Aalto University
School of Science

CS-E4640 Course Management Spring 2025

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

Course Git:
https://github.com/rdsea/bigdataplatforms

Aalto Mycourses:
https://mycourses.aalto.fi/course/view.php?id=44822

MS Teams:
CS-E4640 Big Data Platforms | General | Microsoft Teams

Lectures, tutorials and meetups

- **Lectures**
  - Key concepts about principles, models, methods, and technologies
- **Tutorials**
  - Practical, concrete tools and hands-on discussions
- **Meetups**
  - Adhoc, not mandatory contents but useful tips/experiences
- **Nr. of lectures + tutorials != Nr. of slots in the course agenda**
  - Backup dates (e.g., in case of sickness) & on-demand face-to-face discussions

All dates in the agenda must be booked!

Schedule

Remember the schedule:

https://github.com/rdsea/bigdataplatforms/blob/master/schedule.md

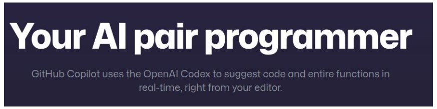
Wed: 12:15-14:00 (lecture)

Thu: 10:15-12:00 (hands-on/meetup)

Try to also use Thu slots for concept discussions and networking!

Communications

- **Course discussion (no moderation!)**
  - Microsoft Teams (pls. register, see the link in MyCourses)
- **Find the ideas/answers from the Internet – no problem**
  - Everyone knows stackoverflow
    - also chat.open.ai and GitHub Copilot



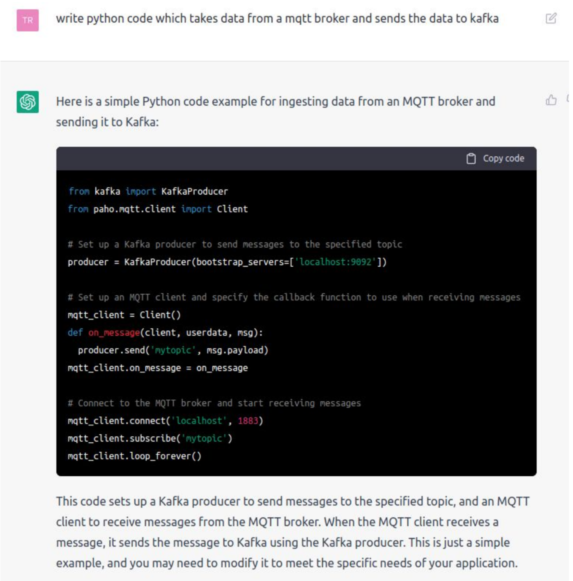
Your AI pair programmer

GitHub Copilot uses the OpenAI Codex to suggest code and entire functions in real-time, right from your editor.

- **Everyone should help sharing the knowledge w.r.t course topics**
- **We try to react as soon as possible but don’t expect real-time responses – people have different speeds and workload!**

AI assistance: Good or bad hints?

Our code detection might flag it and if you cannot explain or make the information clear, it is a problem



TR write python code which takes data from a mqtt broker and sends the data to kafka

Here is a simple Python code example for ingesting data from an MQTT broker and sending it to Kafka:

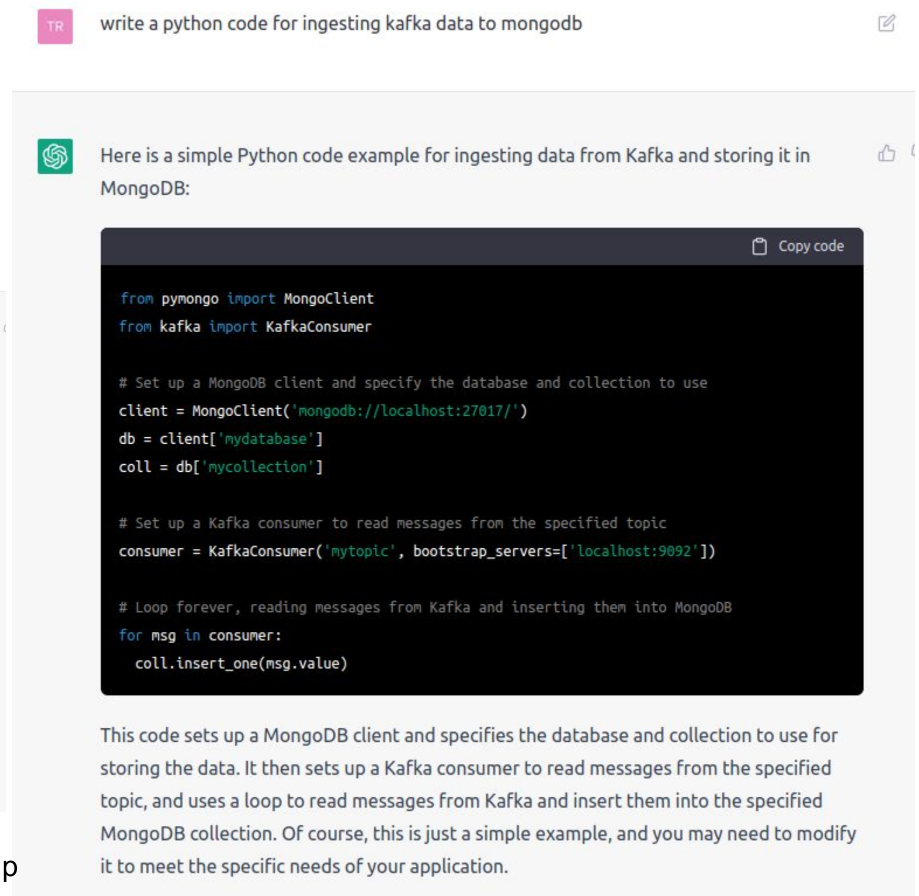
```python
from kafka import KafkaProducer
from paho.mqtt.client import Client

# Set up a Kafka producer to send messages to the specified topic
producer = KafkaProducer(bootstrap_servers=['localhost:9092'])

# Set up an MQTT client and specify the callback function to use when receiving messages
mqtt_client = Client()
def on_message(client, userdata, msg):
  producer.send('mytopic', msg.payload)
mqtt_client.on_message = on_message

# Connect to the MQTT broker and start receiving messages
mqtt_client.connect('localhost', 1883)
mqtt_client.subscribe('mytopic')
mqtt_client.loop_forever()
```

This code sets up a Kafka producer to send messages to the specified topic, and an MQTT client to receive messages from the MQTT broker. When the MQTT client receives a message, it sends the message to Kafka using the Kafka producer. This is just a simple example, and you may need to modify it to meet the specific needs of your application.



TR write a python code for ingesting kafka data to mongodb

Here is a simple Python code example for ingesting data from Kafka and storing it in MongoDB:

```python
from pymongo import MongoClient
from kafka import KafkaConsumer

# Set up a MongoDB client and specify the database and collection to use
client = MongoClient('mongodb://localhost:27017/')
db = client['mydatabase']
coll = db['mycollection']

# Set up a Kafka consumer to read messages from the specified topic
consumer = KafkaConsumer('mytopic', bootstrap_servers=['localhost:9092'])

# Loop forever, reading messages from Kafka and inserting them into MongoDB
for msg in consumer:
  coll.insert_one(msg.value)
```

This code sets up a MongoDB client and specifies the database and collection to use for storing the data. It then sets up a Kafka consumer to read messages from the specified topic, and uses a loop to read messages from Kafka and insert them into the specified MongoDB collection. Of course, this is just a simple example, and you may need to modify it to meet the specific needs of your application.

See also:
https://github.com/rdsea/bigdataplatforms/tree/master/tutorials/llmbdp

The Teaching Assistant team

Hong-Tri Nguyen

Postdoc researcher
https://hong3nguyen.github.io/

Korawit Rupanya

Doctoral researcher
https://smart-portfolio.netlify.app/

Debayan Bhattacharya

Doctoral researcher
https://www.linkedin.com/in/debayanbhattacharyavit/

Minh-Tri Nguyen

Doctoral researcher
https://research.aalto.fi/en/persons/tri-nguyen

Anh-Dung Nguyen

Master thesis researcher
https://www.linkedin.com/in/dungnguyenanh0706/

Reaching us via Teams to the team can support you!
Avoid sending emails!

Personal discussion

- **Arrange face-to-face meetings**
- **Using Microsoft Teams to chat and get meeting slots for one-to-one calls**
- **Discuss your problems with the professor in charge**
  - after the lecture/tutorial time and in the office
- **Try to have personal discussion with our Teaching Assistants as well!**
- **Share your problems in the Teams so that we don’t need to repeat the similar questions**

Assignments

- **3 assignments**
  - Each divided into 3 parts (design, implementation, and extension)
  - Within a part: an objective is evaluated in the 0-5 scale, then multiplied by a pre-defined weighted factor (based on the part)
  - No final exam!
- **Assignment evaluation**
  - Real world design, development, reporting, and demonstration
  - But not a "production" outcome
  - **No automatic grading**: we will check your code and do reproducible test
- **Face-to-face explanation for all assignments**
-

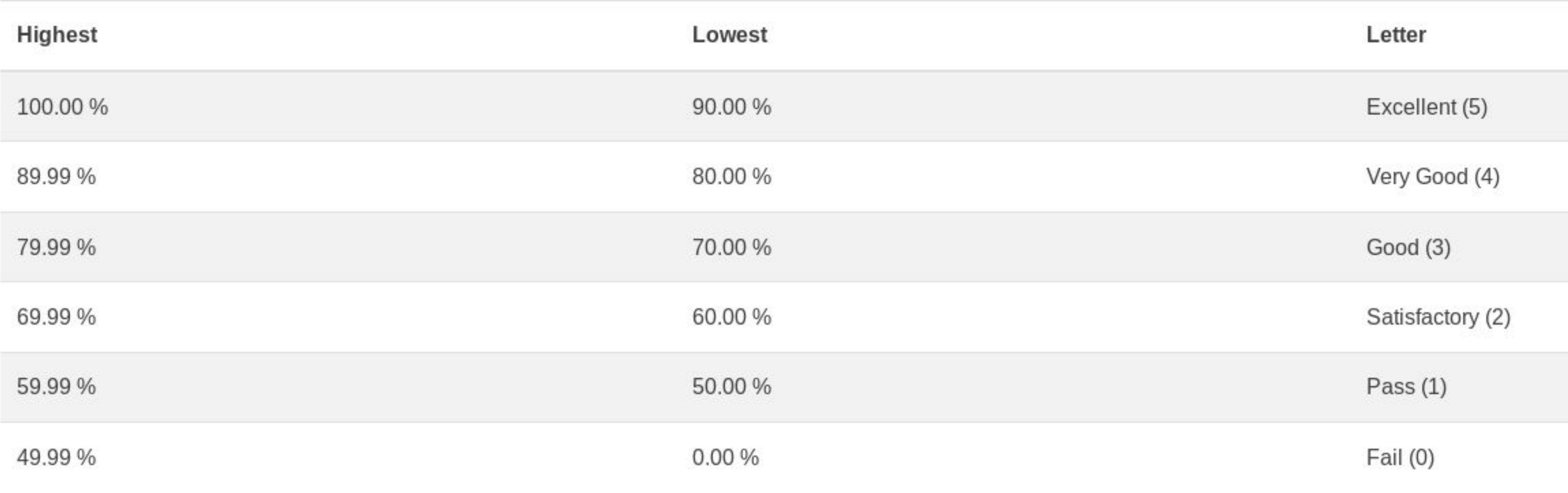
Assessment for each assignment

- **Software artefacts**
  - e.g., code and configuration
- **Data**
- **Written reports in Markdown (https://en.wikipedia.org/wiki/Markdown)**
  - *For explaining your design, evaluation and installation*
- **Records of running results: logs/screenshots**
- **Each part might have a weighted factor of 2 or 3 (e.g., 5*3 = 15 points, with the weighted factor=3)**
- **An assignment should be managed as a git project by yourself**

Assignments

- **Academic honesty**
  - Follow the university rule, peer discussion is OK but *creating your own solution*
  - Check the serious consequence of academic violations here https://github.com/rdsea/bigdataplatforms/blob/master/violations.md
  - Pay attention to the use of chat.openai or GitHub Copilot (reuse and attribution principles)
- **All deadlines are hard**
- **Flexible face-to-face to discuss your assignment submission**
  - you demonstrate your understanding of your solution!

Final grading mapping



Highest	Lowest	Letter
100.00 %	90.00 %	Excellent (5)
89.99 %	80.00 %	Very Good (4)
79.99 %	70.00 %	Good (3)
69.99 %	60.00 %	Satisfactory (2)
59.99 %	50.00 %	Pass (1)
49.99 %	0.00 %	Fail (0)

Some incomplete statistics of previous years can be found in the course Git space!

Flexibility versus limitation

- **Can use Java, JavaScript/TypeScript, Python and shell scripts only**
  - We are elastic but we cannot handle all possibilities
- **Use the recommended dataset and technologies**
  - But you can propose your own dataset
- **Deadlines are hard (don’t be surprised!)**
  - We cannot be flexible in order to guarantee the grading on-time
  - Special exception handling is case-by-case (e.g., sickness, family issue)

Resources

- **Check hints from the course Git/Mycourses**
  - Main Git https://github.com/rdsea/bigdataplatforms
  - Assignment template: gitlab
- **Computing infrastructures and data**
  - Google Cloud Platform
  - Many tests can be run in your own computers with virtualization technologies enabled
  - Try to use Cloud free services (see course materials)
  - CSC if you can get the resource: https://rahti.csc.fi/

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io