TA meeting—Didactics and organizational notes

Programming for Life Sciences

Tuesday, 2023-03-14 11:00

Present

- Matthijs
- Roxana
- Clemens
- Ilja
- Oana
- Erik
- Anton
- Veronica
- Steve
- Tsjerk
- Kim

> **Note** For those who could not make it to the meeting: I hope these minutes are sufficient for getting you up to speed. However, many of the points you see we discussed below arose from questions. Thus, I expect there will be more questions. Do not hesitate to ask for clarification in the group or in a private message or email!
>
> —Koen (k.s.s.westendorp@student.rug.nl)

Agenda

- Grading
  - Assignments
  - Project
- General notes
- Tutorials and Lectures
- Didactics

Grading, assignments, and project

The course is divided into 4 blocks over 3 weeks. The first three blocks require the student to hand in an assignment. In the last block, students make a project.

Students work together in groups on the projects and assignments. They can enroll themselves for a group in the Brightspace page. Groups of one person or three persons are also possible, but we generally aim for two participants.

Assignments

For every block, 4 different assignments are available. Students are free to choose one of those 4 assignments to work out and hand in.

Assignments evaluation

Students evaluate the work together with a TA.

Koen, Steve, Veronica will start out doing the evaluations with students. We hope to teach a couple more of the TAs how to do these evaluations, because this will take some of the pressure off of the others. For example, you can sit together with an evaluating TA and learn about what questions they ask and how they interact with the student.

This year we also grade the assignments (see below). This means that the evaluation of assignments will require a bit longer than they did before.

Project

All example projects have a couple of paths the student can walk. These are different ways of working towards the same end goal, or diverging final outcomes. This means that there is a great diversity of projects possible from those starting points.

Students can brainstorm with TAs about their project choice. But, students need to ask Kim and Tsjerk (see *responsiblities*, below) for **written acknowledgement of their project choice**. This is a final check on whether the project choice is viable and of a level that allows the student to show what they are capable of.

A project should not necessarily be *finishable*, but it should be *modular* in a sense.

If the project is modular, it can be extended or shortened to still result in a nice result the student can be proud of, depending on how the student progresses. When a student runs into trouble, they can still finish it satisfactory if the thing they didn't get to is a later 'module'/part of their project. If they are capable of *more* after finishing the goal they initially came up with, they can continue by extending their project.

Some students will propose to make a game. The danger is that they are more concerned with the game mechanics and aesthetics than with the programming techniques. We want them to show they are capable of the latter, and the former is not relevant for this course. A viable option is to make a game that

has pre-established rules and mechanics, because that will show the students capabilities rather than game-design skills.

Grading projects and assignments

Last few years we graded *only* the project, and not the assignments. This year we will grade the project *and* assignments.

- **20%** the accumulated grade for the three assignments.
- **80%** the grade for the final project.

This used to be the case a few years ago. Back then, the main objection was: assignment 1 is due in the first couple of days, which might be too short of a time to give everybody the chance to get up to speed.

In order to account for this, the following structure is in place. Over the assignments, students accumulate 25 points worth of grading, of which 20 points can be selected to count towards the 20% of the grade the assignments represent. This means that if one of the assignments does not work out as well as the others, this grade is not detrimental to the final grade.

Responsibilities

- Tsjerk is responsible for the Biology students.
- Kim is responsible for the Life Science & Technology students.

Evaluation

After and during evaluation, it is important to keep track of a couple of things. Write these down!

- Who was it (name, student number)
- What did they learn
- What do they struggle with
- What was the feeling you got?
- … and more, Whatever stands out …

If something sticks out about somebody (positive or negative), write it down! This is really important information to track, because it allows us to give students what they need. This could be more more involved help or even more challenging work.

General notes

IDEs

We ask students to use Spyder, because it is easy to install on all platforms.

But, students are free to install their own software and IDEs and editors.

Style

We teach in Python written in accordance with pep8. (That is not to say that we tell them about all of the details regarding what pep8 is or about code formatting.) Quickly read through the specification if you are not familiar with it. You will find it is a way of formatting Python code that will be very familiar to you—most people write it this way.

This is easier to read and comprehend, for both students and teachers.

In practice this means that we *show* how they can write their code well and in good style. The code we write to show them is in this style.

Tutorials

Tomorrow morning (Wednesday 2023-03-15), Erik will send out a form where TAs can specify their preferences for teaching tutorials. This will aid Erik and Koen in knowing who is available for which tutorial.

The tutorials, in order, are:

1. Palindromes, text handling
2. Functions
3. Data types
4. NumPy (slicing, indexing, available functions, with a focus on how to use the NumPy documentation)
5. Classes (OOP, thus a bit more advanced, mosaic exercise and encapsulation)

The person assigned to tutorial 5 will sit down with Kim and take a look at how they can do the tutorial. Generally, there will be ample opportunity for teachers of tutorials to sit down with others to talk through how they can get the most out of their time and the students.

Lectures

The first lecture will be on 09:00 20 March 2023. We recommend all TAs who can make it be there, because it is nice to intruduce everybody.

After the lecture, almost nobody will be capable of doing assignment one, yet. First, they will work through the book and do the book exercises.

If you spot somebody diving head-first into the assignement and getting stuck, point them towards working through relevant sections of the book. The book exercises are very valuable practice leading up to the assignments.

The lectures, in order, will concern:

1. Introduction
2. Functions
3. Data types
4. Numpy
5. OOP (optional)

Didactics

Mental capacity

Humans only have limited mental capacity. When this 'loading bar' is full, everything added onto it will tend to get lost. Try not to overload your students, and if in doubt, check with others.

One way to prevent overloading a student, is to go *one* step at a time. When they ask for help, help them to find their own way towards the next step, as opposed to giving them a whole sprawling roadmap. The latter would overflow them! (But if you give them enough information, it will overflow and the student just starts at 0)

Helping students

Be conservative with what information you give them. Alleviate roadblocks by pointing them to documentation, the book, ask them to think back to tutorials/lectures. This will not only make them understand one little part—it also introduces them to ways of finding their own way when you are not there.

Some students will be really geared towards getting answers, but what we provide are ways for them to get to the answer.

If you get stuck with a student, call somebody else over.

We are not here to give students the answers, we are here to help students to find the answers themselves.

Ask, 'what if the computer isn't here, and you were to do this by hand, step by step?' Some students really get a problem at *that* moment. Guide them towards *that* moment.

In case you think a student grasps the syntax, but not the way to solve a complex problem in small steps: 'how would I teach this with just pen and paper?' A lot of problems happen because the student skips an implicit step. By walking through their trouble very slowly and explicitly, you can uncover these implicit steps.

A document on didactics

Tsjerk and Koen are working on a document outlining some important thoughts, techniques, and perspectives on teaching (programming). This document should see the light of day by tomorrow afternoon (Wednesday 2023-03-15).