Advanced Scientific Programming in Python

a Summer School by the G-Node and the Physik-Institut, University of Zurich

September 1–6, 2013. Zürich, Switzerland

Evaluation Survey Results

Method

The survey has been administered with a web interface created with the LimeSurvey software available at: http://www.limesurvey.org
All answers have been submitted by October 07, 2013.
No answer was mandatory.
The free-text answers have not been edited and are presented in their original form, including typos.

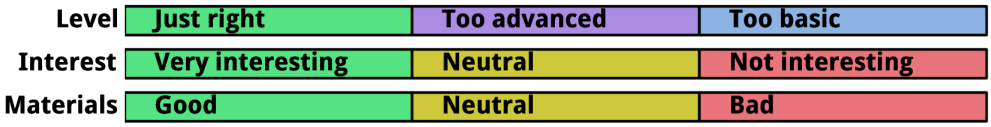
Attendants and Applicants Statistics

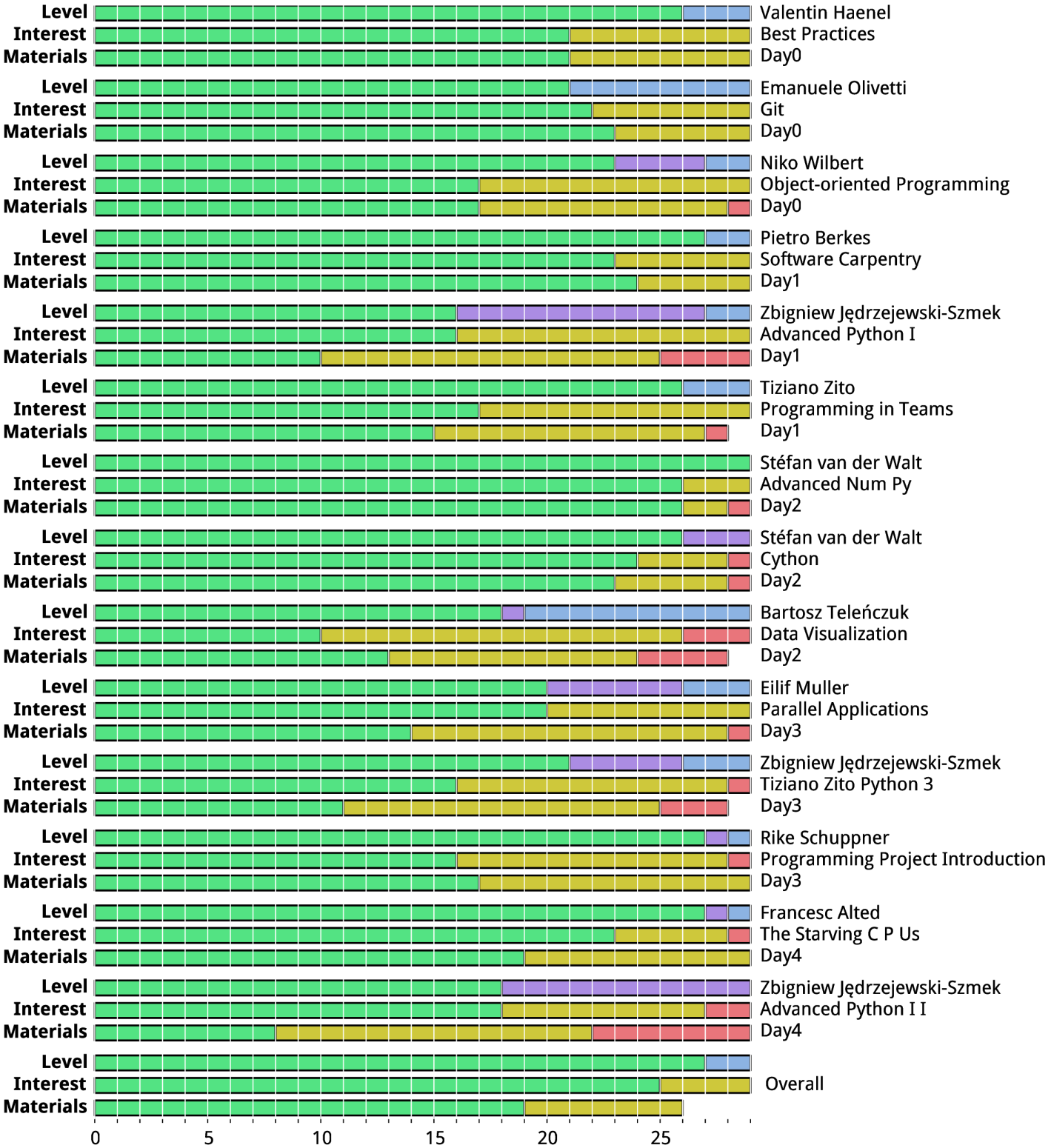
	Attendants		Applicants	
	30	17%	177	
Different nationalities	13		39	
States of affiliation	11		28	
Female	11	37%	41	23%
Already applied	18	60%	26	15%
Bachelor Student	0	0%	5	3%
Master Student	1	3%	23	13%
PhD Students	22	73%	97	55%
Post-Docs	4	13%	21	12%
Technician	1	3%	2	1%
Employee	0	0%	10	6%
Others	2	7%	13	7%
Completed surveys	29	97%		

More stats about attendants are available at: https://python.g-node.org/python-summerschool-2013/students

Lectures & Exercises

Q: Grade the level of the lectures
Q: Grade how interesting were the lectures
Q: Grade the quality of the teaching material provided by the lecturer, e.g. the clarity of the slides, references given, exercises etc.





Q: Are some of the topics presented in the lectures not relevant for a programming scientist?

1. I think the team programming concept, while important in a production environment, is almost entirely irrelevant in fields I'm familiar with. There is typically only one "programmer" (e.g. prototyper) per project. It was helpful to learn from more advanced partners, but would have probably have been better to facilitate the teamwork without the (in the end, fiction) of 2 programmers per computer.
2. For me, "using Cython" and "writing parallel applications" is less relevant than other topics...
3. The section where we did matrix calculations in groups for the paralisation lecture was a bit tedious, and the outcome was always going to be obvious. Other than that, all parts were relevant. Maybe not for my particular field, but I can see a use for them, so it is important to keep them in.

Q: Are there further topics relevant to the programming scientist that could have been presented, given that the total time is limited

1. Data visualization lecture/exercises could be expanded.
2. ** Python3 Why should I care? Debate for python 2.7 vs 3.4 was really well done. But, some simple comparison between python2.7 and python3.4 should help to better understand the difference. Something highlighting the differences form both version could also help. Add some material, link or whatever that could help to write python2.7 code compatible with python3.4 (in the aim of a future evolution).
3. I would have profited from an overview over efficient data storage and retrieval.
4. pyOpenCL or pyCUDA for GPU computing
5. How to organize your own code in order to set your own library.
6. Would have liked more exploration of threads and parallelism. The detail presented was great, but I think I'm missing a whole portion of the possibilities. I'll let you know when I've digested this, in about a year!
7. ipython notebook? pandas?
8. I would have liked a (short?) overview of other, relevant, programming languages during the introduction. Just a summary of what's out there, what's being used and maybe cases where other tools are better suited, or Python is particularly well suited.
9. more advenced visualisation libraries like Mayavi
10. It could have been interesting to go further with data visualization. I think about vtk or hdf5 data files which are useful to visualize data like 2D/3D flow fields, neuronal connections etc ...
11. Organisation of the python project and code, using init, setup and config files. This doesn't need to be a separate lecture, but rather an introduction to naming/organizational conventions used when coding. From my perspective, it'd fit perfectly in Valentin's lecture.
12. How to make our work more accessible and usable by non-programmers / less skilled programmers. e.g. argparse, basic GUIs or how to make setuptools packages. Portability issues to ensure that code works on Windows/Max/Linux
13. More advanced visualization: how to visualize large datasets (tools and best practices?) , rather than Matplotlib API details, which one can figure out quite easily. Also, the latter is not very well suited for large problems (it is slow!) Scientific programming is also about saving/retrieving/querying data to disk. This issue is not particularly easy given dataset, especially regarding efficiency. Perhaps something about PyTables/Pandas?

Q: Do you think that pair-programming during the exercises was useful?

Yes, I have learned from my partner / I have helped my partner	79% (23)
No, it was a waste of time for both me and my partner	3% (1)
Neutral. It was OK, but I could have worked by myself as well.	14% (4)
Other	3% (1)

Other:

1. Yes, but it really depends of the partner sometimes

Q: What do you think of the balance between lectures and exercises? When answering, please keep in mind that the overall time is limited ;-)

Lectures were too long, there should be more time for exercises	11% (3)
Lectures were too short, there should be more time for lectures	0%
The time dedicated to lectures and exercises was well balanced	86% (24)
Other	4% (1)

Other:

1. Different in different circumstances, sometimes I felt the balance tipped too much for the lectures, other for the exercises

Q: Any further comments about the lectures and exercises?

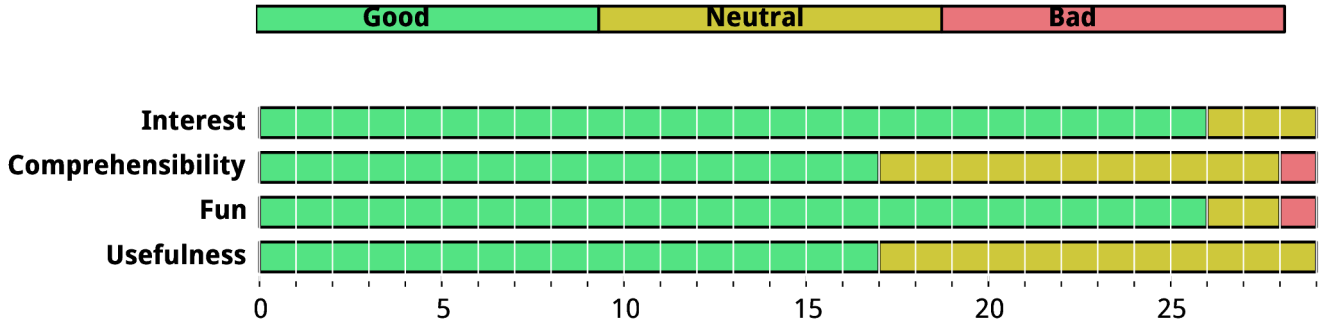
1. First, an apology: what follows is a bit of a rant; solet me state for the record that the course was very educaational, useful and enjoyable! I was taught a lot and learned some of it. I hope to start using what I've been taught in the very near future, and expect that most of the problems I had will become clearer with practice.That said: MORE _WORKING_ EXAMPLES!!! Many of the lecturers would gloss quickly over a demonstration with a code snippet, but in practice the out of context snippet was hard to use int he exercises until the whole surrounding structure could be implemented. I came in as a beginning Python user, so many times I had to refresh my my memory of the basic structure of python before I could understand how the example would actually work in context. I know much of the conceptual stuff from reading or other languages, but some of it really threw me because I never saw a concrete demonstration of how and why something worked. The whole section on decorators went by me like a TGV, leaving me, in the end, more confused than I started. I understand functions, basically get classes, but now I'm being thrown a function that's a class, or is it a class of functions, or WHAT? The provided computers, while nice enough, were less than useful, especially since we hadn't been given the background as to our working environment. I know the instructors have a variety of tasks, but many of us WILL be working in environments other than Linux, so perhaps using some of the Pelita-coding (sleeping) instructors a little support for our own environments and pre-class preparation could largely eliminate the need to stock so many computers and allow us to work in environments with which we're prepared and comfortable.
2. I think that the 1.5hr lectures were some time too long. Particularly in the afternoon sessions I dont think it was possible to concentrate properly for the whole lecture so maybe switch to 1hr lecture 1hr exercises for the afternoons.
3. In my personal experience, I was not able to complete all problem set per session.
4. I think lectures should have covered more live demos. Lectures dont help so much. Also Data visualizasion lecture, personally I was expecting more fancy things, there was not anything special in the lecture. If tutor could teach how exercises could be done in lecures that could help me more. Of course in one week we can not be good programmers in python but giving advise on how to deal with challenging issues could be usefull too. Thank you very much for nice and serious job. Yours Sincerely
5. I guess most scientist are more interested in speeding up algorithm than in in advanced programming structures like classes and metaclasses etc. However, I liked that there was the hint and a reference to the "Stop writing classes"-Talk by Jack Diederich from pycon US 2012. Most people I know have access to GPUs but no acccess to clusters, simply because a todays GPUs are a lot cheaper then a a bunch of computers. Therefore I think it could be interesting to have a lecture on pyOpenCL or pyCUDA (first one prefered)
6. It was really nice when the teachers showed interactively some examples to explain what they teach, before exercises. Concerning Python 3, it was really interesting to know the state of its development, but for the moment I think it is better to use Python 2.7 for the lectures and exercises. Concerning tutor's consultation time, maybe it could be useful to move this slot after lunch for example : I think we were too tired and too focus on exercises to have a good use of tutor's consultation at the ends of days!
7. There are many fantastic things about this school; first, the choice of topics is very good and it fits perfectly needs of a scientist. All of the lecturers are very enthusiastic about what they teach and are able to transmit this energy to students. Also, tutoring and interaction with students was well done. Trade-off between lectures and exercise is perfectly fine, but for me sometimes it was harder to concentrate during the second half of the day: e.g. data visualization was quite an interesting lecture, but I didn't have any brains left for the exercise. Thumbs up and keep up the good work :)
8. I generally profited a lot from all lectures. However, especially in the Advanced Python lectures I could have used a bit more motivation and examples. I gained most from the introduction of unittests (I already implemented it in my current project and it makes my life so much easier) and the tips about speeding up ones code.

9. Some of the lecturers did this, but I would have really liked it if we could have gone through some of the exercises as a class after we had made an attempt at them. I know that there were more exercises to give more advanced people more of a challenge, but it would have been nice to have had some set out that were the 'task' to finish before we went through them all together. The extra exercises could have been for the people who were extra keen who way prefer homework to fondue and beer.
10. I think some improvement can be made in presentation skills. Most of the lecturers are clear and concise. But few had a hard time projecting the voice and conveying information in a simple way. In some classes, the slides are not very helpful unless you are listening to what the speaker is saying. Would be nice if the slides were organized almost as tutorials, step by step, so it could be used later as good reference material. I would avoid having lectures after lunch (between 13 - 15:30) since during this time I was very braindead. I would prefer to be involved in some exercise during this time.
11. I think the lecture/exercise time balance was good, but maybe the structure could be slightly different. The lecture/exercise structure could be broken down a bit more so that there is less material presented before each exercise, i.e.: First part of lecture/First exercise THEN Second part of lecture/Second exercise
12. ** Matplotlib: Level of the talk was not well adapted to the large spread of level in the room. May be a more basic introduction could be done in few words, followed by a deth presentation of Matplotlib capacities (a gallery for e.g.). on the contrary, exercices were very interesting. ** Decorator: Using decorator is not easy and need an introduction or at least some base. For me the level was a little too hight. Add a more basic introduction could help low levelled people to follow the decorator. ++++ A very nice week. I'd like to attend for more Python courses to learn more on that fantastic language. It was really nice to speak with people Python addicted ;-) A fantastic organisation during the week help us to feel very well and enjoy this perfect week ! Thanks for all and continue!
13. It was hard to concentrate on the lecture just after 1.5 hours of exercises. In this case maybe it's better to make a short brake. I didn't have the problem with exercises, it is much easier to concentrate on the exercises even if they are just after the lecture.
14. Thanks, I learned a lot!
15. I would have liked to see more real-world examples, especially of design patterns where I find it hard to map classes based on animals or shapes onto scientific problems. Case studies of real-world use of design patterns and discussion of why this design pattern is better than possible alternatives would have been very interesting even if it was a bit more time-consuming than simplistic examples. Some participants did not know how to e.g. change the keyboard layout setting or enable SSH public key authentication for connecting to the git server. I think a few minutes dedicated to these would have made their time easier.
16. ASPP was so far the best summer school for which I ever attended!
17. In general, exercises were matching the course content quite well, and gave a nice way to get more familiar with the main lecture points. Of course, more time for them would have been a plus, but it couldn't fit in the given time. Lectures were interesting, and time went very fast, without time to be bored!
18. more excercises in working with Git (version control at all) and unittests (the whole week repeating would be great to get used to the techniques)
19. The lecture/exercises balance was very good. I would only vote for including the first exercise of the school a bit earlier, since the beginning was lectures only. Two random annotations: I think it would have been helpful to mention on the website that personal laptops are not required to do the exercises, and that people from outside Switzerland will probably need a power cord adapter. These were the only things I wish someone would have told me sooner ;)
20. Great teachers. I learned a lot and got a lot of motivation. Sometimes I would have liked to have more time for the exercises, but without cutting down the time for the lectures. So I guess this could be difficult to implement.

Programming Project

Q: Evaluate the programming project.

Interest: How interesting was the programming project?
Comprehensibility: How clear and comprehensible was the code and the available documentation? Was it easy to work on the programming project
Fun: Was it fun to work on the programming project?
Usefulness: Was it useful to work on the programming project? Do you think you may re-use what you learned?



Q: Do you think the team-programming experience is relevant to your work as a programming scientist?

Yes: 62% (18)
No: 38% (11)

Q: Do you think that the project should be about a real-world scientific problem instead of a video game?

Yes: 7% (2)
No: 93% (27)

Q: Any further comments about the programming project?

1. This experience was really nice and interesting.
2. The project was good but to be honest I did not enjoy to work with my group, but I think it makes part of life and not always things work out as you want. At the end was fun but I was not able to help simply because some of them did not listen.
3. Maybe it would be nice to have some hints or restrictions on how to structure the own group. I had the impression, also from later discussions, that it was hard for some to get into it.
4. I think it would be very difficult to find a real-world scientific problem that could be solved by a not-especially-experienced team in a few days where there isn't already code out there to solve it. We are better off not pretending to be doing something useful. Maybe you could use a different game every year to avoid giving the tutors an advantage...
5. more "intelligent" bots as enemies to test, for example the winner of last year.
6. I think it would help if in the beginning there would be some basic instructions on setting up the environment properly and explaining the structure (e.g. it took quite a long time to realize that the factory is called inside the __init__ file.
7. Unfortunately, I was ill for most of the programming project. But I think that it was a great idea. If anything, the only criticism I would have would be to have more of a mix of abilities. I realise that this can be difficult and that it is easier to assign people randomly, but I did feel that some of the groups were full of very experienced people and others weren't. In my group for example, there was no one that had any experience of git, and we spent the first day just trying (and failing!) to figure it out. Great idea to have a game though I thought... people do way too much science in their careers, so it's good to have a break from the whole mindset once in a while.
8. Our group sort of dissolved and didn't bond, resulting in branch dissolution and a rush to completion. All of our faults, really, as it took quite some time for some of us to even figure out how to make the project work!
9. In the project rather than dealing with something done, it could be better to start a work from zero and a project that is feasible in the given time.

10. A real-world scientific problem would ressemble very much to a video game, without the associated fun. Pelita includes shortest-path algorithms, tree browsing, and gives total freedom for the programmer to implement extra algorithms (among them the suggested Kalman filters, etc.) Given the limited time, it was hard to have all team members at the same level, and be efficient together. This may depend on teams, of course.
11. It would have been nice if the teams would have had the possibility to play against more advanced players for praticing purposes.
12. I was very satisfied with the project and the group I worked in. I liked that we already got most of the code, and in that way could focus on the interesting, algorithmic, part and not spend time on (mostly) annoying trying-to-make-it-run things. The group work went fine for most of the time.
13. Very cool project, and nicely prepared by slicing all kinds of examples into the exercises that would come in handy later. The scope / time limit seemed just right, and the given example bots are very useful.
14. I would like to have more time to be able to bring personal lab problems.
15. I thought the programming project was excellent. The fact that it was a video game gave it a competitive edge which I think increased the level of motivation and engagement. If you picked a real world scientific problem then this might appeal to some people but you would not be able to appeal to everybody so some people might think well this is not relevant to me so I won't bother so much. A video game is probably not relevant to anybody so nobody feels like it more or less of a waste of time for them.

The School in General

Q: How do you overall evaluate the school?

Good: 100% (29)
Neutral: 0% (0)
Bad: 0% (0)

Q: How do you evaluate the general level of the school? Was it too advanced/too basic with respect to* your expectations*?

Too advanced: 10% (3)
Just Right: 83% (24)
Too basic: 7% (2)

Q: How do you evaluate the general level of the school? Was it too advanced/too basic with respect to* what was advertised in the announcement*?

Too advanced: 3% (1)
Just Right: 90% (26)
Too basic: 7% (2)

Q: Did you learn more from attending the school than you would have learned from reading books and online tutorials alone?

Yes: 97% (28)
No: 3% (1)

Q: How do you evaluate social interactions and social activities at the school?

Good: 89% (25)
Neutral: 7% (2)
Bad: 4% (1)

Q: Would you recommend this course to other students and colleagues?

Yes: 100% (29)
No: 0% (0)

Q: How did you hear about the school?

Google Search: 3
Professor/Tutor/Supervisor: 4
Colleague/Friend: 15
Website/Mailing list: 8
of which:
scipy: 1
python: 1
bccn/g-node: 1
incf: 1
other: 4

Q: There might not be further editions of the school unless we find a way to make it a self-supporting event. Would you have attended the school even if a fee were introduced to cover the running costs?

Yes: 97% (28)
No: 3% (1)

Q: If yes, do you think a fee of about 200 € would be appropriate?

OK: 75% (21)
Too high: 14% (4)
May be higher!: 11% (3)

Q: Any further comments or suggestions?

1. I come from an education background (I was a maths teacher before I started my PhD) and I thought that overall the structure and content of the school was excellent. Most educational experiences not achieve such a high level of engagement and learning amongst the students. I think this is partly down to the short turn around between lectures and exercises in the first phase of the course but mainly due to the group project in the second phase. This part of the course provided so many opportunities to implement ideas from the lectures and achieved almost 100% engagement and effort from all students. How many students checked facebook or surfed the web during the group project time? I suspect that compared to a usual working day it was close to zero. Thank you.
2. Thank you so much for this school. Good luck with future editions!
3. Thank you for that great experience! It was really nice to exchange with experts on the things I use as tool everyday for many hours.
4. Good work, it was a really good school! :-)
5. The only reason that I put that it was too advanced was that I thought there was a little too much on how to optimise the code. I would have personally prefered more numpy and scipy based exercises as this is more what I plan to do with my daily work. I do realise however, that there are people who are more advanced and that they really benefit from learning this stuff and that it is really difficult to get a level that is going to be perfectly suited to everyone's needs. I also really liked the bit about the whole test-driven programming... I would have liked a bit more of that as I can see that it has real benefits for people, whatever they are doing. And when I spoke to many people in the social events, it seemed as though not many people were so used to doing this kind of work.
6. Since the time schedule is so tight, there's not much time left to see the city. It'd be nice to introduce some sort of one-day trip to rest our fingers and brains. Apart from that, everything was perfect, you guys rock, thanks :)
7. Great summer school. The absence of fee makes it easier for everybody, regardless of their origin, to attend it. It can be understand that a fee may need to be introduced. Would it really make a difference, or switch the summer school to something bigger and perhaps not as nice? Keep it simple please! Of course, a huge thank you to all people involved and for their precious time!
8. I learned quite a lot about Python, git etc through using and reading about them in the period between applying and attending the school. Nevertheless I found it helpful to use many of the techniques and packages I had previously only read about, with experienced tutors on hand to solve any novice problems.
9. A fee could be ok, it would definitely be worth 200euros, but it might also be good to offer some type of scholarship so that at least one candidate each year who truly can afford the fee would still be able to attend.
10. The 200,- seem a bit high to me but I have no idea about the costs you need to cover or what comparable events ask in terms of fees. On the other hand, most universities will probably cover it anyway, so it makes little difference to participating students I guess.
11. For PhD student paying fee should be difficult. Depending on the country you are coming from and if your lab can pay you.