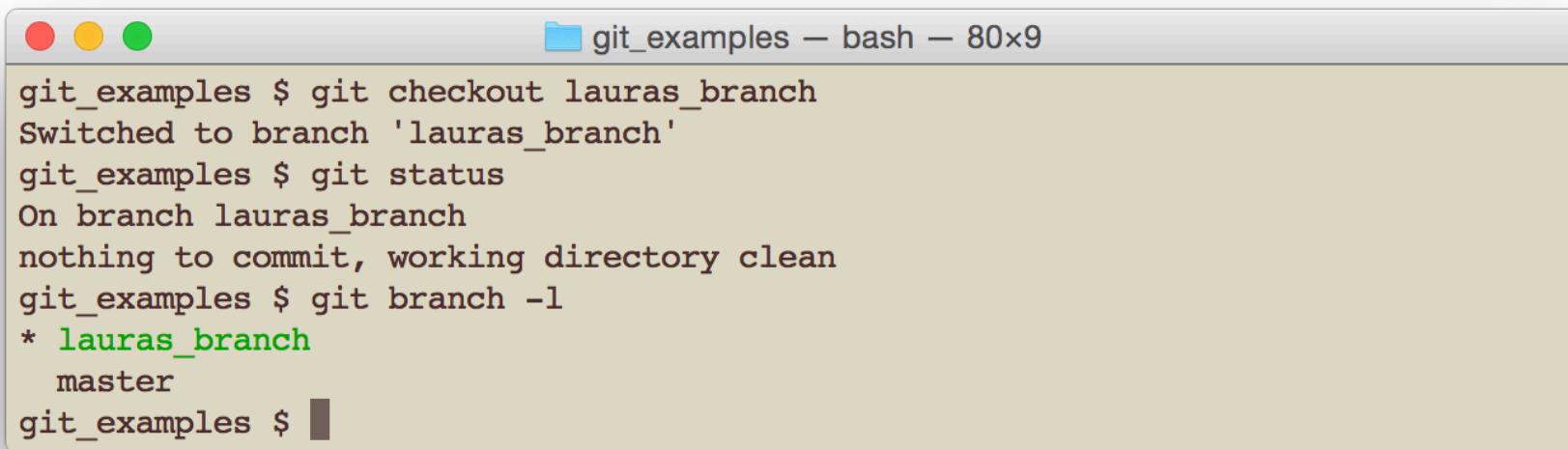
```
git_examples $ git branch lauras_branch
git_examples $ git branch -l
  lauras_branch
* master
git_examples $
```

GIT

git checkout

or,

“let me switch to this branch now”



```
git_examples $ git checkout lauras_branch
Switched to branch 'lauras_branch'
git_examples $ git status
On branch lauras_branch
nothing to commit, working directory clean
git_examples $ git branch -l
* lauras_branch
  master
git_examples $
```

GIT

git merge

or,

“pull in all the commits
from this other branch
to the branch I am on”

```
git_examples $ touch hello_there.py
git_examples $ git status
On branch lauras_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    hello_there.py

nothing added to commit but untracked files present (use "git add" to track)
git_examples $ git add hello_there.py
git_examples $ git commit -m "adding hello"
[lauras_branch 78feldb] adding hello
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 hello_there.py
git_examples $ git checkout master
Switched to branch 'master'
git_examples $ git log
commit 7af33e07919b6167e9d96cc041a4042469560c53
Author: lauralorenz <lalorenz6@gmail.com>
Date:   Fri Sep 16 22:48:58 2016 -0400

  a new file yay
git_examples $ git merge lauras_branch
Updating 7af33e0..78feldb
Fast-forward
  hello_there.py | 0
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 hello_there.py
git_examples $ git log
commit 78feldb884e882c4bd6d32f91f718339dff5a8a1
Author: lauralorenz <lalorenz6@gmail.com>
Date:   Fri Sep 16 22:51:32 2016 -0400

  adding hello

commit 7af33e07919b6167e9d96cc041a4042469560c53
Author: lauralorenz <lalorenz6@gmail.com>
Date:   Fri Sep 16 22:48:58 2016 -0400

  a new file yay
git_examples $
```

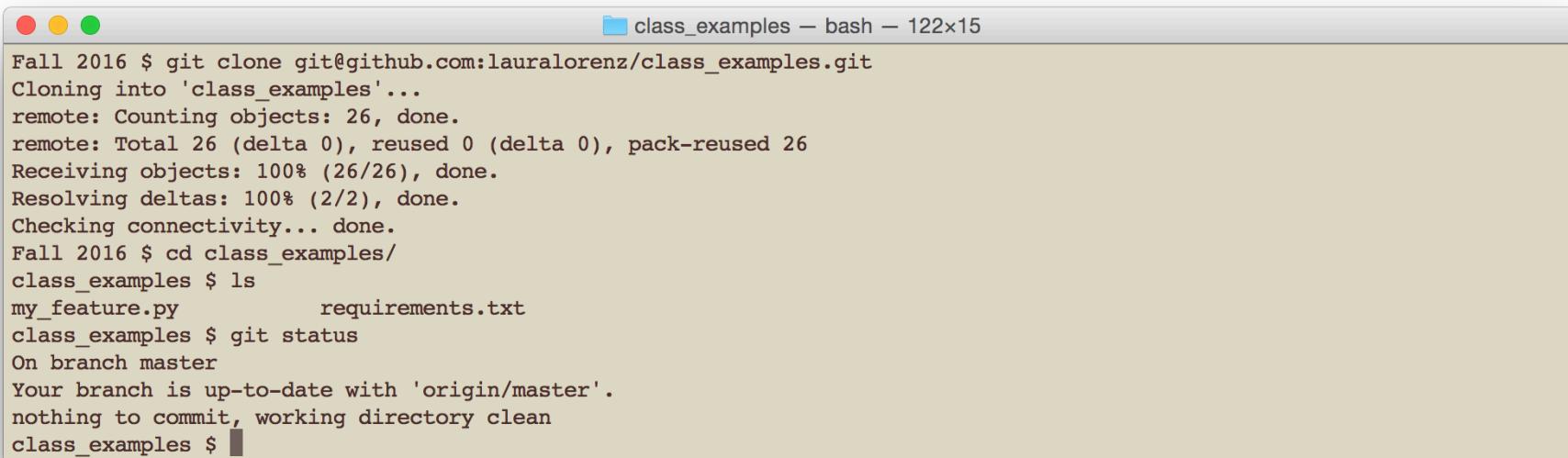
Sharing your code to Github

GITHUB!

git clone

or,

“copy this git repository on the Internet to my local disk”



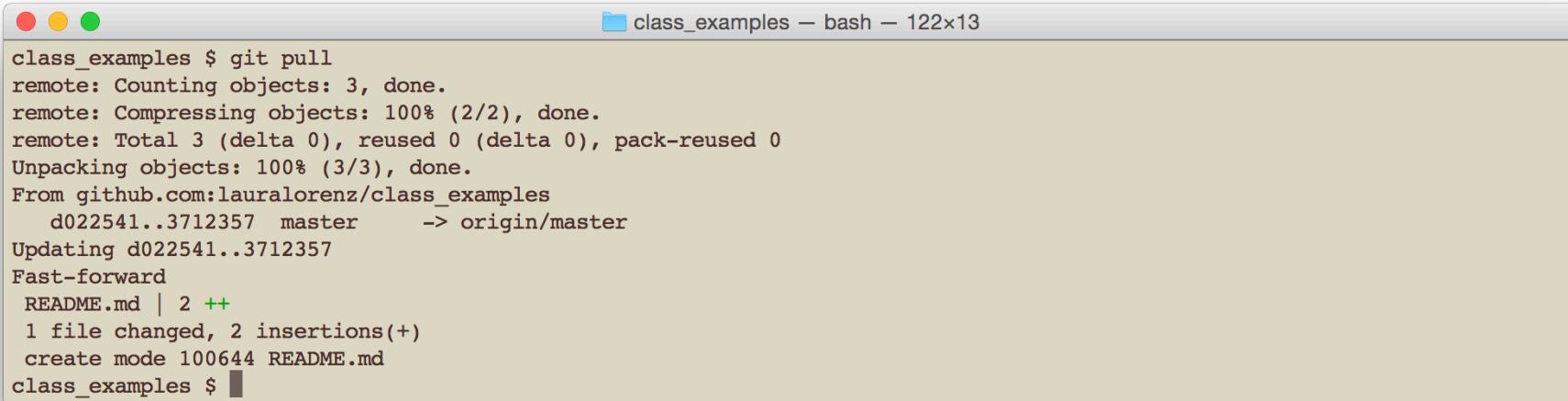
```
Fall 2016 $ git clone git@github.com:lauralorenz/class_examples.git
Cloning into 'class_examples'...
remote: Counting objects: 26, done.
remote: Total 26 (delta 0), reused 0 (delta 0), pack-reused 26
Receiving objects: 100% (26/26), done.
Resolving deltas: 100% (2/2), done.
Checking connectivity... done.
Fall 2016 $ cd class_examples/
class_examples $ ls
my_feature.py      requirements.txt
class_examples $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
class_examples $
```

GITHUB!

git pull

or,

“update the local repository with any changes on the Internet for it”



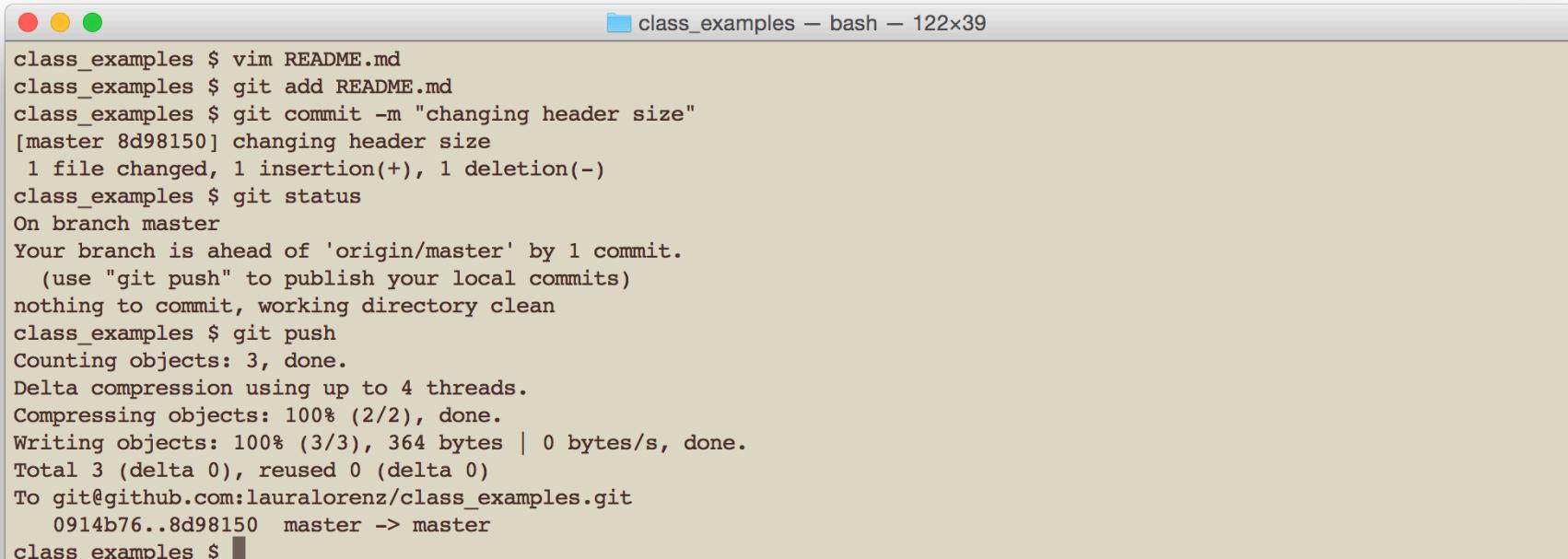
```
class_examples $ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:lauralorenz/class_examples
  d022541..3712357  master      -> origin/master
Updating d022541..3712357
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
class_examples $
```

GITHUB!

git push

or,

“update the remote repository with any changes I made”

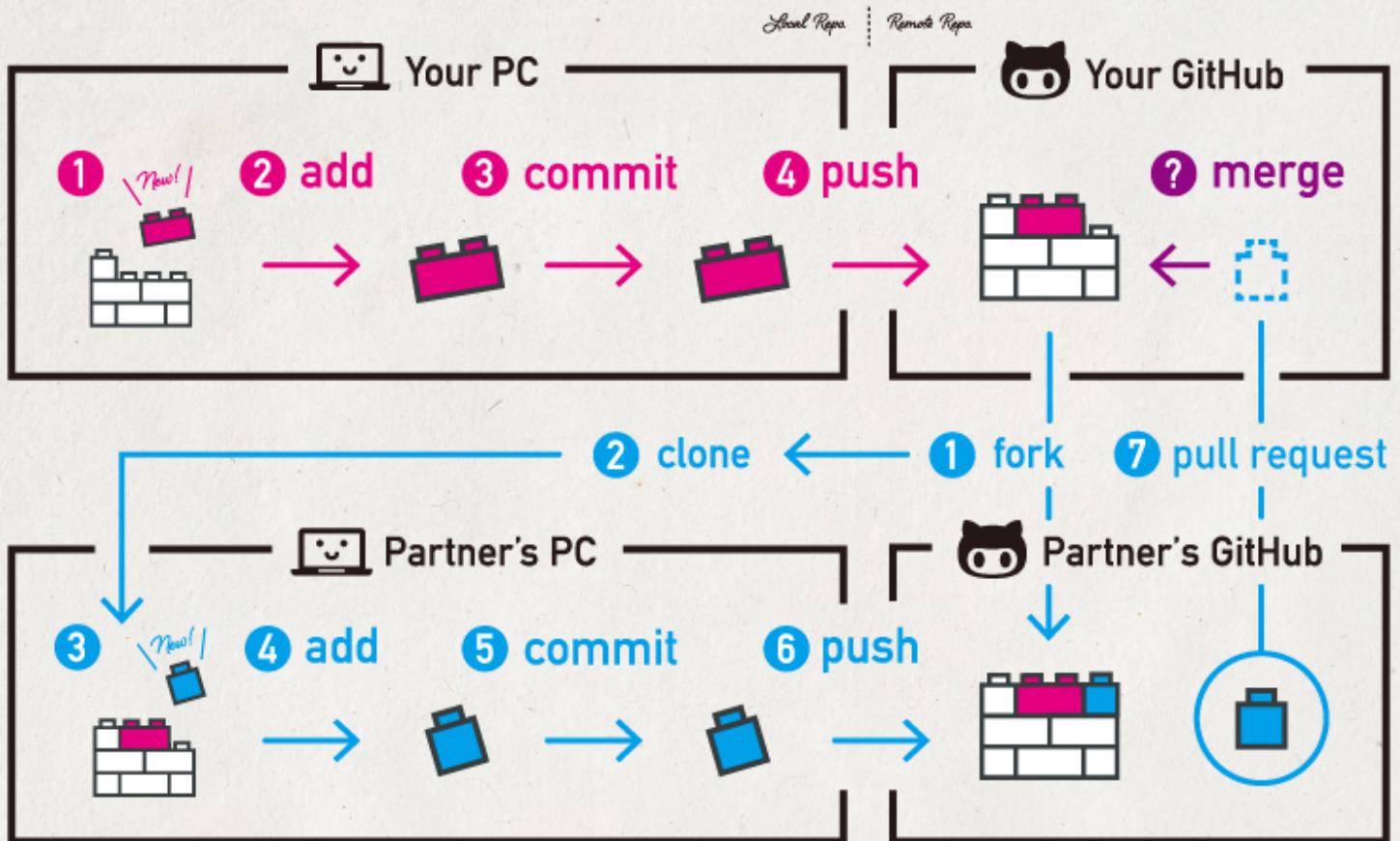


A screenshot of a terminal window titled "class_examples — bash — 122x39". The window shows a sequence of git commands and their output:

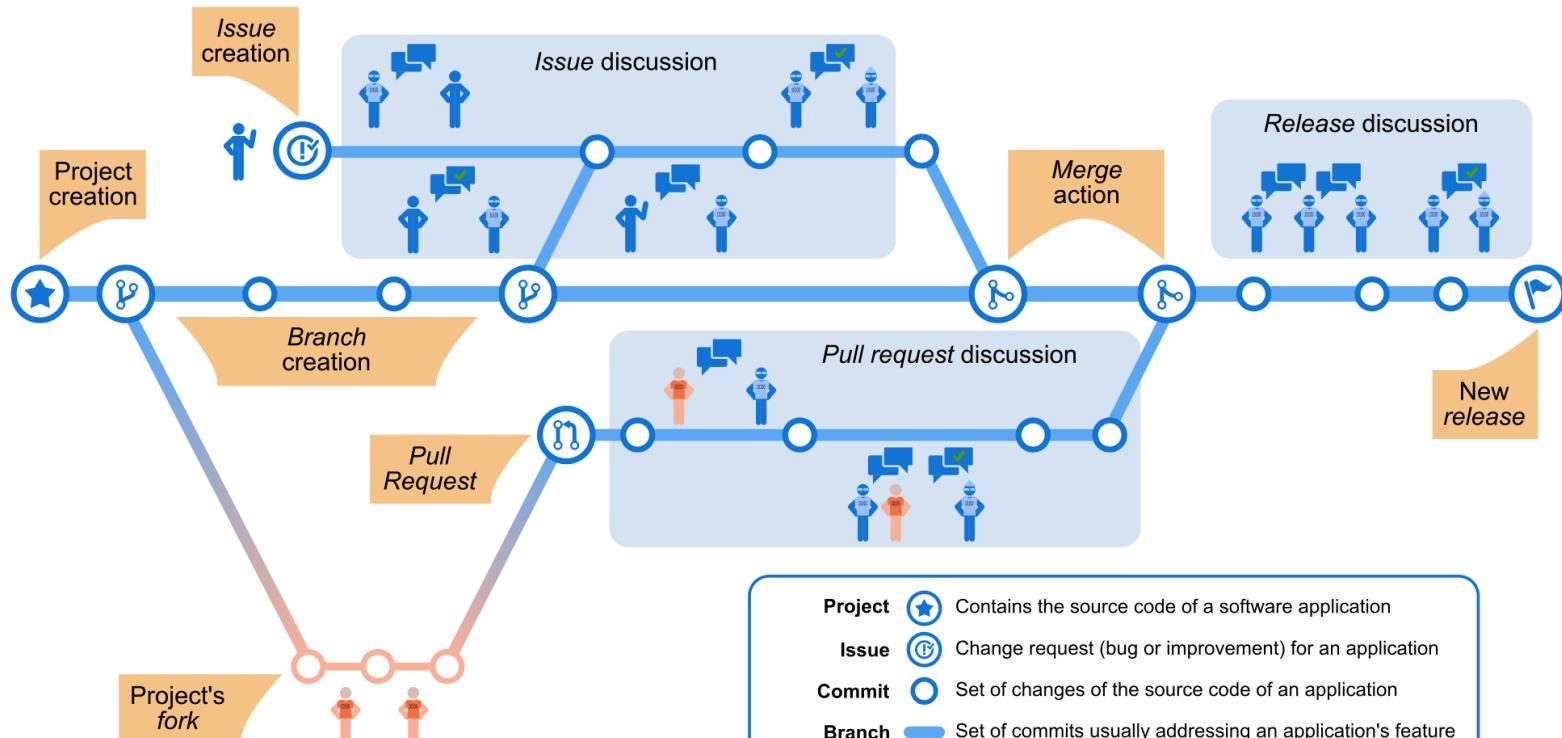
```
class_examples $ vim README.md
class_examples $ git add README.md
class_examples $ git commit -m "changing header size"
[master 8d98150] changing header size
 1 file changed, 1 insertion(+), 1 deletion(-)
class_examples $ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
class_examples $ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 364 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:lauralorenz/class_examples.git
  0914b76..8d98150  master -> master
class_examples $ █
```

Git / GitHub

How to "Pull Request"



How GitHub projects are developed? Where are the main discussion points?



- | | |
|---------------------|--|
| Project | Contains the source code of a software application |
| Issue | Change request (bug or improvement) for an application |
| Commit | Set of changes of the source code of an application |
| Branch | Set of commits usually addressing an application's feature |
| Fork | Creation of a new branch |
| Pull Request | Request to join two branches |
| Merge | Action of joining two branches |
| Release | Publication of a new version of an application |



bcnparticipa.cat/decidimjam



som-research.uoc.edu