INF200_H21_Ju01

May 31, 2022

1 INF200 Lecture No Ju01

1.0.1 30 May 2022

1.1 Today's topics

1. Project overview
2. How we will work
3. Team repositories
4. Project directory structure
5. Development
    - A little bit about agile development
    - Shared repositories, branches and pull requests
6. BioSim — Problem analysis

1.2 1. Project overview

1.2.1 The BioSim Project: See project descriptions file!

1.2.2 Teacher

- Hans Ekkehard Plesser (email: hans.ekkehard.plesser@nmbu.no)

1.2.3 Teaching assistants

- Hans Marius Johnsen
- Hanna Lye Moum
- Mikko Rekstad
- Jan-Eirik Welle Skaar (last week)
- Håkon Mørk (last week)

1.2.4 Exam information

- Oral exam **Monday 20 & Tuesday 21 June**
- Presentation and discussion, approximately 25 minutes
- Each pair of students will **present and discuss together**
- Notify Plesser by email **today** if you have date constraints due to other exams or similar

1.2.5 Schedule

- Mandatory attendance Monday-Friday 09.15-15.00
  - Lectures from 09.15 and meet-ups from 14.30 most days
  - Spontaneous mini-lectures possible if many students encounter similar problems

Date	Event / Milestone
30 May 09:15	Start
30 May 15:00	GitLab Repo and PyCharm Project ready
31 May 15:00	Problem and requirements analysis
02 Jun 15:00	First functional simulation of *herbivores* in one place (no migration)
03 Jun 15:00	Project plan for remaining work
06 Jun	**Pinsemandag—no class**
08 Jun 15:00	Correctly working simulation with herbivores and carnivores in one place (no migration)
10 Jun 15:00	Working simulation of herbivores and carnivores, one type of terrain and simple visualization
15 Jun 15:00	**Hand-in:** Full simulation code incl documentation
17 Jun 15:00	**Hand-in:** PDF and animation for oral presentation
20 & 21 Jun	Oral exam

1.2.6 Deviations from regular lecture schedule

Date	Time	Event
Thu 2 Jun	**13.30**	**Afternoon meet-up**
Fri 3 Jun	**10.15**	Lecture
Mon 6 Jun		**Public holiday, no class**
Tue 7 Jun		**No afternoon meet-up**

1.2.7 Preliminary list of lecture topics

- Lectures will be given as required, usually at 09.15
- The lecture list below is indicative and subject to change on short notice

Date	Topic
30 May	Setting up, Python packages, Team repos
31 May	Some Python features, Packaging
01 Jun	Testing: mocking, fixtures, statistical tests, automated testing
02 Jun	Model Dynamics I, Documenting source code with Sphinx
03 Jun	Efficient visualization
07 Jun	Automated testing

1.3 2. How we will work

1.3.1 Key resources

- GitLab repository with material for the course (https://gitlab.com/nmbu.no/emner/inf200/h2021/inf200-course-materials, `june_block` subdirectory)
    - Project description
    - Lecture notebooks
    - Example code

1.3.2 Daily rythm and work form

Plenary sessions

1. Plenary lectures from 09.15 every day. Depending on content, they will last between 30 and 90 minutes.
2. Short plenary wrap-up sessions from 14.30 every day.
3. **We will check attendance.**

Individual work

1. Individual work in the groups.
2. You shall be in the classroom between 09.15 and 15.00 every workday except for a lunch break.

1.4 3. Team Repositories

- You will work within your `Uxx FirstName1 FirstName2` teams, with team numbers from 01 to 28.
- You will create a repository in this group, see below for details.
- **Remember: Never force a push, always pull before pushing!**

1.4.1 Creating a shared repository on GitLab

The process is similar to creating share repositories in the fall term. A template is available in the `inf200-course-materials` repository. Proceed as follows

1.4.2 Working with the shared repository

1. On your computer, pull all updates from the inf200-course-materials repository.
    - Under `june_block/biosim_template` you will now find several directories (`biosim`, `examples`, `tests` and a `sample.gitignore` file).
2. One of the two partners, shall now do the following on GitLab
    1. In your `Uxx_FirstName1_FirstName2` group on GitLab, create a new project.
    2. Select "Create blank project".
    3. Choose project name `BioSim Uxx FirstName1 FirstName2`. The corresponding "Project slug" will then be `biosim-uxx-firstname1-firstname2`. Here, `xx` is your group number (with leading zero!) and `FirstName1` and `FirstName2` your first names as in the group name.

    4. Check off for "Intialize repository with a README".
3. One of you then does the following on your computer:
    1. Start PyCharm and select `Get from VCS`
    2. Paste the URL from the GitLab clone button and choose a sensible location on your computer.
    3. Once PyCharm has cloned and opened the project, ensure that the correct Python interpreter is chosen (the `inf200` conda environment).
    4. You should now have the project in folder `biosim-uxx-firstname1-firstname2` in your repo.
    5. Now copy all directories and the `sample.gitignore` file from the `biosim_template` folder into the project folder. They should show up in PyCharm after a little moment.
    6. Rename `sample.gitignore` to `.gitignore` by right-clicking the file name in PyCharm and chosing Refactor > Rename.
    7. Commit all new files to Git (either in PyCharm, GitAhead or your preferred git program) and push to GitLab.
4. Now the other partner can clone the project in PyCharm as described above and you are ready to go.

1.4.3 Work planning on GitLab

We will use some of GitLab's project management features: *Issues*, *Milestones*, *Labels*, and *Merge Requests*.

- For each of the milestones related to code developement that is given above and in Table 1 of the project description, create a milestone in GitLab (with due date). You can add more milestones.
- Create labels to mark work related to different aspects of you work, e.g., code, documentation, tests and deployment.
- Create an issue for each thing you need to do and assign it to your project and a milestone and label it.
- Do development work in branches and create merge requests to integrate it into your master branch. Review your requests together with your partner.
- Use the issue board (Kanban board) to manage and track the progress of your work.

For more information, see this video: https://nmbu.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=b8268850-b440-4db1-a7f1-ae1100e63d62.

Important notes:

- Gitlab has changed a bit since the video above was recorded:
    - The "Labels" menu item is now under "Subgroup information" or "Project information", not under "Issues" as in the video.
- In the video, I use GitAhead to work with Git (commits, pushes). You should use PyCharm for working with Git.

1.5 4. Project setup in Conda and PyCharm and directory structure

- A project template directory is available in the course materials repo, see also setup instructions above.
- The template provides the following following directory structure

```
biosim-uxx-<name1>-<name2>
    .gitignore
    README.md
    src
        biosim
            __init__.py
            simulation.py
    reference_examples
        check_sim.py
        mono_hc.py
        mono_ho.py
        sample_sim.py
    tests
        test_biosim_interface.py
```

1.5.1 Notes

- `__init__.py` contains the `__version__` variable for the package.
- Your modules go into directory `src/biosim`.
- Your own scripts using the `biosim` package go into folder `examples` at the same level as `src`, `reference_examples` and `tests`.
- All tests go into `tests`.
- `check_sim.py`, `mono_hc.py`, `mono_ho.py` and `sample_sim.py` test that your code will work with some of my automated tests. The `mono` test work on a single cell.
- We will add more to this structure later.

1.6 5. Development

1.7 A little bit on Agile development

- Agile software development is a modern (2001-) set of software development methods
- Focus on quick delivery, frequent updates, and flexibility, while maintaining quality
  1. Get it to work
  2. Get it right
  3. Get it fast
- We will use in particular three Agile techniques
  - Pair programming
  - Kanban boards (GitLab project)
  - Test-driven development (in a mild version)
- Typical workflow in this project
  - Every morning, review the cards on your board (more often if needed)
    * Any cards “in progress” or “under review” that block further work?

    - Prioritise “to do” cards and select which to start to work on
    - Make sure you do not have more cards “in progress”/“under review” than you can actually work on at one time
  - Move cards between columns as you progress
  - Create an issue when you
    - identify something that needs to be done
    - discover a bug
  - When you create an issue
    - assign it to a milestone
    - assign it to your project and put it into the “to do” column
  - Develop code in branches and integrate them via pull requests
- **Commit often, many times every day!**

1.7.1 A real-life example

For a Github repo of a real-life project using a mild agile approach, see https://github.com/nest/nest-simulator.

1.8 Shared repositories, branches, and merge requests

1.8.1 Developing using branches and merge requests

- In larger and long-living projects, code is precious and must be protected from uncontrolled changes.
- Therefore, changes to the main branch need to be controlled.

1.8.2 Controlled code evolution

- For every change, create a branch
- Develop code in the branch
- Push branch regularly to repo so partner has access, too
- When you think that code is ready for integration to main,
  1. push to repo
  2. go to repo and create a *merge request* (MR, also called *pull request*) to main
  3. let your partner review your code
  4. make changes if required and push, the MR is automatically updated
- Once your partner has approved your MR, merge MR into main

1.8.3 Keeping branches up to date

- Sometimes, branches can live for quite a while
- Meanwhile, `main` will evolve (as MRs are merged in)
- It is then often useful to update the branch with changes in `main`
- To keep the branch up to date with changes in `main`
  1. make sure your branch is checked out
  2. use GitAhead or the git tool of your choice to merge main into your branch
  3. resolve merge conflicts if necessary
  4. push the merged branch to repo

1.8.4 Local and remote branches

- When you create a branch on your computer, it is a *local branch*
- You can only check out and work on local branches
- When you push a branch to the repo, you create a *remote branch*
- By default you local branch will *track* the remote branch
    - You can easily pull changes from and push changes to the remote branch
- Your partner may have to *Fetch* from the repo to see new remote branches you have created
- Once your your partner can see the remote branch in her git tool, she can check it out, creating her own local branch

1.8.5 Exercise

Try this later today if you want to practice a bit with branches

Video: https://nmbu.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=967be389-6e91-4e4b-9484-ae1100e790b3

1. Each of you creates a local branch (choose different names!)
2. Create files `README_<branchname>.md` in your respective branches, fill them with some text
3. Push your branches to the repo
4. Go to GitLab and create a MR from your branch to main
5. Comment on the MR your partner created
6. Make changes to your own branch in response to your partners comments
7. Once your partner approves your MR, merge it to main
8. Let's say A's MR was merged first. Then B should now merge the changes from master into her MR.
9. Now it is time to merge also B's MR into main, provide A is happy with it.
10. Afterwards, both A and B check out main and pull all changes in master from the repo.

Tidy up your shared repository after the exercises! Once you have completed the exercises above

- Delete all branches you created during these exercises
- Checkout `main`
- Delete the `README_<branchname>.md` files
- Edit your `README.md` file so that it gives a brief introduction to your team and project.

1.9 6. BioSim—Problem analysis

- Work in pairs of teams (teams sitting behind each other)
- Questions
    - What are the key components of the "world" to simulate?
    - What are key aspects of running and visualising a simulation?
    - How can you represent this in software?
- We will discuss results when we reconvene

[ ]: