

Bioinformatics 3: Assignment Guide

Winter Semester 2022/23

Due to our diversity of backgrounds, we are all familiar with and used to different procedures, and it is not made easier by the fact that every lecture handles things differently. The goal of this guide is to explain our expectations for Bioinformatics 3 assignments and to set a consistent standard for you and us. With it, we also want to give some starting tips, answer frequently asked questions, and address and clarify issues that come up every semester.

Contents

1	General Information	2
2	Assignment Guidelines	2
2.1	PDF Standards	3
2.2	Code Standards	3
2.3	Templates	4
2.4	Plagiarism and Proper Attribution	4
3	Getting Started	5
3.1	L ^A T _E X	5
3.2	Integrated Development Environments (IDEs)	5
3.3	Python	5

1 General Information

Tutor: There is a specific teaching assistant responsible for each assignment whose name and email address is given at the top of each assignment sheet.

Tutorials: There is going to be a tutorial in Microsoft Teams for each assignment, in which the solution to each task is going to be presented and explained.

Schedule: You can find the assignment and tutorial schedule on the [course website](#).

Graded Assignments: We will try our best to return the graded assignments within a week of submission, before the corresponding tutorial.

Required Points: You can achieve up to 100 points per assignment, for a total of 1,000 points across all 10 assignments. To participate in the final exam, you need at least 500 points (50%) in total. It does not matter whether you achieve that by getting 50+ points in each assignment or e.g. exactly 100 points each in the last 5 assignments and 0 in the first 5 assignments.

Questions: If you have questions, you can send an **email** to the tutor responsible for the current assignment.

2 Assignment Guidelines

Please read these guidelines carefully and make sure you follow them. Not doing so will result in a point deduction depending on the severity.

Language: The assignments have to be written in English.

Groups: While you can work alone, you are advised to work in groups of **two** people. If you do not have a partner, you can use the "Looking for Assignment Group" channel in the Bioinformatics 3 Team in Microsoft Teams. Alternatively, we can help you find a partner after the first assignment has been submitted.

Submission: Submit your solutions per **email** to the current tutor as **two** attachments:

- **BI3_A1_WS22_Lastname1_Lastname2.zip:** The ZIP file should contain all your source code files, potential result files, including plot and image files, and whatever else is needed to generate your solution. You may exclude data files that were provided as part of the supplement if including them would exceed the attachment size limit.
- **BI3_A1_WS22_Lastname1_Lastname2.pdf:** The PDF file should contain your answers **and** your properly formatted source code. Submissions as Word- or simple text files will **not** be considered.

Before you submit, make sure that your submission complies with our [PDF Standards](#) and [Code Standards](#), and that it does not run afoul of our [Plagiarism Rules](#).

Deadline: The deadline for submissions is given on each assignment sheet. Late submissions will **not** be considered.

2.1 PDF Standards

The PDF we ask you to submit is the basis for evaluating your assignments, and it should therefore be well structured and easily readable.

Names: Do not forget to include your name and matriculation number.

Format: It is mandatory to use the \LaTeX -template we provide to generate your PDF. See [Getting Started with \$\LaTeX\$](#) below if this is your first time using \LaTeX .

Answers: If a task asks a question, you need to include your answer in the PDF, not in code comments. Write in complete sentences with proper spelling, grammar and punctuation. If in doubt, run your replies through a spelling checker.

Plots and Images: If a task asks you to generate plots or images, they must be included in the PDF. Make sure that they are of sufficient quality to be readable and that plot axes and curves are properly labelled.

Console Output: Some tasks ask you to print something to the console, in which case the properly formatted console output should be included in the PDF as well. This helps us to quickly narrow down if and where something went wrong and to save points for you. Please do not include console output you just generate for yourself.

Source Code Listing: The PDF must include your properly formatted source code so that we can comment on it and note down our feedback. Yes, that means you essentially submit your source code twice, once in the PDF and once in the ZIP archive. We will **not** layout your source code and merge it into the PDF for you.

2.2 Code Standards

Good coding practices are important to avoid mistakes and so that other people, especially the tutor, are able to understand your code.

Formatting: Make sure that your code is consistently formatted. Have a look at style guides for the language you are using, e.g. [PEP 8](#) for Python. Good [Integrated Development Environments \(IDEs\)](#) will help you with maintaining a good style.

Variables and Comments: Use meaningful variable names, e.g. 'node' instead of 'x', and add comments that describe the gist of what you are trying to do with a certain function, line, or section of code.

Programming Library Usage: Tasks that ask you to implement a function or algorithm should be implemented by yourself. Do not circumvent the task by calling a library that does it for you. For example, if the task is to implement the factorial, do not just return one of the available factorial functions. Whereas, if the task is to implement some formula including the factorial, you do not need to implement the factorial function yourself.

Working Code: Your code should be able to run and finish. While it is always good and worthwhile to consider the performance of your code, you will not lose or gain points based on your code's speed, unless a task specifically states otherwise.

2.3 Templates

In early assignments, we provide templates in the supplementary material for some or all classes and functions that you need to implement. The goal is to provide initial structure and guidance, both to establish a common standard and to not overwhelm students who may not yet be experienced programmers or who are new to Python. Additionally, the first assignments build on each other and thus require a common interface. Throughout the course of the lecture, the templates will become less detailed and eventually will no longer be provided.

The comments and documentation strings specify the expected behaviour of the classes and functions and should be read carefully. Though some of the specifications may seem unnecessary or tedious at first, they exist to familiarize you with different core functionalities and data types, as well as encouraging you to consider and deal with edge cases. The latter is especially important and will save you a lot of time (and points!) throughout the rest of the lecture by alerting you to problems in your code, speeding up the debugging process and helping you avoid costly mistakes.

2.4 Plagiarism and Proper Attribution

We do not expect you to know everything already, on the contrary, we expect you to look things up and doing so is perfectly okay. However, in order to learn, it is important that you engage with and understand what you are looking up, and academic integrity is important as well. Therefore, solutions and code should be written by you in your own words.

What do we consider plagiarism? As soon as you copy something that you have not written yourself and claim it is yours, we consider that plagiarism. It does not matter if you copy it from other groups, books, websites, forums, papers, submissions from previous semester or any other source. It is also plagiarism if you just change the word order a little bit, or replace some words with synonyms, or word for word translate it from another language or use a translator. That also applies to code: Just renaming variables, classes or functions, or removing or changing comments, or changing the order of code blocks also counts as plagiarism.

We understand that in some tasks, e.g. short function implementations, there is only a limited number of ways of stating the correct solution, and we will consider that and similar circumstances before flagging something as plagiarism.

Consequences: Parts that we flag as plagiarism will be marked with 0 points and we will note down the occurrence for tutors of subsequent assignments. Repeat offences may lead to losing points for larger and larger parts of the assignment.

Alternative – Proper Attribution: Should you be unsure if what you have written is too close to the original, or you are stuck and e.g. need to copy a code block from some source to progress with the rest of the task, be transparent and state your source(s). That means the names of the author(s), and if applicable the title, date, journal and link. That way, you can avoid losing points due to plagiarism or due to being stuck, and we will reward the transparency with points where appropriate.

3 Getting Started

3.1 L^AT_EX

We settled on L^AT_EX for the submissions to avoid headaches caused by adventurous formatting and because it is good to be at least somewhat familiar with it, in case you need it later for other lectures or for your thesis. At first, L^AT_EX can appear intimidating, but the template we provide is already set-up and should contain examples of all the commands and structures you are likely going to need.

You can of course install L^AT_EX on your machine and compile it there, but we recommend using something like overleaf.com. It provides free tutorials, online storage, most packages, a PDF auto-compiler, and simultaneous editing by multiple people.

3.2 Integrated Development Environments (IDEs)

While it is possible to write code in simple text editors, we recommend that you use an IDE for the assignments. They help you to organize and structure your code files, provide syntax highlighting, auto-complete variable or function names, perform static code analysis (linting) to alert you of potential issues, errors or style violations, they make it quick and easy to import libraries and run your code, and much more. All in all, they save you a lot of time and frustration.

Many IDEs have good free versions and also offer their full versions to students for free. For example, as a student of Saarland University you can get all professional [JetBrains IDEs](#) for free.

3.3 Python

The programming language Python (version 3.x) is used for the programming tasks of the first couple of assignments. The goal of these early assignments is to establish a common baseline, learn basic Python concepts and to get familiar with network properties. If you are new to Python, there are many examples, tutorials and a good documentation available to get you started, just make sure to write your solutions yourself.

Current Python 3 versions can be obtained from python.org or be installed via macOS and Linux package management systems. Note that line indentation is crucial in Python! All code templates you are given use 4 spaces as tabs, so adjust your editor or IDE accordingly to avoid problems. An example of a good, free Python IDE is [PyCharm](#).

Furthermore, you need to install the popular plotting library “matplotlib”. Linux users are advised to install the library using their package management system, e.g. for Ubuntu:

```
sudo apt-get install python3-matplotlib
```

macOS users can install it as follows on the command line:

```
pip3 install matplotlib
```

Windows users might want to take a look at [this guide](#).