

LESSON PREPARATION FORM

|  |  |  |  |
| --- | --- | --- | --- |
| Lecturer: Helena Rasche | |  | Date: |
|  | | | |
| Group: ATGM/BML | Number of students: 10 | | Classroom: |
|  | | | |
| Subject/lesson: Computational Biology, Lesson 3: Genome Assembly | | | |

**Starting situation:**

*What do the students already know about the subject and what can they already do? How do they feel about it? Have they already gained work experience?*

*Describe the composition of the group. When and where does the lesson take place? And similar.*

Students will attend this lesson with previous theoretical experience doing assemblies, this portion is a review for them. They have gone through the motions of assembly but

not done it by hand, nor understood the intricacies of parameter selection. Students at this stage in their career are planning to go on to a company or research institute wh

ere they will need to apply these skills to analyse genomic sequences and help coordinate and design sequencing projects. When those sequencing experiments occasionally fail

for various reasons, they will need to understand why they failed and how to resolve those issues, be it parameter exploration or resequencing. This lesson should serve stude

nts well as a very practical lesson delving into comparative analysis which provides key information for them.

**Objective/lesson objective:**

*Describe the objective(s) of the lesson according to the 3C model, taking account of the taxonomy level according to Bloom.*

- Compute multiple whole genome assemblies in such a way to develop big data processing skills (Apply+Procedural)

- Learn to evaluate quality metrics so that they can separate good and bad assemblies (Analyse+Conceptual, Evaluate+Procedural)

- Visualise assemblies so that they understand presentation of various failure modes (Apply+Procedural, Evaluate+Conceptual)

**Educational resources:**  
*Which learning materials do you use during your lesson? (book,  
smartboard, whiteboard, paper, etc.)*

Presentation (powerpoint), documentation (webpages), assembly activity (javascript, web-based group activity with paper assembly fallback during in-person lessons.)

**Assessment procedure:**

*Provide a description of the final assessment of the unit of study and at which of Bloom's taxonomy levels the assessment will be carried out.*

Formative assignments using the CoCalc system, we will distribute a homework assignment to them where they do the most basic task as it is the first week:

- Some skeleton of the output will be provided for them, asking them to complete a missing statement demonstrating their ability to predict how the code works and respond with the missing component in order to achieve the desired output (APPLY)

Every piece of code they will write will be evaluated with a number of sample inputs, so they can check their work (to an extent) and make sure it works ok. We will additionally have secret “teacher-only” tests which will ensure that even if the student decided to only handle the cases described, that they’ve properly considered all aspects of the problem and any potential exceptions they might encounter.

**Deviations from Normal:**

*Indicate clearly in the form: a. which teaching method you use (how?) b. why you specifically choose this teaching method (accountability) c. what your role is during the lesson and what activities the students perform (how?) .*

Here we will try Pair Programming in Duos. Pair-programming is often used in corporate programming environments in order to help new employees become more famliiar with the codebase. One person is designate as the “driver” write the code, the other as the “observer” reviews each line of code as it is typed and they discuss the code as they go. They frequently switch roles. This often results in fewer defects in the code as a result of two people discussing and interacting (Cockburn & Williams, 2000). The teacher’s role is to introduce the activity, split them into pairs, and to facilitate their discussions and co-working.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Schedule (how long?)** | **Content (what?)** | **Teaching and learning activities/work forms (how?)** | | **Justify: how will this be used to reach the objective?** |
| **Teacher** | **Student** |
| **Introduction/start:**  **Around 20 min.** | Here we will have a very short introduction to what the python language is, and how it looks and how code is structured. More importantly we’ll discuss why automation is important for them. | Presentation on python  Discussion of how python lets you automate things. | Introductions  Mentimeter / Poll  (“What sort of automation do you think of in the context of technology” / “what would you like to automate in your life”) | The topic is quite large and students need an introductory period where they get to hear and see a bit of the topic before diving on in a hands in manner. Hopefully this step builds the foundation for furthering their knowledge during the rest of the lesson and provides the necessary base of Remember/Understand |
| **~10 minutes** | We’ll introduce students to the CoCalc platform with a series of slides which introduce the basic mechanics of moving around the interface. | Walkthrough/demo | Students log in and access a notebook guiding them through the process | Same as above, separate skill but same requirement of basic understanding/remembering required for subsequent portions of the lesson |
| **Math**  **~20 min.** | - Math (\* / + -) in python, math.sqrt, math.pow  - Translate some known math functions (e.g. euclidean distance, root algorithm) into python | Live-coding (Discuss code and execute cells one-by-one) | Code-along for first portion, executing cells and listening.  **This is followed by break out room where students work in DUOs** (pair programming) and are given a function to translate into python code | Here students gain practice running code and then begin learning to Apply a human description of a function to the transformations necessary for Python. |
| **10 min** | Q&A and discussion, check in with feelings | Discussion | Answering questions, posing their own, integrating knowledge. | Check in with student feelings to ensure their cognitive load is not overwhelming. |
| **Math**  **~20 min.** | - Strings (add / format)  - Translate function into python (exercise) | Live-coding | Code-along for first portion, followed by break out room where students are given a function to translate into python code | Here students gain practice running code and then begin learning to Apply a human description of a function to the transformations necessary for Python. |
| **~15 min** | Break | | | |
| **Functions!**  **~20 Min** | - What is a function (conceptually)  - What do they look like | Lecturing | Obtaining the knowledge | Same as above, but separate skill. The student takes the knowledge from the live-coding portion where they watched the teacher write some code (and did so along with the teacher) |
| **Breakout & Discussion**  **20 min** | - Fill in the missing part of a function (exercise)  - Discussion of the results | Evaluating right and wrong answers, making sure everyone got it right. | Integrating it. | and then apply this in small exercises which are shared with the class. |
| **Functions!**  **~20 Min** | - Write a new function that does a sepcific computation, building on previous portion (exercise) | Live-coding | Students will code-along with the teacher before moving into breakout rooms for the second exercise. | Same as above. |
| **Conclusion:**  **~15 Min** | - Recap of Math + Strings  - Recap of Functions  - Q & A  - “How did you Feel” | Presentation  Questions  Open Discussion | Students will answer pop quiz type questions during the recap presentations | Here we test the lower level Remember skills before the homework assignments will test their higher taxonomgy levels. Here we just want to make sure they internalised the contents of the lesson before they go home.  Use MS Forms to have students construct true/false questions and everyone answers them, and discuss the responses. |