Lesson 2:
Working in Teams

Creating a culture of collaboration

Reference:
Research Software Engineering with Python by Damien Irving, Kate Hertweck, Luke Johnston, Joel Ostblom, Charlotte Wickham, and Greg Wilson
https://merely-useful.tech/py-rse/teams.html

What makes up a project?

- A **dataset** being used by several research projects?
  - raw data, programs used to tidy the data, tidied data, text files describing license and provenance
- A **set of annual reports** written for an NGO?
  - jupyter notebooks, copies of html and pdf versions of the report, a text file containing links to datasets used in the report (which can’t be store don Github since they contain personal identifying information)
- A **software library** providing an interactive glossary of data science terms in both Python and R?
  - files needed to create a package, a Markdown full of terms and definitions, a Makefile with targets to check cross references, etc

What makes up a project?

- One way to decide what makes up a project is to ask what people have meetings about.
- If the same group needs to get together on a regular basis to talk about something, that “something” probably deserves its own repository.
- And if the list of people changes slowly over time but the meetings continue, that’s an even stronger sign.

Establish a Code of Conduct

1. Promotes fairness within a group
2. Ensures all members that this project takes inclusion seriously
3. Ensures that everyone knows what the rules are
4. Prevents anyone who misbehaves from pretending that they didn't know what the did was unacceptable

Option: Create a file called `CONDUCT.md` in the project's root directory.

(Explore and/or adopt the Contributor Covenant, which is relevant for projects being developed online! https://www.contributor-covenant.org/)

Include a License

- A license dictates how project materials can be used and redistributed
- If the license or a publication agreement makes it difficult for people to contribute, the project is less likely to attract new members
- The choice of license is crucial to the project's long-term sustainability

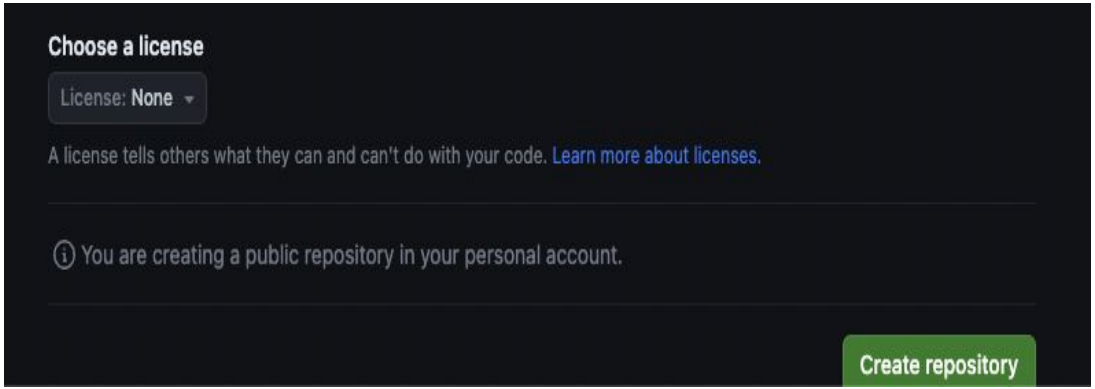
- Members of the team may have different levels of copyright protection.
    - For example, students and faculty may have a copyright on the research work they produce, but university staff members may not, since their employment agreement may state that what they create on the job belongs to their employer.
- Including an explicit license avoids legal messiness, and should be chosen early on. (Licenses do not apply retrospectively)

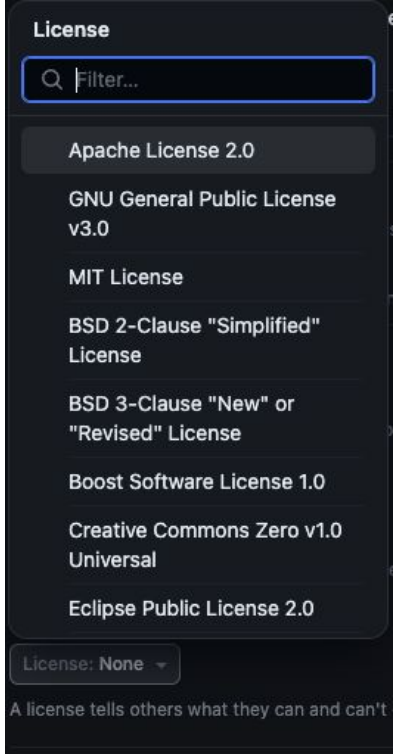
Include a License

1. Do we want to license the work at all?
2. Is the content we are licensing source code?
3. Do we require people distributing derivative works to also distribute their code?
4. Do we want to address patent rights?
5. Is our license compatible with the licenses of the software we depend on?
6. Do our institutions have any policies that may overrule our choices?
7. Are there any copyright experts within our institution who can assist us?

Store `LICENSE.md` in the project's root directory.

Include a License





GitHub allows us to select one of several common software licenses when creating a repository.

https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/licensing-a-repository#choosing-the-right-license

Common Licenses for Software

MIT License:

- Allows use, modification, merging, publishing, distributing, and sublicensing.
- Requires citation of the original source.
- Authors disclaim all liability.

GNU Public License (GPL):

- Similar to MIT but requires users to share modifications under the same GPL terms.
- If someone modifies GPL-licensed software or incorporates it into their own project, and then distributes what they've created, they have to distribute the source code for their own work at well.
- Developed to prevent exploitation of open software without contribution.

Common Licenses for Software

Hippocratic License:

- A newer option, gaining popularity.
- Requires users to avoid harm, aligned with the Universal Declaration of Human Rights.
- Aimed at preventing misuse of software and science.

Common Licenses for Data and Reports

The most widely used family of licenses are those produced by Creative Commons. https://creativecommons.org/

- **CC-0 License ("Zero Restrictions"):**
  - Puts work in the public domain, allowing unrestricted use
  - Ideal for data, since it simplifies aggregate analysis involving dataset from different sources
  - Does not legally mandate, but encourages,citing sources
- **CC-BY (Creative Commons–Attribution):**
  - Allows any use with the requirement of crediting the original source
  - Recommended for manuscripts to ensure credit while promoting wide sharing
- **Other CC Licenses:**
  - ND (No Derivative Works): Prohibits modifications of the work
  - SA (Share-Alike): Requires sharing derivative works under identical terms
  - NC (No Commercial Use): Restricts commercial use without explicit permission

Planning

How do contributors know what they should actually be doing?

Issue tracking systems keep track of tasks we need to complete or problems to fix.

Issues are sometimes called **tickets**, so issue tracking systems are sometimes called **ticketing systems**.

GitHub issues

GitHub allows participants to create issues for a project, comment on existing issues, and search all available issues.

Every issue can hold:

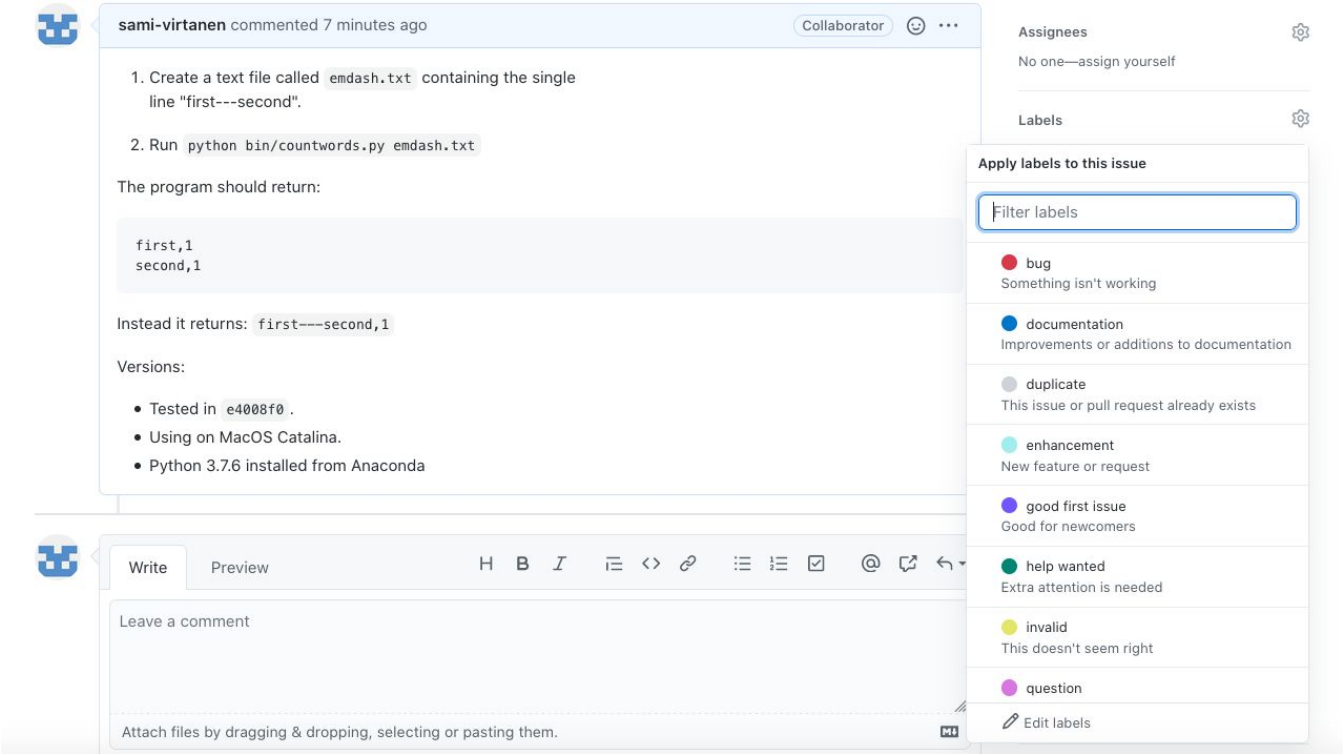
- A **unique ID,** such as #123, which is also part of its URL.
- A one-line **title** to aid browsing and search.
- The issue's **current status**. In simple systems (like GitHub's) each issue is either open or closed, and by default, only open issues are displayed.
- The **user ID of the issue's creator**. Just as #123 refers to a particular issue, @name is automatically translated into a link to that person. The IDs of people who have commented on it or modified it are embedded in the issue's history, which helps figure out who to talk to about what.
- The **user ID of the person assigned** to review the issue, if someone is assigned.
- A **full description** that may include screenshots, error messages, and anything else that can be put in a web page.
- Replies, counter-replies, and so on from people who are interested in the issue.

GitHub issues

Broadly speaking, people create three kinds of issues:

1. **Bug reports** to describe problems they have encountered.
2. **Feature requests** describing what could be done next, such as “add this function to this package” or “add a menu to the website.”
3. **Questions** about how to use the software, how parts of the project work, or its future directions. These can eventually turn into **bug reports or feature requests**, and can often be **recycled as documentation**.

Labels


sami-virtanen commented 7 minutes ago
Collaborator
1. Create a text file called emdash.txt containing the single line "first---second".
2. Run python bin/countwords.py emdash.txt
The program should return:
first,1
second,1
Instead it returns: first---second,1
Versions:
• Tested in e4008f0 .
• Using on MacOS Catalina.
• Python 3.7.6 installed from Anaconda
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Assignees
No one—assign yourself
Labels
Apply labels to this issue
Filter labels
bug
Something isn't working
documentation
Improvements or additions to documentation
duplicate
This issue or pull request already exists
enhancement
New feature or request
good first issue
Good for newcomers
help wanted
Extra attention is needed
invalid
This doesn't seem right
question
Edit labels

Labelling Issues

A small project should always use some variation on these three:

- ***Bug*:** something should work but doesn’t.
- ***Enhancement*:** something that someone wants added to the software.
- ***Task*:** something needs to be done, but won’t show up in code (e.g., organizing the next team meeting).

Labelling Issues

Projects also often use:

- ***Question*:** where is something or how is something supposed to work? These can be recycled as documentation.
- ***Discussion* or *Proposal*:** something the team needs to make a decision about or a concrete proposal to resolve such a discussion. All issues can have discussion: this category is for issues that start that way. (Issues that are initially labeled *Question* are often relabeled *Discussion* or *Proposal* after some back and forth.)
- ***Suitable for Newcomer* or *Beginner-Friendly***: to identify an easy starting point for someone who has just joined the project.

Labelling Issues

Indicating the state of an issue:

- ***Urgent*:** work needs to be done right away. (This label is typically reserved for security fixes).
- ***Current*:** this issue is included in the current round of work.
- ***Next*:** this issue is (probably) going to be included in the next round.
- ***Eventually*:** someone has looked at the issue and believes it needs to be tackled, but there's no immediate plan to do it.
- ***Won't Fix*:** someone has decided that the issue isn't going to be addressed, either because it's out of scope or because it's not actually a bug. Once an issue has been marked this way, it is usually then closed. When this happens, send the issue's creator a note explaining why the issue won't be addressed and encourage them to continue working with the project.
- ***Duplicate*:** this issue is a duplicate of one that's already in the system. Issues marked this way are usually also then closed; this is another opportunity to encourage people to stay involved.

Labelling Issues

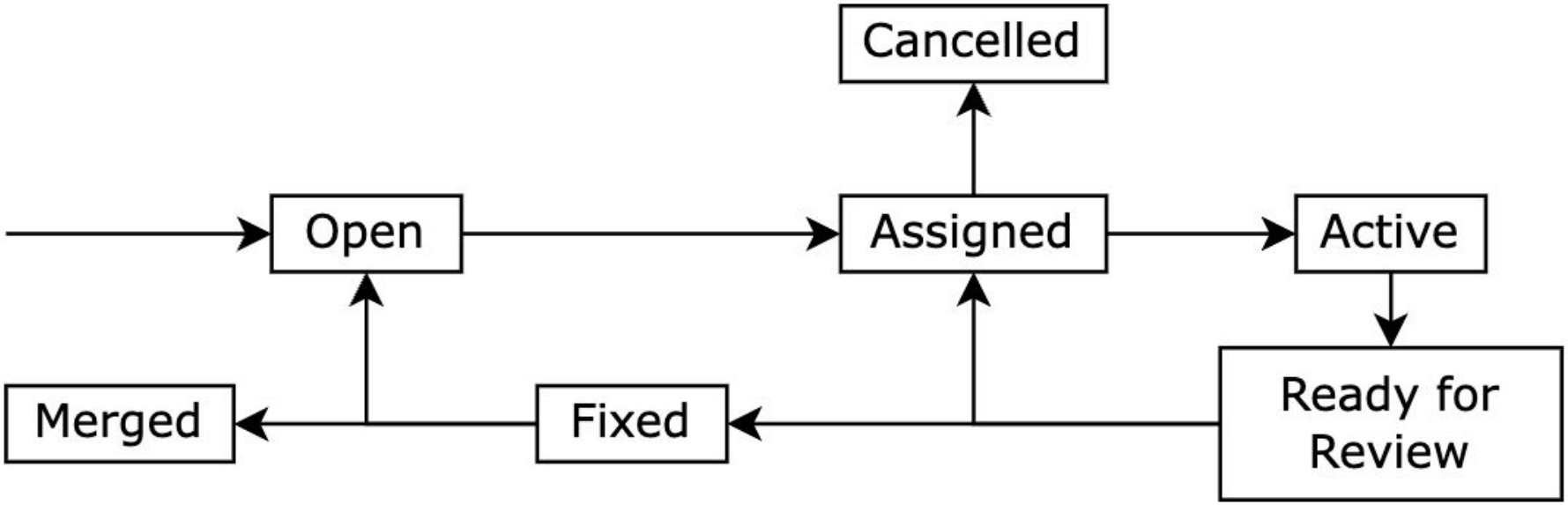


Figure 8.3: Example of issue lifecycle.

Prioritizing

The challenge: Balancing bug fixes, feature development, and project cleanup, especially in research projects where "done" is hard to define.

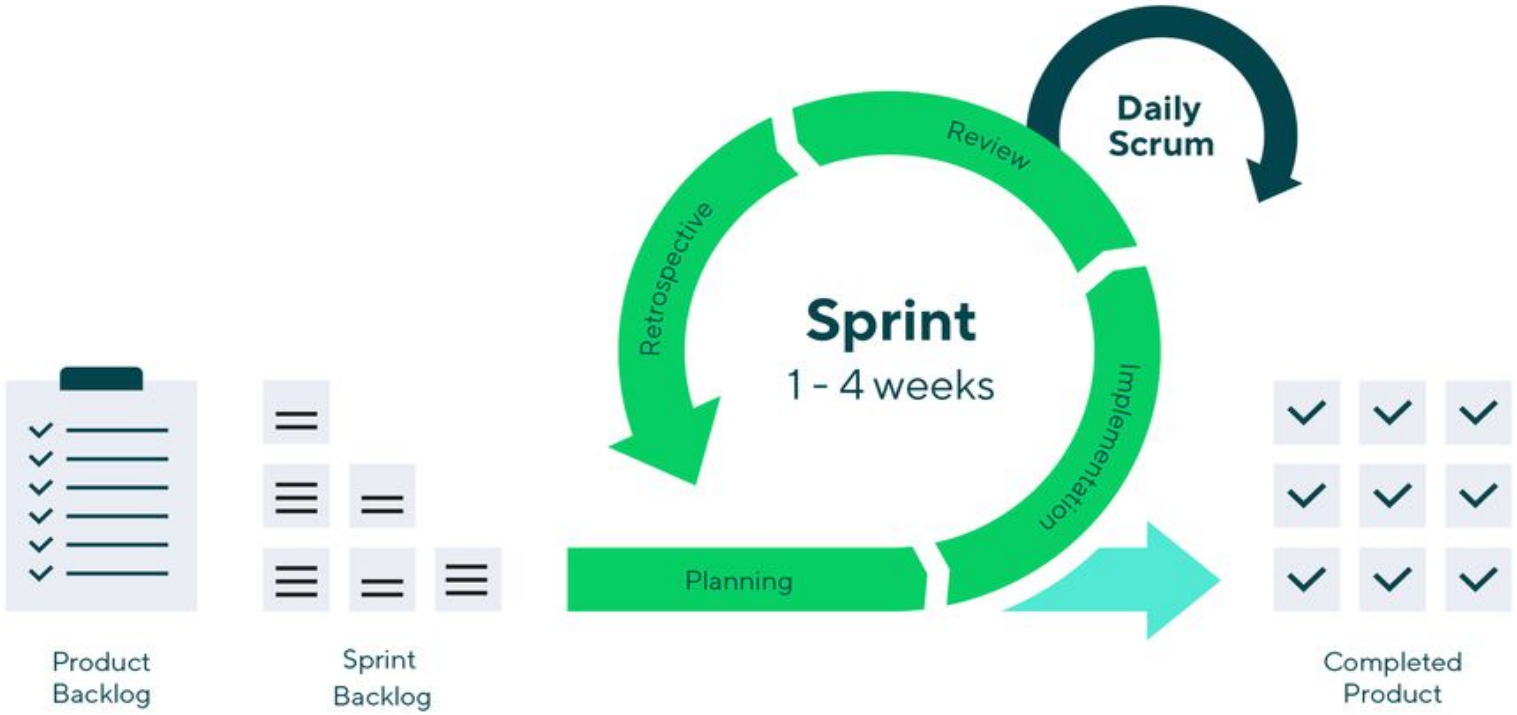
Prioritizing

The challenge: Balancing bug fixes, feature development, and project cleanup, especially in research projects where "done" is hard to define.

The solution: Sprints

- Typically a 1-2 week long cycle where specific tasks are chosen and completed.
- Starts with a planning session in which the successes and failures of the previous sprint are reviewed and issues to be resolved in the current sprint are selected

Sprints


Daily Scrum
Review
Retrospective
Sprint
1 - 4 weeks
Implementation
Planning
Product Backlog
Sprint Backlog
Completed Product

Sprints: Prioritizing

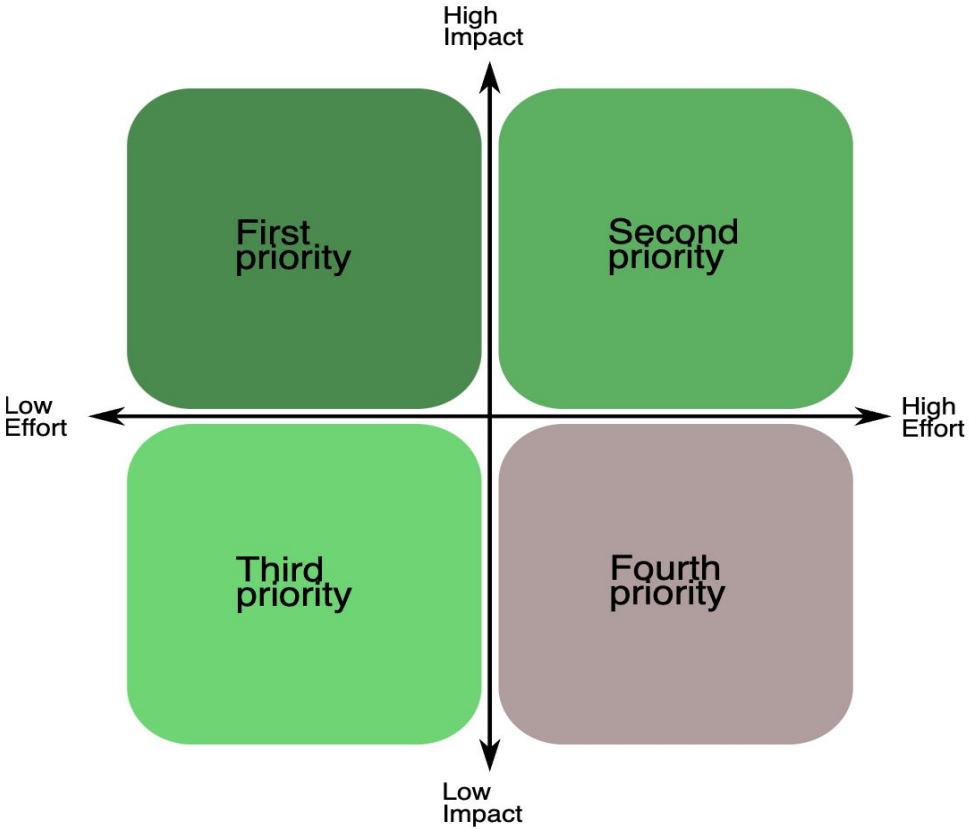


Figure 8.4: An impact/effort matrix.

Decision Making

- Every team has a power structure: formal (accountable) or informal (unaccountable).
- Importance of explicit governance in groups larger than six people.
- Objective: Establish **who makes decisions** and **how to reach consensus**.

Decision Making: Martha's Rules

- Anyone who wants to sponsor a proposal must file one at least 24 hours in advance. It must include:
  - a one-line summary
  - the full text of the proposal
  - any required background information
  - pros and cons
  - possible alternatives
- A quorum is established in a meeting if half or more of voting members are present
- Once a person has sponsored a proposal, they are responsible. The group may not discuss it unless the sponsor is present

Decision Making: Martha's Rules

- After the sponsor presents the proposal, cast a sense vote:
  - Who likes the proposal?
  - Who can live with it?
  - Who is uncomfortable with it?
- If everyone likes or can live with it, it passes with no further discussion.
- If most of the group is uncomfortable, it is sent back to the sponsor for further work. (The sponsor can decide to drop it)
- If some members are uncomfortable, a time is set to discuss, moderated by the meeting moderator.
  - After 10 minutes or so, the moderator calls a yes or no vote.
  - If the majority is yes, it passes.
  - Otherwise, it is returned to sponsor for further work.

Meetings

1. ***Decide if there actually needs to be a meeting.***
2. ***Write an agenda.***
3. ***Include timings in the agenda.***
4. ***Prioritize.***
5. ***Make one person responsible for keeping things moving.***
6. ***Require politeness.***
7. ***No interruptions.***
8. ***No distractions.***
9. ***Take minutes.***
10. ***End early.***

Finally… Make all of this obvious to newcomers!

Document your workflows, meeting rules, governance, licenses and decision making processes in a README file!