# Distributed Systems - Lab Course
# Block 1

Prof. Dr. Pascal Cerfontaine • Zoom 644 2622 6965 • 263566

Technology
Arts Sciences
TH Köln

# Organisation

## Exam admission

To be admitted to the exam you must:

- Obtain 50% of the lab course exercise points
- Pass 2 out of 3 multiple choice quizzes (basically, today is a practice run)
- Pass 2 out of 3 oral quizzes during the lab course (basically, today is a practice run)

## Agenda for Today

1. Take the 10 minute multiple choice quiz under *ILU - Tests - Lab Course Block 1*. The password will be announced in the lab course.
2. Work on the lab course assignment on the next slides (exercise 4).
3. While you are working on the assignment, I will quiz each group for about 15 minutes.

# Important Notes

- Hand in the solution via git until the end of the class. The latest commit before the deadline will be graded, newer commits will be ignored.
- The solution for each exercise should be in a separate, clearly named file. Avoid using special characters and spaces in file names.
- Read this page thoroughly. Then, read each task carefully and completely before you begin working on the solution.
- Work together in your group to develop a joint solution — not separate solutions for individual group members.
- Every group member should contribute and be able to explain the code and make modifications to it during the lab course.
- You might need to look up new Python functions and their documentation yourself.
- Please stick to the standard Python library (do not install additional packages) unless it is specifically mentioned in the exercise.
- Do not use chatbots to for the lab course exercise. You may use search engines like Google.
- Do not use any tools / help for the multiple choice and oral quiz.
- Feedback and grading will be uploaded to your git repo in the branch “grading”

# 4. Chat Client and Server

In this exercise, you will create a simple multi-client chat system using Python's asyncio module. You will implement:

- A chat server that handles multiple clients concurrently.
- A chat client that connects to the server and allows users to send and receive messages.

You can test several clients and the server by running them simultaneously from separate terminals on your local machine.

# a) Server (5 Points)

Write a server `ex_04_server.py` that:

1. Listens on localhost:8888.
2. Accepts multiple client connections concurrently.
3. Prints each client connection and disconnection.
4. Receives messages from any client.
5. Broadcasts each message to all other connected clients.

## Hints

- Use `asyncio.start_server()` to create the server.
- Use `await reader.readline()` to read from a client.
- Keep track of all connected clients in a `set()`.
- Use `writer.write()` and `await writer.drain()` to send messages.
- When a client disconnects, remove them from the client set and close their connection.

# b) Client (5 Points)

Write a client `ex_04_client.py` that:

1. Connects to the chat server on localhost:8888.
2. Runs two asynchronous tasks in parallel:
    - One for reading messages from the server.
    - One for sending user input to the server.
3. Prints incoming messages to the terminal.

## Hints

- Use `asyncio.open_connection()` to connect.
- Use `asyncio.gather()` to run the send and receive tasks concurrently.
- Use `asyncio.to_thread(input, "")` to read `input()` without blocking.