

Governing the Software Commons

Shauna Gordon-McKeon

the tragedy of the commons

individual short term
interest
>
collective long term
interest



- literal commons
- many problems, including climate change
- maybe also free software

“Ruin is the destination toward which all men rush, each pursuing his own best interest in a society that believes in the freedom of the commons. Freedom in a commons brings ruin to all.”

- Garrett Hardin, popularizer of the term “Tragedy of the Commons”, in a 1968 article in *Science*

- dramatic
- many of us instinctively disagree
- but, maybe: realistic acceptance of the inevitably selfish nature of humanity?

“Ruin is the destination toward which all men rush, each pursuing his own best interest in a society that believes in the freedom of the commons. Freedom in a commons brings ruin to all.”

– Garrett Hardin, popularizer of the term “Tragedy of the Commons”, in a 1968 article in *Science*

“Freedom To Breed Is Intolerable [...] To couple the concept of freedom to breed with the belief that everyone born has an equal right to the commons is to lock the world into a tragic course of action. [...] If we love the truth we must openly deny the validity of the Universal Declaration of Human Rights.”

– Garrett Hardin, white supremacist and anti-immigrant activist, in that same 1968 article in *Science*

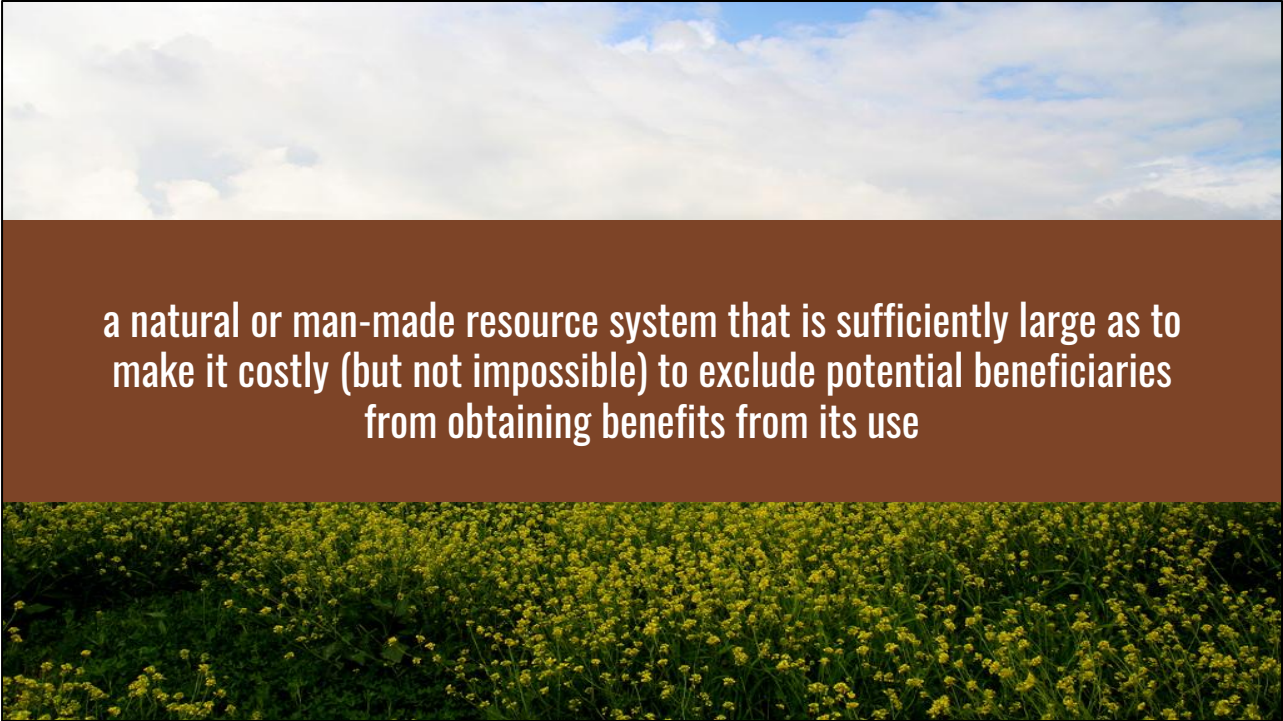
- Garrett was racist, anti-immigrant, promoted racial low IQ theories
- while it's good to be realistic, it's important to recognize where cynicism leads
- if we embrace our selfish sides, there's no bottom
- that's the biggest tragedy

Elinor Ostrom

1933 - 2012
Nobel Prize winner 2009

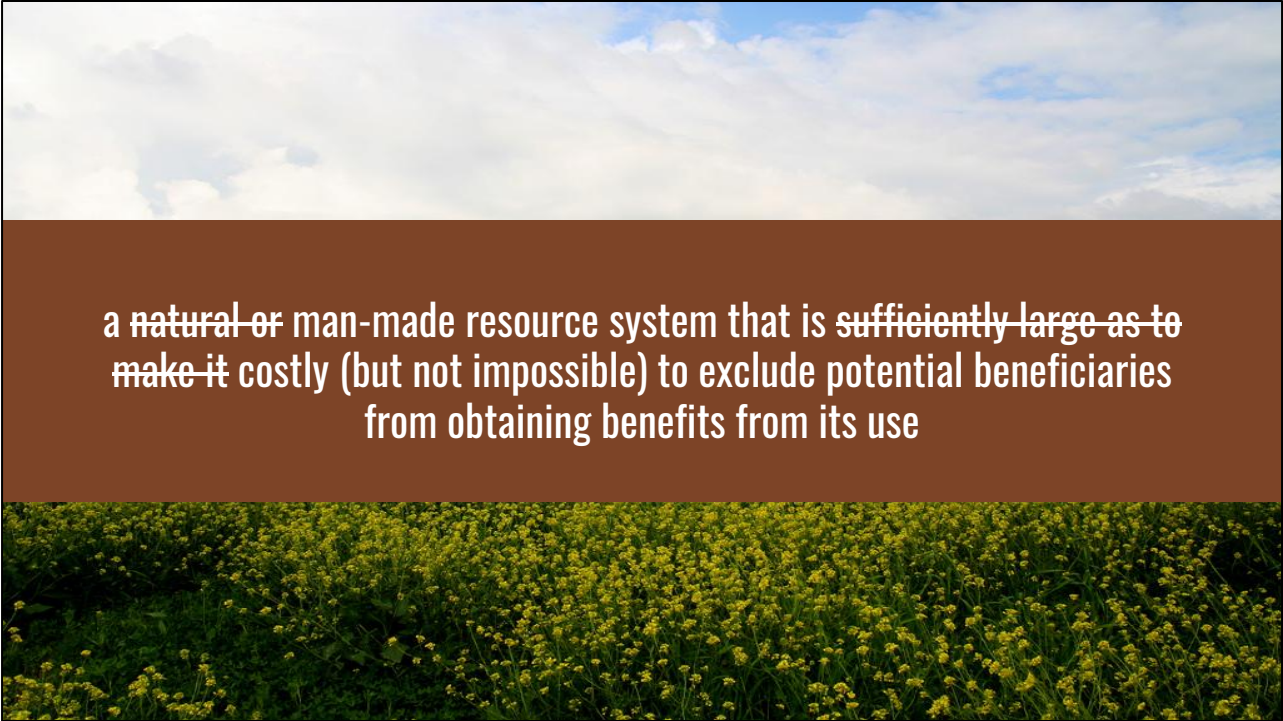


- won a nobel prize for her work (only woman to win economics nobel)
- Governing the Commons analyzes over a dozen commonses
 - from 800 year old land management among Swiss villages
 - to twentieth century Southern California water reserves



a natural or man-made resource system that is sufficiently large as to make it costly (but not impossible) to exclude potential beneficiaries from obtaining benefits from its use

- Ostrom called these commonses “common pool resource”s



a natural or man-made resource system that is sufficiently large as to make it costly (but not impossible) to exclude potential beneficiaries from obtaining benefits from its use

- You may be thinking, “Shauna! It *is* impossible to exclude beneficiaries.” It’s free software! The code must be made available to everyone!
 - Code is not the only benefit of a free software project.
 - Projects are also:
 - social communities
 - sources of reputation and meaning
 - user support systems
 - established customized project management processes
 - resource for fixing your bugs and building your features

“Don’t like it? Just fork it.”

- how many of you have seen people say this, when conflict arises?
- if code was the only thing of value in free software, people would fork all the time. but they don’t. it’s actually pretty rare.
- “pay for support” and “pay for customization” models
 - very popular - for example, Wordpress
 - exclude people from access to labor and specialized knowledge
- think about free software holistically, not just as code base
- back to Ostrom... she came up with 8 principles



Principle 1: Those who have rights to withdraw resources from a CPR must be clearly defined, as must the boundaries of the CPR itself.

- many think only of free software licenses when drawing boundaries. the resource is the code, and the people who can use it is everyone.
- what about:
 - FS as social resource - who has the right to socialize?
 - FS as user support - who gets questions answered or training?
 - FS as new features - who participates in roadmapping?
- might want to say “everyone” again - not bad answer, but not only answer
 - if your FS project is underresourced, in conflict, it might be time to redraw some boundaries



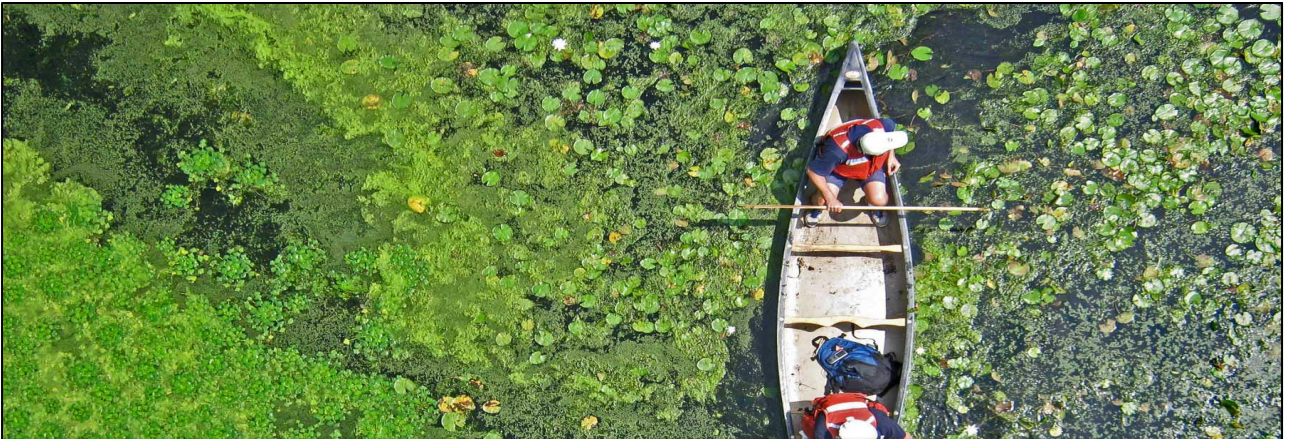
- may not like excluding people
- remember: CPRs draw boundaries based on BEHAVIOR not identity
- FS projects should not be “open to anyone” but rather “open to anyone who behaves in line with our rules and standards”
- this is how CoCs work!
 - different codes & boundaries for different communities is ok
 - in fact, it's vital

Principle 2: Rules in use are well matched to local needs and conditions

- our projects are full of “rules in use”, though many are informal or implicit.
 - “who can push to production” = “who knows the password”, or “who can delete a post” = “who created the post”
 - “who can help build the roadmap” = “who the founder is chatting with on IRC or over a beer”
- it’s good to be INTENTIONAL
- I often recommend projects learn from others or use templates, but you need to adapt them
 - or they’ll do more harm than good

Principle 3. Individuals affected by these rules can usually participate in modifying the rules.

- Huh, principle 3 sure sounds familiar, doesn't it? The people affected by the rules - you could say they're the "users" of the rules - have a "right to modify" them?
- Why is the right to modify so important? Because those who actually experience something - whether it's a law, or a piece of software - know best how it's impacting them and how to fix any flaws. And there will be flaws.



4. Behavior is monitored, and monitors are accountable to the people whose behavior they're monitoring.

- combo of two great cliches - “rules don’t enforce themselves” and “who watches the watchmen”
- Ostrom example: irrigation rotation system
- Jeff Warren & Public Lab:
 - 3 part notification system: transparency, accountability, community knowledge
 - transparency educates community, so they end up gently enforcing the CoC just through mild social pressure
 - “virtuous cycle”

Principle 5: People who violate rules are assessed graduated sanctions, depending on the seriousness and context of the offense.

- context is important!
- Ostrom example: economic distress, monitors remained silent, enhances trust and desire to return to community norms

Principle 6: Members have access to low-cost arenas to resolve conflicts.

Principle 7: The rights of appropriators to devise their own institutions are not challenged by external governmental authorities.

- Ostrom refers largely to state interference
- Not the only way for external pressure to occur
- Community norms that excluding people is “not open” or that charging money for free software is “greedy” can interfere with projects abilities to draw boundaries

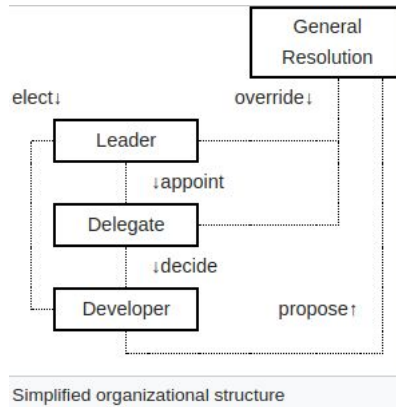
Principle 8: Boundaries, enforcement, conflict resolution, etc are organized in multiple layers of nested enterprises.

- also straightforward, harkens back to second principle of customization

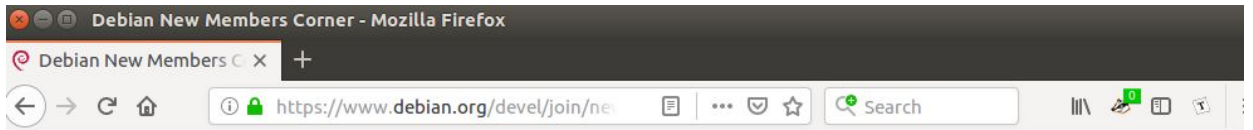
Let's get down to details.



- among oldest & most popular linux distros
- 50,000+ packages
- any DD's in the audience? I'm sure you know way more... I'll do my best



- subset of community members called Debian Developers
 - largely autonomous over their individual packages
 - but Project Leader elected to handle conflicts, overlaps, missing areas
 - PL can overrule individuals but be overruled by DDs through a general resolution although this is very rare
 - PL can delegate their powers to others
 - neat caveats:
 - some things only a delegate can do, such as expel a DD
 - constitution explicitly says this is done to avoid concentration of power in the hands of one individual
 - project leader can't override delegate once decision has been made
- Carefully crafted, customized, aimed at letting people who are affected by the rules create them.



In other words, becoming a Debian Developer grants you several important privileges regarding the project's infrastructure. Obviously this requires a great deal of trust in and commitment by the applicant.

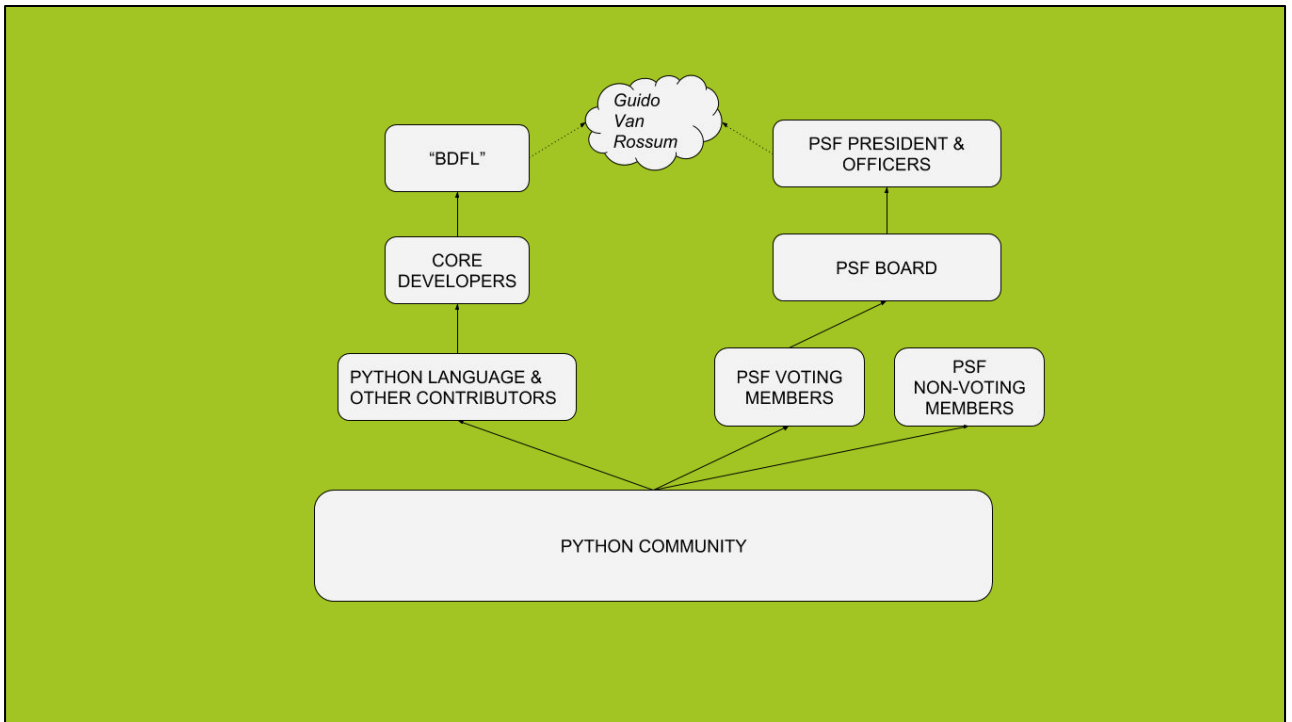
Consequently the whole NM process is very strict and thorough. This is not meant to discourage people interested in becoming a registered developer, but it does explain why the New Member process takes so much time.

- Membership process is unique among free software projects
- Don't have to be a DD to be a maintainer
- Screenshot/image explains motivations
- Steps:
 - Verifying their identity with at least two members through public key signing.
 - Showing understanding of project values, which often takes the form of a written explanation of the Debian Social Contract
 - Demonstrating understanding of Debian's licensing and other project tools and processes
 - Working on various tasks agreed upon by the applicant and the application manager
 - Typically have been maintaining packages for 6 months

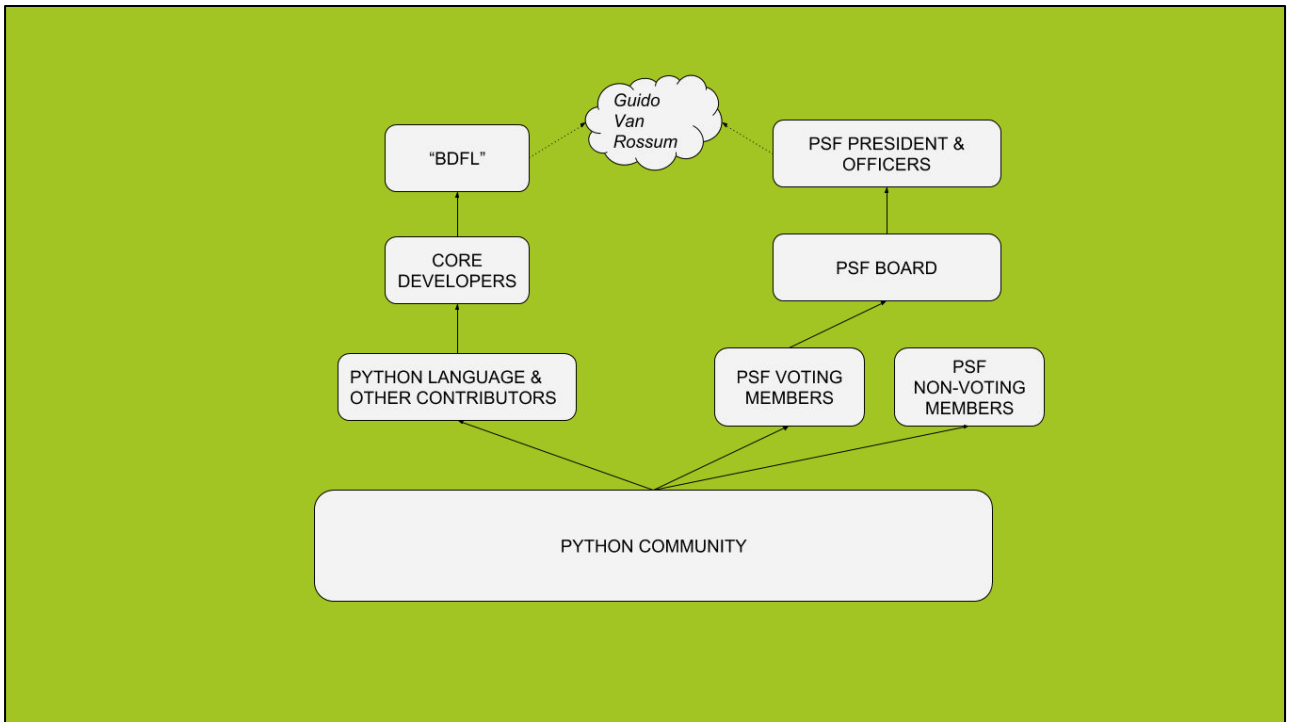
No research, but I bet DDs are more committed and stick around longer.



- 30 years, 3rd most popular programming language (after Java and C)
- fun fact, in 2000 Guido van Rossum used phrase “this Python license is governed by the laws of the ‘State of Virginia’, FSF reached out to him and he changed it, and a year later the FSF gave him 2001 Advancement of Free Software award
 - heads up, 2019 awards will be given out in just a few hours, at the end of the day, in the keynote room



- two branches, PSF and what I'll all the Python language, largely separate
- PSF manages intellectual property
- PSF membership:
 - 5 kinds: basic, supporting, contributing, managing, and fellows
 - all but basic can vote
 - Supporting members “buy the right to vote” but many other ways to get voting rights, and threshold is small (\$99 a year)
 - Voting:
 - must declare at start of year
 - must maintain right by exercising it
 - bylaws: “Voting is a use-it-or-lose-it privilege; missing four votes within a calendar year removes voting privileges for the rest of the year.”



- on the Python language side
 - includes code, issue trackers, etc, but also Python Enhancement Proposals, or PEPs
 - previously Guido had authority to accept or reject PEPs
 - Sometimes BDFL-delegate (you'll note the similar language to PL delegate in Debian)
 - contentious debate, Guido resigned as BDFL, leading to the question:
 - “Who is going to govern the project?” which led them to the question
 - “How do we decide who is going to govern the project?” which led them to the question
 - “How do we decide how we're going to decide who is going to govern the project?”

Step 1: How do we decide what the governance model will be?

- so, okay, step 1.
- PEP 8001, which says that it's summarizing an in person meeting of core developers as well as a discussion thread on discuss.python.org.
- Their answer: we'll ask people to propose various models by submitting PEPs, and then we'll vote on them, via the process outlined in another PEP, 8002.

Step 1: How do we decide what the governance model will be?

Step 2: What will the governance model be?

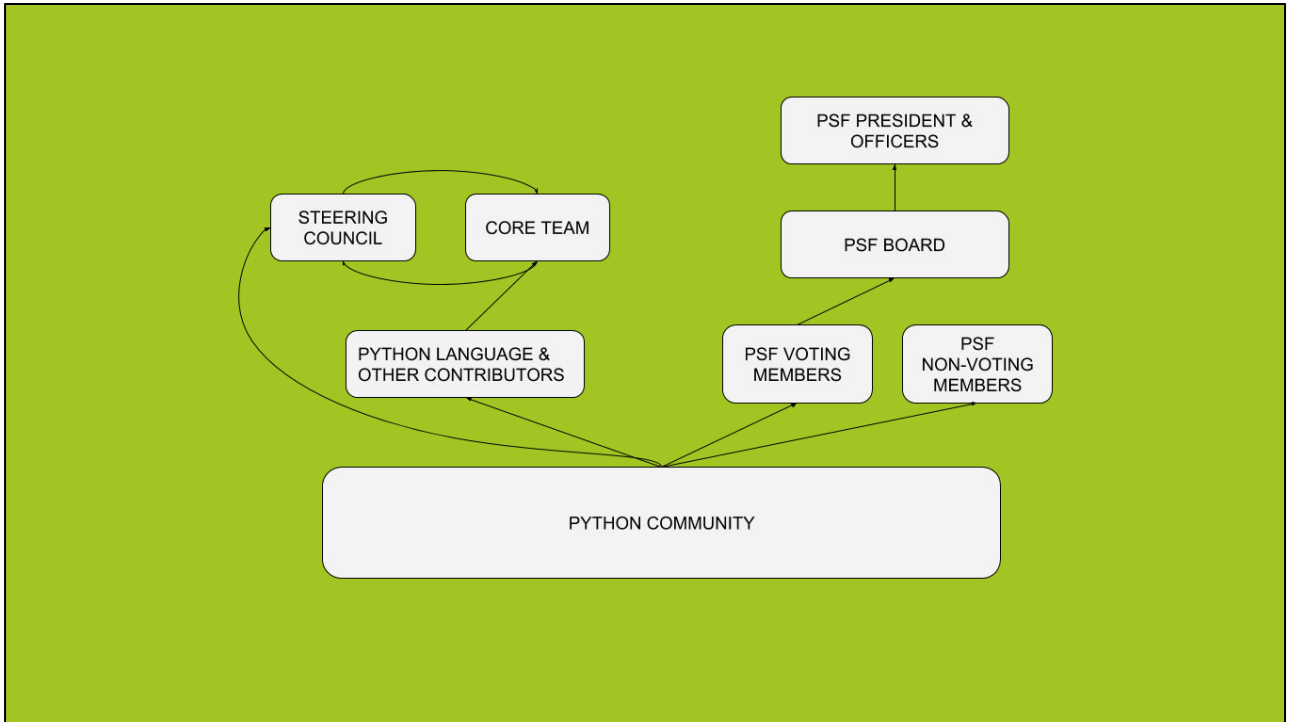
- “open source governance survey” in PEP 8002
- series of proposals in PEP form
- selected in pep 8106

Step 1: How do we decide what the governance model will be?

Step 2: What will the governance model be?

Step 3: Now that we've got the governance model, who will govern?

- First steering council election in January.
- Had to start at a foundational level and build from there.
- Ostrom calls these levels “institutional choice” as contrasted with “operational choice”.



and you end up with something like this.

not enough time to go into details, but a couple of things to highlight.

Python prioritizes “investors”

“we are asking core developers to self-select based on whether the governance situation will affect them directly” - PEP 8001

- Push people who are invested to make the rules:
 - During recent governance changes, PEP 8001 says: “we are asking core developers to self-select based on whether the governance situation will affect them directly”
 - See also PSF:
 - No voting rights for basic members in PSF
 - Must maintain voting activity in PSF

PEP 8002 -- Open Source Governance Survey

PEP:	8002
Title:	Open Source Governance Survey

Python “copies shamelessly”



Working discussion for PEP 8016: The boringest possible steering council mo

■ Ideas ■ governance

- There are lots of major projects that use this kind of mostly-hands-off steering council model (e.g., Django, NumPy, Node.JS, ...), so it's “boring”, in the [good way](#) that mature technology is boring.
- By narrowing our scope like this, it becomes easier to be really rigorous in defining the details, so we can have more confidence we haven't missed anything important.
- By deferring governance details to later, we keep the flexibility to fine-tune processes and experiment with new decision-making workflows *without* having to go through this process again. (For example: if we want to try out PEP 8013, the steering council could start by [appointing an auditor to handle a single PEP, and see how it goes.](#))

- open source gov survey
- steering model “copies shamelessly”
 - but if you look at the threads, you can see they also customized
 - for instance they limited the number of people on the steering council from same employer
 - perhaps reflecting the greater involvement of industry in Python development



The Carpentries - software and data science training aimed primarily at academics

- I want to briefly highlight their nested governance:
 - many lessons across a variety of topics
 - so they've spun off lessons into "Lesson Programs", each of which has to meet requirements for good governance but also has freedom within that
 - and Carpentries themselves have a system of governance



Mapbox itself is not a free software project, but they're a company which is a prolific producer of free software, with 600+ freely licensed repositories, and they provide services on top of them.

I'll be honest, no idea of ratio of proprietary to free code in their services.

Anyway, they have one line in their Terms of Service which caught my eye.

Unlawful and other unauthorized uses

- 30 You agree to comply with all applicable laws, regulations, and third party agreements in your use of the Services. You are not, and are not controlled by an individual, organization or entity organized or located in a country or territory that is the target of sanctions imposed, or on any restricted or sanctioned party list maintained, by the U.S. Government, the European Union or the United Kingdom.
- 31 You may not use the Services in any manner that could damage or overburden the Services or interfere with any other party's use of the Services.
- 32 You may not use the Services to:
1. Disseminate material that is abusive, obscene, pornographic, defamatory, harassing, grossly offensive, vulgar, threatening, or malicious;
 2. Aid or implementing practices violating basic human rights or civil liberties. For the avoidance of doubt, you may not use the Services to assist in the creation of databases of identifying information for any government to abrogate any human rights, civil rights, or civil liberties of individuals on the basis of race, gender or gender identity, sexual orientation, religion, or national origin;
 3. Disseminate or store material that infringes the copyright, trademark, patent, trade secret, or other intellectual property right of any person;
 4. Create a false identity or otherwise attempting to mislead others as to the identity or origin of any communication;
 5. Export, re-export, or permit downloading of any content in violation of any export or import law, regulation, or restriction of the United States and its agencies or authorities, or without all required approvals, licenses, or exemptions;

I for one am so used to seeing long, restrictive, exploitative terms of services that it's hard to think of them as a potential source for good.

But they're drawing a boundary here. Saying this behavior is unacceptable in our community, and if you violate that, you don't get to use our services. They can't do that with the code itself, the licensed code. But they can do that with the services that are built upon it.

“Free software community” as a commons

- beyond individual projects - the community as a whole is a commons
- there are organizations who work across the community as a whole:
 - orgs like the Free Software Foundation and Open Source Initiative
 - people try to support newcomer outreach and greater diversity through projects like Outreachy
 - join groups like Software Freedom Conservancy and Open Invention Network, both of which are collectives which protect projects legally
 - also efforts to support “infrastructure” like the Linux Foundations “Core Infrastructure Initiative”

Acunote's Open Source Initiatives

Unlimited Free Acunote Account For OSS

Jun 5, 2018 - GitLab 

GitLab Ultimate and Gold now free for education and open source

Our top-tier SaaS and self-hosted offerings are now free for education and open source projects. Find out how to apply.

Error monitoring for your open source projects

Acunote is committed to supporting the open source community by providing open source projects with bugsnag for free.

Free testing for open source projects

Sauce Labs relies on open source software for our cloud-based testing platform, so we enable qualified open source projects to use our platform for free. We think that's a fair trade.

PRICING PLANS

- there's also a fair amount of spontaneous treating each other as a community
- most of these initiatives work by saying "if your code is licensed in X way, you qualify"
 - alternatives:
 - "if your project donates 1% of any commercial profits to the Free Software Infrastructure Fund"
 - "if you're an ethical tech certified organization."
 - The key here is that some legitimate authority draws the boundary. What does it mean to be legitimate? Well, ideally you follow the principles of commons management, including clear boundaries, accountability for rule enforcement, self-governance, etc.
 - But in the absence of an established system, people will default to the closest thing to an authority.

Please note that per our service plan, using Open Sauce means your tests and results are open to the public, and usage is subject to verification. Projects that adhere to [best practices](#) from a community perspective, including a license, are also given a leg up.

[About](#)

[Contribute](#)

English ▾

Open Source Guides

Open source software is made by people just like you. Learn how to launch and grow your project.

Background

Open Source Guides were created and are curated by GitHub, along with input from outside community reviewers, but they are not exclusive to GitHub products. One reason we started this project is because we felt that there weren't enough resources for people creating open source projects.

Commons management helps us address systemic problems

- lack of resources for vital projects
- lack of diversity in our community
- the use of free software for unethical purposes

- community stuff can be hard, especially if it's not your forte
- but interpersonal and community management skills can be learned!
- there's plenty of people in our communities already with these skills, but they can be hard to find since we undervalue them
 - how many more people could we bring into the community if we told them how much we value their skills?

project -> community

Some people think that a community forms around a project because it's successful.

community -> project

Other people think that a community is what creates a successful project.

community = project

But to me, the community is the project.

Without the community, code just sits in a repository quickly getting out of date, and falling out of memory of anyone who might have used it.

You build projects by building community, you protect and nourish projects by protecting and nourishing the community. We're all part of the commons. And you, every last one of you, is so much more important and more vital to the community, than whatever code you happen to write.

We've got a lot of challenges ahead of us, and I'm glad to have each irreplaceable one of you with me as we learn how to govern the software commons together.

Thank you.

References and Resources

“The ‘Tragedy of the Commons’ was invented by a white supremacist based on a false history, and it’s toxic bullshit” *Cory Doctorow*, [BoingBoing](#), 3/9/2019.

“Governing the Commons: The Evolution of Institutions for Collective Action” *Elinor Ostrom*, [Cambridge University Press](#), 1990.

Projects referenced:

- Public Lab: <https://publiclab.org>
- Debian: <https://www.debian.org> ([Constitution](#); [Voting Record](#); [New Member Process](#))
- Python: <https://www.python.org> ([PSF Membership](#); [PEP 13](#); [PEP 8001](#))
- Django: <https://www.djangoproject.com> ([Governance](#))
- Jupyter: <https://jupyter.org> ([Governance](#))
- The Carpentries: <https://docs.carpentries.org/> ([Governance](#))

References and Resources (cont)

Cross-project initiatives referenced:

- Free Software Foundation: <https://www.fsf.org>
- Open Source Initiative: <https://opensource.org>
- Outreachy: <https://www.outreachy.org>
- Software Freedom Conservancy: <https://sfconservancy.org>
- Open Invention Network: <https://www.openinventionnetwork.com>
- Core Infrastructure Initiative: <https://www.coreinfrastructure.org>

Image Credits

[“Tragedy and Comedy”](#) by Tim Green, October 16 2012, CC BY 2.0.

[“Press conference with the laureates of the memorial prize in economic sciences 2009 at the KVA: Elinor Ostrom”](#) by Holger Motzkau, 2010, CC BY-SA 3.0.

[“field I”](#) by Jay Erickson, May 15 2008, CC BY 2.0.

[“boundary”](#) by Helen Cook, July 24 2008, CC BY-SA 2.0.

[“Keep out!”](#) by Henry Burrows, May 26 2017, CC BY-SA 2.0.

[“untitled”](#) by Chris Fastie, date unknown, CC-BY-SA.

Debian logo: <https://www.debian.org/logos/> Python logo: <https://www.python.org/community/logos/>
Carpentries logo: <https://datacarpentry.org/blog/2018/02/new-carpentries-logo>