

ASPP 2026



18th Advanced Scientific Programming in Python Summer School 30 August – 6 September, 2026. Prague, Czechia.

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 6 October 2026.

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	19%	154	
Different nationalities	19		43	
Countries of affiliation	9		30	
Gender: other	0	0%	3	2%
Gender: female	18	62%	87	56%
Gender: male	11	38%	64	42%
Already applied	8	28%	16	10%
Bachelor Student	0	0%	7	4%
Master Student	2	7%	25	16%
PhD Students	18	62%	81	57%
Post-Docs	6	21%	20	13%
Professor	0	0%	1	1%
Employee	2	7%	12	8%
Others	1	3%	8	5%
Completed surveys	27	93%		

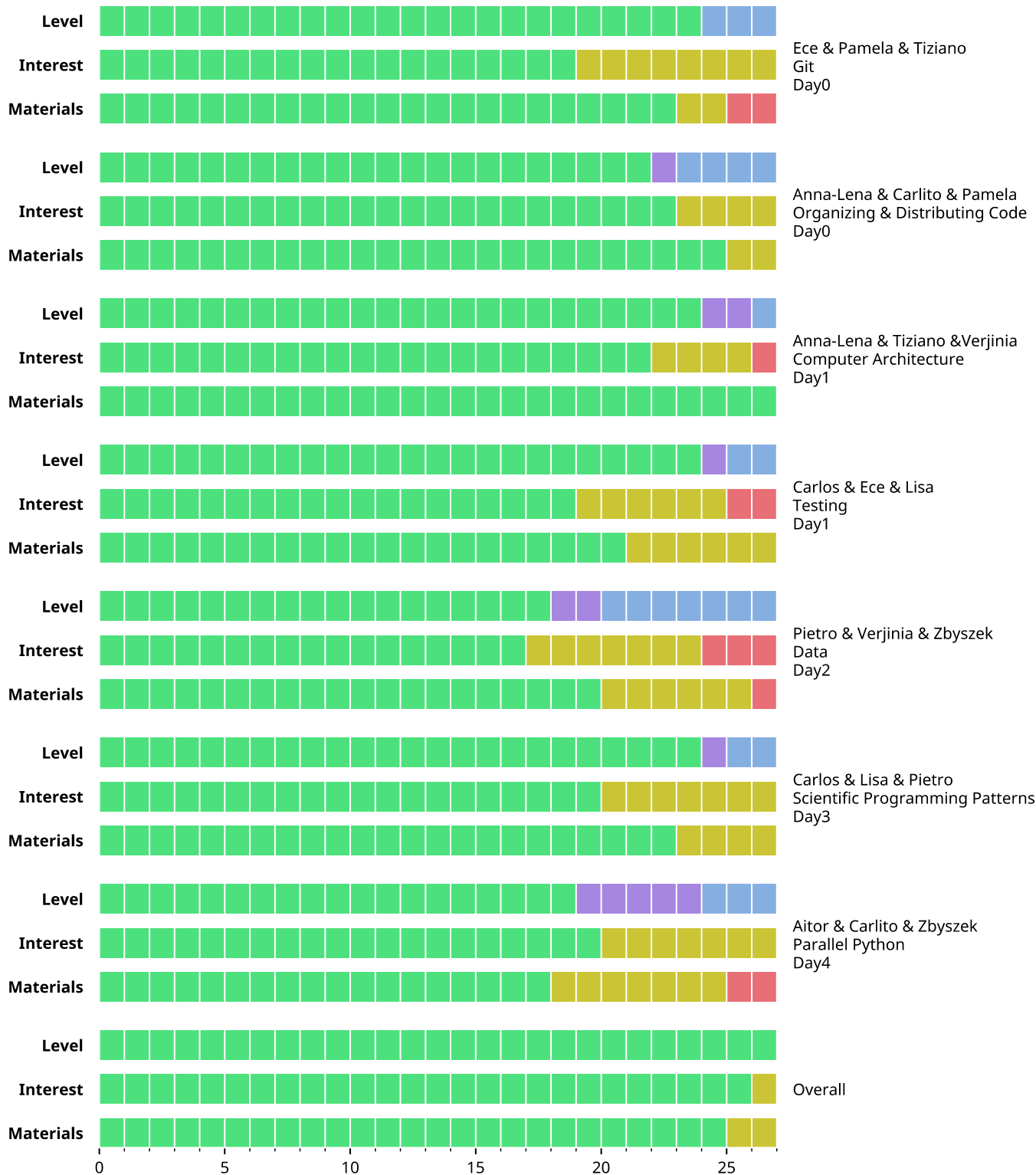
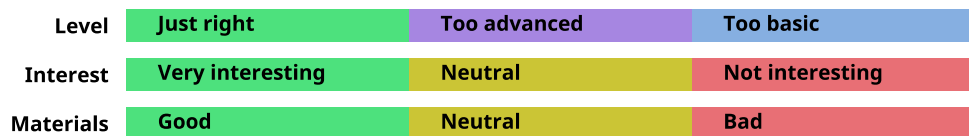
More stats about attendants are available at: <https://archives.aspp.school/2026-prague/students.html>

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. I really liked the lecture on computer architecture but mostly because of my own interest in the topic. I see that understanding how the hardware works/interacts can help debugging code (esp. comp. heavy analysis that are not optimally implemented) but it would suffice to learn the high-lvl do-s and don't-s and not the underlying mechanics.
2. Computer architecture
3. I was not very much interested in df/pandas part in Data, but I knew it well before and I feel that everyone is doing spreadsheeting one way or another anyways.

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. Something on matlab vs. R vs. python, and how to transition from the first two to python. Although I understand this might not be useful for those who were already programming in python exclusively. I would not want to replace anything, but personally the parallel python was the least relevant for me, since I'm not using any big datasets/computationally heavy operations. Additionally, since the programming project was such a big chunk of the schedule, I think a lecture on project management (e.g., scrum, agile, how to organise ourselves, do sprints, etc.) would have been super useful to make the groups work together more efficiently. I would do that right before the project at the project introduction and take a bit of time away from the actual project programming.
2. I felt the lecture/s on data could be expanded upon. They were good, but data handling was a comparatively small portion of the overall course load while arguably a relatively large part of every scientific programmers day-to-day.
3. I would suggest to consider a lecture about to scipy package, maybe instead of parallel python.
For how useful it can be, a scientist, who is generally heavily relying on Numpy, might be more interested in learning Scipy instead of parallel python, which is already handled by Numpy.
4. The data lecture or testing lecture could be extended with an overview of databases and organization of databases. I believe there is a big need to understand how big data can be organized, as this is something that easily falls under the radar and could result in people reinventing the wheel, to avoid being stressed over documentation.
5. Nothing in particular is coming to mind. This was kind of covered to some degree with some other topics but I would have been curious to learn even more about simulations, but not sure what to replace it with.
6. More on debugging and good practices in writing documentation
Good practices for making plots
Linters
7. This is not a new topic, but I thought during the lecture on repo organisation that I would learn more from some hands-on on distributing unorganised badly-named files to folders in a repo; it might help to better understand the repo logic we discussed and this is something some programming scientists encounter from time to time.
I was also wondering if it is possible to add somewhere (e.g. in Scientific patterns part?) a short discussion on what changed in industrial/common Python usage over the last decade(s), e.g. if some approaches or modules are practically obsolete, but stackoverflow is still full of it.
8. I think one thing that comes to mind is making multipanel figures programmatically, i.e., having fully reproducible pipelines so that someone can run my code and obtains the figures in the paper as well (or changes of them for different parameters) I feel this is something I see quite often that code is shared but then the figures are handcrafted in PowerPoint, Inkscape, Figma etc. the scientific data one could maybe be replaced with this?
9. There wasn't much about preparing reproducible code and environments: would love to learn more about in the context of Open Science, not just Open Source.
10. I missed an introduction or an overview to handling big(ger) data, other than a single slide during the data lecture. Although the research standard is Pandas, I think students could benefit knowing more about database systems, for example. I personally found that ODD was given a lot of time and weight, and could use a bit of shaving down - although some others might have found it to be very helpful start to end.
11. Maybe plotting with Matplotlib, this could replace computer architecture or parallel Python. I did find those two topics interesting as well, this is just an idea :)
12. Modeling, more in depth about code organization. The examples presented were too idealized and not realistic for people who work with experimental data and many types of it
13. I believe that the selection of topics was just right, with all important aspects covered.

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

Yes, I have learned from my partner / I have helped my partner	89% (24)
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	8% (2)

Other:

1. I think it could have benefitted some people, but I feel like first implementing and then discussing different implementation in pairs might have been more productive. I felt like in some cases some of my partners needed more time than I did to understand how to implement solutions to the exercises and I think that was infuriating for them
2. I think it was good, but tbh I think I had the best experiences when I worked with someone close to my own skill-lvl and not someone vastly better (in which case they did most of the work) or worse (in which case I did most of the work). Not sure how to improve on that though.

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	19% (5)
Lectures were too short, there should be more time for lectures	0% (0)
The time dedicated to lectures and exercises was well balanced	77% (20)
Other	7% (2)

Other:

1. Overall the balance was good, but I think having even it skewed towards more exercises would have been better. I (personally) learn programming more when doing it rather than being lectured about it :)
2. Lectures are well paced, I wish I had more time for exercises

Q: Any further comments about the lectures and exercises?

1. This was one of the best investments I have made to my scientific career. I am leaving ASPP feeling way more confident than I thought possible in my coding abilities.
2. First the overall experience was far behind my expectations, the wise selection of extremely relevant topics, and each addresses at the level far higher than the content widely available in internet.

Below some minor remarks

Some exercises were taking too much attention for resolving python code problem, than addressing the actual context of given lecture. I think sometimes simpler tasks wholly dedicated to the context would make better job.

Git lectures were fantastic and very relevant to me. It would be useful though that a presenter has behind some map, at least simple local, staging, and remote repos graph. So he could easily use the pointer to remind about which location and direction we are talking as the pace was fast.

I appreciated the reverses structure of some lectures, when first you get a taste of a problem with an exercise and then show theory later, which can be then easily digested.

3. For my experience, level, I found that the lectures were easy to follow. I took notes of certain topics that I would like to explore more because I don't think I understood them well enough. The exercises were supportive of someone like me who takes a long time to write code from scratch, by providing templates for the solutions. I also liked that there were "thinking" or "discussion" exercises which I could do with the pair partner.
4. In some lectures, the balance between lecture and exercise was perfect. In some of them, the chunks of lecture were maybe a bit too long for my poor attention span :'), but perhaps this is more on my end and not shared by the average participant.

5. All the faculty did a great job! All the topics are well designed and very practical. Introducing Git and how to organize the code on the first day is a good idea. The red, yellow, and green flag system is very nice. And I appreciate that course materials from past years are published, so I can learn topics that were not taught this year.

Some small suggestions: 1. It would be nice if the code from live coding sessions could be updated in the course materials after the class finishes. 2. I can understand that AI-generated figures are used in slides for decorative purposes. But some of them are a bit distracting...

6. The structure of the lectures was organized very well. Even though I was unsure about how to use the right syntax or some other basics of python, the exercises helped immensely to improve my trust in it and the pair programming helped to reflect and zoom out. Thank you so much!
7. Felt like I learned the most about exercises. The lecture content didn't necessarily feel too long but sometimes I noticed we didn't actually get to the part of the exercise that was about the topic of the class, but this was almost impacted my own python skills (as in it was taking longer to compile the basic code at the expense of getting to the actual exercise). Still felt like I learned a lot.
8. The git lecture could be more hands on
9. The lectures and exercises are helpful and still, one month after the course I revisit the (very well documented) material.
10. The hands-on exercises were super useful and the implementation of the (handcrafted!) progression flags worked perfectly. I hope each student managed to do at least one pull request (in their pair team) because it was rewarding and exciting. The pair-programming is to me one of the best qualities of the ASPP, no one feels stuck or self-conscious about their own level and everyone can improve and learn so much faster!

Special mention for the human representation of computer architecture, it was so fun!

11. I was surprised how useful were the lectures and exercises for me, even if from the title it was not so obvious. I learned a lot from the lectures and the exercises showed how much the practice differs from theoretical understanding :) I believe that the amount of things I learnt or tried myself during this school was triple more from what I usually did at "programming for non-programmers" workshops.
12. The lectures were super helpful and well-organized, it was clear how much effort and iterations of improvements went into them! Thank you!
13. While I was unevenly interested in lecture topics and exercises, overall the structure, organization, and preparation of classes was outstanding.
14. The lectures were extremely clear and very engaging. I really appreciated the lecture-exercise structure where we were able to try out and apply what we learned immediately. This way I could test my knowledge and ask questions instead of feeling like I understood and moving on without actually knowing how to apply what I'd learned.
15. I would have preferred stock photos/ images instead of AI generated visualisations, those felt a little cringe to look at sometimes, especially since a main purpose of the school is to focus more on human work instead of GenAI output...
16. I really enjoyed pair-programming and I think it is a very strong feature of the school which really benefits both strong programmers and newer ones.

I also really liked alternating between lectures & exercises, and how easy it was to move on to the next exercise even when we hadn't finished the previous step (since the solutions were available and ready to use at the next step).

It felt easy to follow and to integrate knowledge by trying it out right away; I was impressed by how well the sessions catered to different levels (e.g., by having solutions ready for use when adding to an exercise, and by walking us quickly through actual solutions committed by students including mistakes at times). This made me feel like I always learned something both when it was fairly easy for me to finish the exercises and when I was slower.

17. I found the lectures and exercises pretty comfortable to follow. Even for the things I didn't immediately grasp, I was able to understand things better from my pair programming partner or one of the instructors.
- I appreciated that for some lectures, the instructors gave some reasoning on why doing something in a certain way (hands-on) was better than using an LLM for the task. I suggest continuing that trend for the future!
18. Git lecture first day should be more practical; simply showing things is not useful to me because it doesn't stick in memory
19. I really liked the mix of lectures and exercises that allowed us to immediately try out and practice the topics covered in the lectures.

Programming Project

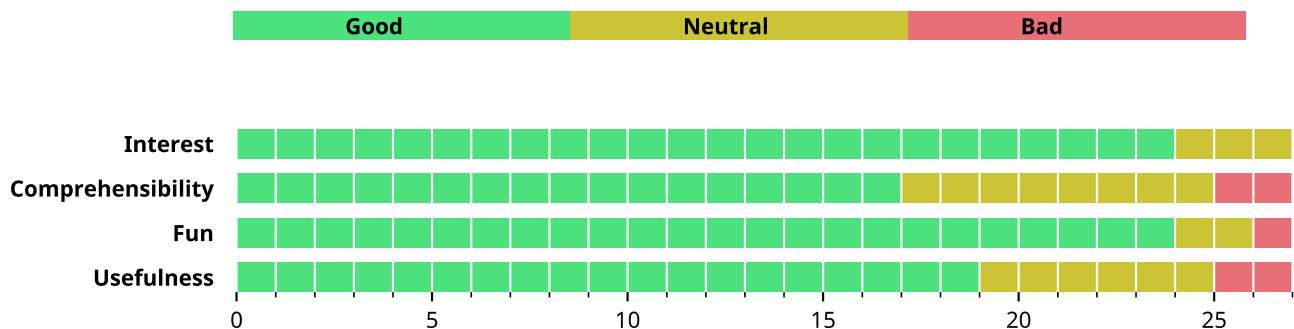
Q: Evaluate the programming project.

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: 85% (23)

No: 15% (4)

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

Yes: 7% (2)

No: 93% (25)

Q: Any further comments about the programming project?

1. The project was very engaging, at first seemed a bit overwhelming but clear documentation helped to progress.
There is one challenge, coming from the fact that people from academia are smarter and more ambitious than regular ones :). Regardless the initial instructions our nature let's us define challenging objectives, which later turn difficult to implement. I would advise to force on the participants solving first some simple exercise demonstrating simple logics of the available functions and methods, and submitting a PR in a neat way. Just to assure, that everyone regardless of initial level, is able to contribute or exercise as least a little. Otherwise there's a risk that the more advanced team members outpace others, and the rest will end up with minor contribution. It would also give the first feeling of how challenging for each of participant is to translate some strategy into code.
2. Very fun. Working in a big team is difficult. Tournament very nice.
3. I really enjoyed the project, and it was nice that pelita was developed just for teaching, so we never thought of using AI for help. The documentation was definitely a lot, but it was easy enough to look things up if I wanted. In future, it may be useful if we could access docstrings of the functions using the typical function? syntax (sorry if this is already possible and I missed it). Most teams' anxiety was regarding unexpected bugs popping up during the tournament, but the demo bots are a good enough foundation to build on and the available background and tests help gain some confidence, so that we have a functioning bot.py in the end. :) I found my calling towards the end of the project when we were pretty happy with our bot's intelligence, so I introduced Shakespearean trash talk for our bot (2BotOrNot 2Bot - that is the question).
4. It was a great project, and it should definitely stay a game, since the skills are transferable and any specific scientific problem would be relevant for some and not for others anyways. The game was also just a lot of fun. What I would have wished for though, is a bit more guidance in how to organise a team where the programming skill levels vary. So, a lecture on project management and different roles you could consider in a group (based on different strengths of different members) would in my opinion be a really good addition. This way, you avoid the disorganised deepdive into the code which may be hard to follow for those less proficient in python programming.
5. It was a lot of fun! The instructions given by Tiziano and the examples in the repo were quite clear for us to complete the projects.

6. It was super fun! While it's maybe less directly applicable to scientific data, I think the programming project is super fun and cool and engaging. Wouldn't change a thing about it.
7. More rules about how we should be working on the project. There is not enough time to work them out in a team.
8. The programming project was great fun. On the one hand it was not a scientific project, so I don't feel the challenge was particularly interesting, however, it took us all zero time to engage, because it really was a common language for people working across different domains. We did learn a lot more on how to work in a team, how to breakdown a project into features, and work on them in together using git.
9. It must have been the first time I truly enjoy programming, it's so fun to work on a videogame! We had so many ideas about different strategies and we felt confident enough on our programming skills thanks to all the hands-on exercises during the week. I am also very grateful for the material and examples given because I often feel overwhelmed when I have to start a project from scratch.

Of course, the team work was challenging as we had very different working styles and programming experience, but we tried to focus on the goal which was learning, and we definitely did :)
10. I was afraid my coding skills were not enough to participate in this part of the school, but in the end it was so much fun and freedom of experimentation in writing the scripts, I liked it very much.
11. I think it was the perfect project to work on. It was not only super fun but also really modular, so we could get something to work but there is also no open end to what we could improve - really amazing!
12. godspeed pelita bots!
13. I was a little sceptical and to be honest, intimidated by the prospect of the video game programming exercise, but after having done it I am 100% convinced that it is the perfect exercise to practice what you learned during the week. Not only is it super fun, but many elements from the week were directly applicable, and even were it wasn't it was easy to think further about how other lecture contents could have been applied. I think the project is also a great way to account for the diversity of fields that the participants come from, and the different python proficiencies they enter the project with. Lastly, because it wasn't connected to research at all, it also made me more confident that I can use python outside of data analysis, which is not something I thought before this school but of course very useful to know.
14. As the Pelita API is quite extensive and there is not a lot of time to really get into it, I don't feel like a lot of the topics from the lectures apart from git were easy to implement/ apply. Either a project with more obvious "problems" related to the lectures or more time to work on it could be good.
15. I really liked it!

I was actually not so excited to work on a game nor in big groups O:) This is not how I typically work so it felt a bit remote from my needs but it actually was very relevant -- I really appreciate it now, and would not have it another way :)

It was a great way to boost my confidence about using things we'd just learn, rather than wait until I got back to the lab and perhaps hesitate to use new skills given time constraints, etc.

It was also great to get us out of our comfort zone and work on something different.

It would have been helpful to get a bit more info on how NetworkX was used in the game (it's a big package and it was hard to find the relevant information on their website, and to figure out which NetworkX functions Pelita used -- e.g., more info on paths and graphs). Otherwise, I found the documentation very good.
16. The project was great. It was a great way to exercise the skills we learned in a fun and competitive environment. Had it been about analysing data, it wouldn't have been as engaging.
17. I think keeping it as a game makes the project more fresh, and ensures that learnings are transferable to any STEM PhD regardless of the domain.
18. For me, what I learned during the programming project was more about the challenge of achieving a task by working in a team with people with different backgrounds and programming levels, then about the programming task itself.
19. I think it would be good to give a bit more instructions about how to organize time during projects.
20. The project offered a perfect opportunity to combine all the knowledge and skills we learned over the summer school. It was a great experience to work as a team and create a fully working implementation in such a short amount of time. It really helped me to get a better understanding of how all the different topics are connected and how they are all relevant in a 'real' project.

The School in General

Q: How do you overall evaluate the school?

Good: 100% (27)
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: 89% (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: 4% (1)
Just Right: 96% (25)
Too basic: 4% (1)

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

Yes: 96% (26)
No: 4% (1)

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

Good: 96% (26)
Neutral: 4% (1)
Bad: 0% (0)

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

Yes: 100% (27)
No: 0% (0)

Q: How did you hear about the school?

Google Search: 1
Professor/Tutor/Supervisor: 9
Colleague/Friend: 17
Website/Mailing list/Social media: 3

1. I believe it was visionlist or cvnet
2. Some excel sheet that circulated on Twitter with summer schools in europe
3. LinkedIn connection

Q: Any further comments or suggestions?

1. ASPP is an incredible opportunity for any self thought programming scientist. I learned so much in such a short time and would recommend anyone who wants to implement good/reproducible practice in their jumble of a code.
2. Just keep going, you are fantastic :)
3. ASPP was a great experience in building confidence in what I know already and the work I've put in to learn programming. My main motivations to attend were to find out what I know, and what I'm missing, and how to access resources that will help me improve. The school provides enough information in all aspects, and I really appreciate that.

I was very happy to meet people from diverse backgrounds and experiences, and it was really nice to get to know everyone. Regarding alternative activities for socialising in the evening, I would recommend some self-

motivated artsy activities that need minimal supplies (e.g., origami), so that people can participate in having fun together without necessarily staying up late, drinking, and going out in noisy environments.

Finally, I'd like to say the the genuine intentions of the tutors to spread their knowledge, and the students' curiosity and interest in becoming better programmers was heartwarming to see (and it restored my faith in humanity to quite an extent :D). I loved how the school provided plenty of room to be unapologetically nerdy!

4. The summer school was amazing and exceeded my expectations very far. It was really a wonderful time and I loved meeting like-minded people who share interests with mine. That is something I don't find so often and can be hard as a PhD candidate, I felt a strong sense of belonging :). Although the level was sometimes a bit too advanced for me (looking back, I think I may have overestimated my skills a bit...), I still had a lot of fun and learned a LOT that I will be applying in my daily life now. What would make the summer school even better is if Carlos would get to lecture a bit more, he's really good at explaining things but we only had one topic taught by him. I was nervous about the pair programming at first, but actually ended up liking it so much. But the biggest thing I want to say is that I believe the faculty deserves so much kudos and appreciation. You guys organised everything so well and you could just tell that you guys were all excited to be there and super passionate about ASPP. I loved that there was no hierarchy (on the welcome activity I had no idea who was faculty and who was attendee), and that everyone was always mingling. It felt like the social-group part was just as important as the lectures. Finally, I want to give a big shoutout to Aitor for being the best local organiser anyone could ever ask for.
5. Thank you for organizing the summer school. Teaching topics that are not relevant to AI is a kind of resistance to the trends of the times, and I appreciate that. Keep up the great work!
6. This summer school was so so cool! I learned a ton and feel so much more confident in my skills and abilities. I lost some of the anxiety I had before of writing scripts and thinking through the logic of the analysis. I came back and immediately started putting my code into a replicable version with a proper documentation and folder structure that makes sense!!!

I cannot put enough "!" and thank you here, but i hope you know how grateful i am for this opportunity and appreciate the dedication you put into it!
7. I didn't expect a Python summer school to be as fun as it was or learn as much as I did! I grew in confidence with my Python and programming skills all the while having a lot of fun with the other course attendees and the faculty. Best decision of my PhD journey to go do ASPP - thank you so much!
8. It was one of the best courses I have ever attended.
9. ASPP was an incredible experience and I am really grateful to the team and students for creating such a safe environment for learning. Almost all teachers had been students in the previous editions, and I could feel their passion and care towards the summer school. Everyone was super approachable, knowledgeable, humble and fun!

I truly benefited from the pair-programming and switching partners for each lecture - I learnt new useful tricks and also shared the ones I knew. I also enjoyed the live coding done by the lecturers because it helped me follow their train of thought and normalise that no one writes perfect code on the first attempt. The lectures were very dynamic and there must have been some sort of spell thrown on us because we were all focused from 9AM until 7PM every day! I think the coffee breaks were helpful for that (thank you Aitor for the amazing food - honestly, it's unheard of at scientific conferences!). On a more philosophical side, I resonate with the "no-AI" policy of the summer camp and I think that it's very valuable that it helped us to gain confidence in ourselves, both for implementing and fixing code.

Based on my personal experience, I have a couple of ideas that could help clarify some concepts. I think it could be useful to have some sort of overall structure to understand at which level each concept applies (data structures, testing, version control, parallel programming, ...) - should we think about it from the beginning of the project/at the exploration stage? Or more when we want to ensure reproducibility and optimise the code, knowing that it works? It would be nice to visualise how the different good-practice methods interact, and how they enable or hinder each other's effective functioning.

Another idea could be to present comparative tables for different tools/techniques that address the same issue - when is one more effective than the other? When is one more straightforward to implement than the other? Can they both coexist? I am thinking here about comparisons such as creating a virtual environment vs a code package, git forks vs git branches, breakpoint() vs print statements, multiprocessing vs multithreading, etc.

(Very minor detail: for the card sorting game in the computer architecture class, I think an initial example could be useful as some of us didn't understand that information was deleted from the memory when moved to the register.)

Overall opinion: 10 out of 10! Thank you very much Tiziano for creating and believing in such a wonderful project, and transmitting your values to the new scientific generations.

10. I really hope there will be next editions of this school in future, it brings joy in academic life.
11. I attended another summer school right after ASPP and immediately recommended it to three of the people I met there, and I gladly would have recommended it to ten more. Anyone who uses python for scientific purposes will immensely benefit from this course, I learned more here than from any online tutorial or any other course I did at my home institution. It made me excited about coding and programming in a way I didn't know I could be, and I actually enjoy writing and reading code now. As if that wasn't enough, it is also just a super fun week with some really cool people. The attendees were all (!) extremely nice, friendly, open, and I really enjoyed getting to scientists from such varied backgrounds, which is unusual as one often stays in their own field for summer schools. And the organizers/teachers were very, very passionate and dedicated, I really cannot thank them enough for all the time and energy they've spent in developing the materials and making the school happen.
12. I really enjoyed the school, which vastly exceeded my expectations! I have tried several times to improve my programming skills by following courses online but they never really fitted my needs as a programmer who learned by doing (having ok general coding skills but lacking structure). This school was finally spot on for my needs -- thank you so much for building it!

Here, I found a really well thought-through curriculum and cheatsheets I am already returning to as I implement some of it in my own work. I also found a really warm and welcoming environment where student programmers of different levels collaborate on everything, which felt very encouraging and helpful for all. I will do my best to model this in my research collaborations :)
13. I understood going in that classes were designed to best accommodate the resources/logistics, lack of funding as well as the limited free time of the instructors - especially since many of them took paid time off to do this, and I am grateful for this. That said, if circumstances were different, more spaced out classes with shorter working days (e.g. 9.00-17.00/18.00 with shorter breaks) would have been nicer. The course is quite intense, but I have no regrets! Big thank you to the organisers once again :)
14. I went through the introductory material and I had the impression that not all of it was necessary to follow the lectures - parts of it were covered again during the lectures.
15. I think the time for lectures could be reduced or made more compact
16. Attending ASPP was an incredible useful experience. The program covered a remarkable amount of practical knowledge and valuable skills in a short timeframe, all of which will be directly relevant to my ongoing work. I am very grateful for the opportunity to have participated. Thank you very much for putting together such a well-run summer school!
17. I would generally recommend the ASPP. It was very valuable for me to dedicate one week in a supportive environment, without assistance of GenAI tools, to learn and practice programming with peers.