Table of Contents

**Introduksjon**

Placeholder

## Hvordan jeg som underviser har utviklet meg

### Begynnelsen

#### Student og PhD student

Min første erfaring som underviser var da jeg var gruppelærer for et kurs i zoologi. Det ble en veldig fin opplevelse: mens studentene så et tversnitt av en kattenyre, så jeg mange flere når studenter ba meg om hjlep med deres. Jeg følte at jeg lærte en god del mer av å undervise faget, da da jeg tok faget selv

Som PhD student fikk jeg mulighet til å bli med som underviser på det europeiske Erasmus kurset “Marine Cell Biology,” som fant sted på et marin biologsk laboratorium i Frankrike (Observatoire Oceanologique, Banyuls-sur-mer, France). Forkergruppen min var en av de som sto for kurset, og mine veiledere ba meg å holde et par foredrag og bidra med labøvelser. Denne erfaringen fortalte meg at jeg faktisk lite å undervise, å stå foran en gruppe studenter og formidle min kunnskup til dem.

#### 2005-2009

Jeg mener at denne erfaring har bidratt da jeg fikk stilling som postdokter ved Bioligisk Institutt (nå Institutt for Biovitenskap) ved Universitete i Oslo. Den jog jobbet for ba meg allered etter kort tid å bidra med noen forelesninger i kurset BIO2120 Evolusjonsbiologi om evolusjon og utviklingsbiologi (mitt doktorgrads fagfelt), og evolusjon av gener og genomer. Jeg ble også bedt å steppe inn en gang for å organisere en av labbene i BIO1000, det stor introduksjonskurset for nye studenter i biologi. Det var en ny erfaring å undervise unge studenter, noe jeg synes var spennende. Men, jeg var fortsatt opptatt av den klassiske undervisningsmåten med 45 minutter lange forelesnignen og ingen studentaktiviserende øvelser.

Som forsker byttet jeg imidlertid fagfelt i retning bioinformatikk og genomikk. Datasettene jeg jobbet med begynte å bli så store at de ikke lenger kunne bearbeides med Excel. Jeg måtte lære meg bruk a kommandolinje-baserte verktøy og supercomputere. I tillegg lærte jeg meg et programmeringsspråk (Perl). Alt dette trengte jeg for å delta i genomrevolusjonen som startet i 2006/2007, der mye instrumenter gjorde det mulig for mindre forskningsgrupper å sette sammen, og undersøke, genomer til diverse organismer.

Jeg merket fort at jeg ønsker å videreformidle min nye kunnskap til mine kollegaer, som hadde samme behov. Jeg ga korte kurs for kolleger i bruk av de verktøyene jeg brukte mest, og i progammering.

### 2009-2016

Fra 2009 ble jeg med i Norwegian Sequencing Centre (NSC), en nasjonal forskningsinfrastruktur for DNA sekvensering. NSC bestemt seg i 2011 for å starte med kursing av sine ‘kunder,’ og satt opp et kurs med en generell introduksjon til sekvensering, og to paralelle praktiske kurs i bruk av bioinformatiske verktøy. Sammen med en kollegaforsker fra England organiserte vi en av disse: en to-dagers workshop om teknikker for å sette sammen genomer, sokalt ‘genom assembly.’ Vi tok bevisst en veldig praktisk tilnærming, der deltakere, PhD studenter og erfarne forskere, fikk mye hands-on erfaring med ekte data og bruk av verktøyene. Dette var en fantastisk erfaring: her følte jeg at deltakere virkelig lærte noe, og at de når de kom ‘hjem’ med en gang kunne bruke det de hadde lært. Dette var starten av en ny periode der jeg ble mer og mer overbevist om at denne hands-on læringen fungerer veldig bra, og er en god mulighet til å koble undervisningen veldig tett på pågående forskning.

Rundt denne tiden oppdaget jeg Software Carpentry, en internasjonal organisasjon av frvilige som holdt workshops om bruk av verktøy for forskere som bruker mye beregninger og programmering i sin forskning (‘research computing’). Målgruppen var forskere som meg, som senere i karrieren sin oppdaget behov for å kunne bruke disse verktøy, uten å ha hatt dette i sin opplæring. Jeg så at deres tilnærming var grundig forankret i pedagogisk forskning, og at målet deres var å undervise kunnskap opparbeidet gjennom årene, og forhindre å finne på hjulet på nytt. Jeg merket at jeg ønsket å delta på en slik workshop og tok kontakt. Resultatet var at deres direktør FIXME [?] Greg Wilson kom til Oslo og holdt den første Software Carpentry workshoppen i Norden. Ikke bare lærte jeg mye om hvordan jeg kunne bli mer effektiv i min forskning, jeg bled - gledelig - rekrutert til å bli instruktør for Software Carpentry. Etter et (online) instruktør treningsprogram med stor fokus på pedagogikk, ble jeg og en kollega sertifisert får å kunne holde Software Carpentry workshops selv.

Denne erfaringen endret mitt syn på undervisning totalt. Plutselig ble jeg bevisst på viktigheten av motivasjon for læring, det å begrense kognitiv last, formativ vurdering osv.

#### INF-BIO5121/9121

På denne tiden var det flere grupper på UiO som organiserte korts kurs eller workshops rundt analyse av DNA sekvenseringsdata. Alle sammen ble de bedt av det som på denne tiden var Computational Life Science initiaivet, en satsning fra fakultetet for å fremme bioinformatikk forksning og undervisning, å laget et poengivende kurs på master og PhD nivå basert på disse kursene. Vi ble enige å slå oss sammen i et 5 poengs intensivkurs der studentene fikk en generell introduksjon til DNA sekvensering og data analyse, etterfuglt av moduler om anvendte analyser tilsvarende våre opprinnelige workshops. Jeg ble bedt om å være kursansvarlig. I 2013 ble jeg tilbudt en 20% 1. amanuensis II stilling ved Institutt for Informatikk for blannet annet å drive dette kurset. Kurset ble en suksess og tiltrakk seg mange studenter, flest biologer, men alltid noen informatikere. Fra og med 2014 endret vi kurset til en 10 studiepeongskurs, siden vi opplevde at mange studenter droppet eksamen fordi de bare fikk 5 studiepoeng ut av det.

Dette var min første erfaring som kursansvarlig (emneansvarlig), som var noe nytt for meg. Kurset var er teamarbeid med meg som leder, og jeg lærte mye om å drive fram et slikt prosjekt. Vi brukte studentevalueringer akitvt med et egetutviklet nettskjema som studentene fylte ut etter siste kursøkt. Dette førte til at jeg besluttet å bytte ut noen av forelesere, som fikk veldig dårlig tilbakemelding fra studentene. Jeg synes det var en vanskelig, men helt nødvendig beslutning å ta, og det fikk det ønskede effekten med bedre læringsutbytte for studentene for den aktuelle modulen. Erfaringen som emneansvarlig var positiv, og førte til en bedre forståelse for hele kursets løp, fra forberedelse, gjennføring, eksamen og evaluering. Jeg fikk også bedre innsikkt i studentene perspektiv på unversitets kurs.

Våren 2014 tok jeg fellesdelen av Universitetspedagogisk Basiskompetanse. Jeg skrev et utviklingsarbeid om hvordan å øke studentdeltakelse i undervisningen i INF-BIOS5121/9121 kurset. Dette fikk jeg gode tilbakemeldinger på. Siden jeg var ansatt i en 20% 1. amanuensis II stilling trengte jeg ikke å gjenomføre hele opplegget og jeg valgte da bort kollegaveiledningen.

Jeg fortsatte å involvere meg i The Carpentries. Vi fikk til et samarbeid med realfagsbiblioteket ved UiO. dette førte til at vi kunne tilby flere workshops. I 2016 ble jeg instruktør trener for the Carpentries. Dette innbærer at jeg kan lære opp nye instruktører. Til det har The Carpentries utviklet et instruktør trenings [opplærings materialet](https://carpentries.github.io/instructor-training) som er forankret i noen elementære pedagogiske forskning (pedagogisk psykologi) rundt hvordan vi lærer. Sentrale begreper er motivasjon, kognitiv last, novise-ekspert, mentale modeller, tilbakemelding og summativ versus formativ vurdering.

* Mine erfaringer
* refkleksjoner rundt det
* hvordan undervisningspraksis har endret seg over tid
* inkluder bruk av studentevalueringer

ha med Carpentries, TTT, og inflyttelsen!

### 2017-2020

#### BIOS1100

2017 saw the introduction of new bachelor programs in all science disciplines at UiO, and with it, now also the bioscience program was going to introduce computational modeling early in the program. It was decided to expose students to this topic already in the first semester with the compulsory course BIOS1100 - “introduction to computational modelling in bioscience.” I was given the responsibility to design and organise the course, which started in 2017. It is the first such course in Norway, and perhaps in the world.

BIOS1100 aims to teach simple (mathematical) modelling, implementing these models using the programming language Python, while all the time focussing on problems relevant for bioscience students. The focus on biology aims to ensure students see the relevance of the material taught, which is important for student motivation and learning. Problems ranging from population growth and dynamics, inheritance, DNA analysis and disease epidemics are used to gradually introduce more complex programming and modelling.

A textbook has been, and still is being, developed for BIOS1100. It introduces new programming concepts, illustrating their immediate usefulness in the context of a biological problem.

The course is a first-semester course, obligatory for all students taking the Bioscience program. It consists of weekly lectures (2 hours), compulsory group sessions (4 hours) and optional Participatory Live Coding sessions for students new to programming.

FIXME: legge til Participatory Live Coding

## Min undervisningsfilosofi (Teaching Statement)

FIXME fix references

### Introduction

My approach to developing my teaching is based on the observation that education is its own science, and a conviction that we should take the results of that science seriously when we develop courses. Just as prior research informs us when we develop our own research, so should educational research inform us when we develop our own teaching.

Educational research is a vast field and there is a lot to learn. But, one can not use all there is to learn, one has to limit oneself and focus on a few areas at the time. For me, these areas are i) Cognitive Load Theory, ii) Formative Assessment and Peer instruction, and iii) Constructive Alignment. In the following, I will introduce what I have learned from these areas and describe how I have implemented this in my teaching. I will focus on a new course that I have developed in the period since 2017, BIOS1100 – Introduction to computational models for Biosciences. When developing, and improving the course, I increasingly tried to incorporate what I have learned, and am learning, from educational science. This will be described in the following three sections.

### Cognitive load theory and the need to manage cognitive load

#### Theoretical background

Knowing some of the research of how humans, and especially university students, learn can (and should) inform us on how to organise, plan and execute our teaching. The study of mental processes, including learning, is called Cognitive Psychology. Part of this field is concerned with cognitive load theory.

Learning theory starts from acknowledging that humans have two memory systems: *working memory* and *long-term memory*. Working memory is where new information is processed, and coupled to pre-existing information present in long-term memory. It is said that learning happens if this new information is transferred to long-term memory. While long-term memory can contain vast amounts of information, working memory is considered small, and has a capacity of about “seven plus or minus two” pieces of information [1]. Cognitive load theory is “a set of learning principles that deals with the optimal usage of the working memory” [(Caspersen and Bennedsen, 2007)](https://www.zotero.org/google-docs/?XtsuNg). This theory, as defined in a recent review on the subject [(Sweller et al., 2019)](https://www.zotero.org/google-docs/?VMxIhb), “aims to explain how the information processing load induced by learning tasks can affect students’ ability to process new information and to construct knowledge in long-term memory.” The theory argues that the limited capacity of working memory severely restricts how much new information can be processed at any one time. When too much is asked from this working memory, there is a risk of overloading it, hampering learning. Overloading working memory inhibits the effective transfer of new knowledge to long-term memory, which is required for learning. It is argued that instructional methods need to take these limits into account.

#### What does this mean for BIOS1100?

Learning how to program is an important part of the teaching in BIOS1100. Learning to program is generally considered difficult [(Guzdial, 2015; Jenkins, 2002; Robins et al., 2003)](https://www.zotero.org/google-docs/?OjF63O). Reducing cognitive load for students then becomes an important goal. I have always felt students can not learn programming from looking at a slide presentation of programming concepts, and then asking them to start programming themselves.I have experienced this approach myself at some time and it did not work for me, nor did it seem a useful approach. One reason for this is that this approach would result in a large cognitive load: students would be required to retrieve the factual knowledge presented during lecture and apply it to solve complex problems without any guidance on how to approach the problem. There is thus a need for alternative ways of teaching the fundamental building blocks of a programming language. One that is more based on Guided Instruction [(Fisher and Frey, 2013)](https://www.zotero.org/google-docs/?dw50Jz).

My main technique for reducing cognitive load when learning programming is called Participatory Live Coding.

**Managing cognitive load in teaching programming: Participatory live coding**

Participatory live coding is a guided instructional technique “in which a teacher or instructor writes and narrates code out loud as they teach and invites learners to join them by writing and executing the same code” [(Nederbragt et al., 2020)](https://www.zotero.org/google-docs/?NDSa2v). The instructor reads what is being typed out loud, explaining the different elements and principles. Teacher and students all execute the commands or program, leading to an immediate evaluation of the results - hence the term ‘participatory.’ Crucially, the session contains regular, often short, exercises, where students are asked to solve a small relevant problem on their own or in pairs or small groups.

I am convinced that participatory live coding can help reduce this cognitive load for students learning programming. This approach works better than lecturing about programming, or relying on students reading a textbook or compendium. What is taught is immediately applied and the execution of the program being written provides immediate feedback. This helps student couple programming code with its result. Students’ questions arising during the session can immediately be addressed. During this form of guided instruction, students are shown not only what to program and how each element works, but also how to program, i.e. how to go from a problem formulation to a working solution (the thinking process). It also slows the teacher down relative to using slides to show the concepts and code, giving students more time to actively engage with the material before moving on to the next concept. Interrupting the live coding with exercises enables immediate practice using the material.

The main drawback of using this technique is that it takes time to develop appropriate material and to teach it. Additionally, it does not scale too well as a single student with a problem that keeps them from following along and that takes some time to fix may hold up the entire group. Finally, students may think that they learn enough by simply following along. They should be made aware that to properly learn how to program they should practice, for example by doing exercises.

#### Participatory live coding in BIOS1100

The first edition of BIOS1100 (2017) relied on the students using the textbook for learning the programming concepts needed each week. Programming was not taught during lectures for reasons described above. Exercises were handed out during group work where students could practice applying programming to simple biological problems. Two-thirds into the semester it became clear that a significant group of students did not master the Python programming, and were in danger of failing the course. I then decided to not introduce any new Python material, but instead offer some extra teaching using Participatory Live Coding. I had learned this technique from being an instructor for Software Carpentry. Software Carpentry, now of the global non-profit called “The Carpentries” (<https://carpentries.org>), “teaches researchers the computing skills they need to get more done in less time and with less pain” and is mostly aimed at researchers at the PhD and postdoc level. Participatory live coding is the main method of teaching in the two-day workshops, and it is part of the training and assessment for becoming a certified instructor [(Wilson, 2019)](https://www.zotero.org/google-docs/?qilqqq)

In BIOS1100, I thus offered sessions re-teaching the Python material with Participatory Live Coding, with the effect that many students reported that they now finally understood it.

I had previously considered using Live Coding for BIOS1100, but felt it would not scale to such a large group. The experience in 2017, and the student’s feedback, convinced me that I had to find a way to adapt the technique of Participatory Live Coding to a course with 200 students, starting from the 2018 edition of BIOS1100. I decided that it should be taught in small groups, not with the entire group of students, and that I could not teach it all by myself. My solution was to develop a completely new set of teaching materials to teach Python programming in BIOS1100 using Participatory Live Coding, and train Teaching Assistants to be able to teach using this technique. One of the things taught during Instructor Training workshops for The Carpentries is Participatory Live Coding. I am a certified Instructor Trainer for The Carpentries, meaning I already had taught Participatory Live Coding to incoming instructors. I reused the material developed by The Carpentries [(Erin Alison Becker et al., 2019)](https://www.zotero.org/google-docs/?1KrEqk) to train enough Teaching Assistants so that they could teach Python using PLC during the group sessions.

The results were that students in 2018 had a much greater confidence in Python programming. To start learning programming using Live Coding helped them overcome the initial barrier (sometimes fear) of programming, and led to a feeling of mastery early on. Students really loved the Live Coding (‘samkoding’ in Norwegian), some reported they learned the most there. A drawback of this approach was that much, sometimes all of the 4 hour group sessions were used doing Live Coding with the students. Too little time was left for students working on their own with exercises. A master student who studied the BIOS1100 students for his project that year concluded that students lacked good problem solving skills because of this [(Håland, 2019)](https://www.zotero.org/google-docs/?sHmFMo).

In other words, while in 2017, we did not help students enough with learning programming, in 2018 we helped them too much and did not challenge them enough to apply what they learned.

From 2019 onwards, the Participatory Live Coding was moved to voluntary sessions, two hours each week. Students new to Python were strongly encouraged to participate. In group sessions, organised activities led by Teaching Assistants were limited to the first two hours, which left two hours for students working on their own solving problems. During the first two hours, some Live Coding was done to further explain concepts, or for so-called worked examples. Worked examples “provide a full problem solution that learners must carefully study” [(Sweller et al., 2019)](https://www.zotero.org/google-docs/?Etqt1z), and are another technique for reducing cognitive load.

Participatory Live Coding was what made introducing Python programming to new Bioscience students possible. I believe this technique can be used in many settings where students who traditionally do not have to learn programming are introduced to it.

### Formative Assessment

#### Theoretical background

It would be a mistake to assume that students have learned the thing you just presented to them. Formative assessment is concerned with informing both the teacher and the learner about how much they understand about a topic, and discover any misunderstandings. This allows for addressing misunderstandings promptly, which helps learners to progress through the material. Formative assessment is not graded, although sometimes teachers make participation count towards being able to pass a course. Note that graded assignments are what is called part of a course’s summative assessment. Ideally, formative assessment can be done quickly and in an easy way for students and teachers to participate and see the results.

There are many forms for formative assessment, but a very common one is multiple Choice Questions. A well-designed Choice Question poses a problem with one or more correct, and multiple incorrect answers, so-called distractors. Ideally, distractors should not be too obviously wrong, but rather be indicative of possible misconceptions.

**Formative assessment combined with Peer Instruction**

Originally created by Eric Mazur at Harvard [(Mazur, 1997)](https://www.zotero.org/google-docs/?vCsJAA), Peer Instruction is an evidence-based method where students are actively discussing the material amongst themselves based on prompts provided by the teacher. By asking students to reflect on course material together in their own words, a student that just understood the material is able to explain it in a way that matches better a student that almost understands the material, than the way a teacher would explain it. Thus, *Peers* are *Instructing* themselves.

Often, Peer Instruction is combined with formative assessment through Multiple Choice Questions. There are different approaches on how to do this, but commonly, a Multiple Choice Question is posed and students consider the different answers for themselves. They then vote for the answer they think is the correct one. The tally of votes is shown, without revealing the correct answer. If there is a spread of answers among the correct one and one or more of the distractors, students are asked to discuss the question in pairs or small groups. They then vote again (individually, not as a group). More than often, the results show many more students converging on the correct answer. If needed or desired, the teacher can go over the different answers and explain how they are right or wrong.

Peer instruction has also been studied in the context of programming, with positive results on student learning [(Crouch and Mazur, 2001; Porter et al., 2013, 2016)](https://www.zotero.org/google-docs/?vC9WNx).

#### What does this mean for BIOS1100?

Also in BIOS1100 there is a need to investigate student learning and check for misunderstandings. This can be partly achieved by Teaching Assistants helping students during group work, and by studying the obligatory assignments students hand in during the course. But, also the technique of Peer Instruction through Multiple Choice Questions is an ideal addition to this.

Teaching Assistants are a vital resource for student learning. In BIOS1100, they have the most direct contact with students (during the 4 hours group sessions and the non-compulsory live coding sessions). This means that they have a lot of experience that could inform me as a teacher about student progress and misunderstandings. The challenge is then how to ensure this information reaches me as a teacher, in other words, how to implement formative assessment through Teaching Assistants.

#### Practical implementation

I use the following Formative Assessment techniques in BIOS1100:

**Multiple Choice Questions and peer Instruction**

I have written a set of around hundred Multiple Choice Questions for BIOS1100. Some of these are based on common misconceptions I found in the scientific literature. Often, when I observe students displaying a misconception, or are told about one by the Teaching Assistants, I use that as inspiration for writing new ones.

I have used Multiple Choice Questions, through the online Student Response System Mentimeter (<https://mentimeter.com>), during lectures and group work. I always combine this with Peer Instruction, using the approach described above. I regularly observe a mix between questions that are ‘too simple’ (a large majority of students get it right at the first try) and questions that reveal a misunderstanding in a significant proportion of students, that then gets largely resolved in the group discussion. Students and Teaching Assistants really like the “Menti’s,” as they are fondly called. It is an easy form of Active Learning that helps create a dynamic group session or lecture. A drawback is that executing Multiple Choice Questions take time. I usually use no more than 4 questions, and those easily take up half an hour or more. Ideally, I would be able to always see the tally of votes for all questions to be able to filter out the ones that are too easy for next time, but is it challenging to collect that data from all Teaching Assistants.

It is fairly straightforward to adopt Multiple Choice Questions to an online teaching setting, provided the platform used allows students to work in small groups (for example, in so-called Breakout Rooms that tools like Zoom offer). In my experience, this works best if the students in these groups know each other from before, otherwise it is much more challenging to get them to discuss the question.

**Obligatory assignments**

In BIOS1100, students hand in 5 obligatory assignments spread evenly throughout the course. These are meant to ensure students work with the material throughout the course. The assignments are deliberately modeled after exam problems, so as to help prepare students for the exam. These “Oblig’s” are graded pass/fail by the Teaching Assistants, and students have to pass 80% of them to be able to take the exam. When students fail on their first attempt for an assignment, they get two additional chances.

In the first edition of BIOS1100, not only were there 11 assignments, I intended these to be only used for summative assessment. I had attended a presentation where it was argued that one should not try to combine formative with summative assessment as assignments can not effectively serve these two purposes. So I instructed the Teaching Assistants to grade them without leaving any comments. It quickly became clear that it made much more sense to give students feedback on their assignment, regardless of whether they had passed or not. Having the option to hand in some of the work students doing, and receiving constructive comments on it, is very useful for student learning. Using the obligatory assignments for this is really a very good way for *all* students to receive such feedback, and one of the few ways to offer this to all students throughout the course. I thus concluded that I deliberately want to use the obligatory assignments for both formative and summative assessment. From then on, I asked Teaching Assistants to give feedback to all delivered assignments.

Ideally, I as a teacher would also study the deliveries to distill common misunderstandings, in other words, use them as a proper formative assessment tool for myself. Unfortunately, there has as of yet not been enough time during the course to do this.

**TA experiences document**

From the very beginning of the course, I have used a shared document (a Google Doc) where I asked Teaching Assistants to note down their experiences immediately following their teaching sessions. The other TA's can then read this so-called 'erfaringsdokument' (experiences document) and use that to help them prepare for their teaching. I use it to immediately address mistakes in the teaching materials, or propose solutions when things don't work as I intended them. This has worked very well, and much useful feedback, or inspirational suggestions, have been written down by the Teaching Assistants throughout the years.

Using such a form for formative assessment shows Teaching Assistants that their feedback is being taken seriously, and that their input to the course is very welcome. It provides them ownership in the teaching, something they also give me as feedback during the course evaluations at the end of the semester.

When preparing for the next year's edition, I heavily use this document, and a private ‘erfaringsdokument’ that I write myself during the semester, to further improve the teaching materials and adjust teaching sessions where necessary. These documents turned out to be essential for improving the course.

### Constructive Alignment

#### Theoretical background

Constructive Alignment is concerned with aligning the learning activities with the intended learning outcomes. Following Biggs [(2012)](https://www.zotero.org/google-docs/?jWdTHb), we start with determining:

1. What are the desired learning outcomes, these are the objectives
2. How to measure whether desired learning has been achieved, assessment
3. What (teaching and learning) activities can we use that engage students in a way that leads to learning

Biggs calls this Constructive Alignment, which is an aligned system of instruction where “the objectives define what we should be teaching; how we should be teaching it; and how we could know how well students have learned it” [(Biggs, 2012; Biggs and Tang, 2011)](https://www.zotero.org/google-docs/?cBFxN3).

#### What does this mean for BIOS1100?

Following Biggs' approach, we should start by determining our objectives.

The learning outcomes of BIOS1100 are described on the course homepage <https://www.uio.no/studier/emner/matnat/ibv/BIOS1100/> (English: <https://www.uio.no/studier/emner/matnat/ibv/BIOS1100/index-eng.html>). But since these are deliberately kept short, there is a need to expand on them.

The next step would be to determine the summative assessments, in other words, make exam questions and any obligatory assignments before the course starts. In practice, this is not often done.

Finally, we need to design appropriate activities that help new students learn the mix of biology, mathematical modelling and programming that BIOS1100 aims to teach.

#### Practical implementation

To align how we teach programming In BIOS1100, we use Jupyter Notebooks for everything. Jupyter Notebooks (<https://jupyter.org>) are a form of ‘computational’ notebooks combining text, media, programming code and the ability to execute that code and include the results of running it in the same notebook. Teaching materials, including exercises, are delivered as Jupyter Notebooks. During Live Coding sessions and group work,students as well as teachers and Teaching Assistants , do all their programming in them. Obligatory assignments are also handed out, and handed in and graded, as notebooks.

When students, assistants and teacher all are seeing and commenting code in the same environment, discussing problems and helping each other, this reduces the extra cognitive load of switching programming environments.

We use a cloud-based server, called JupyterHub, to provide students with this programming environment. An additional benefit of this programming environment is that it saves the students from installing software on their own laptop: as long as they have a working internet connection they can log in (using university credentials) to the server and start working.

In the first two years of BIOS1100, however, there was one component that did not use these notebooks. During the exam the students had to use a different environment which did not allow them to test and run their code and programs. The four hour exam was done in a UiO's digital exam environment Inspera, which did not have a technical solution for running programming code. This was initially a deliberate choice. In dialogues with my colleagues who teach beginner programming courses at the Department of Informatics, I became convinced that not being able to run code during the exam is of benefit for the students. Informatics students until recently even handed in their programs during the exam on paper. It prevents them from getting stuck with a relatively minor error (in the syntax, for example), and having to spend a lot of time fixing it - which can be hard. I thus decided to follow the same strategy, the motivation for which I also explicitly explained to the students.

In dialogues with students it became increasingly clear to me that not being able to test and run programs they write during the exam caused a lot of stress for the students. They had not experienced this way of working during the entire semester. I also learned more and more about the benefits of Constructive Alignment, and concluded that the exam introduced a major mis-alingment in the course. I thus decided that the situation needed to change.

Thus, in the fall of 2019, with help from the university's IT department and the faculty's Inspera team, it became possible, for the first time for any digital exam at the university, for students to submit exam assignments in the form of Jupyter Notebooks. This means that the students could now also run and test their code before handing it in. This led to a much increased Constructive Alignment between teaching and examination methods.

A large drawback to this approach is that students have to work in two different systems during the exam, the Inspera system and the JupyterHub browser. There is a risk of uploading the wrong notebook or the wrong version of it. Experience so far has shown students that are able to manage this satisfactorily. In 2020, the exam was changed to a 4-hour digital home exam, but otherwise organised as before. Incidentally, this aligned the exam situation even more than the 2019 exam, as it allowed students to use all available resources, as they are used to when working with (obligatory) assignments.

### Conclusion

After four years of continued development, I am very satisfied with BIOS1100. i have found the right ‘form’ for this course, with lectures, Participatory Live Coding sessions and group sessions. The material written for the course, the set of problem exercises and Multiple Choice Questions is of sufficient quality and amount.

Along the way, educational theory has informed me for the many choices a teacher has to make. It has made me choose methods to lower cognitive load, for example by successfully scaling up Participatory Live Coding. It has shown me the importance of formative assessment, how there are different ways to be informed about student progress and how collecting this information can guide course adjustment immediately, or in between course editions. Finally, it has led to a much better alignment of the way students are exposed to programming, and work with it, with the summative assessment, by making the exam situation as similar as possible to the rest of the activities in the course.

Students also report more satisfaction with the course now than in the beginning. In the student evaluation of 2019, for the first time the words feeling of mastery (‘mestringsfølelse’) appeared in the open questions.

I plan to continue to explore new areas in educational science to make adjustments in my teaching. I am convinced that using it as a basis for trying out things is the most fruitful way forward.

## Fokus på studentenes læring

Placeholder

* hvilke pedagogiske tilnærminger bruker jeg
* hvordan det bidrar stimulerer studentenes læring
* koble til teaching statemen, men mer konkret, eksempler

## Bruk av forskning i undervisningen

Placeholder

* hva vil jeg at studentene skal gjøre med forksning jeg presenterer
* bruk av forsking: Parson’s problem, peer instruction, constructive alignment, labelled subgoals
* her kommer også studentevalueringer in
* master prosjekter
  + transdisciplinær veiledningsteam!

### Bruk av resultater fra forsning om læring i undervisningen

Å lære å programmere anses generelt som vanskelig [2] [3] [4].

For et motargument (og et utmerket sammendrag av forskningen som støtter denne posisjonen), se imidlertid [5]. Å lære et programmeringsspråk er litt som å lære et nytt skriftspråk: man må ikke bare lære seg syntaksen, men også hvordan man kombinerer ord for å gi meningsfulle setninger (‘grammatikken’). Når man programmerer fører feil ofte til feilmeldinger som ikke er skrevet for nybegynnere (Denny et al., 2014). Kode som kjører, men som ikke gir forventet eller korrekt resultat, dvs. har ‘bugs,’ er vanskelig å rette - feilsøking er en ferdighet i seg selv (McCauley et al., 2008). Å be en student om å ‘skrive et dataprogram som gjør X’ fra bunnen av, krever mange ferdigheter samtidig: å vite hvilke kodingskonstruksjoner og datastrukturer som skal brukes, å kunne bruke syntaksen for dataprogrammeringsspråk, å kunne designe og teste programmet, og sørger for at løsningen faktisk løser problemet som blir adressert (dvs. ‘X’), for et problem som kan kreve kunnskap fra et bestemt domene (f.eks. biologi når det gjelder BIOS1100-studentene mine). Det er trygt å si at det er lett å komme til en situasjon hvor mengden informasjon som må behandles samtidig er for mye for en nybegynner å takle. Læring å programmere fører med andre ord raskt til det som i kognitiv psykologi kalles høy kognitiv last.

Kognitiv belastningsteori er

A set of learning principles that deals with the optimal usage of the working memory – Caspersen and Bennedsen, 2007

Denne teorien, som definert i en nylig review av emnet [6], “aims to explain how the information processing load induced by learning tasks can affect students’ ability to process new information and to construct knowledge in long-term memory” Teorien hevder at den begrensede kapasiteten til arbeidsminnet begrenser sterkt hvor mye ny informasjon som kan behandles til enhver tid. Når det blir spurt for mye fra dette arbeidsminnet, er det en risiko for å overbelaste det, noe som hindrer læring. Overbelastning av arbeidsminne forhindrer at ny kunnskap overføres effektivt til langtidsminne, noe som kreves for læring. Det hevdes at instruksjonsmetoder må ta disse grensene i betraktning.

Selv om få studier måler kognitiv belastning under læring av hvordan man programmerer direkte (f.eks. Morrison et al., 2014), prøver mange å utvikle inngrep som reduserer kognitiv belastning, og gjør forskning for å påvise eventuelle effekten av ingrepene.

Jeg har brukt en del av disse resultater for å forbedre utbytte for mine studenter når de lærer seg programmering i BIOS1100.

**Worked examples**

“Worked examples provide a full problem solution that learners must carefully study” [6]. Worked examples kan betraktes som en form for læring fra eksempler. Når løsningen på et problem er utarbeidet ved demonstrasjon, blir studentene bedt om å løse lignende problemer selv. Basert på begrenset eksisterende forskning om emnet, (Skudder og Luxton-Reilly, 2014) fokuserer jeg på to typer bearbeidede eksempler: \* eksempel-problem-par, hvor hvert utarbeidede eksempel er parret med et lignende problem for studentene å løse umiddelbart etterpå \* “faded working examples,” der etter å ha presentert en utarbeidet komplett løsning på et problem, blir det presentert problemer med flere trinn som studentene blir bedt om å løse, og slutter med et problem som studentene fullt ut skal løse

**Parsons problems** En annen viktig type oppgave jeg bruker er såkalte “Parsons problemer.” Disse innbærer å tilby et komplett program, men med randomisert rekkefølgen på linjene (instruksjonene), eventuelt supplert med noen unødvendige kodelinjer for å løse problemet (Parsons og Haden, 2006).

### Forskning på egen undervisning

## Kollegialt samarbeid om utvikling av undervisning

### Underverk

* min rolle i programrevisjon biovitenskap 2017-
* min rolle i CCSE

I 2019 tok kollega Tone Gregers ved IBV initiativ til UndervisningVerksted ved IBV, også kalt Underverk. UnderVerk er “en workshopserie der alle som er involvert i undervisning ved Institutt for Biovitenskap skal få mulighet til å arbeide med alle aspekter av undervisning i et kollegialt felleskap.”[[1]](#footnote-66)

Sammen med kollega Hans-Petter Hersleth har jeg vært medorganisator for UnderVerk. Med støtte fra MatNat fakultetets studiekvalitetsmidler har vi organisert flere workshops. Noen av temaene har vært

* hvordan å skrive LæringsUtbytte Beskrivelser
* Læringsmål og baklengsdesign
* praktisk bruk av pedagogisk teori i undervisningsplanlegging
* studentaktive læringsformer
* bruk av digitale verktøy
* vurderingsformer

Under Corona-krisen vår 2020 organiserte Underverk et ukentlig treff i Zoom for erfaringsutveksling.

Mitt bidrag har vært å hjelpe med planlegging og gjennomføring, samt bidra konkret til noen av workshoppen:

* kognitiv last teori (oktober 2019)
* bruk av Zoom for digital undervisning (mars 2020)
* digital hjemmeksamen i BIOS1100 (november 2020)

UnderVerk oppleves som veldig nyttig av undervisere ved instituttet. Det er varierende oppmøte, men deltakere synes altid dat har vært verdt å delta. Jeg opplever UnderVerk som et fint tilbud for å dele erfaringer.

### Undervisning i et felleskap med gruppelærere og studenter

* studentrepresentanter - felles med BIOS1110

#### Gruppelærere

* pedagogisk opplæring
  + a la TA programmet

Jeg ser på gruppelærere som undervisere. Det er de som har mest kontakttimer med studentene gjennom semesteter. Derfor synes jeg det er flere viktige aspekter med å jobbe med gruppelærere

* jeg skal hjelpe de til å bli gode undervisere
* de skal oppleve at de jobber i et kollegialt felleskap
* jeg skal aktivt involvere dem i å forbedre undervisningsopplegget

Det første året hadde jeg en sesjon før kursstart (fæør somemrferie) men da blie det mye informasjon for lang tid før det skulle brukes

De siste årene har jeg brukt de ukentlige møter med gruppelærere der jeg tok opp et tema som er pedagogsik relevant, eller inviterte en annen person til å presentere temaet.

Brukte aktiv læring (tenk-par-del osv)

Noen av dette er inspirert av the Carpentries instruktør trening.

Temaene vi vanligvis diskuterer:

* introduksjon til rollen som gruppelærer
* lærinsgmål og baklengs undervisningsdesign
* motivasjon (student *og* underviser)
* kognitiv last - hvorfor er det viktig å ta hensyn til det
* noviser og eksperter
* mentale modeller

Etter at jeg innførte denne måten å lære opp gruppelærere i noen pedagogiske begreper mm har jeg intrykk av at nivået på deres undervisning, og måten de samarbeider med studentene på, har blitt forbedret. Det er som om de tar jobben mer seriøst når de forstår at det er en vitenskap bak det å lære og undervise.

følelse av tilghørighet og eierskap til kurset

Både PhD, MSc og BSC studenter

#### Evaluering

* erfaringsdokument og bruken av det

#### Kursevaluering

#### Studenter som bidragsyttere i det faglige innholdet

CSE sommerstudenter

## Dokumentasjon (Vedlegg)

FIXME oversett til norsk

* ha med utviklingsnotat fellesdelen
* ha med ten quick tips?
* jupyter canvas projsekt
* ‘emneplaner’ jeg har utviklet
* sluttraportene BIOS1100
* lenke til AMBP
* digitale læringsressurser?
* MNT 2019
* lenke til BIOS1100 canvas sidene for alle år
* mine blog posts?
* github.uio.no sider for gruppelræere

### Universitetsundervisning

* flytt til utvikling.do.txt

#### Emneansvar

BIOS1100 - Introduction to Computational Modelling in the Biosciences: <http://www.uio.no/studier/emner/matnat/ibv/BIOS1100>; new first-year bachelor course that I am coordinator and main teacher for; the course is the first implementation of the ‘Computing in Science Education’ project for the Biosciences study program, teaching python programming and modelling of real biological phenomena, 2017-

INF-BIO5121/9121 - High Throughput Sequencing technologies and bioinformatics analysis Univ. of Oslo: <http://inf-biox121.readthedocs.org/en/2015/>; I organise the course, coordinating with the five other teachers, and teach the assembly module: <http://inf-biox121.readthedocs.org/en/2016/Assembly/>, 2012-2016

#### Annen universitetsundervisning

MBV-INF4410/9410 Bioinformatics for Molecular Biology (Univ. of Oslo): lecture on “The bioinformatics of sequencing and assembling genomes,” and “What does it mean to do bioinformatics?” 2013-2016

BIO9905MERG1 - Bioinformatics for Metagenomic Analyses and Environmental Sequencing (Univ. of Oslo): lectures on “Next Generation Sequencing techniques and data relevant for metagenomics analyses” and “Assembly of metagenomes,” 2011

BIO2120 Evolusjonsbiologi (Univ. of Oslo): lectures on “Evolution and Development” and “Evolution of Genes and Genomes”; group work, 2006 - 2007

Erasmus ICP course Marine Cell Biology (Observatoire Oceanologique, Banyuls-sur-mer, France): lectures on “Fundamental aspects of development” and “Cell cycle changes during development,” 2000

#### Workshops

FIME - og kopier til /Users/alexajo/current/CV/to\_add.md (!)

2011 NSC course: High Through-put Sequencing: technology basics, applications and bioinformatic analysis (Oct. 25-27 2011) Course webpage: <https://www.sequencing.uio.no/events/2011/nsc_course_2011.html>

Workshop on Assembly Workshop webpage: <https://github.com/lexnederbragt/denovo-assembly-tutorial/>

Next-Gen Sequence Analysis Workshop ‘week 3’ (intermediate and advanced skills) (invited) 2015, Michigan State University <http://angus.readthedocs.org/en/2015/week3.html> Univ. California Davis Assembly Masterclass (invited) 2013 <http://davis-assembly-masterclass-2013.readthedocs.org/en/latest/> *De novo* genome assembly, Univ. of Oslo, 2011 *De novo* genome assembly (invited), Univ. of Gothenburg, 2011

### Software og Data Carpentry

[Software Carpentry](http://software-carpentry.org/) and [Data Carpentry](http://www.datacarpentry.org/) are teaching researchers in science, engineering, medicine, and related disciplines worldwide computing and data analysis skills through short workshops using volunteer instructors. I am a certified instructor of and organize and lead workshops in Norway and Sweden. Together with Karin Lagesen and with support from the Science Library (UiO), we have established Software and Data Carpentry at the University of Oslo, see [uio.no/carpentry](http://uio.no/carpentry), organising workshops more and more frequently, and increasing the number of certified instructors. I am also a certified *instructor trainer*, which enables me to give workshops for researchers that want to become instructors for Software or Data Carpentry.

#### Software Carpentry workshops taught

University of Oslo: 2012, 2013, 2015, 2016, 2017 Netherlands eScience Centre: 2017 University of Bergen: 2014 Science for Life Laboratory, Stockholm, Sweden 2014

#### Software/Data Carpentry instructor retreat

I gave a session on Interactive learning techniques, recording: <https://www.youtube.com/watch?v=QptHJgzooU0>.

### Liste over vedlegg

* SWC, DC certificates
* utviklingsarbeid /Volumes/GoogleDrive/My Drive/archive/uniped/Fellesdelen\_V2014

## Litteraturliste

* **G. A. Miller**. The Magical Number Seven, Plus Or Minus Two: Some Limits on Our Capacity for Processing Information., *Psychological Review*, 63(2), pp. 81-97, [doi: 10.1037/h0043158](https://dx.doi.org/10.1037/h0043158),

* **M. Guzdial**. *Learner-Centered Design of Computing Education: Research on Computing for Everyone*, Morgan & Claypool,

* **T. Jenkins**. On the Difficulty of Learning to Program, *Proceedings for the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, Loughborough University,

* **A. Robins, J. Rountree and N. Rountree**. Learning and Teaching Programming: A Review and Discussion, *Computer Communications*, 13(2), pp. 137-172, [doi: 10.1076/csed.13.2.137.14200](https://dx.doi.org/10.1076/csed.13.2.137.14200),

* **A. Luxton-Reilly**. Learning to Program Is Easy, *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ACM,

* **J. Sweller, J. J. G. v. Merri"enboer and F. Paas**. Cognitive Architecture and Instructional Design: 20 Years Later, *Educational Psychology Review*, [doi: 10.1007/s10648-019-09465-5](https://dx.doi.org/10.1007/s10648-019-09465-5),

1. Miller GA. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review. 1956;63: 81–97. doi:[10.1037/h0043158](https://doi.org/10.1037/h0043158)

2. Guzdial M. Learner-Centered Design of Computing Education: Research on Computing for Everyone. Morgan & Claypool; 2015.

3. Jenkins T. On the difficulty of learning to program. Proceedings for the 3rd Annual conference of the LTSN Centre for Information and Computer Sciences. Loughborough University; 2002. pp. 53–58.

4. Robins A, Rountree J, Rountree N. Learning and Teaching Programming: A Review and Discussion. Computer Communications. 2003;13: 137–172. doi:[10.1076/csed.13.2.137.14200](https://doi.org/10.1076/csed.13.2.137.14200)

5. Luxton-Reilly A. Learning to Program is Easy. Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education. New York, NY, USA: ACM; 2016. pp. 284–289. doi:[10.1145/2899415.2899432](https://doi.org/10.1145/2899415.2899432)

6. Sweller J, Merriënboer JJG van, Paas F. Cognitive Architecture and Instructional Design: 20 Years Later. Educational Psychology Review. 2019. doi:[10.1007/s10648-019-09465-5](https://doi.org/10.1007/s10648-019-09465-5)

1. <https://www.uio.no/for-ansatte/enhetssider/mn/ibv/underverk/>, se også <https://www.mn.uio.no/kurt/undervisningsutvikling/underverk/> [↑](#footnote-ref-66)