We had trouble with pushing repos to gihub if the local email did not match the github email address. github rejected to do this to protect the private email from being exposed.

On newer windows jupyter is started with “python -m notebook”

Also in the Git bash the path was wrong and did not find python.

Do not forget to print cheatsheets (Radovan forgot to do this for Git, again).

Make clear that Jupyter needs to run on local machine - I got a question whether people can use a distant linux machine and I did not realize that Jupyter will then not trivally work.

Is it better to first finish the git intro and then switch to Jupyter? Maybe not a problem if people can change their focus from git to jupyter and back to git without effort

Keep Jupyter as last session so that the group work doesn’t have to end abrubtly and people can stay longer if they wish?

Quaker terminal on linux somehow adds /home/user/.local/bin to PATH even though the last line of .bashrc is “export PATH=/home/user/anaconda/bin:$PATH”. This created jupyter/pip issues for one person. It worked fine in the native terminal

Add “pip install rpy2” to Installation instructions

Reduce the number of things demonstrated in Jupyter introduction (particularly some magics), and list them instead for later reading by learners, to save time for group work

Could we have groups by picking random names of fruits or animals? No need to assign numbers, forces them to think about clashes (should one pick a name someone else is likely to pick or something really exotic?). That way we could accidentally get

Get the afternoon coffee already after lunch and get morning break coffee at the beginning of the course so that people can grab coffee whenever they like and so that it does not arrive too late.

PyCharm lesson depends on integration with GitHub but that can be very tricky on Windows (PyCharm did not find git.exe of the Git bash). I would recommend decoupling this. Otherwise Windows users spend time installing things and get completely lost in the lesson and miss all essentials.

Avoid \_\_name\_\_/\_\_main\_\_ stuff in the lesson, for Python newbies unnecessary complication.

Should we perhaps \*not\* mention the 3-6 month feedback, so that it doesn’t somehow bias people? [Radovan: :-) good idea, agree, let us then perhaps not mention it next time to set no bias at all]

Meditating anecdote in the testing session was good :-)

We should try to convey the testing session trying to minimize jargon.

I think we still spend too much time on testing theory - we really need to start with exercises sooner in the session, ideally 10-15 minutes into the session.

In the <https://github.com/coderefinery/cr-ide-test> part, dependencies were introduced that we did not prepare for in the install instructions (pytest-mock). We cannot introduce dependencies at this late stage - it will throw half the people off the session. Many people do not have virtualenv on their machine, most people run Anaconda (since we recommended it).

I think we have too many industrial examples in the testing part - people at the workshop may then think that this is still not relevant for them and that they might get away without testing - we need research examples only.

We have too much text in the testing part -> extract this into best practice guide and massively reduce the slides. Instructors tend to follow this material and we spend 45 minutes on this, way too long.

Testing example starts with existing code base and tests and it was argued that we never start from zero but actually most people in our workshop do not have any tests in their codes so adding tests from zero is really relevant for them.

If we introduce new exercises “late”, then other instructors cannot really help because they also see it for the first time (but I - Radovan - have done that, too, so also guilty).

People expected to see red tests when things fail. I think we should not use “incomplete” tests in our exercise examples.

Radovan: I think we should keep IDE and testing part decoupled, it actually increases complexity IMO since more things can now break and it increases cognitive load. It is OK to demo an example in both parts referring to the other section but it was hard to follow.

We always need to get the essence across. If we show many interesting facts but the essence is not conveyed, then it is a pity.

We have to be careful not to use any shell aliases since then nobody can follow on their terminal.

In the ccmake example it was not clear where it was run from.

We should mention RTD accounts in the install/setup section to avoid that people need to set it up during the session.

pybind11 example on one osx machine could not detect pytest, although pytest was installed in an Anacoda environment - possibly the CMake detection probed the wrong Python. My recommendation is to install pybind11 before hand and not let people install it during the course as this is error prone. The participant fixed it by passing the correct -DPYTHON\_EXECUTABLE to the cmake command.

We need to test all examples we use automatically on Travis, both on Linux and OS X since we always have hiccups on OS X or the “other” platform.

Pybind11 example seems to require Python 3. Lots of trouble in the class room imporing lib under Python 2.