Advanced Scientific Programming in Python

a Summer School by the G-Node and the Centre for Integrative Neuroscience and Neurodynamics, School of Psychology and Clinical Language Sciences, University of Reading, UK

September 5 – 10, 2016. Reading, 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 10, 2016.
No answer was mandatory.
The free-text answers have not been edited and are presented in their original form, including typos.

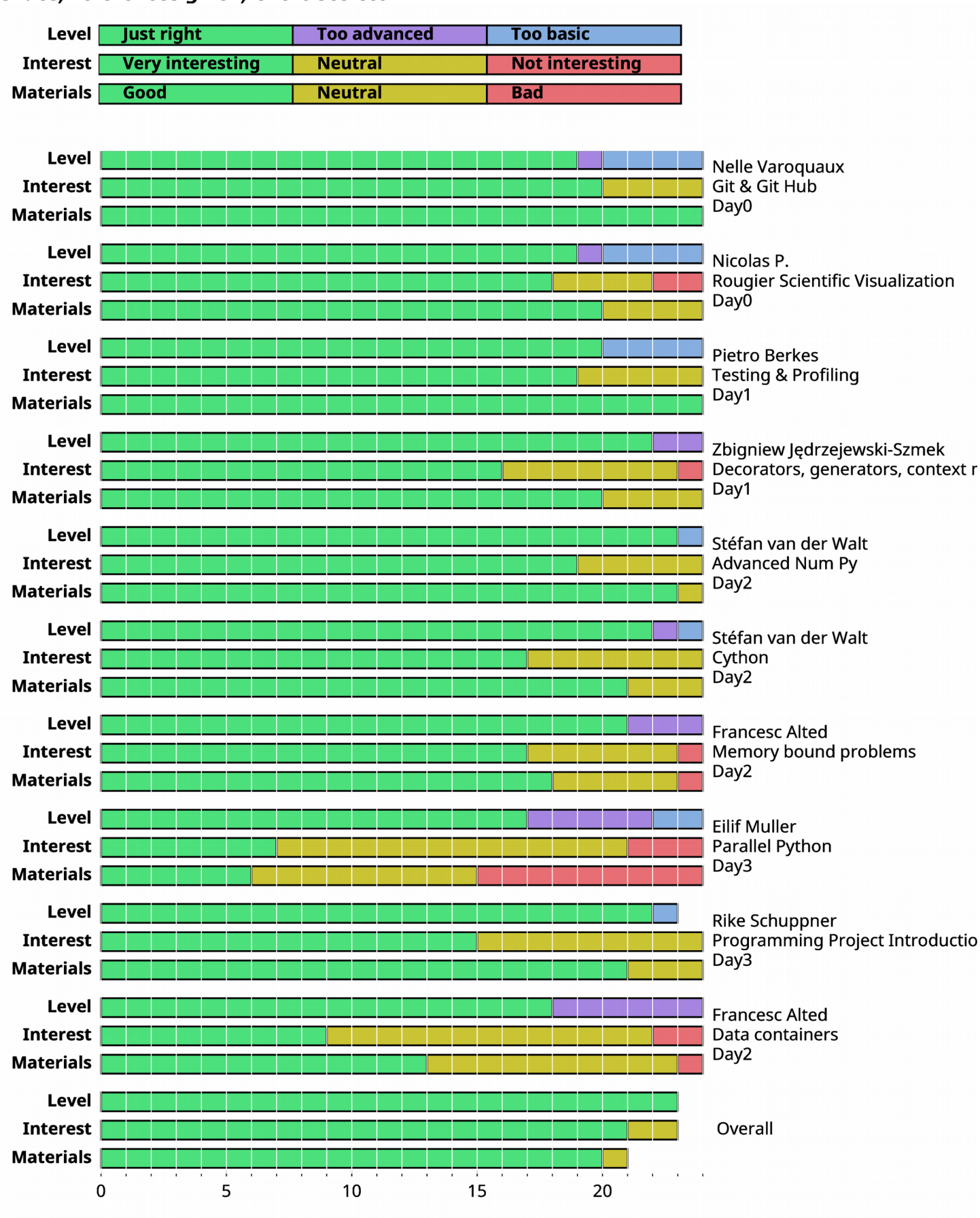
Attendants and Applicants Statistics

	Attendants		Applicants	
	28	9%	303	
Different nationalities	18		54	
Countries of affiliation	7		37	
Female	13	46%	101	33%
Male	15	54%	202	67%
Already applied	10	36%	47	16%
Bachelor Student	0	0%	6	2%
Master Student	1	4%	31	10%
PhD Students	20	71%	177	58%
Post-Docs	4	14%	46	16%
Professor	1	4%	3	1%
Technician	0	0%	1	0%
Employee	0	0%	17	6%
Others	2	7%	19	6%
Completed surveys	24	86%		

More stats about attendants are available at: https://python.g-node.org/python-summerschool-2016/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 did not see the immediate applicability of decorators onto science code during the class, but I enjoyed the the lecture very much.
2. While being very interesting from an educational point of view, the "memory use profiling" could maybe be shortened (in case more time is needed to have a slot for other stuff). I didn't really see the practical application of this, given that python memory usage seemed to be very hard to control anyway. For what I usually try to do, I'm not very interested in how much memory my object needs exactly. It's rather just generally about 'what to do if I don't have enough'.
3. After the lectures I do not see the direct relevance of the data containers lectures. It would be nice to have a more hands-on lecture on one topic (e.g. crashcourse pandas).
4. I felt like going into the details of how Python handles memory might be too much. The other approaches we learned about seem much more significant in terms of improving on performance that I wouldn't try to mess with the pointers.
5. Decorators, generators, context managers
6. All topics were highly relevant taking us from a good starting point to quite advanced. My only slight criticism: although I really enjoyed Francesc Alted's topics and his immense knowledge and deep understanding, I found his 2nd day (Data containers) quite challenging, mainly due to problems accessing the server. Maybe for his 2nd day, Francesc can focus on delivering a demo and allocating more time responding to questions (which he does incredibly well) rather than hands-on exercises.
7. Almost all topics were very well chosen and extremely relevant. Only at some point during the Advanced Numpy Lecture I wasn't sure if the advanced slicing and indexing are something we can apply or if it was more about trivia.

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

1. Data management and organization. We learn how to process data efficiently, but not how to organize it in an efficient way.
2. Maybe a bit more on actual software engineering with respect to larger (planning) projects would have been nice.
3. Besides Cython, I feel it would be interesting to know the interfacing with R, JavaScript, Julia etc. or even a simple subprocess.call()
4. I would have liked a more specific lecture about data containers where we are taught how to work with a specific data container. Maybe also something on how to find the right packages/write packages so that others can use it.
5. Data processing pipelines
6. Simple tasks for parallel python. I believe most of us need to learn how to write a simple script to send a simple file to run in a cluster before understanding more complex libraries and jobs management.
7. Data structures, algorithms
8. I feel it was all brilliant and well-designed with very thoughtful topics, experienced tutors and overall great fun.
9. - Unit testing - debugging was originally in the schedule, but due to time limitations was not covered. Though all the other topics were also interesting, I would have found this more useful than others like Parallel Python, Data containers or cython.
10. Programming more specific to running experiments - correctly storing results, presenting stimuli, synchronizing several software/hardware etc.
11. How to organise code, file structures, gpu parallel processing, clean code good practice guide, collaboration strategies (would tie in wit git).
12. OOP
13. A separate introduction to commonly used bash comamnd line tools and e.g. jupyter notebooks (magic functions) could have been a good addition. But most of these things were anyway mentioned on the side, so it was fine :)
14. Perhaps a little bit of debugging, or a quick overview of pros and cons of different IDEs.
15. Topics were perfectly chosen, but I would probably rethink the specific contents of the lectures about visualization and parelization. Also the two talks of Francesc could probably be merged to be more focused and less redundant in the theoric content, and extended with better exercises.
16. In the visualisation part, maybe there could be a bit more about the general structure and idea behind matplotlib (and other packages? Seaborn?). In the end, the thousand possible ways of how to plot something (regarding syntax and package) still seemed to be confusing to me and others. And what about a bit of xarray and dask in the data containers / memory bound probs lecture? I found these quite useful in combination and would've liked to hear a bit about use cases and potential tips and tricks ;)

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

Yes, I have learned from my partner / I have helped my partner	92% (22)
No, it was a waste of time for both me and my partner	0% (0)
Neutral. It was OK, but I could have worked by myself as well.	4% (1)
Other	4% (1)

Other: I would say neutral, but working in pairs had also 'social' function.

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	4% (1)
Lectures were too short, there should be more time for lectures	0%
The time dedicated to lectures and exercises was well balanced	92% (22)
Other	4% (1)

Other: The time was well balanced, but some exercises were lacking a solution and explanation of the exercise, befor continuing with the next one.

Q: Any further comments about the lectures and exercises?

1. Preferred the exercises that required a little bit of thinking over those which were just read-understand-click-and-see.
2. For git a typical teamwork-workflow with detailed examples might have been useful.
3. I liked the pair programming a lot, it helps to get in contact with all the participants and see hoe. At certain times during lectures I was not sure if I should just listen to the lecture or follow along with the coding. In my opinion this should be stated very clearly by all lecturers.
4. Overall excellent, although some times it was challenging to get remote servers up and running. For such exercises requiring remote servers, tutors should be more alert to potential problems/issues.
5. As I said above, the are three topics that I found not very well focused:

   -Visualization. The talk was well prepared and performed, and interesting in general, but for this course I think that a reflection on visualization aesthetics is not so relevant. Probably more advanced topics of toolkits (or just discussing and using the main capabilities of the different topics), like 3D animation, integration with QT, interactive plotting, web plotting... are much more in focus for a programming course.

   -Parallel programming: I use it everyday, so this obviously essential for scientist, but the talk was a real waste of time. More like Amazon advertisement. We could perfectly focus on working locally using two or three tools (threading in Cython, multiprocessing, Theano...), because this is an environment that we will all have in our work. Clustering is usually dependent on the infrastructure of each lab, and is usually assisted by the in house technical staff anyway. So not much point in trying to get into the details of any particular system (much less Amazon). If anything about distributed computing a more agnostic and barebones presentation of mpi is of general interest (i don't think many people use ipython notebooks for working on mpi clusters, and this also locks you into magical commands).

   -Memory and data containers: The topic was essential also, but there was quite a lot of redundancy between these talks, and the exercises were mostly about checking out that there was actually boundaries in the machines. So merging the theory of both, and focusing on methods to detect and solve these issues would make them much more useful.

   As a general comment, I think that practice should have a more central role, and the programming project could be used exactly for that. If we start with the project from day one, and interleave the contents on the classes in the morning with applying them to the work in the project in the afternoon, that would give much more coherence to the talks, and help fixating the concepts. In my previous attempt to fill this form I did a detailed description of a possible schedule (just to show this idea a bit more), but this is too much now. If you are interested in this idea we can discuss it by email. It would be a lot of additional work for the teachers, so I understand that it may not be feasible.

   Just to comment a bit, both git and testing should have more weight and be the central skeleton of the course through the programming project. The rest of the talks are more about specific topics all of them relevant, but much less about attitudes and work-flow. Specific topics can be learned reading a tutorial and using the tools, but attitudes and work-flow...well, that's totally another beast.

   I also missed other software engineering topics, like discussing about python anti-patterns, good abstractions

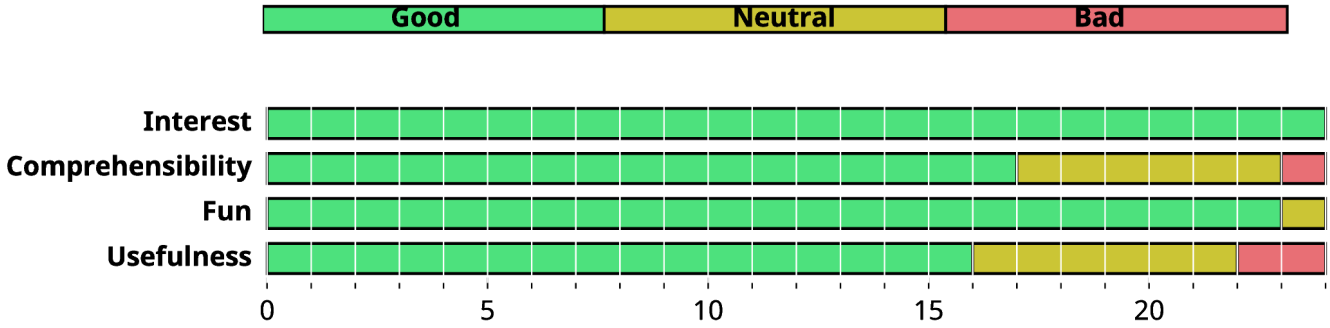
to build efficient libraries, and this kind of stuff.

6. I actually find it good that there were (a couple) more exercises than time allowed for. For me, that means I can still take a look at the topics I found more interesting or relevant.
7. I really liked the structure of most lectures where we learned some theory and immediately used it in the exercises. One general comment is that there should be enough time for each group to do the exercises and think about them before the lecture moves on. In most lectures this was okay, but in some we just had time to run the code provided and did not have time to discuss.
8. When coming to the Data containers lecture, I was expecting to hear more about data structures. The lecture was mostly about how the memory is organised, not much about how to handle data. I enjoyed the ratio of lectures/exercises - it's very good to do something on your own, very soon after you've heard of it. I wish there were no problems during the lecture about parallelism, as I really wanted to learn, how to do that, but because of the chaos, I do not remember much.
9. In a few lectures, we spend more time trying to set up the system (for example, setting up connections) than actually doing the exercises. Most problems that appeared were far too difficult to be solved by the average student, and I ended up confused and just copying/pasting lines of code without understanding what I was doing (but I had to do it in order to quickly continue with the lecture). Besides that, most lectures had a great balance between lecture time and exercise time. Last, maybe 8:30 is too early to start?
10. Git commands are hard to remember, I think a GUI would would be helpful for students that do not familiar with command-line. The ipcluster and the px magic could be more emphasized.
11. Enjoyed the week :) all of the helpers were lovely and always willing to help.
12. My overall impression of this course, including all the lectures and exercises was very good. Well done! The parallel computing exercises were suffering a bit from the problems with the cluster. As far as I understood, most of it was related to a shaky cluster, so not much you could do about it, maybe you have more luck with cluster next time. What could have been explained a bit better is how running an ipython notebook on a remote server, then opening a tunnel on a port and displaying in a local browser works. Especially because this might be a very useful workflow for people working with clusters. It would be nice to have the tutors' alloted time for general questions (session dedicated to students' own research questions) expanded a little bit.

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: 100% (24)
No: 0% (0)

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

Yes: 4% (1)
No: 96% (22)

Q: Any further comments about the programming project?

1. As explained in the previous page, for me it should be the conducting thread of the course, and the software engineering topics should appear along the week and be used for the project. It's true that as it is, the project does not really allow to use the scientific toolkits and techniques learned during the 'domain specific' classes. This is an issue because then these talks would be less integrated but there are ways to find workarounds. However, the pelita software leaves a challenge for the students that is very specific, and defined within a programming environment quite well developed (which further reduces the need to deal with uninteresting stuff). So I would say that it makes a lot sense to help focusing on interesting problems and practices explained in the classes. The environment requires a bit of time to get used to it, which also reinforces the idea of starting the first day with the project. With six whole days, the morning could be used for two talks with the bare minimum exercises to make sense of the concepts explained, and a well planned use of them in project during the afternoon. That allows for 12 2-hour talks in total, which is quite close to the current amount. Considering this two big talks per day scheme, a simplified schedule would be:

   -Monday: Git / Unit testing-Debugging

   -Tuesday: Git / Unit testing-Debugging

   -Wednesday: Decorators / Antipatterns-Software Engineering,

   -Thursday: Numpy / Cython

   -Friday: Boundaries / Parallel

   -Saturday: Visualization / ? Profit :)
2. I think it would be more useful if people concentrate on the team programming and use of programming tools we were learning (e.g. git, testing), than on the score of the game. Not sure if you can achieve it easily with the video game (some people are extremely competitive), but some extra encourage from the tutors might help.
3. The documentation for pelita could be improved, but that is now of course also our responsibility.
4. The programming project was really fun. But also a problem with a known solution would be fun, and it would be easier to code to a fixed goal.
5. I enjoyed the project very much! And I think it's important to learn, how to organise a group to work together, so that everyone does something useful. In our group in the end it didn't really work this way, but that's how we learn!
6. In the pelita project we made the initial mistake to work on one file and it became pretty confusing:

Everybody was pushing something all the time and I was somewhat stressed by having to keep my stuff up-to-date and merging conflicts (in terms of 'lesson-learned' this was excellent of course :)). I guess for me it would have been useful to have a bit more guidance in the first steps of teamwork organization and distributing work wisely. In the end we all worked more or less on separate machines and spent the most time fiddling with git. But still it was fun!

7. I appreciate this is the tricky one where people may have strong views. Beforehand, I thought that a game, as opposed to a real-world problem would be a waste of time. Having done it, I realise that "it was all about the Journey, not the Destination". I found the team-work quite challenging and I think we could do with some more support. The tutors were checking out progress a couple of times per day, but our team would have benefited from closer "supervision". No complains though: I learned a lot from the video game problem!
8. Each team should have a chance to find and fix their bugs. It could be fun to fight with last year's players.
9. In my opinion the time given for the project, the task as well as the size of the team was well balanced. Also, the fact that there were no instructions on how to organize the work within a team was good, because it gave us the opportunity to figure this out by ourself. I feel like I have learned a lot about how to work in teams and also what not to do. Overall, this was an excellent experience.
10. It's a great idea and was very interesting. Keep the same project.
11. I think it's a really fun game, and I learned a lot. It might be helpful to introduce development strategies, especially for the team work. Our team didn't really have a strategy for development, we didn't make UML diagrams/planned the structure of the code. A small lecture on that may be useful so that people could try it on the team project.
12. Would benefit from a examples of how "proper programmers" collaborate with one another, so when teams decide how to work, they have some guidance.
13. The first lectures were more relevant for the programming project. Topics of the advanced lectures were not needed for the programming project.
14. I really enjoyed the project. I think the game is interesting, challenging, but not too difficult, such that we were able to think about and concentrate the tools (such as Git, testing, etc.) rather than get stuck understanding some niche area of science, or a complex algorithm.
15. The project was great, but a few slides about "how to work in groups" might be useful to us all :)

The School in General

Q: How do you overall evaluate the school?

Good: 100% (24)
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: 0% (0)
Just Right: 96% (23)
Too basic: 4% (1)

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: 4% (1)
Just Right: 92% (22)
Too basic: 4% (1)

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

Yes: 92% (22)
No: 8% (2)

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

Good: 100% (24)
Neutral: 0% (0)
Bad: 0% (0)

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

Yes: 100% (24)
No: 0% (0)

Q: How did you hear about the school?

Google Search: 2
Professor/Tutor/Supervisor: 6
Colleague/Friend: 10
Mailing list: 9

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: 92% (22)
No: 8% (2)

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

OK: 86% (19)
Too high: 5% (1)
May be higher!: 9% (2)

Q: Any further comments or suggestions?

1. Notify students when the accommodation could be an issue. No pillow for the first night was a terrible experience.
2. The school was of the exact level I was looking for! And it's not easy to find one like that. Very often summer schools and courses are either too simple or too hard. This one was exactly to the point! Thanks!
3. You are obviously becoming very popular, perhaps consider splitting into 2 events, possibly run in consecutive weeks, splitting the topics. A few of the people I spoke to were confident with some of the topics.
4. I think the content received at the school is worth MUCH more than 200 EUR (in a perfect capitalistic world where everything has a price, whatever that means), on the other hand (fortunately, that's not the world we live in (yet)) this could leave out people coming from less capitalistic countries. There may be a solution where people who can, contribute with a fee, and people economically limited, apply for a grant. Maintaining the balance between these two has to be taken care, so that money doesn't impose a bias into the (already successful and super optimised) selection algorithm.
5. The level of most lectures was just right for me. However, some of them were concentrated too much on basic usage and we didn't have enough time to explore more advance features (e.g. git, pytest). Parallel Python is hard to grade, at the end the tutor spent most of the time talking to and solving technical problems of individual groups.
6. I hope this project exists for many more years! This course is among the most well organized ones I have seen. Keep up the good work!
7. I would say the problem with the level of the classes is that very often the first 30 minutes was used to repeat the most basic concepts of the topic, even if we were supposed to know them from the introductory material. This reduced the amount of advanced concepts that could be explained. But in the announcement was clear that the level of average, so probably fits with its own definition. Also very important was to be able to interact with people active in the python ecosystem. We probably never go to programming events to meet you guys, and it is great to put face and voice to the people that contribute to our tools. This advances in the sense of belonging to a community, which is probably one of the best outcomes that I had from the school. About the money, I would say a general fee of 300-400 would be OK because many people could obtain funding from their projects/institutions, and is money that the institutions should be spending in formation already. However, the fee should be scaled in way that allows at least one third of people to attend for free, those who cannot obtain funding in any other way. Controlling who receives this maybe a bit complicated, and open to discussion.
8. 200 euros would be fine if it included the food as well. As the campus was far from the city center, dinner options were very limited and we lost a lot of time just looking for places to eat in the evening. Also, it would be nice if the accommodations could offer bedding and pillows. Carrying those is extremely difficult due to luggage limits in the flights.
9. Great opportunity, thanks to all who organized it, I learnt a lot, and have already starting to update my code to be better!
10. Thank you very much for organizing this course. I learned many things and almost all of them are directly relevant to my PhD. I will try to use and practice all of these things and make others do that around me as well. Maybe leaving some more time for people to work on their own (such as PhD) problems with help could be helpful, or ask people to bring a real problem to solve for themselves that they can work on the whole week with guidance? I think the course is good as it is. :)
11. I had an awesome time and really felt at home with the crowd you selected: So many very different people but most of them extremely likable and interesting. Well done!
12. This School was brilliant, no doubt about it! For future schools it is essential that you keep the number at around 30, unless you can recruit many more tutors. I appreciate this will disappoint some 300 applicants, but the only other alternative is to run it twice a year. Regarding a fee: I am in support of having some sort of fee if the viability of the School depends on this. I would have happily paid up to €300 for the School, but this is because I can afford it and I am aware that not everyone can. I would like to propose some sort of gradual means-tested fee, where PhD/early researchers pay €100-150 and mid/advanced career pay €250-300, with a few bursaries for outstanding applicants who cannot afford it.
13. Overall, I both enjoyed the course and learned a lot. The team work was perfect and the project later on fun. Great team and thank you so much. The one improvement idea I would have is to maybe link the different lectures a bit more. For example, by shared exercises - a lot focussed on improving speed. If they all used the same basic structure of the code in their exercises, they would be more relatable to each other. People could even share their times online in a spreadsheet on github or so to have an element of repetition and interleaved learning, maybe ending in an overall exercise to make that example run as quickly as possible by using the techniques learned over the last day. The game was a lot of fun and a great way to make us interact with github again but not many of the other techniques were used. It would have been nice to come back principles introduced at earlier points at the end of the lecture.