Bioinformatics 3: Assignment Guide

Summer Semester 2024

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

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. However, the return may occasionally be delayed by a couple of days due to other obligations, life circumstances or an unusually high number of submissions.

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 general questions about the assignment or the tutorial, you can ask them publicly in the "Public Questions" channel in the Bioinformatics 3 Team in Microsoft Teams. If you have questions about your assignments, submissions or other matters that you do not wish to ask publicly, you can send an **email** to the email address given on the assignment sheet.

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: Due to the high number of participants, 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.

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

- **BI3_A1_SS24_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_SS24_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.

A1 stands for Assignment 1, so do not forget to increment it to **A2** for Assignment 2 and so on. 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, matriculation number and email address.

Format: It is **mandatory** to use the LaTeX–template we provide to generate your submission PDF. See Getting Started with LaTeX below if this is your first time using LaTeX. Take the time to look at the LaTeXtemplate we provide and follow the structure and examples set.

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 submission through a spelling checker.

Equations: Equations, formulas and proofs must be formatted in LaTeX and should never be provided in handwritten form. See the provided LaTeX–template for examples.

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, that they have a caption, that plot axes and curves are properly labeled, and that you reference them in the corresponding exercise.

Console Output: Some tasks ask you apply your implementation to an example and 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.

However, do **not** include console output you just generate for yourself, be that for your own examples, debugging purposes or any other reason!

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.

Add a brief caption to your listings and reference them in their respective exercises. Furthermore, make sure that you clean up your code before listing it in your PDF. That means removing code pieces that you have commented out, debugging messages and the like, and adhering to the code standards outline below.

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.

Doc Strings: When implementing your own functions, create doc strings for them that give a brief description of what the function does, what the parameters are, what (if anything) it returns and what (if any) errors it raises. Similarly, when changing the provided code templates, e.g. by adding, removing or changing the parameters or raising additional errors, make sure to adjust the doc strings accordingly. See the provided code templates for examples.

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. See the provided code templates for examples.

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 some assignments may not provide a template at all.

The comments and documentation strings specify the expected behavior 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.

Furthermore, take our templates as examples of what we expect good documentation strings and code comments to look like, and emulate them when writing your own classes or functions, or when commenting your code.

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 if from other groups, books, websites, forums, generative AI, 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 LaTeX

We settled on LaTeX 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, LaTeX 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 LaTeX 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. An example of a good, free Python IDE is PyCharm.

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.

3.3.1 Python Beginners

If you are new to Python, there are many examples, tutorials and a good documentation available to get you started. Since everyone has a different experience level, language and learning style preference, it is recommended to search for tutorials that suit you personally if you feel you need additional training to complete our assignments.

In some semesters, there is also a Python course offered to master students that will introduce you to the language while providing credit points, so make sure to check the lecture list for the current semester if you are a beginner.

3.3.2 Installation

Current Python 3 versions can be obtained from python.org or be installed via macOS and Linux package management systems. If possible, install the latest stable version of Python (3.12 as of writing this guide) or at least Python 3.10 or newer.

Furthermore, you need to install the popular plotting library "matplotlib". You can find installation instructions for Linux, macOS and Windows on their website.

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.

3.3.3 Being Stuck

When you are stuck on the technical aspects of specific parts of your implementation and the Python documentation is not sufficiently helpful, you can typically find answers through an internet search, often leading to websites like Stack Overflow.

Just make sure to either adapt and rewrite the code pieces yourself, or to credit the author by providing a link to the answer in the code comments if the piece of code cannot be adapted or improved upon by you. Again, make sure to follow our Plagiarism and Proper Attribution guideline.