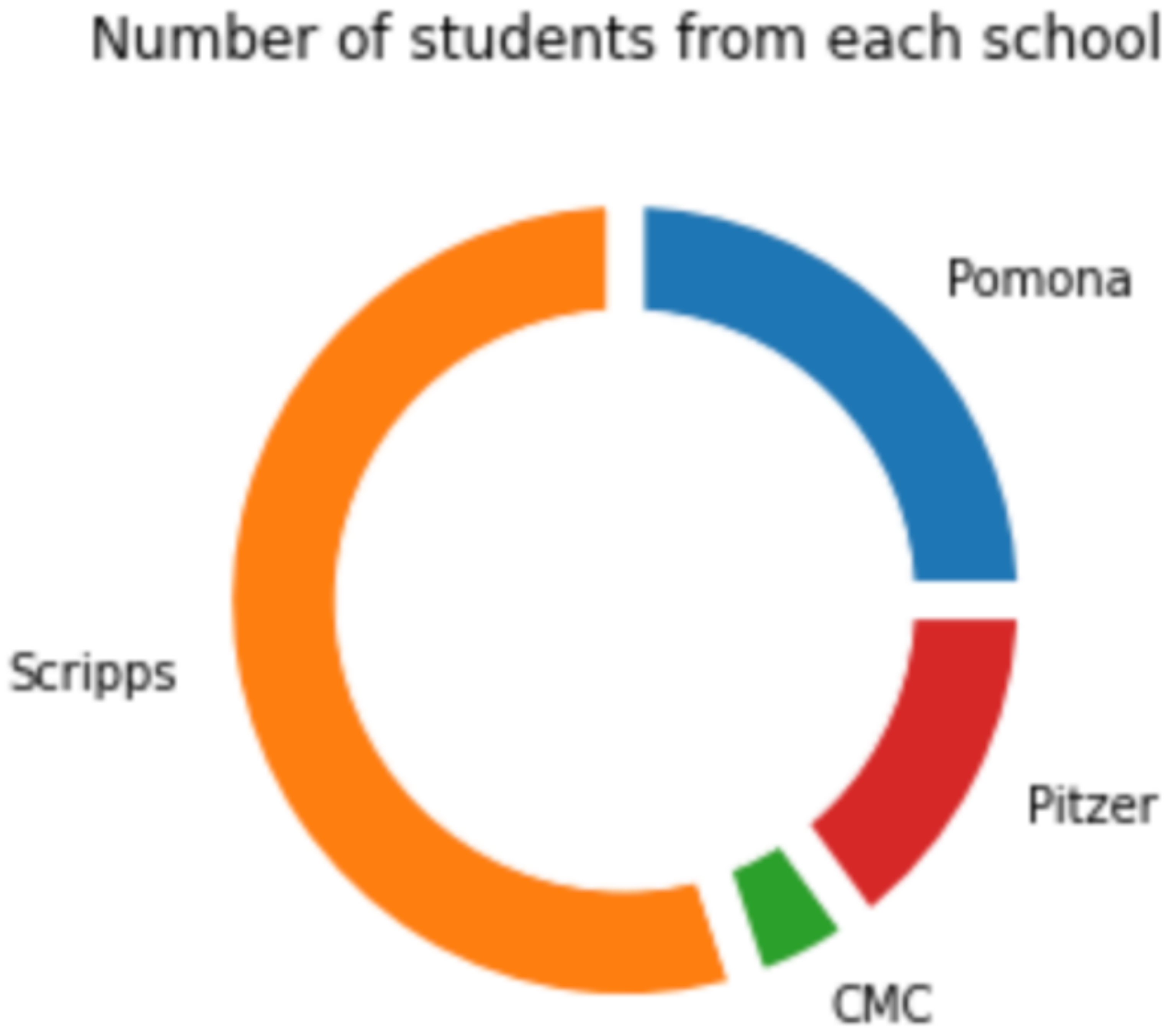
DS002.3.1: GitHub, Python Review 1



Administrativia

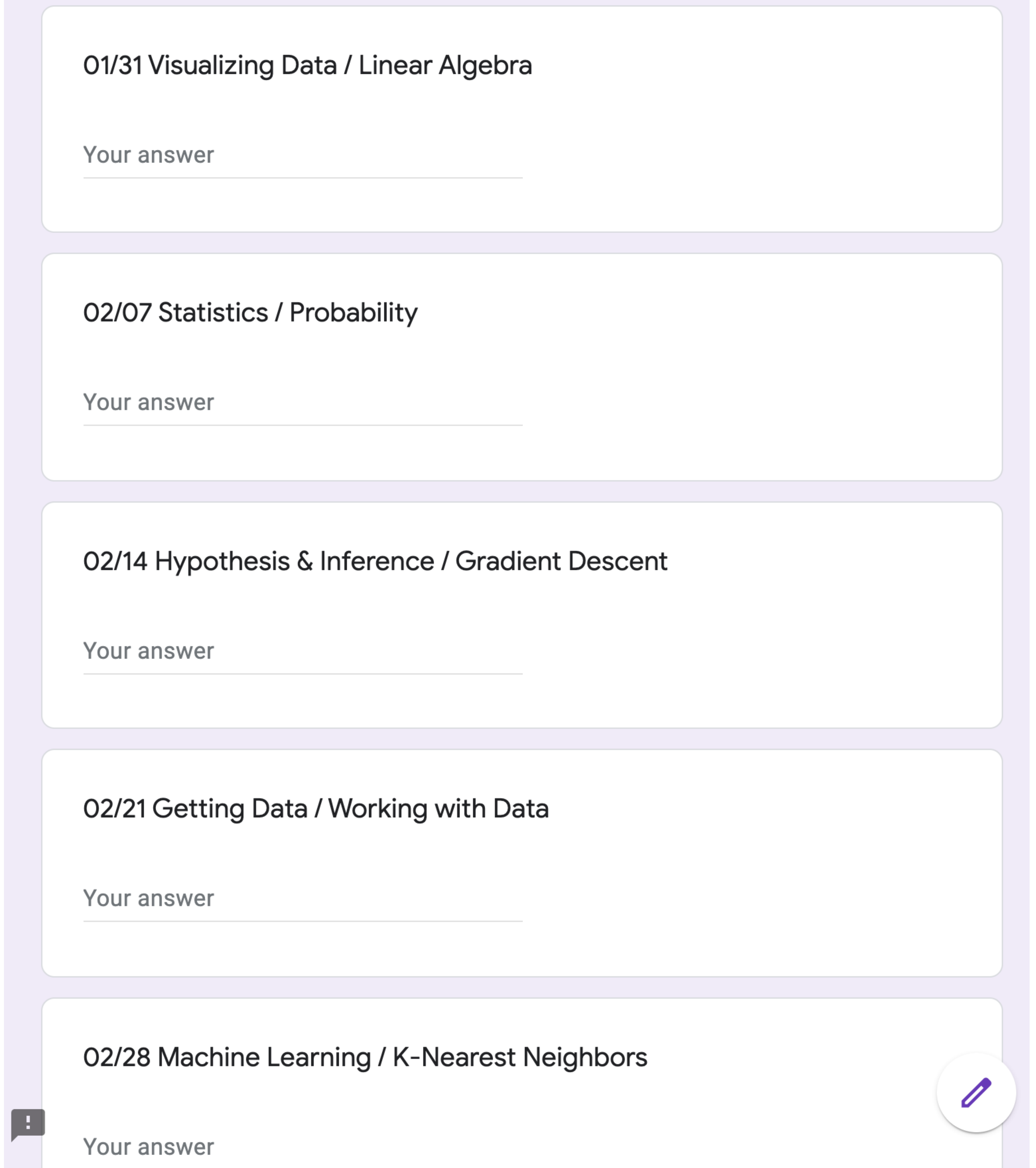
1. In-person classes
2. Attendance: double the number of classes you may miss
3. Computers
4. Recordings ok?

Sharing code is hard, especially for people in the back of the room. Last year students had trouble hearing me over the room fans. I want to try using Zoom in the classroom to share my screen to address these problems. I will also use Zoom to record lectures in case you can't come to class. Good?

Remote questions

I like to walk around the room to help you during exercises and lab time, and so probably can't handle remote questions or help synchronously.

Presentation slots are going fast!



01/31 Visualizing Data / Linear Algebra

Your answer

02/07 Statistics / Probability

Your answer

02/14 Hypothesis & Inference / Gradient Descent

Your answer

02/21 Getting Data / Working with Data

Your answer

02/28 Machine Learning / K-Nearest Neighbors

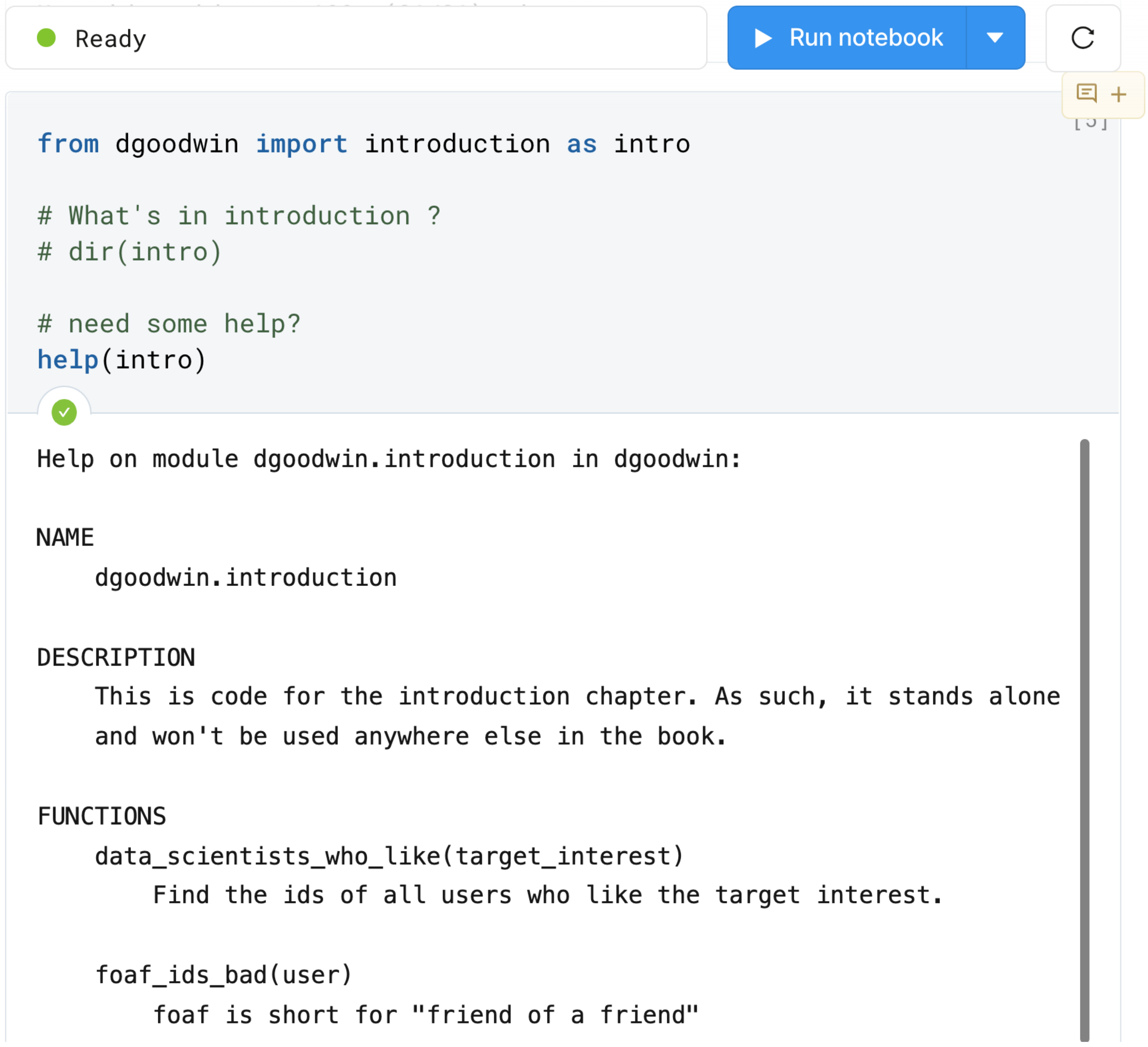
Your answer

GitHub: file edit, commit changes, push!

Open the Deepnote notebook that we created on Monday

Import your code and run

`help(intro)`


Ready

Run notebook

```
from dgoodwin import introduction as intro

# What's in introduction ?
# dir(intro)

# need some help?
help(intro)
```

```
Help on module dgoodwin.introduction in dgoodwin:

NAME
    dgoodwin.introduction

DESCRIPTION
    This is code for the introduction chapter. As such, it stands alone
    and won't be used anywhere else in the book.

FUNCTIONS
    data_scientists_who_like(target_interest)
        Find the ids of all users who like the target interest.

    foaf_ids_bad(user)
        foaf is short for "friend of a friend"
```

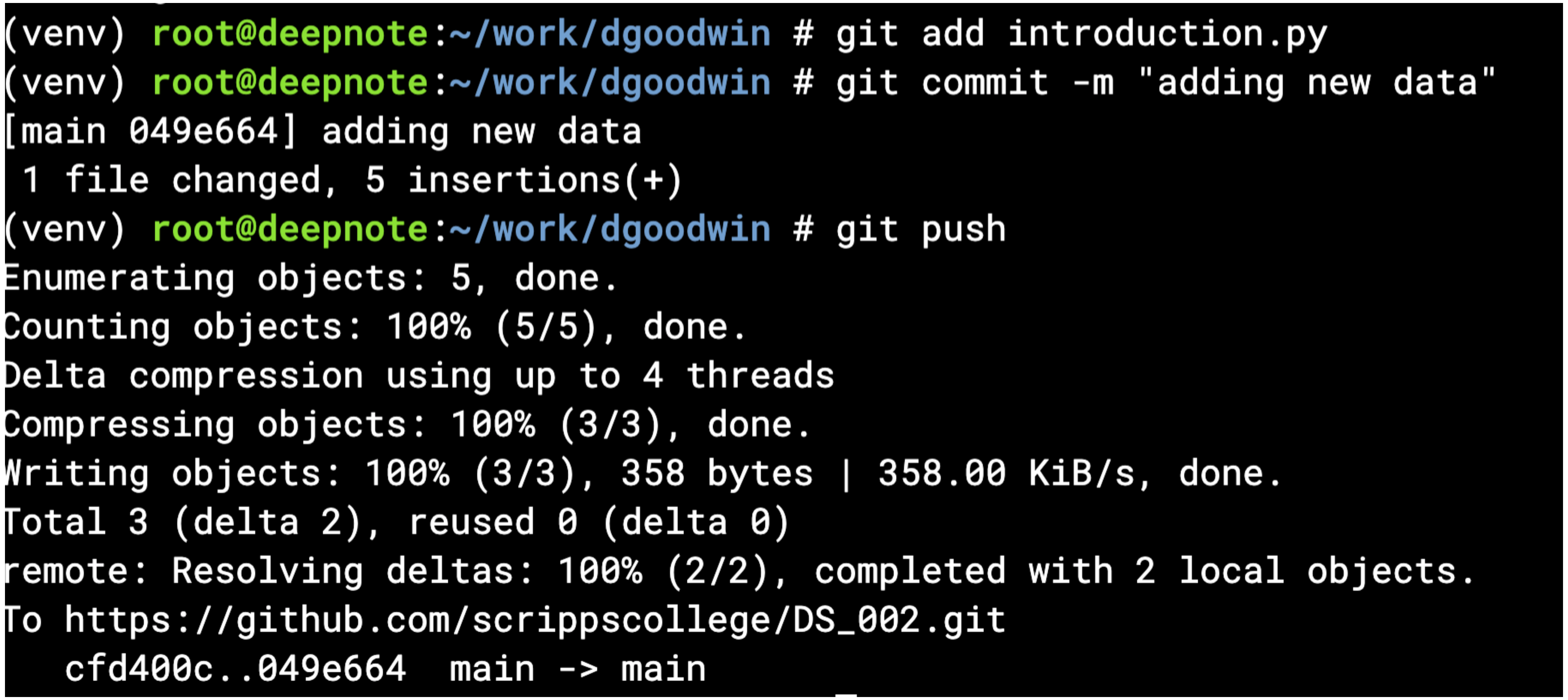
Copy **this data** to `introduction.py`

This data came from the form you filled out last week.

```
schools = ["Pomona","Scripps","Scripps","Scripps","CMC","Scripps",
    "Scripps","Scripps","Pitzer","Pitzer","Scripps","Scripps",
    "Pomona","Pomona","Scripps","Pitzer","Scripps","Pomona",
    "Pomona","Scripps"]
```

add, commit, and push your changes

Deepnote allows you to edit, save, and commit changes to your GitHub repository. Let's try it out!



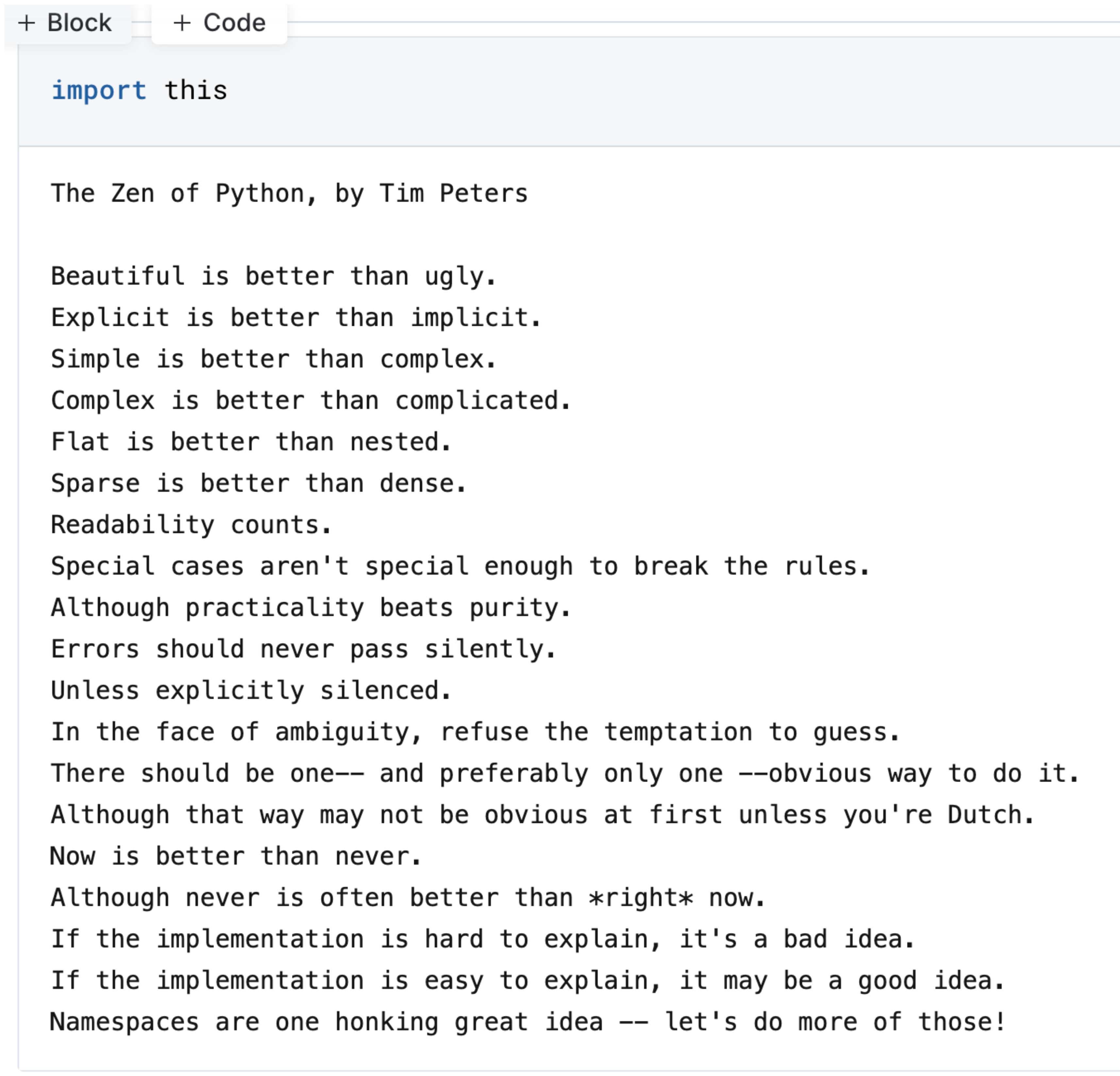
```
(venv) root@deepnote:~/work/dgoodwin # git add introduction.py
(venv) root@deepnote:~/work/dgoodwin # git commit -m "adding new data"
[main 049e664] adding new data
 1 file changed, 5 insertions(+)
(venv) root@deepnote:~/work/dgoodwin # git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 358 bytes | 358.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/scrippscollege/DS_002.git
   cfd400c..049e664  main -> main
```

Cycle your notebook, re-import your repository, import your new data

Can you print `intro.schools` ? Great!

START THE PYTHON REVIEW

Deepnote notebook



```python
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Very important Python features for DS

1. **Lists** `myList = ['Hi','there!']`
2. **Dictionaries** `myDictionary = {'firstWord': 'Hi', 'secondWord': 'there!'}`
3. **looping `for w in myList: print(w)**
4. **functions:**

```
def sayHi(myName='Anonymous'):
    print('Hi! My name is %s', myName)

>>> sayHi('Doug')
Hi! My name is %s Doug
```

Collect data: how will you store it?

data type	question
NOMINAL	**Your preferred pronoun**
ORDINAL	**How do you feel about chocolate? HATE**
INTERVAL	**What is the current temperature in your hometown? (in F)**
RATIO	**How far way are you from your hometown? (in miles)**