# General feedback

* IDE was not so interesting
* The whole workshop: very good. High quality. Thank you! Very good to have all on github.
* Too much python. It would be good if you advised before that Windows support is limited because Linux would have worked equally as well for me. But too late to change at the course.
* Give to the audience time to go into the exercises.
* In overall this was a nice experience! Thanks guys!
* Good with the short presentation and exercises in the Git part [Radovan: it is not clear which Git part is meant]

“Hey. Thanks!   
It was really fun and interesting.”

# Jupyter session

* Lots of very useful information with examples which runs. Good with access to an experience with loaded notebooks.
* Went too fast and it would be nice to be able to type along, but then we would only go through a very small fraction of what we did.
* Very nice introduction - great with the many exercises and the solutions.
* Might have been nice with a bit more time - but as the exercises are there we can always look on our own.
* was interesting; P.S. Albert Fert appears twice (with same id?) in Nobel prize list
* please use python 3.5 :)
* Nice introduction to a useful tool
* pd/melt and some of these tools went very fast
* Broad, nice introduction
* Self-explanatory text and exercises.
* Too much down time i.e. a bit slow [Radovan: this was in Jupyter folder but I am not sure this is the right folder]
* Well explained!
* Good exercises!
* Nothing to complain about!
* Only one thing: Just an example showing Jupyter being a better tool than the regular IDE.
* Really cool useful tool
* Lots of information and well-structured lesson in NOT enough time.
* Explicit, previously prepared examples
* Too fast
* it would help not to talk/explain, while you press the buttons
* Good coverage of different topics
* Want more time
* The mix between going through a lot of things and trying to make small exercises was a bit too much. Either less content and exercises or more content and walkthru the examples.

# Git introduction

* Scrolled a lot around in lesson 1 and that was confusing.
* Overview of git features + workflow
* Make sure people have the right prerequisites
* Presentation and exercises in one “document” (instead of a slide show and an exercise document) Easy to find via workshop page and schedule
* One or two small breaks for stretching legs would be good.
* Interactive teaching
* slow start due to person not knowing how to open a terminal -> this could be a specific question alongside “did the installation go well?” Would also give more time at later git topics
* Easy to follow introduction to git
* Using a tool to visualize branch structure would make it easier to understand.
* Very good introduction to Git for beginners - nice pace
* Perhaps even more exercises - even just for homework
* well structured, easy enough to follow for a beginner (1. part)
* **a printout of all git commands used during the day** [so one does not have to go through course material if one forgets something :)]
* How you demonstrated answers to questions
* Last part went too quick
* Examples easy to follow
* Straightforward explanations with examples that build up throughout the session
* Interactive
* Updated material online
* Nothing, as a beginner was a good pace for me. But maybe boring for more advanced users.
* Speak loud or just increase monotonically the volume of your voice
* Very didactical but a bit messy. Maybe you can split the session in two parts.
* Many information => Many questions that I would prefer to do after double check the info.
* Could be more dynamical but it was an excellent starting session for not experts.
* **that we did some of the exercise ourselves**
* we could do more complex examples where we also delete files and maybe better understanding of the staging area
* clear, not rushed
* ?
* Recipe example works very well
* Talk a bit louder

# Complexity and modular code development

* Everything in lesson 2
* Too complicated scripts. It too all the time to read and understand many of the code examples.
* A bit terse
* The slideshow was a good overview to take a closer look at a later time, but
* it went a bit fast and was a bit difficult to understand
* Lot of material
* **Would be good with some hands-on**.
* Good topic. Useful things to know. Unfortunately I did not get too much of it.
* First part: mentions a lot of things but not really why/how/when to use it.
* Last part: too many examples. Again not really explaining when to do what.
* Relevant topic.
* **Would have liked clearer recommendations**.
* Too much python focus.
* figures illustrate concepts, useful tips
* maybe a pdf printout of talk so one can make notes?
* use only a few examples throughout the talk consistently, so time is not wasted reading new example on each slide (similarly as using recipe example in git part)
* A lot of very useful tipes
* A few exercises could have been good - like guess what the code does.
* Explicit examples
* Pictures on slides
* Use “complete” examples
* Explain abbreviations (API)
* Spend more time to link examples to concepts/vocabulary
* make better connections between different topics
* don’t speed up, when you explain things

# Git archaeology and branch design

* Very useful topics
* Could have spent more time ...
* Good
* Once again bigger screen needed.
* Useful tricks & fun exercises
* /
* Emphasize that we do not need to remember, but see what is possible and understand why. You could emphasize this even more for the entire workshop and make it dominant in all sections of the workshop.
* Practice git clone and many git commands
* SUPER USEFUL!
* bisect exercise
* making nice commits (squashing)
* clearer formulation of the motivations for blame, ... and the differences would be great
* Too git-technical, I would have liked a more general discussion applicable also on svn
* Good quality

# Collaborative Git

* We need a bigger screen. I can barely see what was on the screen.
* It was great to see the merging of PRs!
* Everything
* Nothing
* Exercise in pairs
* Talk is too fast
* Good exercise + introduction to forking/ merge requests
* **Could use another interactive exercise**
* Good structure with terminal work and webpage
* Using sticky notes and branches, head
* Lots of useful information
* Good tempo
* The fork and remote wasn’t completely clear. The difference between centralized system with branches and forks.
* Fun exercise, seems like a lot of useful info, good presentation/ explanations
* Not clear if we were supposed to clone the repository and follow the examples interactively or just listen/ look at the projector. I would have prefered to try it out simultaneously with the talk but it was a bit too late for that.
* Hands-on and interactive
* **Working in pairs** enhances discussion to better grasp the lesson
* Maybe longer lesson to practice more, but NOT needed, already pretty good.
* A lot of really useful information and very illustrative exercises.
* **More exercises would be nice. Perhaps modify timeplan so nothing needs to be rushed - rather fewer things.**
* I finally understand how to elegantly updating when working with forks!
* Exercises were fun and teaching necessary parts without being complex.
* Pictures illustrating commits, branches, ... are all very similar and after several topics it’s hard to remember the changes wrt the other ones.

# Testing

* Many useful informations (and good exercises)
* The schedule was too ambitious
* Internet need to be improved. It’s not good to have troubles with the presentation because bad internet connection. [Radovan: we need to be able to serve without internet]
* Very useful topic
* Need to work on presentation technique ...
* Nice exercise
* **Too long intro, too little interactivity in first part**
* Good to have introducing questions
* Automated tested exercises
* Better than yesterday, but even clearer when switching between topics might be of advantage
* formulate questions clearer, closer to our experience
* Very important topic
* C++ exercise did not work (**because of Windows**) [Radovan: that is my bad]
* Hands on exercise with forking, issues, and pull requests
* Spends too much time on motivations. Would be nice to spend more time on examples.
* Not very structured and difficult to understand from a lot of text.
* I didn’t understand the concepts of unit testing, integrated testing. **Would be nice with some graphical or schematical presentation**.
* **The “slides” don’t support presentation very well**.
* More examples/exercises for the testing/concepts part, please
* To be a not python expert I am confused if what was explained applies to other language like C++
* Should be presented more clearly because in my opinion this is a very important subject.
* More dynamical presentation including **slides without many text.**
* Exercises are good and help a lot but should be linked to the content of the presentation.

# Documentation

* Markdown exercise included in the documentation exercise would be nice
* very good
* nothing
* From this section I realized that it is more practical give to the audience time to go into the exercises.
* Small exercise with repositories to clone to try out making the websites easy from github/local.
* **Super convincing that the CodeRefinery project uses the same tools**.
* Relevant, interesting, easy to follow, motivating.
* **Maybe a few illustrative examples in the talk part (screenshots, pictures, etc.)**.
* Documentation is very important. Glad you included it!
* Relevant topics.
* Can’t think of any. It was really good.
* Nice with hands-on example for some distribution platforms.
* **No introduction to the syntax used by Sphinx et al. and how to configure them**.
* Hands on examples.
* The code on the home page is very disconnected from what we learned during the workshop -> **maybe use a markdown example?**
* Good useful exercises.
* Really nice introduction.
* Perhaps a bit more about how it works with webhooks and git webpages in general if you have extra time.

# IDEs/PyCharm

* Interactive, type along
* Tables with shortcuts next to PyCharm (would be even better with two screens)
* Motivation took a long time. Would be nice to spend more time in PyCharm to see some of the more advanced features which would be useful. This would be the best way to convince people.
* Again, the screen is a limiting factor.
* As the previous section this part was very dense but still well explained.
* PyCharm is great **but I need the key reference**.
* Well explained the basic features to dig into the tool.
* Builds on [-unclear-] things already learnt along the workshop.
* **Show the support for other languages, i.e. RStudio**.
* I loved the features in PyCharm that you showcased.
* The speaker was clear and thorough in his talk.
* Good introduction.
* Could skip the 1st part?
* Good to see how comfortable they are.
* Too much focus on Python.
* Pretty much all of it.
* Perhaps a bit more type along.
* Interesting to see.
* Not relevant for me. I love my vim.
* It is difficult to follow, when you use short cut keys (because one can’t see what you are doing)
* Contradicts the idea of doing only one thing but do it well.

# CMake

* Very interesting and important.
* MUCH TOO FAST
* Step by step
* Start with makefile
* Incomplete on the homepage.
* Too fast.
* You lost me when starting to import file in main.cpp -> you additionally sped up.
* Rather let us try to do it by ourselves -> allows to remember what we did
* Important topic
* **Much too fast**, no exercise. It would have been better with less. And a chance to try it myself. Now it’s mostly confusing. Will I be able to reproduce at home?
* Compelling intro for people who don’t develop in non-compiled languages.
* Really cool tool & empiric examples.
* /
* I will no longer be scared when I see Make/CMake!
* A lot of material was not relevant for me. Would be good with a more “bare bone” course for python-devs.
* In my opinion could be go to follow the example but there are so many things. My suggestion is stop at some point and divide the talk in two parts. Or focus in some part and leave the rest for home.
* interesting
* **Less is more**: it’s better to include less and not have to rush through everything. When it’s hard to follow, it’s hard to take away enough from it.
* Fast, lots of very useful information.
* Fast. I chose to not type along (because I got lost in the beginning) and then the tempo was good. Too fast to type along.
* Nice to build it up from scratch!
* Useful introduction to basic CMake functionality.
* Nice simple example.
* Last part was very fast.