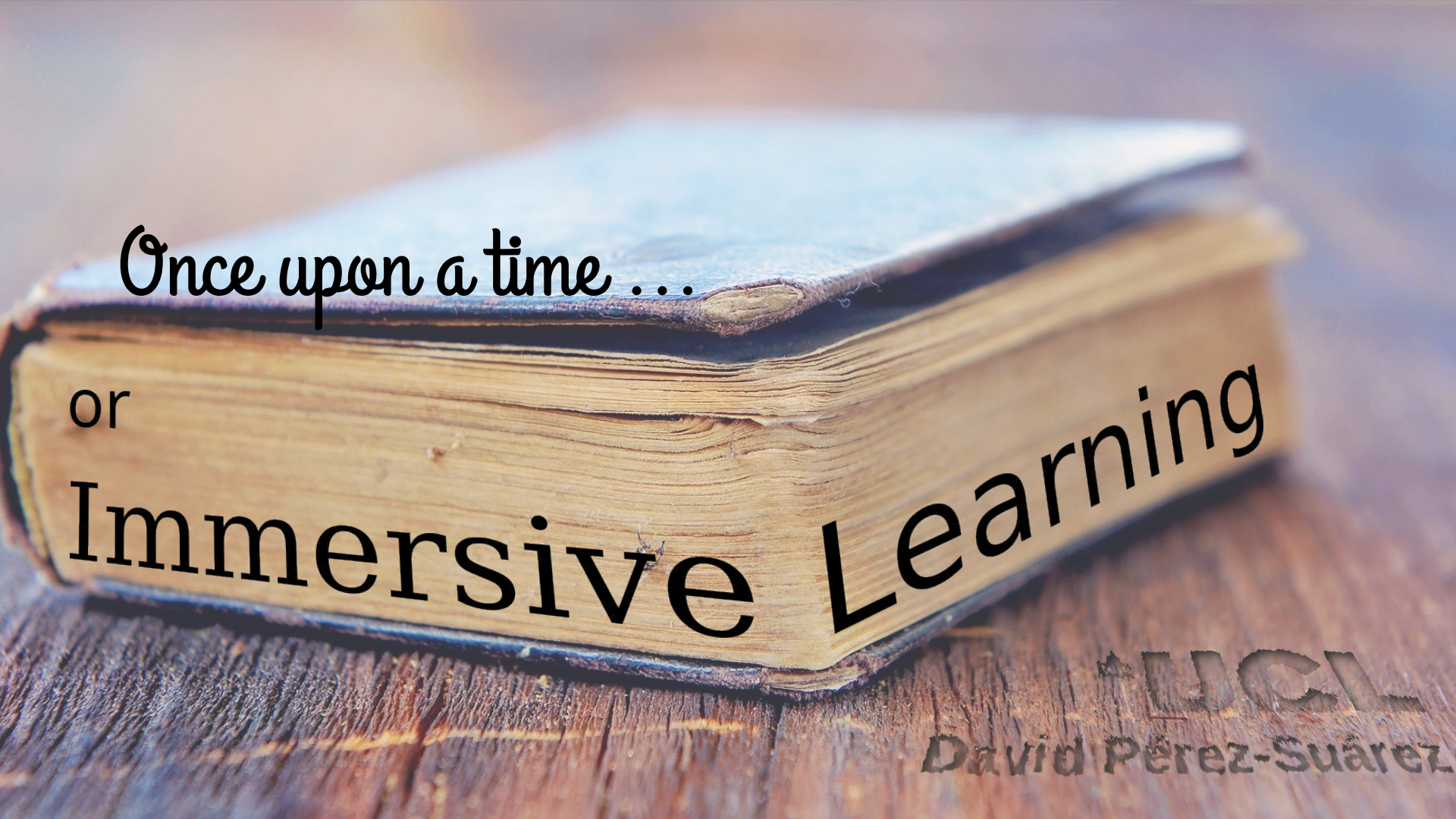
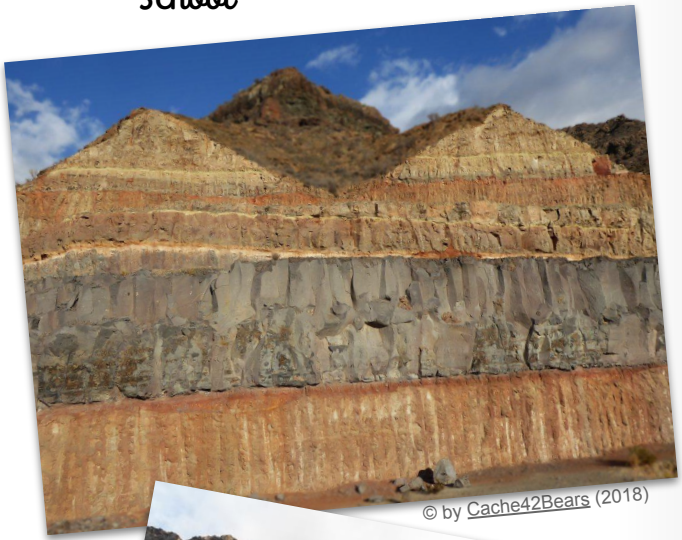
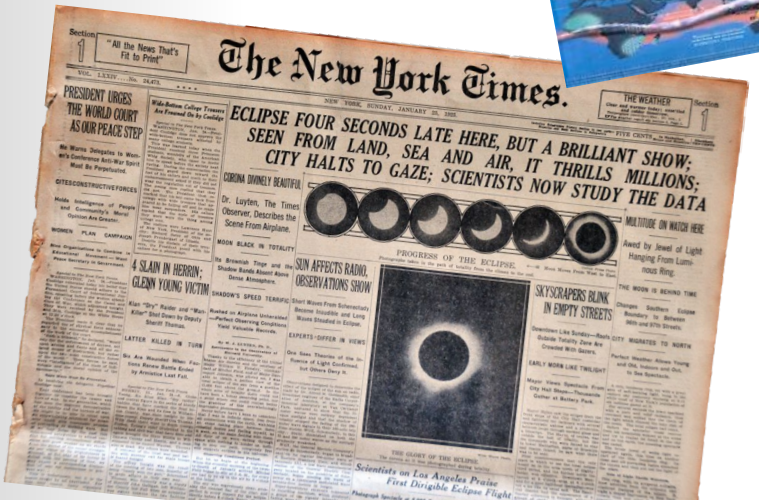

Once upon a time ...
or
Immersive Learning
UCL
David Pérez-Suárez

School

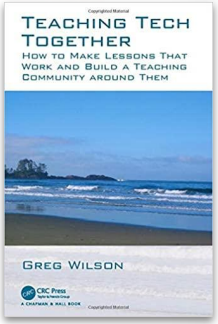


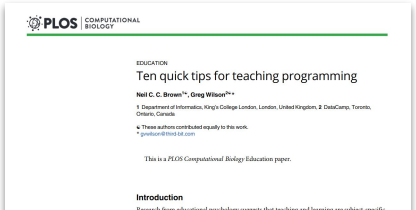
University



Greg Wilson

software carpentry




PLOS COMPUTATIONAL BIOLOGY

EDUCATION

Ten quick tips for teaching programming

Neil C. C. Brown[1]*, Greg Wilson[2]*

1 Department of Informatics, King's College London, London, United Kingdom, 2 DataCamp, Toronto, Ontario, Canada

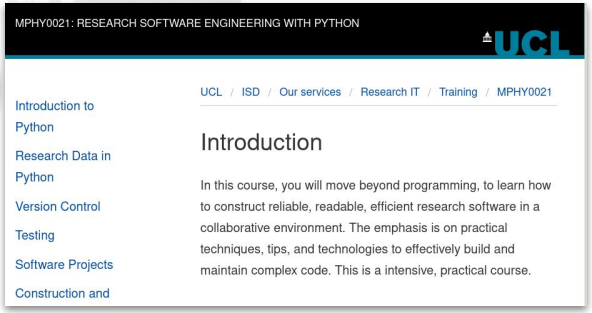
* These authors contributed equally to this work.
* gvwilson@third-bit.com

This is a *PLOS Computational Biology* Education paper.

Introduction

James Hetherington

UCL RESEARCH SOFTWARE DEVELOPMENT


MPHY0021: RESEARCH SOFTWARE ENGINEERING WITH PYTHON

UCL

Introduction to Python
Research Data in Python
Version Control
Testing
Software Projects
Construction and

UCL / ISD / Our services / Research IT / Training / MPHY0021

Introduction

In this course, you will move beyond programming, to learn how to construct reliable, readable, efficient research software in a collaborative environment. The emphasis is on practical techniques, tips, and technologies to effectively build and maintain complex code. This is a intensive, practical course.

Home Code of Conduct Setup Episodes Extras License Improve this page Search...

The Unix Shell

The Unix shell has been around longer than most of its users have been alive. It has survived because it's a powerful tool that allows users to perform complex and powerful tasks, often with just a few keystrokes or lines of code. It helps users automate repetitive tasks and easily combine smaller tasks into larger, more powerful workflows.

Use of the shell is fundamental to a wide range of advanced computing tasks, including high-performance computing. These lessons will introduce you to this powerful tool.

Prerequisites

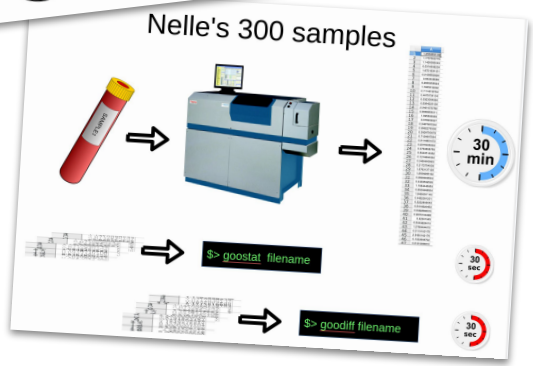
This lesson guides you through the basics of file systems and the shell. If you have stored files on a computer at all and recognize the word "file" and either "directory" or "folder" (two common words for the same thing), you're ready for this lesson.

If you're already comfortable manipulating files and directories, searching for files with `grep` and `find`, and writing simple loops and scripts, you probably want to explore the next lesson: shell-extras.

Nelle's Pipeline: A Typical Problem

Nelle Nemo, a marine biologist, has just returned from a six-month survey of the North Pacific Gyre, where she has been sampling gelatinous marine life in the Great Pacific Garbage Patch. She has 1520 samples that she's run through an assay machine to measure the relative abundance of 300 proteins. She needs to run these 1520 files through an imaginary program called `goostats.sh` she inherited. On top of this huge task, she has to write up results by the end of the month so her paper can appear in a special issue of *Aquatic Goo Letters*.

The bad news is that if she has to run `goostats.sh` by hand using a GUI, she'll have to select and open a file 1520 times. If `goostats.sh` takes 30 seconds to run each file, the whole process will take more than 12 hours of Nelle's attention. With the shell, Nelle can instead assign her computer this



MPHY0021: RESEARCH SOFTWARE ENGINEERING WITH PYTHON

UCL

Introduction

In this course, you will move beyond programming, to learn how to construct reliable, readable, efficient research software in a collaborative environment. The emphasis is on practical techniques, tips, and technologies to effectively build and maintain complex code. This is a intensive, practical course.

Exercise: a Maze Model.

Work with a partner to design a data structure to represent a maze using dictionaries and lists.

- Each place in the maze has a name, which is a string.
- Each place in the maze has zero or more people currently standing at it, by name.
- Each place in the maze has a maximum capacity of people that can fit in it.
- For each place in the maze, you can go from that place to a few other places, using a direction like 'up', 'north', or 'sideways'

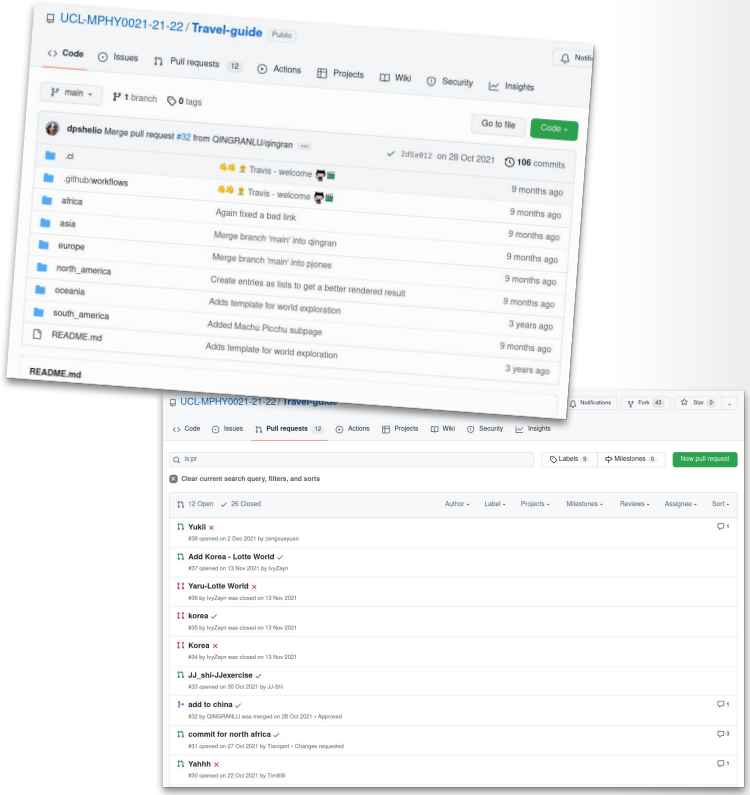
Create an example instance, in a notebook, of a simple structure for your maze:

- The living room can hold 2 people. James is currently there. You can go outside to the garden, or upstairs to the bedroom, or north to the kitchen.
- From the kitchen, you can go south to the living room. It fits 1 person.
- From the garden you can go inside to living room. It fits 3 people. Sue is currently there.
- From the bedroom, you can go downstairs. You can also jump out of the window to the garden. It fits 2 people.

Make sure that your model:

- Allows empty rooms.
- Allows you to jump out of the upstairs window, but not to fly back up.
- Allows rooms which people can't fit in.

```
house = {
    'living' : {
        'exits': {
            'north' : 'kitchen',
            'outside' : 'garden',
            'upstairs' : 'bedroom'
        },
        'people' : ['James'],
        'capacity' : 2
    },
    'kitchen' : {
```

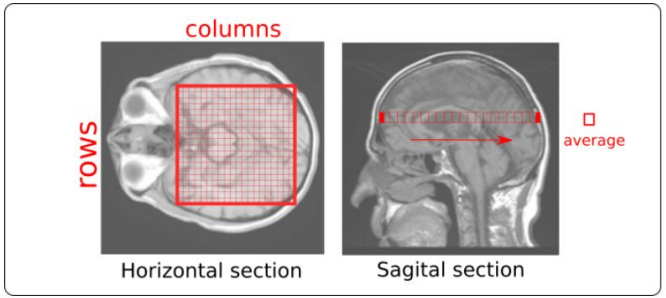


Finding bugs in history #25

Open **dpshelio** opened this issue on 5 Nov 2021 · 2 comments

dpshelio commented on 5 Nov 2021 • edited by stefpiatek — Contributor

Charlene Bultoc has just started a post-doc at an important neuro-science institute. She is doing research on a new methodology to analyse signals in our brains detected through a combination of CT and MRI. Using image processing techniques she can simplify the whole dataset into a grid of 20x20 arrays.



Her theory is that the average of such signals through the sagittal plane is constant over time, so she has written some software to calculate this. She decided to write that software in Python so she could share it (via GitHub, sagittal_average) with people from other labs. She didn't know as much Python when she started as she does now, so you can see that evolution in her program.

Charlene is an advocate of reproducibility, and as such she has been keeping track of what versions she's run for each of her results. "That's better than keeping just the date!" you can hear her saying. So for each batch of images she processes she creates a file `versions.txt` with a content like:

```
scikit-image == 0.16.2
scikit-brain == 1.0
git://git.example.com:brain_analysis.git@dfc801d7db41bc8e4dea104786497f3eb09ae9e0
git://github.com:UCL-MPHY0021-21-22/sagittal_average.git@d8bc3ebaecd0cc7a2872da4c81d30b56f9b746ad
numpy == 1.17
```

With that information she can go and run the same analysis again and again and be as reproducible as she can.

However she's found that `sagittal_average` has a problem... and she needs to re-analyse all the data since that bug was

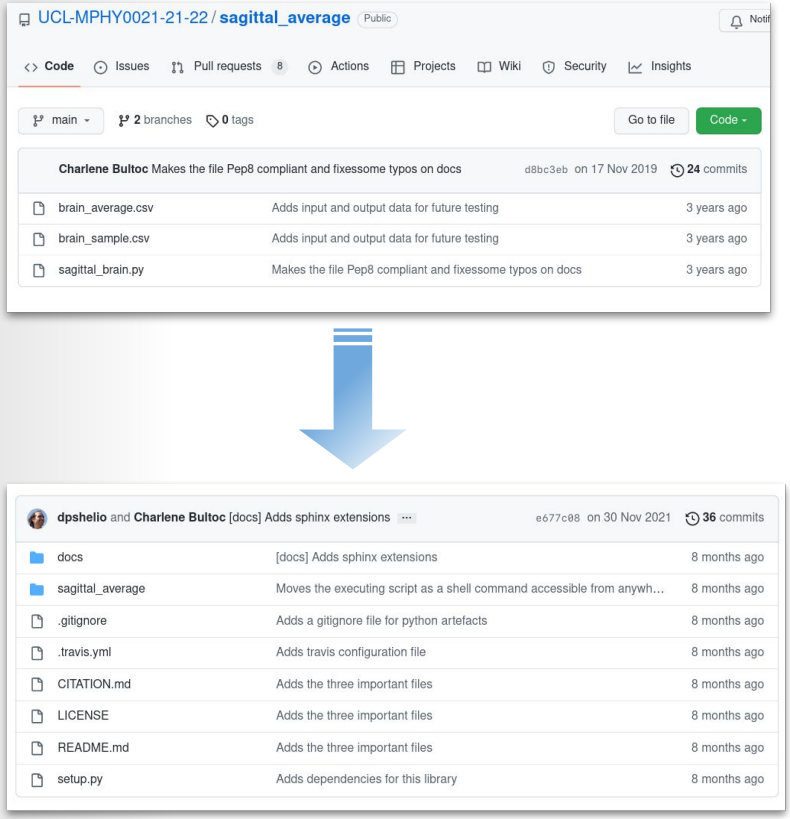
Creating a 🐍📦 #36

Open **dpshelio** opened this issue on 25 Nov 2021 · 0 comments

dpshelio commented on 25 Nov 2021 • edited by stefpiatek — Contributor

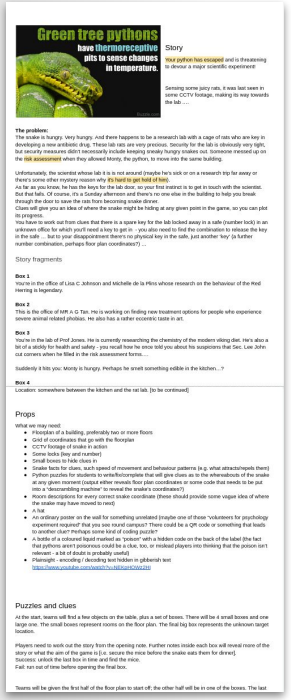
Help Charlene to create her repository into a package (remember to commit after each step).

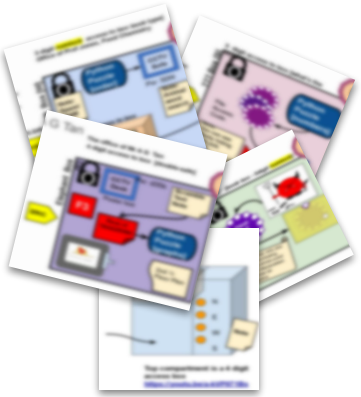
1. Choose who in your team is sharing now! (make sure you've got a **fork** and a local copy of Charlene's repository)
2. Add a `.gitignore` file to the repository to avoid adding artefacts created by python or your text editor. You can use gitignore.io to generate a file for your needs.
3. Modify the repository directory structure to make sagittal_average as an installable package (don't forget to add empty `__init__.py` files, but there is no need to add the `.md` files (yet!)).
4. Add a `setup.py` file with the information needed.
5. Try to install it by running `pip install -e .` from the command line, in the filesystem location where the `setup.py` is.
6. Share your solution as a pull request to Charlene's repository mentioning this issue (by including the text `Addresses UCL-MPHY0021-21-22/RSE-Classwork#36` in the pull request description), remember to mention your team members too! (with `@github_username`)
7. Congratulations, you've created a Python package! Now, let's see what can be improved about it in the subsequent issues!



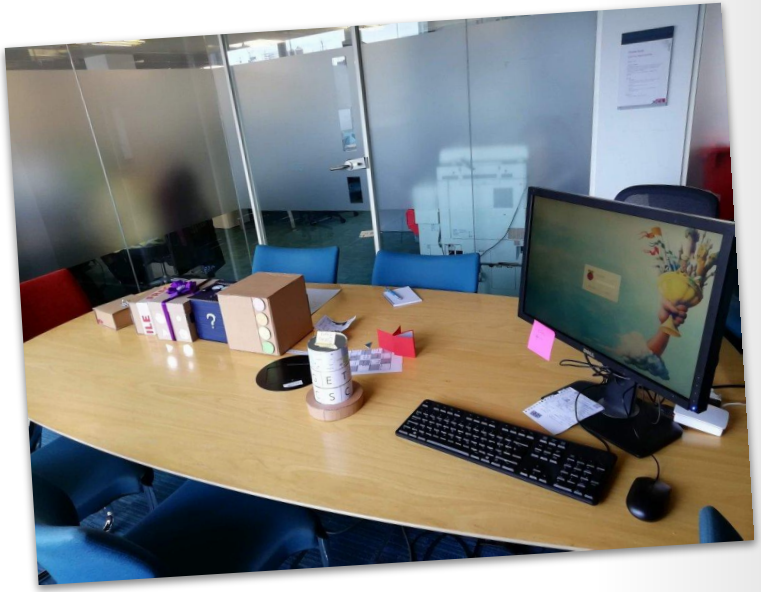
And then …

a escape room





a escape room (with real students!)



Online version... on the making

It was a real challenge (almost three weeks of full-time work) but also a very rewarding experience and I learnt a ton :) Thanks for such a great module and such an enriching learning experience!

Thank you to the MPHY0021 team for being great teachers and making us to enjoy the lectures. I think that it was the best coding lecture course I have attended.

Thank you all for teaching such an amazing module - UCL would have highest student satisfaction in the world if all modules were like this.

Thank you teachers, this is a bravo class!

Summary:

- Tell a story
- Let them be part of the story