12th Advanced Scientific Programming in Python

a Summer School by the G-Node and the University of Camerino

2–7 September, 2019. Camerino, Italy

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 10 October, 2019.
No answer was mandatory.
The free-text answers have not been edited and are presented in their original form, including typos.

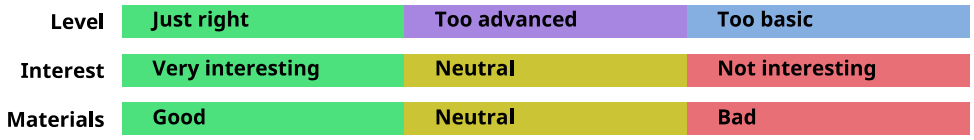
Attendants and Applicants Statistics

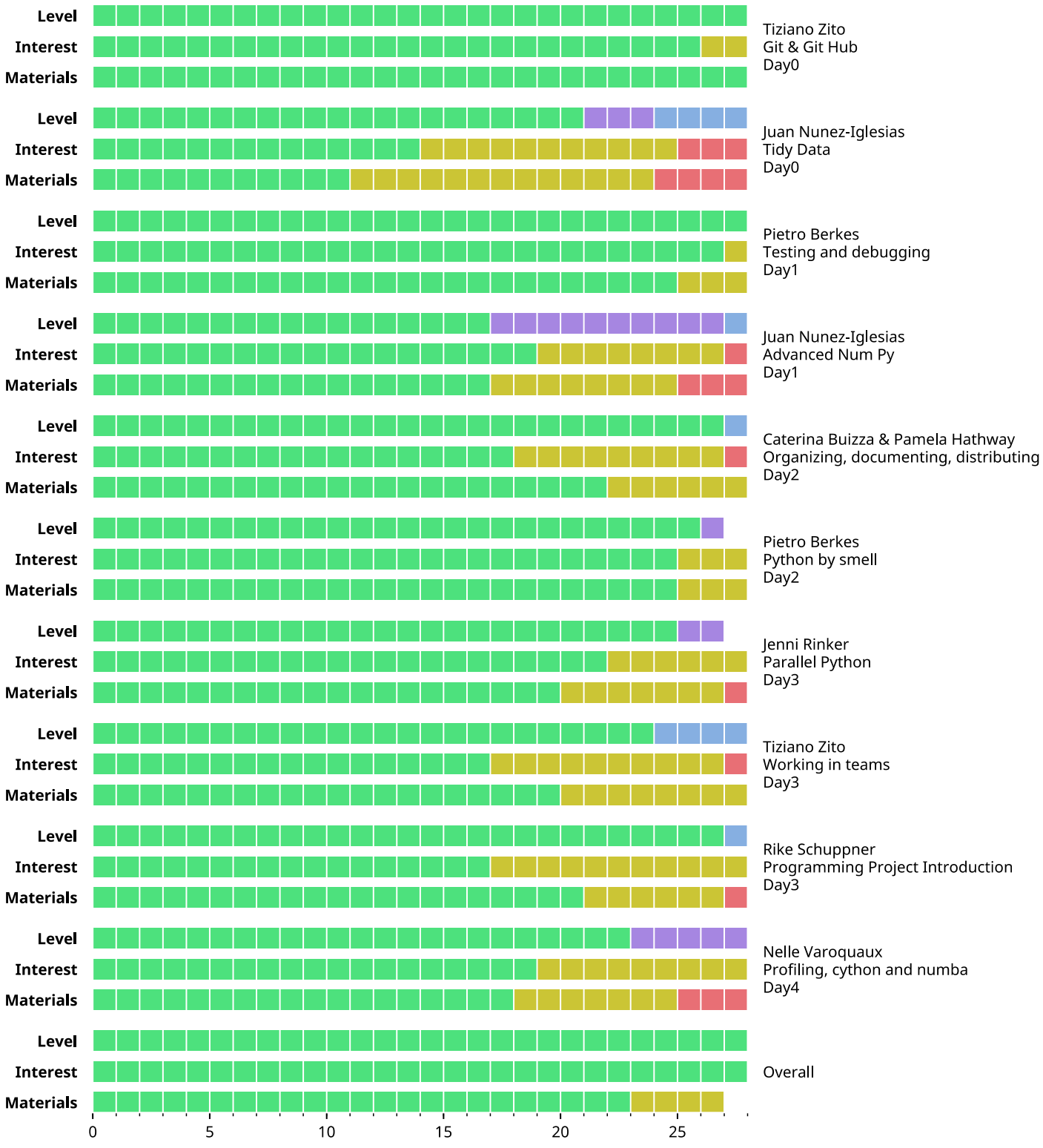
	Attendants		Applicants	
	29	20%	147	
Different nationalities	17		42	
Countries of affiliation	13		26	
Gender: other	0	0%	1	1%
Gender: female	14	48%	58	40%
Gender: male	15	52%	88	59%
Already applied	10	35%	22	15%
Bachelor Student	0	0%	2	1%
Master Student	1	3%	21	14%
PhD Students	20	69%	86	59%
Post-Docs	4	14%	18	12%
Professor	0	0%	0	0%
Technician	0	0%	0	0%
Employee	2	7%	11	8%
Others	2	7%	9	6%
Completed surveys	28	97%		

More stats about attendants are available at: https://python.g-node.org/python-summerschool-2019/students

Lectures & Exercises

Q: *Grade the* ***level*** *of the lectures*
Q: *Grade how* ***interesting*** *were the lectures*
Q: *Grade the quality of the presentation style and/or of the teaching* ***material,*** *e.g. the clarity of the slides/code, the exercises and the solutions, etc.*





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

1. First half of "Tidy Data" and "Advance NumPy" were relevant. The second half of each was too advanced and hardly useful in practice. I would suggest to shorten these 2 topics. All other topics were great.
2. I am not sure why you taught us about numpy strides in the numpy lecture; also I find it a bit strange that you teach us stuff like parallel execution of python code, profiling and optimizing and then you tell us that before doing any of that we should learn how to write better algorithms.

   Given the complexity of this topics, and the fact that you only showed us basic usage, I wonder why didn't you focus on teaching us how to write better algorithms!

   Honestly, I would have loved to have a an extended version of Pietro's lecture on testing, debugging (which is super important but wasn't mentioned), generators and context managers instead of the last two lectures. I would have love to learn properly about classes and OOP instead of how to make my code faster and less heavy on memory.

   You said this course was for non-programmers, but I believe only the first half was.

   If I can't write good algorithms what do I care about profiling and optimizing and PP ??
3. Advanced numpy did not appear very relevant to me. The techniques shown were partly too advanced and a very specific, things I would have to look up anyway when I should need them again. More emphasis on data visualisation would have been great, since it also may provide a work flow.

Q: Are there further topics relevant to the programming scientist that could have been presented, given that the total time is limited. Please also mention which topics should be replaced by the new ones.

1. Basic SQL/data structure/data engineering; Scientific visualization; Computer architecture; useful shell scripts

   I may shorten the Git lectures a bit for these...
2. I would include one session for specific questions to tutors, related to our current research for example, at the end of the lectures but before the group project, because another way the questions will be related to Pelita. I had a couple of questions related to some old codes of mine but didn't find time to properly ask them. In this case, I would replace the "Python by smell" lecture since this can be introduced directed in examples from other lectures.
3. I would have liked to learn more about pandas, plotting and visualization
4. In my opinion, all the topics were very relevant and useful for all scientists and regardless of the research topic.
5. Shorten: 1. Tidy Data 2. Advance NumPy - As explained above second half of each of these two topics was advanced and not very relevant in practice.

   Add/Elongate: 1. Testing and debugging - Testing was nicely covered and relevant. However, there was very little or nothing about debugging. I suggest to cover some basic debugging practices like breakpoints which are practically useful and save much time. 2. Parallel Python - Nice presentation and illustration of dusk. Still, I felt that the last part with dusk cluster was not much relevant (20 mins wasted). I suggest to cover also some other parallel processing utility (e.g., multiprocessor). At least give a skeleton example of using something else besides dusk.
6. Not for the limited time that was available. The selection of the topics was for the one week very good!
7. You already blew my mind with topics I didn't even know existed, so no, nothing new. TBH better to teach stuff properly then having a thousand extra topics.
8. - There could be more scipy/scikit-learn lectures or some advanced techniques working with matrices/vectors (I was actually expecting to learn some functions like einsum etc.)

   - Profiling/cython/numba were actually good topics to learn as an introduction but at the end it was not so clear how and when we would utilize these approaches. Maybe it can be replaced by scipy/scikit-learn or just be shortened to a 20 minute introduction where students are shown some examples without exercises.
9. It would have been more helpful if the whole section on documenting (which was really basic) could be ditched for more on cython and numda. That was too fast and I could've really used another afternoon figuring out how to use both tools from an actual project.py file and not just from a notebook.
10. I really missed a lecture on proper debugging. I use print statements way too much and don’t really know how to efficiently use breaking points etc. I would’ve been nice to have a lecture on a debugging workflow (and maybe also 1 IDE covered or pro and cons of different systems). I would get rid of the advanced NumPy to make space for that, because I think more people would benefit from that topic, also to a greater extent.

11. I believe automatic differentiation and the basics of one of the many available frameworks would be very useful. Automatic differentiation goes far beyond deep learning applications. Much of what scientists do is optimization. They optimize models to describe data in every field and the current preferred methods are based on scipy or scikit-learn which do provide nice functions that take as inputs the function to be optimized, and its gradient. However, this places in the researcher the responsibility of finding analytical formulations of the gradient. This is often not possible in a compact and easy way. Although there are many methods out there, automatic differentiation methods provide with unprecedented ease an automatic way of finding optimal solutions for highly nonlinear functions that are also driving the deep learning revolution. Thus, I suggest teaching the basics of automatic differentiation and the basics of either Pytorch or Tensorflow. If I had to choose, I would replace it with parallel python.
12. Having been introduced to the testing workflow it would have been great to also learn something about debugging (which tools to use for debugging, debugging workflow etc).

    As mentioned above, data visualisation could have been nice in place of the pandas or advanced numpy lecture.
13. There are several topics which would be interesting to study, however given the limited time available I think the choice of topics in ASPP is the most reasonable.

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% (0)
Neutral. It was OK, but I could have worked by myself as well.	7% (2)
Other	11% (3)

Other:

1. it really depends on the topic... for some things it was useful, for others I would have needed peace and quiet.
2. hit or miss, but could be consistently useful if proper instructions are provided (please see further comments for more info)
3. It is necessary to create the summerschool expierence as a whole. It gets people together.

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	7% (2)
Lectures were too short, there should be more time for lectures	7% (2)
The time dedicated to lectures and exercises was well balanced	75% (21)
Other	11% (3)

Other:

1. I really liked that exercises were mostly set as a simulation of how to contribute to open source code. This really helps consolidating the proposed git workflow. However it's also a bit time consuming, hence i would suggest to spend some more time for exercises.
2. Depends on the lectures. The mix between exercises and lectures for Pietro was perfect! I would have liked to have more excercises for the Github lecture.
3. Exercise are good, but instead of going thorugh infinite notebooks I would have preferred more solid and lengthy explanations.

Q: Any further comments about the lectures and exercises?

1. I totally appreciate the pair-programming approach! Definitively keep it like this!
2. The lecture "Tidy data analysis and visualization" required some previous knowledge of Pandas, which was not advertised as part of the "Introductory material".
3. In my opinion, I would have spent more time on the parallelization and cython lecture, since they were a little bit advanced for me and I would have needed more time to work on the exercises.
4. The lectures, teachers and exercises were a great package of subjects and people, which you will not find anywhere else (really, I would like to emphasize that multiple times!). I truly learned a lot and I will start rehearsing all cool stuff I have seen and learned during the summer school.:
5. Sometimes I had the feeling that there was too little time to complete the exercises, which was a bit frustrating.

6. Overall, the lectures and the exercises were well designed. All of the lecturers and tutors were enthusiastic and engaged. One specific comment is that some of the Cython exercises were difficult to accomplish. The functionalities of the Cython functions were not explained in depth before we started to do the exercises, and we needed to go through the documentation in depth and consult other sources (e.g. Stack Overflow) to search for the solutions. I would like to have more specific directions for those exercises. However, the improvement of computational time after implementing the Cython techniques was astonishing to witness, so I still liked that session a lot.
7. Everything was very good
8. The paired programming was really hit or miss 50/50 and I think it could've been a lot more useful if we had some instructions on the roles. I looked them up myself and that improved the experience for me, but sometimes there was definitely a 'watch the master' situation going on, and that it was difficult to avoid. I am a very vocal person and I did ask to be the driver or asked the driver for more information, but if this consistently did not happen I ended up disengaging. Paired programming is a very useful tool and therefore I would recommend to include some background focussed on the pairing variations. That way it is easier to know if you're effectively applying the technique or not. Jenni asked me to include the information I thought was useful, so please find those URL's below:

   Information on pair variations:

   https://en.wikipedia.org/wiki/Pair_programming

   https://www.agilealliance.org/glossary/pairing/...

   Pair programming styles

   https://stackify.com/pair-programming-styles/

   Tutorial:

   https://medium.com/@weblab_tech/pair-programming-guide-a76ca43ff389

   This cool article is extra reading:

   Lui, K. M., & Chan, K. C. (2006). Pair programming productivity: Novice–novice vs. expert–expert. International Journal of Human-computer studies, 64(9), 915-925.
9. I suggest to reconsider time allocated for lectures/breaks. 2 hours session without break may be too long. 30 mins coffee breaks and 2 hours lunch breaks are also too long. I would suggest 80 min sessions, 20 min coffee breaks and 90 min lunch breaks. For a full lecture day I would suggest something like this:

   08:50 - 10:10 (80 mins) session 1
   10:10 - 10:30 (20 mins) coffee break
   10:30 - 11:50 (80 mins) session 2
   11:50 - 12:10 (20 mins) coffee break
   12:10 - 13:30 (80 mins) session 3
   13:30 - 15:00 (90 mins) lunch break
   15:00 - 16:20 (80 mins) session 4
   16:20 - 16:40 (20 mins) coffee break
   16:40 - 18:00 (80 mins) session 5
   18:00 - 18:20 (20 mins) coffee break
   18:20 - 19:20 (60 mins) session 6

   This way there is more time for sessions (more topics can be covered), they are not very long (attendees lose focus on a 2 hours lecture without break), and the full lecture time is almost the same (08:50 - 19:20 ~= 09:00 - 19:00)
10. This has been the most rewarding experience I've had as a PhD student. I think the course was perfectly designed and the lectures were useful, engaging and entertaining. I learned a lot from the lecturers and the other students. I was wary of the pair programming concept but I now think it doubles the learning you get from the course. Hard to describe with words other than I would give the whole experience 100/100. Whatever that algorithm does to filter and select students, it does that perfectly.

11. I felt it a pity that the Pandas and tidy data concepts were eclipsed by git usage, but this was probably the overall feeling. It felt like the people were stuck in the most simple exercises (which most people already knew about) and the class could not advance to the really interesting parts.

    I would have liked to learn more about good programming practices (Pietro's second lecture), but honestly I do not see when this could have happened, since I would not remove anything else.

    The parallel python was really interesting but using one or two other packages would have been also really interesting. Jenni did her best and was very well prepared, this is just a suggestion for future improvement.

12. It happened many times that the exercises were left without finishing them since time was running and the lecture material had priority, so it would make sense to choose more carefully the exercises to fit in the adequate time windows.

13. Yes. When you ask us to write functions first, in order to be able to use them in the exercises, please ask us to write easier functions.

    As in, i wasted a lot of time trying to write functions that I didn't know how to write, and then did not have time to do the actual exercise.

    I didn't submit a single pull request when this was the structure of the exercise.

    More in general, it's not that useful to just try out every single method we learn about (for example in the pandas and numpy lectures). I am going to forget about it pretty soon, and will have to look up documentation anyways so ....Hope this is useful.

14. The reading "Tidy Data" was too advanced for students who did not already use Panda. Maybe some exercises and advanced features can be removed to spend more time on the basics and learn something.

15. You managed to teach us at multiple levels: one obvious level concerned the python programming tricks and advanced topics. Another important one was the whole git workflow and how to contribute to open source projects. Finally there was the collaboration with other students. All of this culminated in the final project. In my opinion this was extremely well contrived and powerful.

16. I liked the entire structure. Just thought the Numpy is a tad advanced. Overall it's a whole new experience and a lot of learning. Enjoyed it all. Thank you everyone whoever made it possible.

17. The ASPP school has met and even largely exceeded my expectations!

    The lectures cover a wide range of topics enabling you to make better use of Python as a scientific programming language, enabling you to write both more efficient and more maintainable code. They incorporate a large amount of exercises enabling you to quickly put into practiced your newly learned skills (taking advantage on the interactivity offered Jupyter notebooks).

18. Overall, I found the lectures excellent and well prepared. The exercises consolidated the new knowledge very effectively!

19. Two of the lectures covered too much material for the allotted time and thus became quite rushed towards the end.

    All of the lectures were excellent and well-structured and the presentation style with in-class exercises allowed for maximum retention of the content and greater understanding of the concepts presented.

20. I appreciated the overall balance between lectures and exercises. However a few lectures, especially "Tidy Data" and "Profiling, cython and numba", had way too many exercises. 'Endless' jupyter notebooks give the feeling of rushing through, in particular among acquainted students.

    I also think that the lecture "Profiling, cython and numba" was really packed of interesting content. I think it could benefit from a bit of extra time.

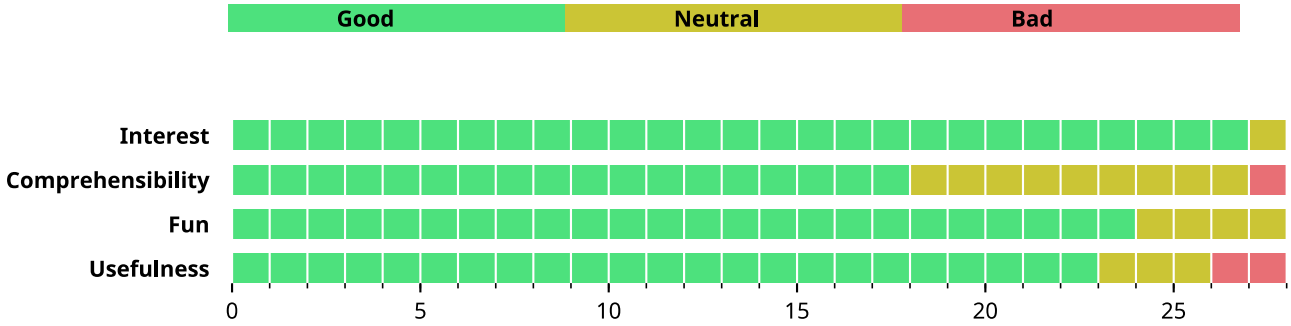
    None of the comment above should be considered serious problems, but just recommendations for fine-tuning the lectures.

21. Generally, I preferred the lecutures with separated learning and exercise material. I think the combination of both in a jupyter notebook can lead to information overload since there are too many things condensed into a small space.

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

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% (27)

Q: Any further comments about the programming project?

1. The programming project will certainly remain the most memorable experience of the school!

   It is however a pity that it takes almost a day to understand what should be done, leaving very little time to actually start turning "wining strategies on paper" into functional Pythonic code, putting into practice everything we learned during the lectures.

   I would for example recommend to increase the clarity of what is expected from us when presenting the game for the first time:

   - making clear that we don't have to generate the terrain nor the bots and their transformation from Ghost to Pacman when entering enemy territory etc...

   - making very clear that we only need to write the move() function

   - maybe mentioning that there are already several "stupid bots" that can be used as a starting point to create our own "slightly less stupid bot" (maybe discussing a bit how the "stupid bot" was programmed, the key functions that can be used as well as the bot.attributes)

   All this should increase the time spent on actually writing code instead of figuring out how the game works (which is okay if you have a week, but very demanding if you only have two days worth of programing project).

   ...But still, it has been an awesome experience!
2. It was very useful in all terms. For most of the groups, it was one of the first experiences of working in teams. It has helped us to be able to understand other's code and workflow, and also to work as a team with a common objective. And it has helped me to become more confident about using git!
3. The core of the programming project is to learn how to work together both technically and as individuals in a group. The pelita game is a perfect, leightweight, fun, and yet competitive way to simulate this environment including emotions like fear, anger, stress and euphoria. Therefore, the Pelita programming project is irreplaceable.
4. I loved the programming project! Maybe it would've been useful to get a full lecture about how to code in distributed teams.

5. I really enjoyed the project but to be honest I couldn't contribute any code to the end product.

   I realized that working collaboratively requires many years of practice. Our code was actually written by only one person. Of course I also had fun with my paired mate, trying to implement a specific utility but it was not even included in the bot code at the end. I felt I would be better off coding on my own at some point, although that would miss the main point of the project which is to code together.

   I say, there should be some main points to implement to the game. Let's say, you give six small implementation ideas and tell people to distribute these tasks between group members. The difficulty of the implementations should be between a wide range because some people might not be a confident programmer or another would be an expert so s/he would be bored. Then everybody would implement their own functions first and then they can go on and explore new tricks to enhance their bot. In this way, I assume, everybody would be satisfied and trust their own contributions.

6. I have no idea how you create the working groups but maybe it could be possible to improve them. One suggestion would be to measure the students willingness to work together at the beginning of the school (or in the application phase) and use this information to balance the group formation.
7. pelita is an awesome didactic tool.
8. I would verify that all members of the course are able to contribute (at least minimally) to the code and have sufficient programing skills.
9. I thought it was super fun. It particularly helped with Git and testing. Working in teams was hard but completely worth it.
10. Other than Git, applying the programming techniques learned in the lectures was hard. I personally spent a few hours to read through the documentation (with distractions from group members). I finally understood the purpose and the limits of the move function by the end of Friday, which left me less than 24 hours to code. Writing tests and detailed documentations in such a short time was hence impossible. I think the introduction session to the project was a bit short, and students should have the chance to read the documentation beforehand (either before the workshop started or by the end of day 2 of the week). Therefore, we would be more familiar with the game and the move function, and we would ask intelligent questions during the introduction session. Moreover, working with people I barely knew (I did not get to pair up with most of my group members in lectures, nor I hanged out with most of them outside of class) caused complications and consumed more time than expected. We finally had the right group dynamics by Saturday morning, which was late. The learning experience of the group project was also more on the social side than on the technical side. For example, our group did not really exchange technical knowledge or skills within the group because the code development process was so slow and inefficient, and because we also did not know each other well enough before Thursday. Students in the future should be allowed to form groups with people they already knew and befriended, which actually could enhance knowledge exchanges (e.g. if I find a group member unbearable, why would I spend extra time to interact/teach/learn from him/her?). But I understand why the organizers choose to form random groups: we gotta work with people we dislike in reality.
11. The programming project was easily the most demanding and stressing part of the school. Nevertheless it was designed in the first place to reflect a real-world work scenario. Therefore, I think the programming project was a success. Moreover, the tournament at the end was pure fun.

    10/10 would recommend.

12. Great work:
13. The videogame idea is nice because it makes it more fun.

    Regading the teams..... I guess I was pretty unlucky with mine, maybe give us the opportunity to choose some companions...? I guess its not easy to implement, but I really wasted a lot of precious time where I could have been coding and learning stuff, to manage fights within the team. For me, managing this team work was completely useless because I am a scientific programmer, and when does it ever happen that in science you work with five people on the same code???? Generally in science, different analysis go together in a paper, and thats entirtely another sort of teamwork. Honestly it just gave me an headache and removed learning opportunities.

14. I believe that people learned way more about group dynamics and teamwork than programming through this project. It felt a bit laid back in the sense that nobody looked at our code to evaluate how good or bad it was. The task was complex for the little available time and the code structure and good practices were mostly ignored for the sake of time, which is the opposite of what we were told the whole week.

    The students were under way too much stress and pressure towards the end. I do not think that having a live countdown is a good idea at all. Also the remote bots undermined the morale of the team very much.

    I would in future editions provide a couple more examples (with a switching bot for example), from which the students can create their strategies. Too much time was spent in understanding the dynamics of the game itself.

    I think it was a very fun project and enjoyable experience, it was very much fun to work on something different, with such an immediate output.

15. There was a high level of independent programming in my group, so I felt I could've learned more if paired programming was mandatory instead of just advised. Again, this only works if the proper way of paired programming is instructed at some point. Additionally, it would’ve helped to have a very general timeline for the project, as none of us were used to working in groups and integrating code. Alternatively, an earlier deadline/advised time to merge might be a good idea.
16. Pelita is love, Pelita is life!
17. It was an amazing experience, especially since it also involved a real-life social component.
18. One could maybe complain about the non-science nature of the project because stuff you learn is less applicable. Personally I think it might be hard to find a real-world project that is inherent to the variety of fields people come from. Also the current solution is more fun :-)
19. Some of the demo bots were not so easy to understand as there was very little documentation in the demo code.

    The video game project allows all participants to practice what was taught during the course and ensures that no student is disadvantaged due to a lack of knowledge of a specific scientific domain. Students can later apply the concepts of the course to their our scientific domain after having practiced and gained an understanding of the key course concepts with in-class exercises and on an accessible exercise like the video game project.
20. I thought it was awesome. Great fun!
21. I think the programming project is really well thought. Each person involved, whatever his field of expertise, can give his contribution with his knowledge on basic Python coding. Moreover the tournament at the end of the week mimics really well the pressure of real life situation making the whole project experience a good training exercise for later scientific research activities.
22. Maybe it could be useful to extend a little the documentation and give some more insights about heuristics that can be added/implemented to make the code work better.

The School in General

Q: How do you overall evaluate the school?

Good: 100% (28)
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: 4% (1)
Just Right: 93% (26)
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: 0% (0)
Just Right: 100% (28)
Too basic: 0% (0)

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: 93% (26)
Neutral: 7% (2)
Bad: 0% (0)

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

Yes: 100% (28)
No: 0% (0)

Q: How did you hear about the school?

Google Search: 7
Professor/Tutor/Supervisor: 7
Colleague/Friend: 8
Mailing list: 6
Other: 4 (twitter, scipy, BCCN-Berlin, ASPP mailing list)

Q: Any further comments or suggestions?

1. The ASPP team is really amazing. The instructors always had a generous and humble attitude towards everyone, making the school a pleasant experience, both from a social and learning point of view. You all passed the tests successfully ;-) The student group was also very nice and fun. I am very grateful to have been part of this remarkable social experiment.
2. Thank you so much for the super fun time with great and very dense input. It is amazing that you all choose to put your time into this and we get to benefit from your insights and experience. It has been a very exhausting week that I would love to repeat if I could :-)
3. Keep doing such amazing job guys!
4. The organization of the course was amazing, you created a wonderful atmosphere and a great week! Such a different experience from any other type of scientific meeting. It has been a pleasure to be part of this course, the perfect balance between learning and enjoyment!
5. This was one of the greatest expierences I have had in a long time. I learned an incredible amount of things that I would have not been able to scrape together from books and online courses. Furthermore, you created a safe, friendly and fun environment, which resulted in an incredible positive [insert a row of superlative adjectives here] vibe amongs both students and teachers. This made it possible to push ourselfs from 9AM to 7PM and beyond. I feel really grateful to have been part of ASPP 2019.
6. Everything was great! Never had such experience before.

7. Thank you so much for this experience. It was awesome and incredible. Now I'm willing to code again to improve the way I code.
8. Thank you to the organizers, instructors and assistants for an excellent course - all your hard work and dedication is greaty appreciated. I have learnt a lot during ASPP - everything that was covered during this course is directly applicable to my scientific programming career. The lecture/practical balance was perfect; interspersing the lectures with hands-on in-class practical exercises allowed for better understanding and retention of the concepts presented. The hands-on approach allowed us to practice what we had just been taught and to experience possible errors that we could encounter. When learning new programming concepts in isolation these errors can be discouraging and take considerable effort to resolve, but this was not the case during the ASPP course. The assistants were always on hand to explain why an error occurred and how to resolve it. The instructors and assistants are all very knowledgeable experts; exposure to so many experts with such experience is absolutely priceless. I highly recommend this course to other young scientists who wish to become better scientific programmers.
9. The course was really great! I learned a lot during the one week and I had at the same time a really good time and I met a lot of interesting people. I can highly recommend this course!
10. It was all so good.
11. This Python school has been a memorable experience, as much on the lecture side, the location (Camerino) and of course the social side.

    I learned a lot about Python... almost reconsidering staying in research now to put into practice all these new skills ;)

    The organizers are amazing and the selection algorithm is indeed working: social interactions have been great throughout the entire week!

    Furthermore the remoteness of Camerino certainly played a role in the "social bonding", I think it's important to keep doing this school in somewhat remote locations without too many distractions.

    The absence of participation fees (other than the travel costs, very affordable housing and food) is I think also an important aspect promoting social diversity.
12. It was really great! If I didn't live in one of the most expensive places on the globe (aka Geneva) I'd be super happy to help you all next year. Let me know if I can do anything else to help out!
13. This is somewhat related to my previous comment on paired programming: I had a lot of different ways in which I encountered feedback this week. Although I am not that bothered by blatant critique, I think it is important to include a small note on communication in the course because it is essential to communicate clearly and professionally, especially in collaborations. Some inspiration could be found in the standard feedback rules (and for more depth, the rose of Leary)

    (Feedback rules:

    https://www.learning.ox.ac.uk/media/global/wwwadminoxacuk/localsites/oxfordlearninginstitute/documents/overview/rsv/Guidelines_for_giving_and_receiving_feedback.pdf

    Rose of Leary: https://en.wikipedia.org/wiki/Interpersonal_circumplex)
14. I applied to the ASPP thinking I will not be a part of it. Then I was selected and starting to think I will never live up to it. Well, I had to retreat my opinion. One of the best aspect of the ASPP was the feeling of being accepted, of being worth it. Even when some arguments arose there was always someone acting as a peacemaker and by the end we, the students, were always ready to make a step back rather than continuing argue. So by the end I can surely assert that ASPP has been a key point for the development of both my human and programming skills and I would recommend it to all scientists who program.
15. This is a fabulous experience. I repeat, this is a fabulous experience. The instructors were phenomenal. On the technical side, I may be one of the few "average" students you were looking for: I had experience with a lot of the topics covered but I was not very proficient at most of those. I found even re-learning the "basic" concepts useful, especially for Git and testing. After this workshop, I walked away with a lot of new concepts, techniques, and tricks, and I am certain I have improved my programming knowledge. Of course, some of the topics were way over my head (cough, Juan's advanced lectures, cough). The difficult parts would require revisiting and revising after the workshop, yet this is the purpose of learning. So, I would not drastically modify the difficulty of the lectures and exercises. To me, this is perfect, or perfectto. On the social side, everyone there was friendly and genuine. I had an enormous amount of fun working with many peers in class as well as spending time with them outside the classroom to learn about their lives. It has been difficult to find a cool group of science nerds during and after graduate school, so I am glad I had the opportunity to connect with so many fun and kind people in this program. I must say I am going to keep in touch with many of them long after. All in all, this workshop was extremely fruitful and unbelievably amazing in many ways. Thank you all very much for organizing and teaching, you rock!
16. Yes, it would be nice not to have every single social activity revolve around alcohol use.

    Honestly, do we really need that? There is a number of reasons why somebody would maybe want to restrain from alcohol use, and you make it difficult for these people not to feel that social pressure. Also, on a related note, maybe avoid mental-illness related jokes. Please know that in academia the prevalence of these illnesses is even higher than in other fields, and its really not best practice to just make light jokes about them. I am sorry to be so brutally honest, but this doesn't help at all the sufferers.