Scientific Programming with Python in the life sciences
Vitenskapelig programmering med Python i biovitenskap
Vitskapleg programmering med Python i biovitskap

Location:
Tromsø

Discipline:
Biology, Biotechnology, Medicine, other Life Sciences

Course type:
Enkeltemne

Semester:
Spring 2023, see schedule

Target audience:
PhD candidates in the life sciences; researchers

Type of course:
This is a standalone course to introduce PhD candidates and researchers in the life sciences to scientific programming.

Work load:
10 ECTS pts
2 weeks active participation and ~ 1 week full-time working on the project
Course includes ca. 20h of lectures and ca. 100h of active coding during the seminar plus the time required to set up one's own PC beforehand.

Exam type:
Work assignment that is completed independently and graded pass/fail.

Teaching and exam language:
English

Course level:
The course is designed for PhD candidates in the life sciences.

Course overlaps with:
M.Sc. level version of this course.

Recommended knowledge:
There are no required courses.

Course contents
The first week introduces the participants to basic computation in Python. It includes all the basics necessary to get started writing working Python code. Programming concepts and techniques in Python are introduced with plentiful exercises gleaned, as far as possible, from the scientific praxis. After the

first week the participants will have a good understanding of general computation in Python. They will have also created some first projects using the Python language's base capabilities and basic file handling tasks.
The second week then further introduces students to the most common aspects and tasks of scientific coding. Participants learn to use many of Python's scientific packages in realistic settings. Exercises again will mostly be taken from the life sciences and include complex datasets from published research. Lastly, students shortly learn about the most important good coding practices. These include needs for documentation and maintainability, as well as techniques for quality assurance. Emphasis is put on the importance of these techniques for good scientific practice.
The more detailed sections of the course are:

- Introduction to computing and Python
- The command line, Interactive shell, Scripts
- Basics, variables, string handling
- Functions & control flow
- Object-Oriented Programming
- File in- and output
- Error handling
- Libraries and foreign code
- Commonly used packages
  - Jupyter Notebooks
  - Data handling with Pandas and SciPy
  - Plotting with Matplotlib and Seaborn
  - Sequence analysis with Biopython
  - Text search with Regular Expressions
  - Generally useful packages
- Using Blast with own code
- Best practices: effective and efficient coding
- Maintainable coding, testing, and debugging
- Resources for Python programmers

What do you learn?

Knowledge

- Understand the core principles of the Python programming language
- Apply common scientific packages in Python
- Understand common strategies to solve problems
- Apply strategies to familiarize themselves with new techniques and tools
- Apply criteria for good documentation
- Understand the need for maintenance of code
- Understand factors that make code efficient, maintainable, and clean
- Identify need for testing in own code
- Know how to find resources for further study and skill development

Skills:

- Dissect larger data sets
- Isolate and solve complex problems
- Identify core challenges of larger data analysis tasks
- Build and manage larger data analysis projects
- Develop programming-based problem solving skills

- Test and correct one's own thinking and engineering in a targeted manner
- Adapt, extend, and improve existing code
- Build simple data analysis pipelines
- Compose proper documentation
- Create appropriate tests for one's own code

Competences:

- Rephrase scientific problems as computational problems
- Automate and systematize everyday tasks
- Plan and organize computational work
- Estimate difficulty and duration of coding tasks
- Build advanced logical and systematic thinking

Evaluation & exam:
The participants need to be present for and actively participate in at least 80% of the sessions. For the homework project participants will be required to create a short bioinformatics pipeline to analyze complex data and document the pipeline accordingly.
Evaluation criteria are functionality and reproducibility of the pipeline, code documentation, coding practices in light of readability, maintainability and scientific quality. The first three criteria will be weighted strongest. Participants should show that they have considered testing for their project. During the course smaller data analyses problems to be completed before the end of the course. Completion of these problems as well as course attendance is required to participate in the exam. Grading for the homework exam is pass/fail. Participants have 4 weeks to complete homework project.
The course include 20 hours of lectures and 60 hours of computer practical plus 40 hours independent course work.

Relevance to the degree:
Participants will be trained to make full and efficient use of computation for their work. This is an essential skill in modern science, especially in the increasingly data-driven life sciences.

Minimum and maximum participants:
Minimum 3. maximum 30

Contact person:
Ines Heiland (AMB)
Yin-Chen Hsieh (AMB)