

# Algorithms and Data Structures

## A1. Organizational Matters

Gabriele Röger

University of Basel

February 28, 2024

# Algorithms and Data Structures

February 28, 2024 — A1. Organizational Matters

## A1.1 Organizational Matters

## A1.2 About this Course

# A1.1 Organizational Matters

# People



Gabriele Röger



Salomé Eriksson

## Lecturer

Gabi Röger

- ▶ **email:** `gabriele.roeger@unibas.ch`
- ▶ **office:** room 04.005, Spiegelgasse 1

## Assistant

Salomé Eriksson

- ▶ **email:** `salome.eriksson@unibas.ch`
- ▶ **office:** room 04.005, Spiegelgasse 1

# People



## Tutors

Flurin Baumann ([flurin.baumann@unibas.ch](mailto:flurin.baumann@unibas.ch))

- ▶ Friday, 14.15-16.00, Pharmazentrum, U1075 and
- ▶ Wednesday, 10.15-12.00, Biozentrum, Room U1.193

Giovanni Utzeri ([giovanni.utzeri@unibas.ch](mailto:giovanni.utzeri@unibas.ch))

- ▶ Wednesday, 10.15-12.00, Pharmazentrum, U1075

Renato Farruggio ([renato.farruggio@stud.unibas.ch](mailto:renato.farruggio@stud.unibas.ch))

- ▶ Tuesday, 14.15-16.00, Pharmazentrum, U1075

# Time & Place

## Lectures

- ▶ **Wednesday:** 14:15–16:00, Biozentrum, lecture hall U1.131
- ▶ **Thursday:** 14:15–16:00, Biozentrum, lecture hall U1.141

## Exercise Sessions (starting March 1/5/6)

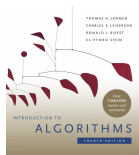
- ▶ **Tuesday, 14.15-16.00, Pharmazentrum, U1075**
- ▶ **Wednesday, 10.15-12.00, Pharmazentrum, U1075**
- ▶ **Wednesday, 10.15-12.00, Biozentrum, Room U1.193**
- ▶ **Friday, 14.15-16.00, Pharmazentrum, U1075**

# Resources

- ▶ **Adam:** central starting point and exercises  
<https://adam.unibas.ch/>
- ▶ **Website:** course information, slides, notebooks
- ▶ **Discord:** for your interaction with each other
  - ▶ Idea: course participants help each other.
  - ▶ Lecturers and tutors can help by request.
  - ▶ Feel free to use a **pseudonym**.

# Textbook

## Textbook



Introduction to Algorithms  
by Thomas H. Cormen, Charles E. Leiserson,  
Ronald L. Rivest and Clifford Stein  
(MIT Press, Fourth Edition)



# Programming Languages

- ▶ Lectures: Mostly Python  
→ Advantage: compact and direct, ideal for smaller programs
- ▶ Exercises: Java or Python (indicated on exercise sheet)



We don't require any previous knowledge about Python!

# Exercises

## Exercise sheets (homework assignments):

- ▶ theoretical and programming exercises
- ▶ on ADAM every Thursday evening
- ▶ may be solved in groups (we recommend groups of 2-3)
- ▶ group members should be in same exercise group
- ▶ due Friday the following week (23:59)  
(upload to Adam at <https://adam.unibas.ch/>)
- ▶ discussion and **individual feedback** in exercise meeting

# Exercises

## Exercise sessions:

- ▶ introduction of/questions about the current exercise sheet
- ▶ discussion of previous exercise sheet (common problems)
- ▶ questions about the course
- ▶ if time: work on the homework assignment
  - ▶ support with the current exercise sheet
  - ▶ technical support (Java/Python, programming environment)
- ▶ participation voluntary but highly recommended

**important:** please fill in the survey on ADAM for the group assignment until **tomorrow 15:15** (February 29).

- ▶ One registration per team (please list all names).
- ▶ All team members will be in the same exercise session.

# Course Format

- ▶ 6 ECTS main course + 2 ECTS exercises
- ▶ separate enrolment and evaluation
- ▶ can and should be taken in parallel

## Enrolment

- ▶ <https://services.unibas.ch/>
- ▶ register today for the course, so that you get all relevant emails and access to the ADAM workspace
- ▶ enrolment for exercise after we made the group assignment

## Prerequisites

- ▶ basic programming skills (ideally Java or Python)

# Evaluation of Main Course (6 CP)

- ▶ **written exam**, 6 ECTS credits, graded 1-6
- ▶ 10 June 2024, 14:00-16:00
- ▶ admission to exam: **no prerequisites**
- ▶ must **register** for exam during April 8 – April 22  
    ↪ see <https://philnat.unibas.ch/de/examen/>
- ▶ grade for course determined exclusively by the exam
- ▶ if you fail: **one** repeat attempt (within one year)

Last lecture (May 30): Q&A session for exam preparation

## Evaluation of Exercises (2 CP)

- ▶ midterm exams on April 10 and May 15
- ▶ in the usual lecture hall (Biozentrum)
- ▶ pass/fail evaluation based on the accumulated marks from the midterm exams

# Laptops

Small exercises during the lecture: please bring your laptop.

But stay focused:

*Research Article*

## **Logged In and Zoned Out: How Laptop Internet Use Relates to Classroom Learning**

**Susan M. Ravizza, Mitchell G. Uitvlugt, and  
Kimberly M. Fenn**

Department of Psychology, Michigan State University, East Lansing

### **Abstract**

Laptop computers are widely prevalent in university classrooms. Although laptops are a valuable tool, they offer access to a distracting temptation: the Internet. In the study reported here, we assessed the relationship between classroom performance and actual Internet usage for academic and nonacademic purposes. Students who were enrolled in an introductory psychology course logged into a proxy server that monitored their online activity during class. Past



Psychological Science  
2017, Vol. 28(2) 171–180  
© The Author(s) 2016  
Reprints and permissions:  
sagepub.com/journalsPermissions.nav  
DOI: 10.1177/0956797616677314  
www.psychologicalscience.org/PS  
The SAGE logo consists of a circular icon with a stylized 'S' inside, followed by the word 'SAGE' in a bold, uppercase, sans-serif font.

# Jupyter Notebooks

web-based interactive computational environment for Python  
(and some other languages)

- ▶ illustrating algorithms and concepts
- ▶ implementing algorithms for experimenting and studying at home
- ▶ small exercises during the lecture

The screenshot shows a Jupyter Notebook titled "selection\_sort" with a last checkpoint 54 seconds ago. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and code execution. The notebook content is as follows:

```

Selection Sort

Let's include the selection sort algorithm from the lecture. You can uncomment the #print statements to get some output on its computation.

171 def selection_sort(array):
    n = len(array)
    for i in range(n - 1):
        # print(array)
        min_index = i
        for j in range(i + 1, n):
            if array[j] < array[min_index]:
                min_index = j
            # print("swapped element at position", i, "-", array[j] - 1,
            #       "at", array[min_index], "at position", min_index)
            # print("swap if with", array[i], "at position", i)
        array[i], array[min_index] = array[min_index], array[i]
        # print(array)

Using it in an example:

181 test_array = [7,3,3,3,2]
    selection_sort(test_array)
    print(test_array)

[2, 3, 3, 7, 8]

To illustrate how we can use it to sort more complex data than just numbers, we create a class Person that store the name and age of a person. We compare persons only by their age.

184 from functools import total_ordering
@total_ordering
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self):
        # creates string representation like "Maxwell (25)"
        return "%s (%i)" % (self.name, self.age)

# the comparison operators only consider the age
def __eq__(self, other):
    return self.age == other.age
  
```



## A1.2 About this Course

# Algorithms and Data Structures

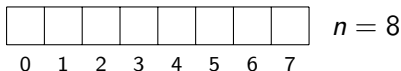
- ▶ some basic building blocks are needed again and again in programming projects, e.g.
  - ▶ sorting algorithms
  - ▶ search trees
  - ▶ priority queues
  - ▶ shortest path in a graph
  - ▶ ...
- ▶ oftentimes provided by libraries
- ▶ here you learn ...
  - ▶ how all this works internally.
  - ▶ how to select suitable building blocks.
  - ▶ tricks to achieve efficient programs.
- ▶ independent of specific programming language

# Example: Algorithms for Sorting

- ▶ task: sort a sequence of elements in increasing order, e.g.  
input [5, 9, 3, 5] → result [3, 5, 5, 9]
- ▶ 1960s (and a long time afterwards): a quarter of all commercial computation time used for sorting
- ▶ naive algorithm: **selection sort**



# Selection Sort: Informally



- ▶ identify smallest element at positions  $0, \dots, n - 1$  and swap it to position 0
- ▶ identify smallest element at positions  $1, \dots, n - 1$  and swap it to position 1
- ▶ ...
- ▶ identify smallest element at positions  $n - 2, n - 1$  and swap it to position  $n - 2$

# Selection Sort: Example

3	7	2	9	7	1	4	5
---	---	---	---	---	---	---	---

1	7	2	9	7	3	4	5
---	---	---	---	---	---	---	---

1	2	7	9	7	3	4	5
---	---	---	---	---	---	---	---

1	2	3	9	7	7	4	5
---	---	---	---	---	---	---	---

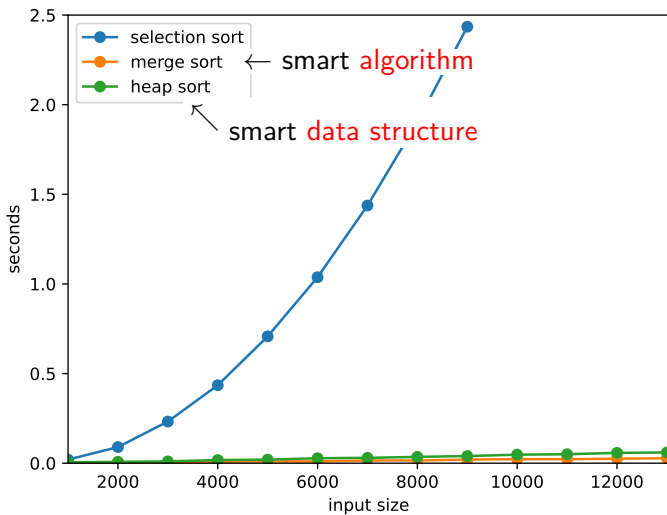
1	2	3	4	7	7	9	5
---	---	---	---	---	---	---	---

1	2	3	4	5	7	9	7
---	---	---	---	---	---	---	---

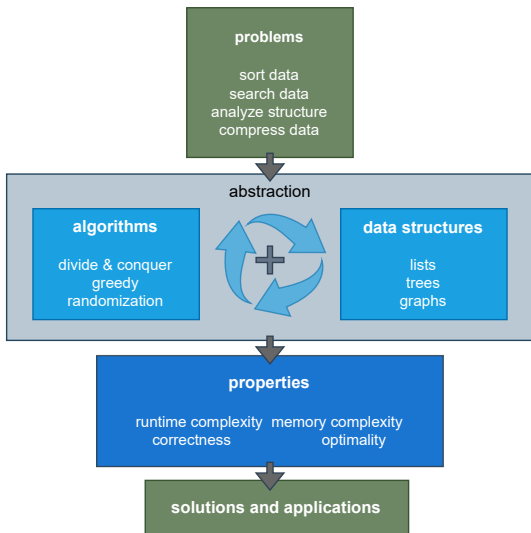
1	2	3	4	5	7	9	7
---	---	---	---	---	---	---	---

1	2	3	4	5	7	7	9
---	---	---	---	---	---	---	---

# Algorithms for Sorting: Runtime



# Algorithms and Data Structures



# Content of the Course

