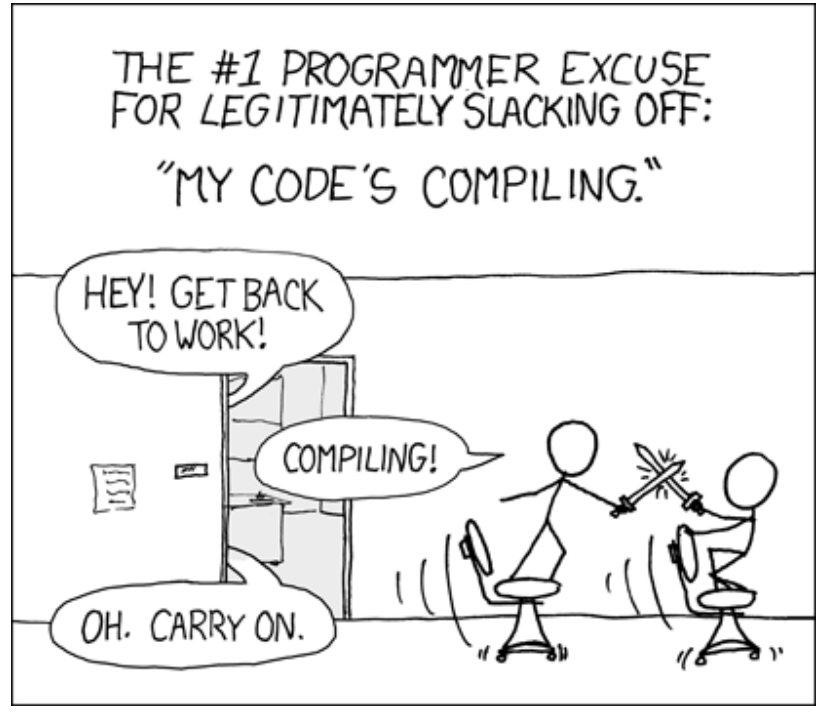
Introduction to Python for Earth Scientists

1. About the author;
2. Why programming;
3. Earth Science data: seismology-bias;
4. Why Python.



About the author

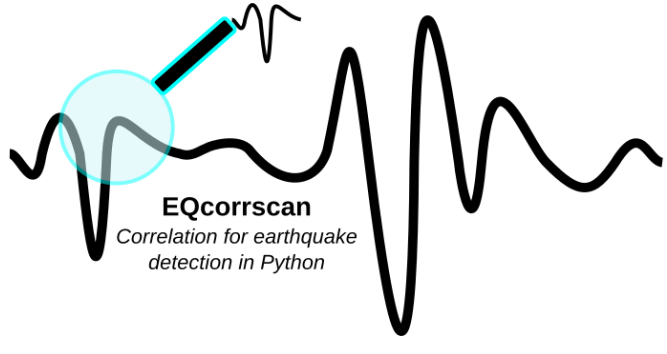
- Dr. Calum J. Chamberlain
- Room CO528 (do come find me if you have Python questions)
- Postdoctoral Research Fellow at VUW
- Completed PhD at VUW in 2016 and haven't left yet...

Research interests

- Earthquake Physics;
- I like thinking about how earthquakes start, stop and interact with other earthquakes and tectonic processes;
- Use large datasets of seismic waveforms and earthquake catalogues to tackle these problems;

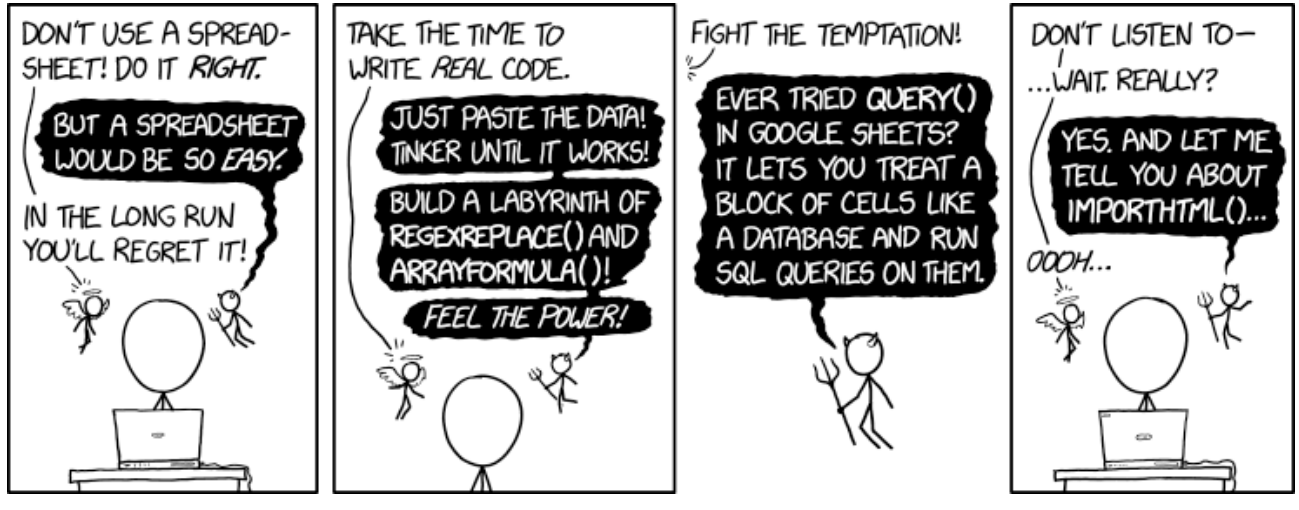
- Python has made my life simpler and my science faster and more reproducible;

    - I contribute to obspy (obspy.org) and lead the EQcorrscan (https://github.com/eqcorrscan/EQcorrscan) project.



Why programming (what's wrong with Excel!?)

Reproducibility, Safety, Speed, Complexity, Data scale



Why programming (what's wrong with Excel!?)

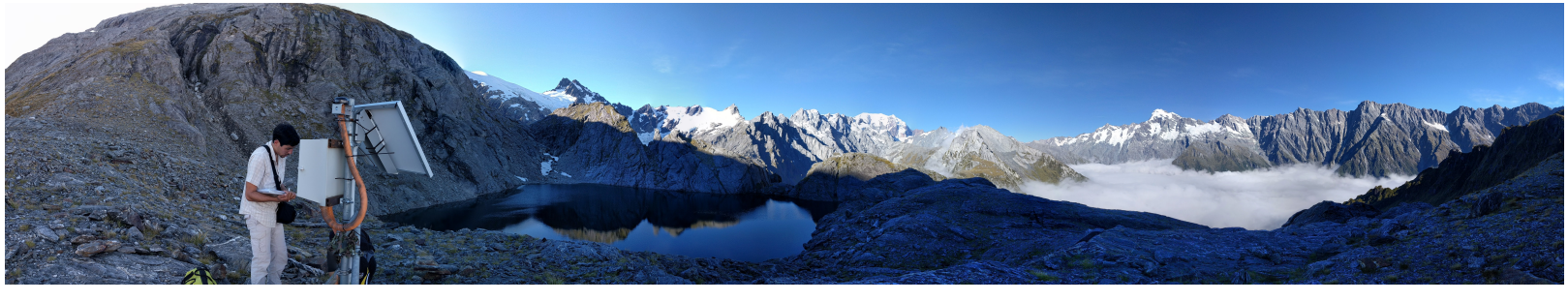
- **Reproducibility:** If someone can't replicate your work, why should we trust it to be true?
- **Safety:** Your data and your processing should not overlap. Your raw data should be sacred.
- **Speed:** You want a result, and you want it yesterday... Learn how to write good code (and change the clock on your computer) and you can...
- **Complexity:** Being able to solve complex problems logically, in a way that others can follow (and reproduce) is essential to natural sciences. *Hint: Writing good code is as much about the quality of your documentation as it is about the quality of your code*.
- **Data scale:** Data in natural sciences is noisy, and large. Ideally to understand the natural world we would have data from every place at every time throughout the Earth. We don't have that, but our datasets are growing...

Data scale: seismology bias

Seismographs deployed for:

- long durations (multi-year);
- multiple locations;
- modest sampling rates.

For example: SAMBA (http://ds.iris.edu/mda/9F/?starttime=2008-01-01T00:00:00&endtime=2020-12-31T23:59:59) is the Southern Alps Microearthquake Borehole Array around Mt. Cook has been recording since 2008.



SAMBA records at 200Hz (200 samples per second). How many seconds per day?

In [1]:
```
seconds_per_day = 60 * 60 * 24
print(f"There are {seconds_per_day} seconds in a day")
```

```
There are 86400 seconds in a day
```

How many samples per day?

In [3]:
```
sampling_rate = 200.0
samples_per_day = seconds_per_day * sampling_rate
print(f"SAMBA records {samples_per_day} samples per day")
```

```
SAMBA records 17280000.0 samples per day
```

So, > 17 million samples per day. But that is just for one channel: SAMBA seismographs have three channels, a vertical and two horizontals, so how many samples per day for one station?

In [4]:
```
number_of_channels = 3
samples_per_day_per_station = samples_per_day * number_of_channels
print(f"One station records {samples_per_day_per_station} samples per day")
```

```
One station records 51840000.0 samples per day
```

Nearly 52 million samples per day per station. SAMBA is made up of 13 stations, so our dataset gets bigger still:

In [8]:
```
number_of_stations = 13
samples_per_day_total = samples_per_day_per_station * number_of_stations
print(f"SAMBA records {samples_per_day_total} samples per day")
```

```
SAMBA records 673920000.0 samples per day
```

673 million samples per day across the network. So what is that over the first 10 years of operation?

In [11]:
```
days_per_year = 365.25 # Roughly
samples_per_year = days_per_year * samples_per_day_total
print(f"SAMBA records {samples_per_year} samples per year.")
samples_per_decade = samples_per_year * 10
print(f"In 10 years of operation SAMBA recorded {samples_per_decade:e} samples")
```

```
SAMBA records 246149280000.0 samples per year.
In 10 years of operation SAMBA recorded 2.461493e+12 samples
```

2 Trillion samples.

Try working with that in a spreadsheet...

Why Python?

Our first notebook covers this a little more, but the key things for me are:

1. Open-source, community driven (free) software;
2. Simple syntax, fast to make mistakes and helpful error messages;
3. Community libraries to do lots of complex tasks (e.g. obspy (https://github.com/obspy/obspy/wiki) for seismology)

How to use these notebooks:

There are a series of Jupyter notebooks on here. You can run them interactively in your browser. You should run through them, change some values, see what works, try and play with variables and experiment. There will be sections that you are expected to fill in marked as **Exercise:**. Shout out if and when you have problems.

Let us know if you want to play around with any other data.

Remember that this course is supposed to be a brief look over some of the key ideas in Python and useful libraries, it is not complete!

Plan for the day:

- 9 - 9.30am: Intro (this);
- 9.30 - 10am: Set-up;
- 10 - 11am: Do some Python;
- 11 - 11.30am: Elevensies (*very important*);
- 11.30am - 12.30pm: Do some more Python;
- 12.30 - 1.30pm: Lunch;
- 1.30 - 3pm: Do even more Python;
- 3 - 3.30pm: Afternoon tea;
- 3.30 - close: Haven't you had enough Python yet?!

Getting set-up:

1. Log into your Linux account
2. Set-up your conda virtual environment

```
startconda
conda activate esci451
```

   - for those not using a lab computer, skip this and go onto 3 then 3a

3. Clone the notebook repository

```
git clone https://github.com/calum-chamberlain/ESCI451-Python.git
```

   - If you are not using a lab computer you need to follow the install instructions here: [https://github.com/calum-chamberlain/ESCI451-Python (https://github.com/calum-chamberlain/ESCI451-Python)](https://github.com/calum-chamberlain/ESCI451-Python)

4. Start Jupyter:

```
cd ESCI451-Python
jupyter notebook
```