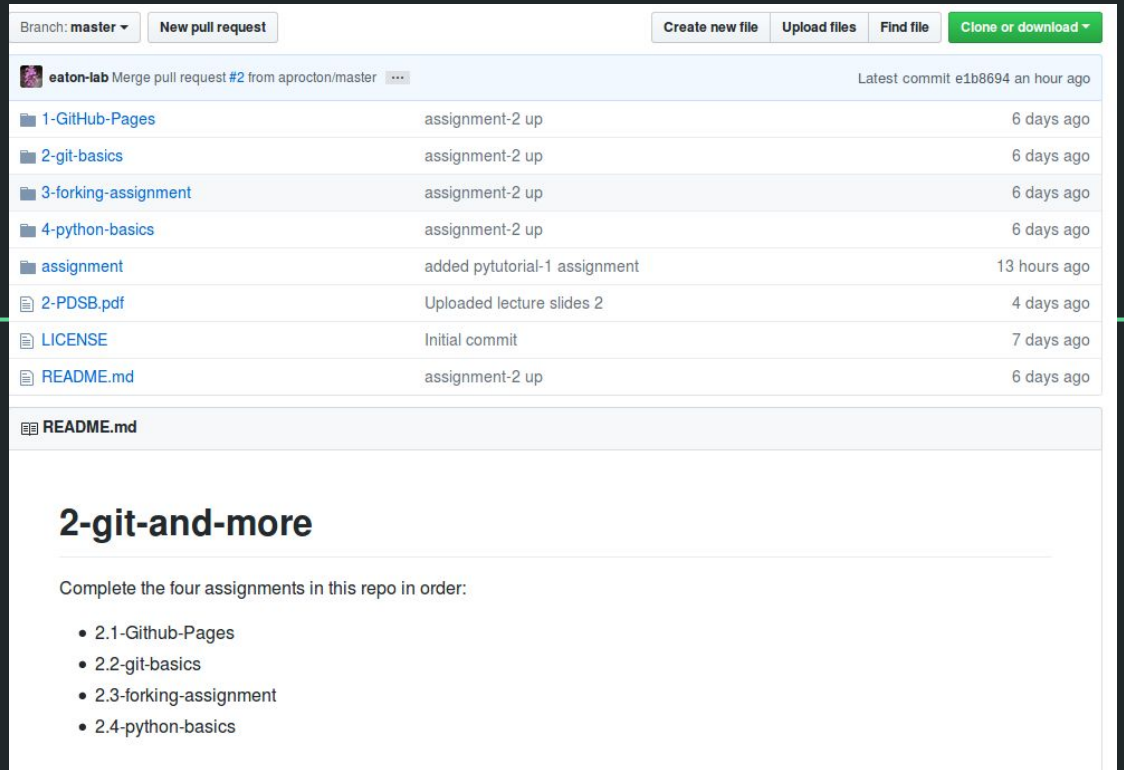
Programming and Data Science for Biology (PDSB)

Session 3
Spring 2018

You should have completed assignment 2

Link to session 2 repo



Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

eaton-lab Merge pull request #2 from aprocton/master ··· Latest commit e1b8694 an hour ago

1-GitHub-Pages	assignment-2 up	6 days ago
2-git-basics	assignment-2 up	6 days ago
3-forking-assignment	assignment-2 up	6 days ago
4-python-basics	assignment-2 up	6 days ago
assignment	added pytutorial-1 assignment	13 hours ago
2-PDSB.pdf	Uploaded lecture slides 2	4 days ago
LICENSE	Initial commit	7 days ago
README.md	assignment-2 up	6 days ago

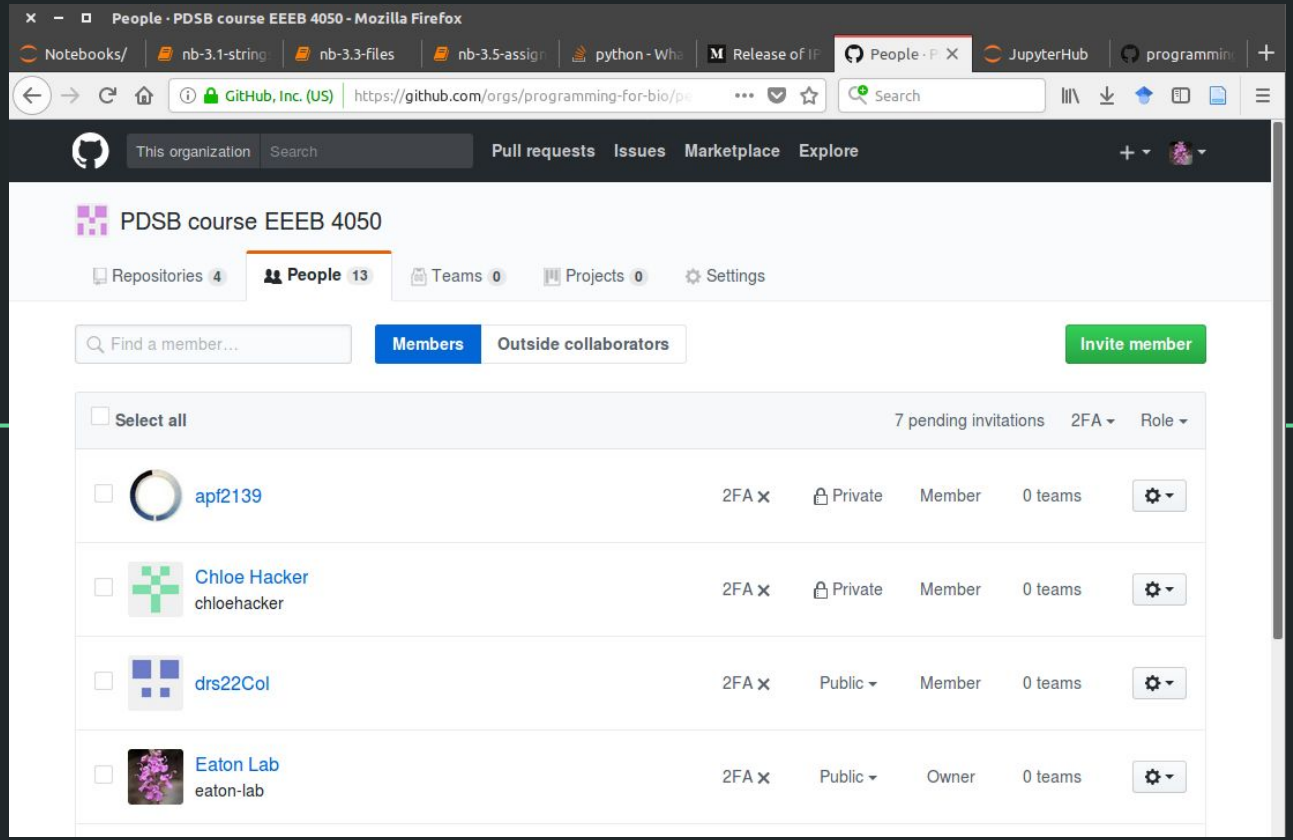
README.md

2-git-and-more

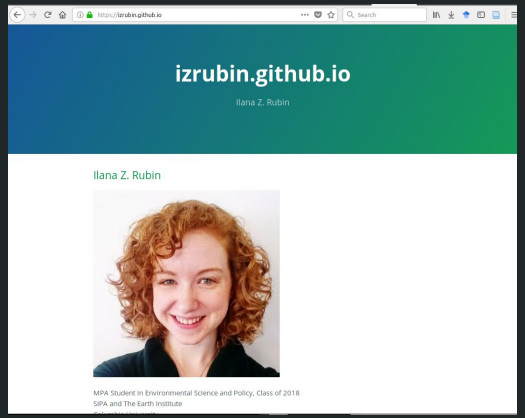
Complete the four assignments in this repo in order:

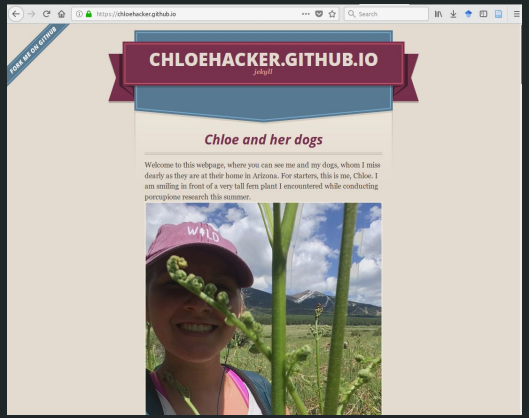
- 2.1-Github-Pages
- 2.2-git-basics
- 2.3-forking-assignment
- 2.4-python-basics

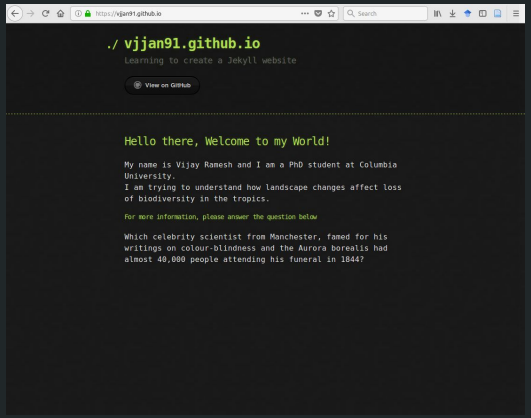
Class Organization “People” tab

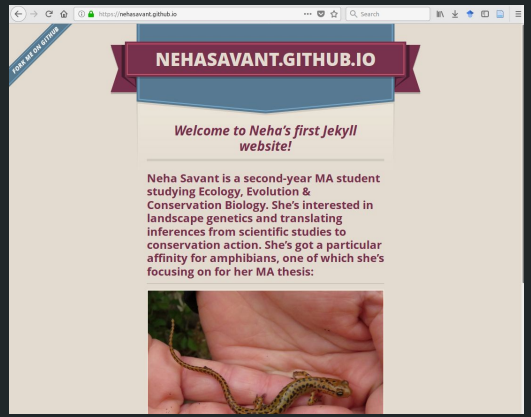


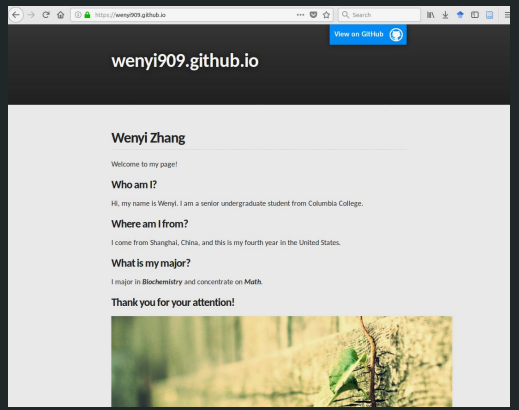
GitHub pages: static websites

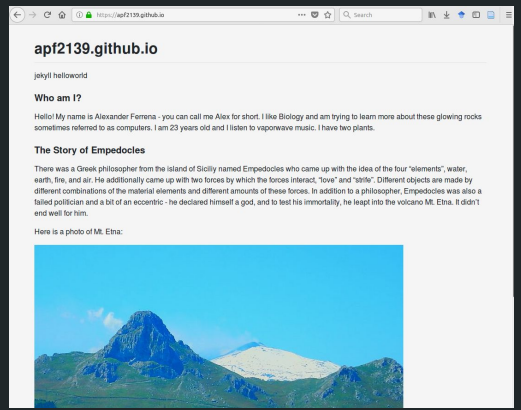












The git assignment:

Why are we learning to use *git* again?
To learn good coding practices from the start.

I didn't really get it, can I just forget it and move on?
No, we're going to be using git for every assignment going forward. And the same is true for most concepts in this class.
If you fall behind, ask for help!

Office hours:

Professor Eaton: Fridays 10-12am (Scherm. ext. 1007)
Patrick McKenzie: Thursdays 12-2pm (Scherm. ext. student office)
Ask questions on gitter http://gitter.im/programming-for-bio

Python basics: variables, expressions, operations, and data objects

Now that we are all using git:

Go to the GitHub repo for today's class and fork the repo
`https://github.com/programming-for-bio/3-Python-basics`

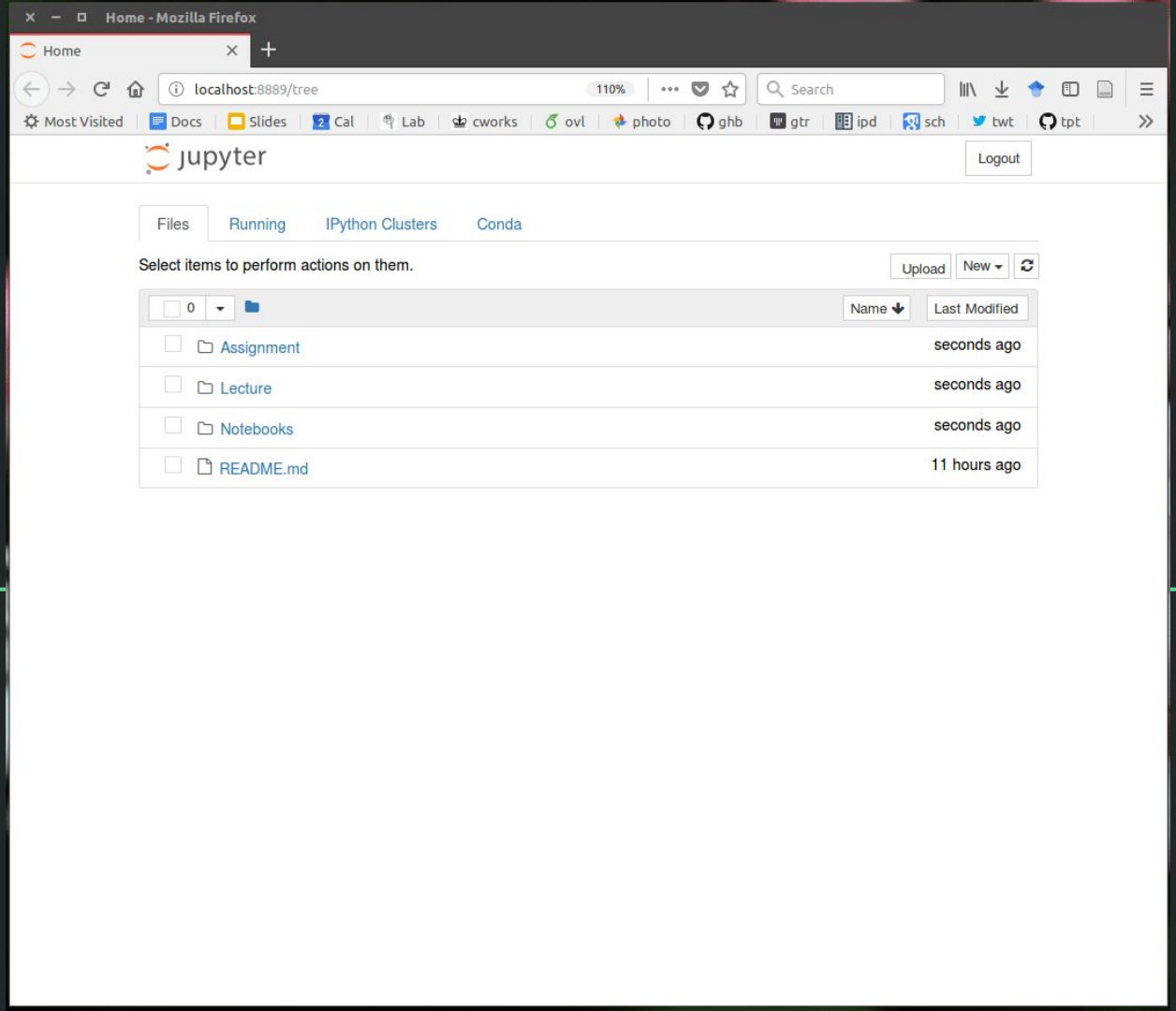
Then clone your copy onto your laptop so we can each access the documents in the repo.

```
> git clone https://github.com/<username>/3-Python-basics.git
```

Today’s repo

```
Lecture/
 +   PDSB-3-Lecture.pdf

Notebooks/
 +   nb-3.0-arithmetic.ipynb
 +   nb-3.1-strings.ipynb
 +   nb-3.2-lists.ipynb
 +   nb-3.3-files.ipynb
 +   nb-3.4-functions.ipynb
 +   nb-3.5-code-review.ipynb

Assignment/
```

```
## You should have installed ‘notebooks’.
## Also install the ‘requests’ library
conda install notebook
conda install requests

## cd into the repo
cd ./3-Python-basics/

## start a notebook server
jupyter-notebook
```


Home - Mozilla Firefox
Home
localhost:8889/tree
110%
Search
Most Visited
Docs
Slides
Cal
Lab
cworks
ovl
photo
ghb
gtr
ipd
sch
twt
tpt
jupyter
Logout
Files
Running
IPython Clusters
Conda
Select items to perform actions on them.
Upload
New
0
Name
Last Modified
Assignment
seconds ago
Lecture
seconds ago
Notebooks
seconds ago
README.md
11 hours ago

Python numeric types

Boolean, integers, and floats are the basic types used for arithmetic operations and comparisons.

```
## integer types
x = 3
x = int(3)

## float types
x = 3.0
x = float(3.0)

## boolean
x = True
True == 1 == bool(1)
```

Python operations

Arithmetic operators in Python.

```
+    = addition
-    = subtraction
*    = multiplication
/    = division
//   = integer division
%    = modulus
**   = exponent
```

```
## addition
x = 3 + 3      ## 6

## subtraction
x = 3 - 3      ## 0

## multiplication
x = 3 * 3      ## 9

## division
x = 3 / 2      ## 1.5

## exponent
x = 3 ** 3     ## 27
```

Order of operations

PEMDAS

Parentheses first
Exponent second
Multiplication and Division are third
Addition and Subtraction are fourth

Operators with same precedence are executed left to right.

```
## parentheses to add before multiply
x = 2 * (3 + 3)     ## 12
x = 2 * 3 + 3       ## 9

## division before exponent
x = (3 / 2) ** 3     ## 3.375
x = 3 / 2 ** 3       ## 0.375

## nested operations
x = 10 + ((3 ** 3) / 9)   ## 13
```

Python numeric types

Comparison operations:

```
==     = equal to
>=     = greater than or equal to
>      = greater than
<      = less than
<=     = less than or equal to
!=     = not equal to
in     = in sequence object
not in = not in sequence object
```

```
## comparisons
x = 3
x == 3

## float and int are not same type but
## values are equal. Can be confusing.
3 == 3.0

## boolean
x = [1, 2, 3]
3 in x
```

Assignment to variables

The “=” character is used to set a variable to *something*.

You can later reassign the variable.

To ask whether two variables or values are equal Python uses the “==” operator.

```
## set x
x = 2 * (3 + 3)

## reassign x
x = (3 / 2) ** 3

## reassign x using value of x
x = x**2
```

Assignment to variables

The “=” character is used to set a variable to *something*.

You can later reassign the variable.

To ask whether two variables or values are equal Python uses the “==” operator.

```
## set y
y = x

## y and x are different variables, but
## store the same value
x == y       ## True

## reassign y it changes, but x is same
y = 10
x == y       ## False
```

Variable naming

There are some **hard rules**, such as no spaces, dashes, or some special characters in variable names.

There are also some *conventions* that are commonly used for variables. Use lowercase, descriptive words. If multiple words combine with underscores.

```
## ‘y’ is not a great var name since it’s
## not descriptive
y = 3

## better to name it something longer
my_var = 3

## Class, or globals are named with caps
TurtleClass          ## CamelCase
SETTINGS_DICT        ## ALLCAPS
```

Python data types

Over the next two weeks we will cover the core data types in Python, while also learning some other useful tools at the same time. Today we are learning **strings, tuples, lists,** and **range.**

```
## strings are immutable objects
x = “abcdefg”
y = “the cat jumped over the fence”

## tuples are immutable containers
x = (1, 2, ‘a’, ‘b’)
y = x[1:3]

## lists are mutable containers
z = [20, 0.13, ‘apple’]
z[0] = 0.05
```

Python iterators

Most of the standard Python data objects are a type of *iterator*, meaning that we can efficiently sample sequential elements from the object using a for-loop.

The key terms in a for-loop are **for** and **in**, which updates a variable on each pass through the loop.

```
## The standard for-loop structure
for {var} in {object}:
    {something}

## iterate over a string object
for base in “AGATCGGA”:
    print(base)

## iterate over a list
for spp in [‘species-A’, ‘species-B’]:
    print(spp)
```

Notebooks: 3.1 and 3.2

Let’s now open notebooks 3.1 and 3.2 to learn about strings and lists. Like the last notebook these have some code already written that you can execute, and/or modify.

In addition, these notebooks have challenges for you at the end. Read and execute the notebook and try to complete the challenges while improving your skills with using jupyter.

Notebooks: 3.3 files and data

Notebook 3.3 focus on getting data from files. The first is about how to read and write files. It also covers the `requests` library briefly, which is used to get data from the web.

```
## requests to query data from the web
response = requests.get(url)
data = response.text

## file objects, writing and reading
with open('myfile.txt', 'w') as out:
    out.write(data)

with open('myfile.txt', 'r') in indat:
    dat = indat.read()
```

Notebooks: Python packages

The Python standard library includes loads of packages for doing all sorts of tasks. These packages are atomic, each in its separate place, and must be **imported** to access their functions.

Many excellent Python libraries are not part of the standard library, such as **requests.** These can be installed separately (e.g., with conda) and loaded the same way.

```
## import a package from standard library
import os

## import a third-party package
import requests

## access functions in a library
requests.<tab>
```

Code reviews

Learn by looking at code. You may not answer questions the same way as someone else.

This week, you will push a notebook to your repository in which you critique the answers of your peer.

```
## clone repo of your assigned peer
## and open their notebooks to edit
git clone <url> 3-pdsb-peer

## or, just look at their code online
## and create and push your own notebook.
git push
```

Assignments and readings

Assignment is due Friday at 5pm

Code review is due Monday by class.

Assignment: Link to Session 3 repo

Readings: See syllabus

Collaborate: Work together in this gitter chatroom