School of Psychology
University of St Andrews

G-Node German Neuroinformatics Node www.g-node.org

Advanced Scientific Programming in Python

a Summer School by the G-Node and the

School of Psychology, University of St Andrews

September 11–16, 2011. St Andrews, UK

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 11, 2011.
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	19%	154	
Different nationalities	14		42	
States of affiliation	12		28	
Female	10	33%	28	18%
Undergraduates	2	7%	17	11%
PhD Students	18	60%	89	58%
Post-Docs	5	17%	23	15%
Professors	2	7%	4	3%
Others	3	10%	21	14%
Completed surveys	28	93%		

Lectures & Exercises

Q: Please grade the lectures and all lecturers according to the provided criteria. In case the lecture did not have exercises, just leave the corresponding field blank. Please refer to the schedule for the list of lectures.
[Grades: "Bad"=−1, "Neutral"=0, "Good"=+1]
The criteria are defined as follows:
Interest: How interesting was the subject of these lectures?
Comprehensibility: How clear and comprehensible was the material presented? Was it easy to follow the lecturer? Did he take time to answer students' questions?
Material&Exercises: Evaluate the quality of the teaching material provided by the lecturer, e.g. the clarity of the slides, references given, scripts, exercises etc.

	Interest	Comprehensibility	Material&Exercises	
Overall	0.93 (27)	0.74 (27)	0.89 (27)	0.85 (81)
Valentin Haenel Best Practices	0.93 (27)	0.78 (27)	0.62 (16)	0.80 (70)
Zbigniew Jędrzejewski-Szmek Advanced Python	0.74 (27)	−0.18 (28)	0.38 (26)	0.31 (81)
Bastian Venthur Git	0.78 (27)	0.85 (27)	0.62 (26)	0.75 (80)
Bartosz Teleńczuk OO Programming	0.71 (28)	0.68 (28)	0.77 (26)	0.72 (82)
Pietro Berkes Software Carpentry	0.93 (28)	0.82 (28)	0.78 (27)	0.84 (83)
Tiziano Zito Programming in Teams	0.61 (28)	0.57 (28)	0.17 (18)	0.49 (74)
Pauli Virtanen Advanced NumPy	0.93 (28)	0.57 (28)	0.81 (27)	0.77 (83)
Pauli Virtanen Cython	0.89 (28)	0.86 (28)	0.89 (27)	0.88 (83)
Bartosz Teleńczuk Data Visualization	0.61 (28)	0.71 (28)	0.33 (27)	0.55 (83)
Bastian Venthur Parallel Applications I	0.93 (28)	0.82 (28)	0.81 (26)	0.85 (82)
Francesc Alted Parallel Applications II	0.89 (28)	0.86 (27)	0.78 (27)	0.84 (82)
Rike Schuppner Programming Project Intro	0.86 (28)	0.67 (27)	0.67 (18)	0.74 (73)
Francesc Alted The Starving CPUs	0.79 (28)	0.82 (28)	0.56 (27)	0.73 (83)
Francesc Alted Data Persistence	0.68 (28)	0.64 (28)	0.57 (21)	0.64 (77)

Note: values in gray are computed as a weighted average over the grades in all criteria.

Q: Are in your opinion some of the topics presented in the lectures not relevant for a programming scientist? Leave blank if you think all topics were relevant.

1. -command line debugging-cpu starving
2. GUI programming
3. Some emphasize on working effectively as a single programmer was missing. In terms of use of GIT, unit testing and working with scripts (not interactive programs)
4. I thought the topics were all really useful potentially, but especially if you are a self-taught then you struggle a lot with the things like CPU and API and things, you might need to explain these and why they are important first.
5. All topics were relevant. Perhaps parallel was over-emphasized?
6. Advanced python, the Starving CPU and Data Presistance went into a lot of details which are not that useful for a scientist. To really understand these memory, databases or decorators a lot more time would be necessary. In all these cases a short overview should be given, such that we know that these things exist. (not how to implement or use them) People interested in these can then take a lecture/book dedicated on that topic.

7. It really depends. Many scientists do not strictly need OOP, decorators, parallel computing and git. It would have been nice to focus a bit more on where these concepts really help (especially w.r.t. to OOP). I don't mean this be offensive, I know how hard it is to teach this. Nevertheless I thought it might be worth saying.
8. Parallelization and optimization for speed is not relevant for me, but that doesn't mean that it's not relevant for programming scientists in general. Some of the advanced Python stuff presented by Zbigniew is probably also more advanced than I'll need every day, but still interesting. On the other hand, most of the object oriented stuff by Bartosz was so basic that I doubt that anyone who didn't know it already would be able to follow any of the other lectures.
9. particular students will find some bits more or less relevant (i.e. I'm unlikely to use Cython any time soon) but all topics useful to know something about
10. I guess the lecture on 'advanced python' could have covered a bit broader range of topics.
11. in general it was very interesting and useful but a bit too deep in the programming itself instead of on using it as a tool.

Q: Are there further topics relevant to the programming scientist that could have been presented?

1. Maybe some GPU-programming would be interesting, but I understand that it is hard to put in the tight schedule and I there is not one lecture that I would make shorter or even put out ...
2. -data persistance with relational databases (schema development)-personal productivity (tool chains, atomistic workflows)-(toy/surrogate) dataset generation-numpy/scipy receipts, api/sdk introduction (after all its called lalal in python, so teach the vocabulary along with the grammar :)
3. I would have loved to have more about programming. I also thought that for me without a computerscience background, the way how a software sould be developed (softwarearitecture) (modules, blackbox-idea etc) would be very importante.
4. Maybe something on accuracy of numerical representation / numerical hygiene? I.e. what can go wrong when storing and calculatting number (e.g. Doubles vs floats, rounding and casting, order of operations, etc.)Also maybe a bit more advice about choosing an editor / IDE that is apppropriate for you and/or the project/team.Could also be scope for mentioning using other flavours of Python such as Jython, IronPython etc. to help work with Java and MicroSoft based products (e.g. if lucky can sometimes choose IronPython rather than VBA/VBScript)
5. Just a few comments about:- documentation- ide's
6. 1)advanced text parsing with re module2)basics of gui developement3)basics on how expose datas/results on the web
7. I would have like more detailed instruction on the data vis part - instead of design principles for graphics, more meat on how to do it in Python - we were just left to do it by trial and error in the excercises
8. Clever forms of representing your data so you can quickly do the operations you want on them would be interesting, but I don't know if there are general solutions to this. Some tipps on how to use matplotlib interactively and efficiently for data exploration could have been interesting in the data visualization part.
9. I think a lot of the students had come from a Matlab background. I know this was about python, but I think it would have been very useful to briefly describe or include hints for Matlab implementation if possible and where not possible or a lot more complicated to point out the superiority or ease of python. That way I would know for what sort of problems should I consider switching to python.
10. Refactoring was mentioned but could be presented in more detail, also in light of for instance object-oriented programming.Because researchers often have a lot of legacy code that needs work.
11. A little bit more background - we don't all know in detail how a computer works, it would make more sense if a little less was assumed, like the terminology. If you just say what these are at the beginning it would help a lot.
12. Something more on software architecture for scientific programming? For example: how to structure code that deals with multiple small experiments, put it all in one big script? multiple scripts with common functions and classes? configuration files? dedicated computational experiment software (e.g. sumatra)? Something about this would be useful!
13. Although I do not want to start a war on which editor or EDI is the best, I think it might have been useful to have a short overview or comparison between different tools available for programming Python scripts. In my opinion, I think spyder should definitely be on that list as well: interactive python IDE which integrates tools as pyflakes, pep8, rope, etc.
14. Personally something on imaging processing, and tips on how to solve more practical (mathematical) issues one encounters when doing science with computers

Q: What do you think of the level of the "Advanced Python" lecture?

It was OK	36% (10)
It should be extended with more time for exercises	18% (5)
It should become an introduction to Python and advanced topics should be skipped	11% (3)
It should be skipped completely and its time slot should be used for the other lectures	4% (1)
Other	29% (8)

Other:

1. Give a bit more introduction, but include these advanced things also
2. Ok, but better given later in the course
3. I would move the lecture to a later day and add 10 minutes to motivate the different topics a bit more.
4. It just seemed a bit ouf of place and the presentation was poor
5. I would have put an intro to python on the first day (basics on functions, classes, modules, etc...) and then move this one to one of the last days, so everybody is more or less familiar with the python structure""
6. I didn't get the motivation for the generator part. Some introduction to why this is useful would be nice
7. it was interesting, but I'm not sure of also useful

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

Yes, I have learned from my partner / I have helped my partner	82% (23)
No, it was a waste of time for both me and my partner	0%
Neutral. It was OK, but I could have worked by myself as well.	7% (2)
Other	11% (3)

Other:

1. for me it was very useful, but it depends a lot on the person who works with you,
2. Sometimes it worked, sometimes it didn't, depending on who I worked with. When the communication is good it's a good idea, but else it's waste of time.
3. My partners have learnt from me, but I have learnt little from them. So I suppose it was OK.

Q: What do you think of the balance between lectures and exercises? Note that the exercise sheets were not meant to be finished in a single day: they were meant to offer a diverse set of tasks, among which every student may choose the ones they prefer. When answering, please keep in mind that the overall time is limited ;-)

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

Other:

1. I think the exercises should focus more on situations where this or that tool comes in handy. Maybe start the day with exercises before the lectures, then show how e.g. decorators solve the problem and then have some exercises about decorators.
2. The time dedicated to lectures and exercises was well balanced, except for parallel programmin 1, that was to fast and to short
3. Lectures were the appropriate length, maybe more time to do exercises

Q: Any further comments on lectures and exercises?

1. I enjoyed the course and learned a lot. For me the lecture about datavisualization was not so interessting, this could have been skipped.
2. More time for everything would have been great to rethink what has been taught in the lectures and to work (and finish) all the exercises.
3. The lecture on visualization was too short. I was expecting to see more visualization in action, and the conceptual part could be reduced.
4. Thanks for organizing this great course! I am looking into the possibility of getting my institute (Risø-DTU from Denmark) involved in this process. In some departments people are starting to switch to Python. I think this course would be of great value for a lot of my colleagues. Maybe there could be two Python courses a year: the free one we just had and a institutional one (with a paid lectures and fee from participants)? Anyway, I will need to do some lobbying at my institute first. To be continued.Regards,David
5. The pair programming was useful to me, but I feel I was a huge drain on the person I was working with. I found the course a huge struggle, it could be made a little easier if terminology is explained (even just include what these things are on the slides). I am not an expert in Python but I think you would need to be to get much out of the course the way it is at the moment.
6. I enjoyed the exercises and the pair programming very much. I learned the most from the pelita tournament and had a great time. A great summer school!
7. It was really well balanced
8. This was the first summer course I attended that actually had exercises, which was one of the main reasons for applying. I got to understand the concept while practicing a lot more through the applications than I would have otherwise. It is always hard to implement something if you don't have any examples, which is why it was excellent that the solutions were also posted with the exercises. Well done!
9. In all I found the topics covered a lot of area, which is good. There was enough to make progress in on the moment and enough to think about later on.Sometimes topics were quite specialized, (cpu starvation, visualization, cython) and there the main merit is to touch on the subject not so much use it right away. These lectures can do with less exercises, I think.
10. Overall I was very happy with the quality of the lectures and exercises.Just to avoid confusion about my ratings above: I think data visualization is a very important and interesting topic, maybe the most relevant for my day-to-day work. Only the lecture wasn't all that interesting .
11. Don't use "pass" in examples (where some real code should go), use real code or "[...]"*best practices*Very nice and helpful overview, a lot of material in very short time (hard to follow)*Advanced python*interesting but not that useful for everyday lifeespecially since it violates KIS/readability in any research environment (no one else will understand these things)good to know about these things if one encounters them in code*OOP*good lecture, but python is just slightly broken for proper OOP :-)*TDD*Very usefull introduction and nice summary on cheetsheet (necessary since a lot was live and not on slides)I was missing a examples/pointers for IDEs*Advanced NumPy*very relevant, since we will be migrating a lot of people from Matlabmaybe assign more time to this lecture and reduce "advanced python"*Programming in Teams Discussion*Messages not clear/concise enough to really be valuable. Everything remained vague and general knowledge. No Discussion happened, probably due to the way the ideas where presented (facts instead of open questions) and the organization of the group (frontal)*parallel*both talks very informative. good to see all the different options and tools that make life easier
12. I definitely learnt something from the Advanced Python lecture. For instance in the future I'm going to better understand any python code I look at with decorators, context managers etc., although doubt I'll be using them that my myself any time soon. However, in that lecture slot a quick reminder run through Python would probably have been more beneficial. Then perhaps with the concepts of the decorators, generatirs context managers etc. could have been worked in later in the week.Given that I work with several different languages I might not be able to put into practice the material presented that is very specific to oure Python as soon as I'd like. However the material on git, memory management, parallel computing, etc. that is also generally applicable to scientific computing is already proving useful. My own unfamiliarity with Debian and the available text editors etc. provided in the virtual machine a bit of a learning curve. Maybe a few more preinstalled editor/IDE options might have speeded things up for people like me.With my area of interest would have liked to have more visualization. The lecture Bartosz gave was all good advice very applicable to certain types of datasets. There's potential for looking at other aspects of visualization that such as "outreach" where the viewer might not be a specialist in the particular area of science from which the data came. I also saw there was a slide at the end on "interactivity" - I assume that if the lecture slot was longer Bartosz might have gone on to look at things like MayaVi and VTK or brushing, linking, across multiple views etc. ?
13. the "Advanced Python" lecture was not well integrated with the rest. If possible, references to advanced topics shoul be given but before you need to explain why and when these topics are usefull.lesson on data visualization was too much generic. A more detailed introduction on matplotlib environment (tricks also) woul be more usefull.
14. -when I think about the pytables lecture today, it still recalls as a marketing talk :) not very instructive, much evaluation.
15. Good job!

Programming Project

Q: Please evaluate the programming project.
[Grades: “Bad”=−1, “Neutral”=0, “Good”=+1]
The criteria are defined as follows:
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?

Interest	Comprehensibility	Fun	Usefulness	
0.93 (27)	0.74 (27)	0.89 (27)	0.74 (27)	0.83 (108)

Note: values in gray are computed as weighted average over the grades in all criteria.

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

Yes: 79% (22)
No: 21% (6)

Q: Do you think the inter-group competition inspired by the tournament had a detrimental influence on your experience at the school?

Yes: 29% (8)
No: 71% (20)

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

Yes: 11% (3)
No: 89% (25)

Q: Any further comments on the programming project?

1. Time available for exercise is understandably constrainted by length of the course. So given small size of task that is feasible within the framework individuals or pairs might actually have better chance of producing a "better" bot than some teams working together. That might not have helped promote the teamwork aspect.However, personally I was quite happy to use the exercise to try working in a new team, practise using the material/techniques presented to us during the week and to take the opportunity to learn things from team mates with more experience in certain areas (e.g. driving git)Exercise did help convince me of the benefits of TDD as our unit tests said "save_moves" function was OK and eventually turned out bug was indeed elsewhere.
2. I loved it. I already started programming a new bot!!!!! And organized a meeting with some others for the next tournament....
3. The programming project was wonderful. In my opinion, it brought together many aspects of python that I didn't know much about and really helped me to learn in a real way.
4. Team-programming would be a nice thing for my work but I think it's not gonna happen for different reasons. The main reason is that I am lacking a partner with whom I could programm. I furthermore believe that you have to find an equal partner. Teaching someone along the way is very difficult.
5. It was useful in the sense that I got acquainted with python a lot better by doing any kind of project. This exact game had not much to do with my work but was a lot of fun and it was inspiring. I wish we had a lot more time to do it, because in the beginning it was extremely hard to start anything since I was totally new to this language.
6. The project was a lot of fun and working as a team was a nice experience. Unfortunately there was no time to really get the hang of the best practices we learned. We sacrificed good design, refactoring, frequent swapping of pairs and other things for speed.It might be good to let people think about how to organize their work earlier in the week (but no programming yet). In addition it might help to get the teams to work faster, if some hints about what might be needed (and the relevant test framework) were provided at the beginning.But all in all the project was great.:
7. in the documentation a section on unit testing would be usefull and in particular some examples on how to test single features of the players.

8. I have the feeling that the outcome of the tournament depended very little on the programming skills of team members or the quality of within-team communication. I don't know if that is very important, but it is something I felt I might mention. Since teams do not know this in advance, it is probably of little practical relevance during the project.It would have been nice to have the API-reference online before the start of the tournament, but on the other hand, I think it is a good experience to learn to work with code that is not perfectly documented online, since this is often necessary when using 3rd party libraries. In that regard pelita may even be too well documented ;)
9. where are the results/replays of the competition?
10. It would be great if after the tournament, all teams would have half a day more to improve their code and then repeat the competition.Or at least we could hear about some strategies used for bots.
11. Although I would have like a "real-world" scientific topic for the programming project, it might be hard to identify a topic that is not trivial, does not skew the individual interest and enthusiasm with the project. Altogether the "pelita" game is a good sandbox to applied the concepts learned in the workshop so far.some impressions:In my team we really had a huge issue to bring everybody on a level where you could discuss about actually implementing. One member conceived from the start, two others secluded themselves in unnecessary tasks and I tried to help the remaining two to actively learn during the programming experience.Most of the concepts taught in the course so far were not recognized in the pelita code, so we started over at "What is this @property thing in here.." and managed to arrive at using list comprehensions and fancy indexing that they came up with on their own. So, they have learned a lot for sure.outlook:Judging from the experience in my group and from conversations, one of the topics not covered in the summer schools the meta-task to transition an abstract problem formulation into a code skeleton. A repetitive story was "I have this mathematical problem solved on my whiteboard, but I do not manage to come up with a program." Mostly this is true for participants without a strong programming background of course. My impression is also, that a large part of the problem is that they don't know the options of the scientific python core solution stac (e.g. scipy, numpy and friends). My suggested cure would be to teach a set of atomistic workflows, what can be chained an stacked and improve familiarity with the core packages.
12. We had some different understandings about team programming in our group. It seemed to be different in the other groups, but I think our group would have worked a lot better if there were more constraints about things like splitting the project up into parts and how to define a design so we could divide the work.With constraints I would mean that you would have to come up, like with a design after the first day, split it up into subprojects and stick to the roles. On the other hand I though that it created a nice atmosphere that the tutors were there to help and not to set constraints, so hard to tell ...I really enjoyed the video as a highly motivating task, but it was not easyly split up into subprojects. Maybe a video game with different characters would solve that ... Than it would be easier to organize the group, because the subtasks would be clearer.
13. I answer yes on the teamworking question, mainly because I for see it's use in the nearby future, not so much the present.It might be emphasized some more in the lectures, that the computerization of research will continue and will in the near future involve groups of programmers and not individuals.
14. I think I learned the most on the programming project - once you see some working code and start using GIT and things then it all starts making a lot more sense. This is due to my teammates for a large part - we had a couple of very good people who were patient enough to explain these things to us.
15. Comment on "Do you think that the project should be about a real-world scientific problem instead of a video game?"Difficult question. I agree with the arguments you put forward at the course: needs to be fun, competition aspect and due to the great varied of backgrounds it will be difficult to find a real-world scientific problem. One could also argue that creating an AI is actually a real-world scientific problem. Due to the limited time it will be difficult to integrate a more complex scientific problem. The time required to start grasping the problem at hand will outrun the time spent on the programming. We already spent a lot of time in understanding the API and all available objects. I realise pelita is an open source project with limited funding, but it would have helped to have more comprehensive documentation.
16. I really like the idea of this programming project and its competitive character. I wish we had a bit more time to additionally use some of the concepts we dealt with during the lectures and exercises.
17. Divide the students into more teams
18. it was very funny, very didactic and very complete. I loved it

The School in General

Q: How do you overall evaluate the school?

Good: 96% (27)
Neutral: 4% (1)
Bad: 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: 7% (2)
Right: 86% (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: 11% (3)
Right: 75% (21)
Too basic: 14% (4)

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

Yes: 93% (26)
No: 7% (2)

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

Good: 89% (25)
Neutral: 11% (3)
Bad: 0%

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

Yes: 100% (27)
No: 0%

Q: How did you hear about the school?

Google Search: 2
Professor/Tutor/Supervisor: 5
Colleague/Friend: 12
Website/Mailing list: 16
  of which:
    connectionists/comp-neuro/ML-news: 5
    python: 1
    debian: 3
    bccn/g-node: 2
    other: 5

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: 86% (24)
No: 14% (4)

Q: If yes, do you think a fee of about 150 € would be appropriate?

OK: 50% (12)
Too high: 8% (2)
May be higher!: 42% (10)

Q: Any further comments or suggestions?

1. The social interactions between students was good.Between students and tutors this can be improved. Tutors should show more interest in the students. Listen to questions, etc.Socially mixing between tutors and students might be a strong factor in this. This might be improved in future editions, possibly through a more interactive opening night.
2. Thanks for organizing this. Even though some of my feedback may sound very negative and critical, that is mainly because I don't think just saying "it was all so great" is very helpful for improving the next edition. So I'd like to let you know that I liked the school, escpecially the lectures by Pauli and Pietro were excellent in my opinion.
3. I would have attended for a fee if it was implemented in Matlab. I did not get any funding for this school unfortunately, because I could not convince my supervisor that this was going to be beneficial for my work. It might have been otherwise, had there been Matlab applications in it.About the level of the school: I was shocked at the very first day with the Advanced Python lecture, because I had not much to build on. I think, it would have been a better structure for that lecture to give examples where the introduced structures make programming easier and then talk about them one by one, so we know what it is used for and what it does. The lecture might have explained it but the message didn't get through. The rest of the topics were also mostly new to me, but the structure of the lectures made it perfectly understandable.
4. Thanks for a great course! I really enjoyed it, both the programming part and the social part. It would be nice to have the option of common lodging organized by the school. This would help on the social part, but I guess it's hard to find a suitable place.
5. I might have paid if it was in California!I would think it would be a valuable experience for most grad-students/postdocs in Computational neuroscience
6. Start the school on say wednesday with the "basic" advanced lectures. Then have the weekend off to let this sink in, form the programming groups, practise a bit what has been learned, and then continue next week with the "advanced" advanced lectures and the project.
7. open the feedback form at the start (then it could be filled out immediately after the lecture). The way it is now I had to try and reconstruct my impression from the first days.would be nice to make the school longer (eg. 12 days), provide more time for everything (and especially exercises); but ok this way as well, we can invest another week after school to go through all the interesting stuff in more detail.I prefer schools with a fee but everything included (everybody living at the same place close to the lectures)
8. I learned a lot on the course, mostly through my partner and the project. I found the lectures a real struggle. I think in its current format it needs to be advertised that you have to be advanced in Python, not just have programming experience in another language and have done the online tutorial. If you were already very familiar with Python, then I would totaly recommend this course, it would be great for you. If you have some programming experience, but not so much in Python, then I would not recommend this course, it will be too hard.
9. Overall lots of fun and very useful. Nice to be in the role of student again evne if just for a week.I've already fought with LF vs CRLF and .gitignore and managed to get git working for my long runnning Jython/Java projects. So having attended the course is already paying benefits. I've another potential upcoming project for which Python/PyTables and Matplotlib might well be a good fit. So was extremely good to get some tutor consultation time with Francesc to discuss how feasible he thought a proposed archetecture might be and in particular how suitable PyTables might be for working with long time series datasets with lots of ragged arrays.Thanks very much to all the tutors and organisers.
10. I would have paid that sort of fee without any issues, (especially since I found it so useful) but I would understand if supervisors of PhD students in particular found a fee in addition to transport and accommodation costs was too much.
11. Keep the school going! I think €150 is a fair fee to pay.
12. I think the fee is not that expensive, however it depends on where you are organizing the school: UK was/is expensive so it was cool that there wasn't a fee.
13. It was a great course for getting inspired which advanced programming techniques I can use for improving my current and future work. I really enjoyed the time regarding to the course as well as the social activities. I expected a lot of this course, regarding to the announced topics, and I was not disappointed. Thumbs up!!!
14. The school was a bit advanced for my level but that is always better than being too basic. I learned a lot and what is more important one gets the idea of the kind of things that can be done, and once you know this you know that you can find the "practical" way of doing it. Also, getting in contact with the tutors is extremely valuable in my opinion.I would have paid a larger amount than 150 euro but then it really needs to become a bit more practical, meaning deal with more daily scientific issues. I don't know if adding more "tutor time", or getting information from the students about what are they doing to try to focus the lectures a bit more on what people is actually working with. Personally I had a great time and it was a very good an useful experience. I would definitely recommend it to my colleagues and I may attend again :)
15. carry on!
16. for me it was a really good experience. I learnt a lot (including that I'm not that bad in programing as I thought) ....