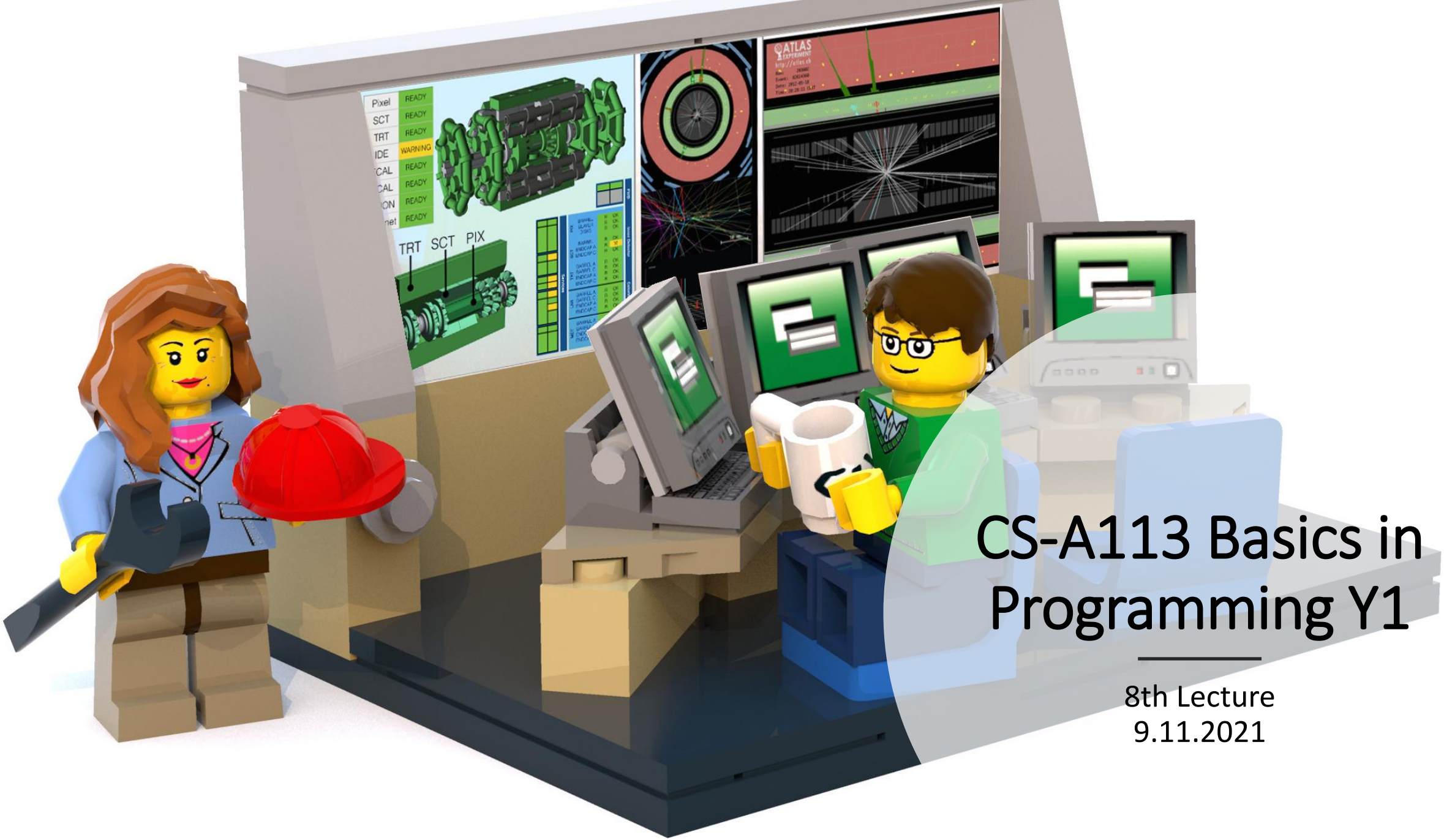


CS-A113 Basics in Programming Y1

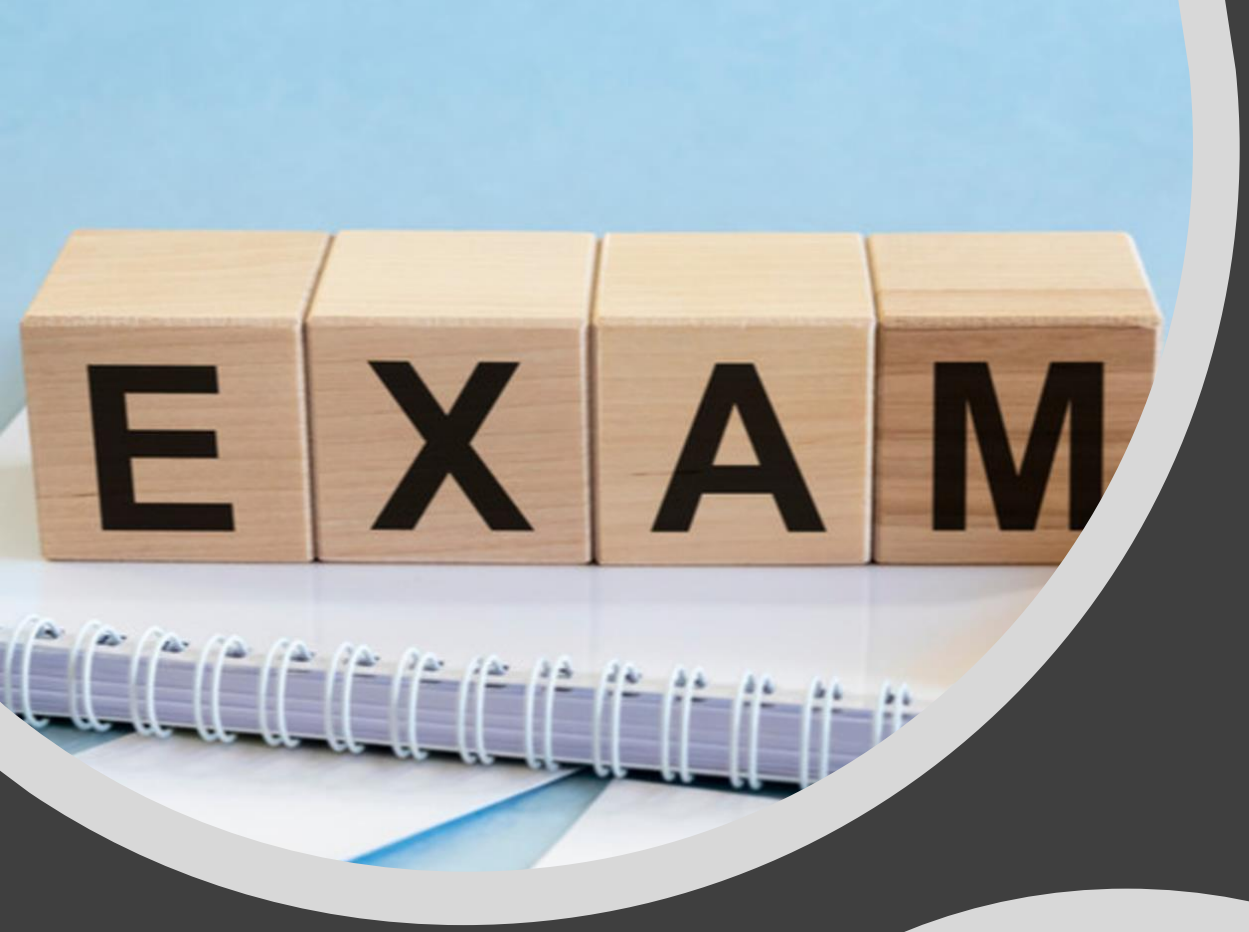
8th Lecture
9.11.2021

The Lecture

- **Join with Video** – Makes my life nicer!
- Feel free to open your microphone and ask questions
- Feel free to write questions into the chat
- We will record the sessions and put it unlisted on youtube.

Interactions Today:

Go to:
http://presemo.aalto.fi/csa1113

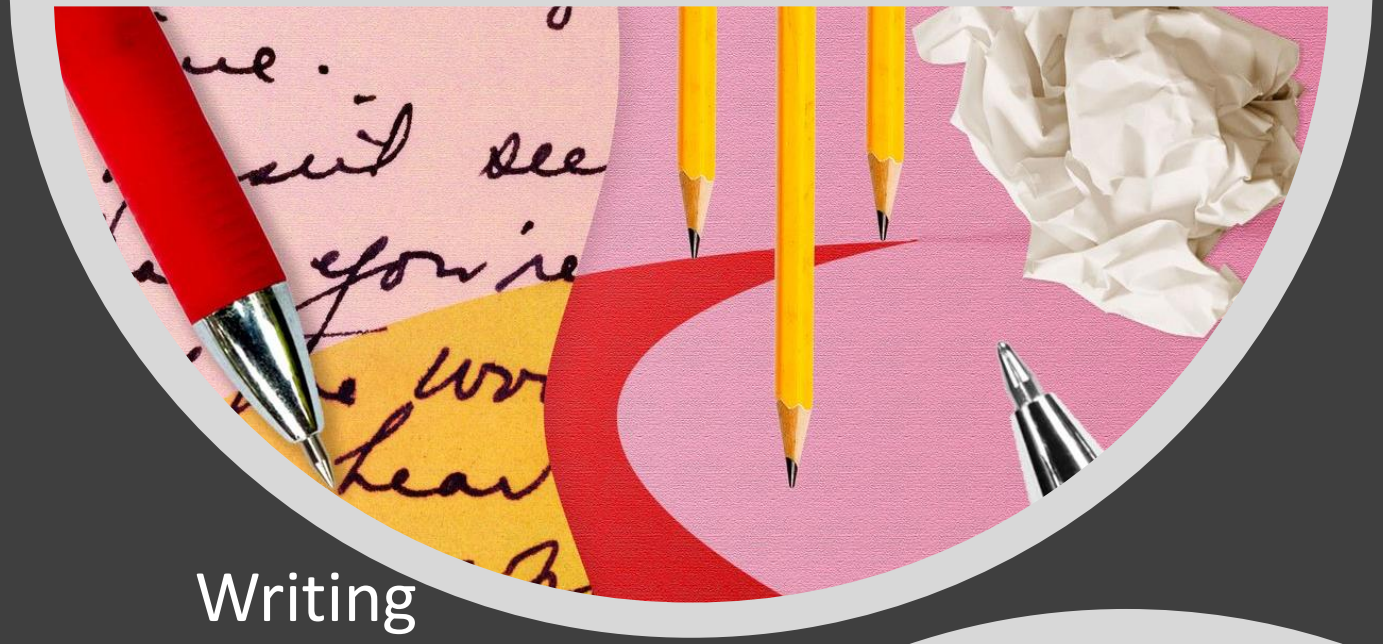


Course Information

- **Exam in EXAM rooms**
  - 8.-21.12.
  - You will need to sign up for it:
    - On SISU
    - AND for the EXAM
  - Instructions on myCourses
  - Mock-Exam available on A+ https://plus.cs.aalto.fi/cs-a1113/2021practiceexam/

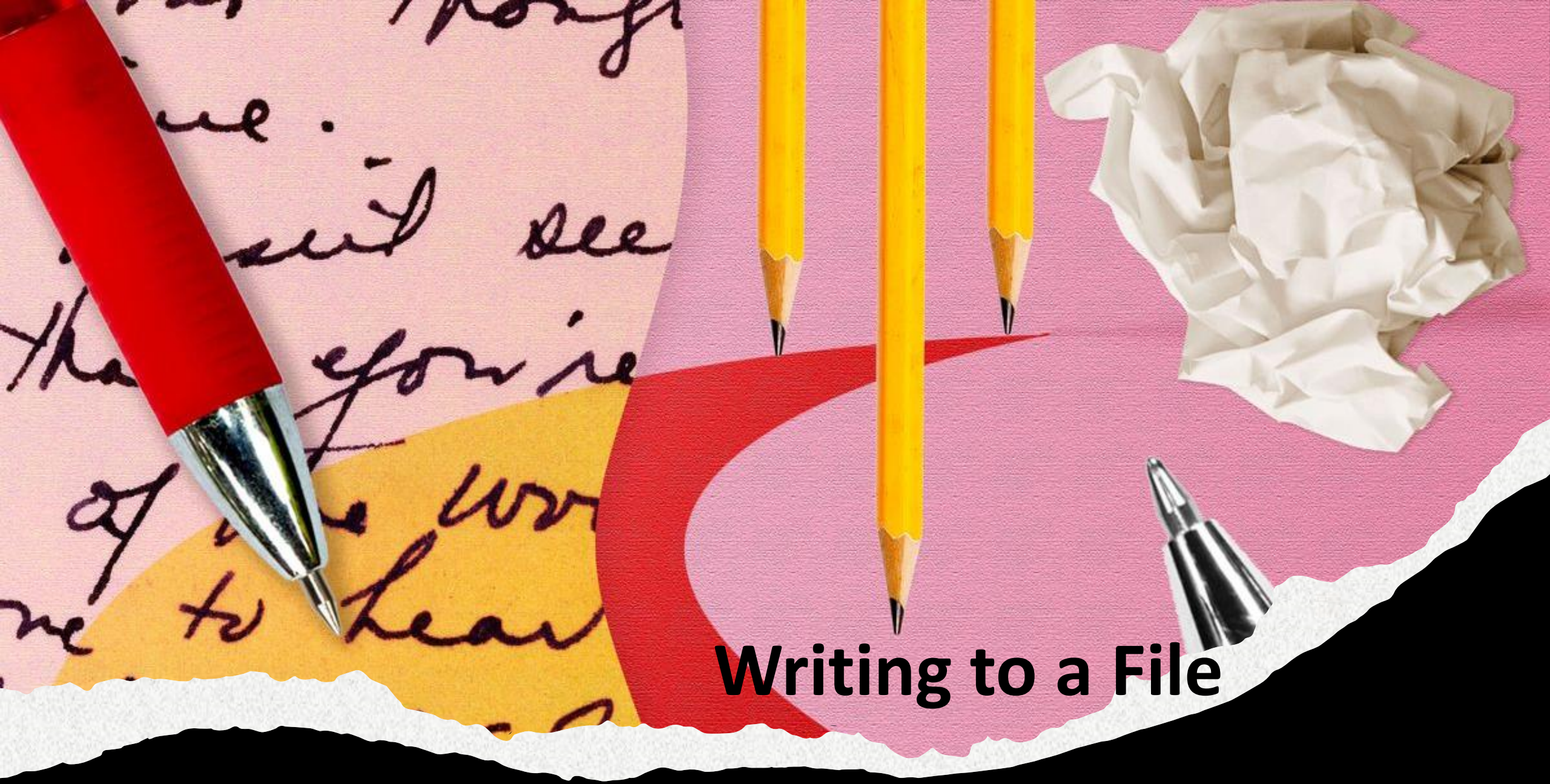
- **Substitute Exercises:**
  - 18.11 – 10-12
  - Substitute as many rounds as you missed exercises
  - You need to reach the passing points of the substitute exercise, but you will only get Minimum passing points of missed exercise (no points are accumulated)

Topics Today



Writing

Objects



Writing to a File

How to Open and Close a File

Open the file

Do some writing

```
newFile = open(”text.txt”,”w”)
newFile.write(”Woohooo, we can write \n”)
newFile.close()
```

Close the file

How to Open and Close a File

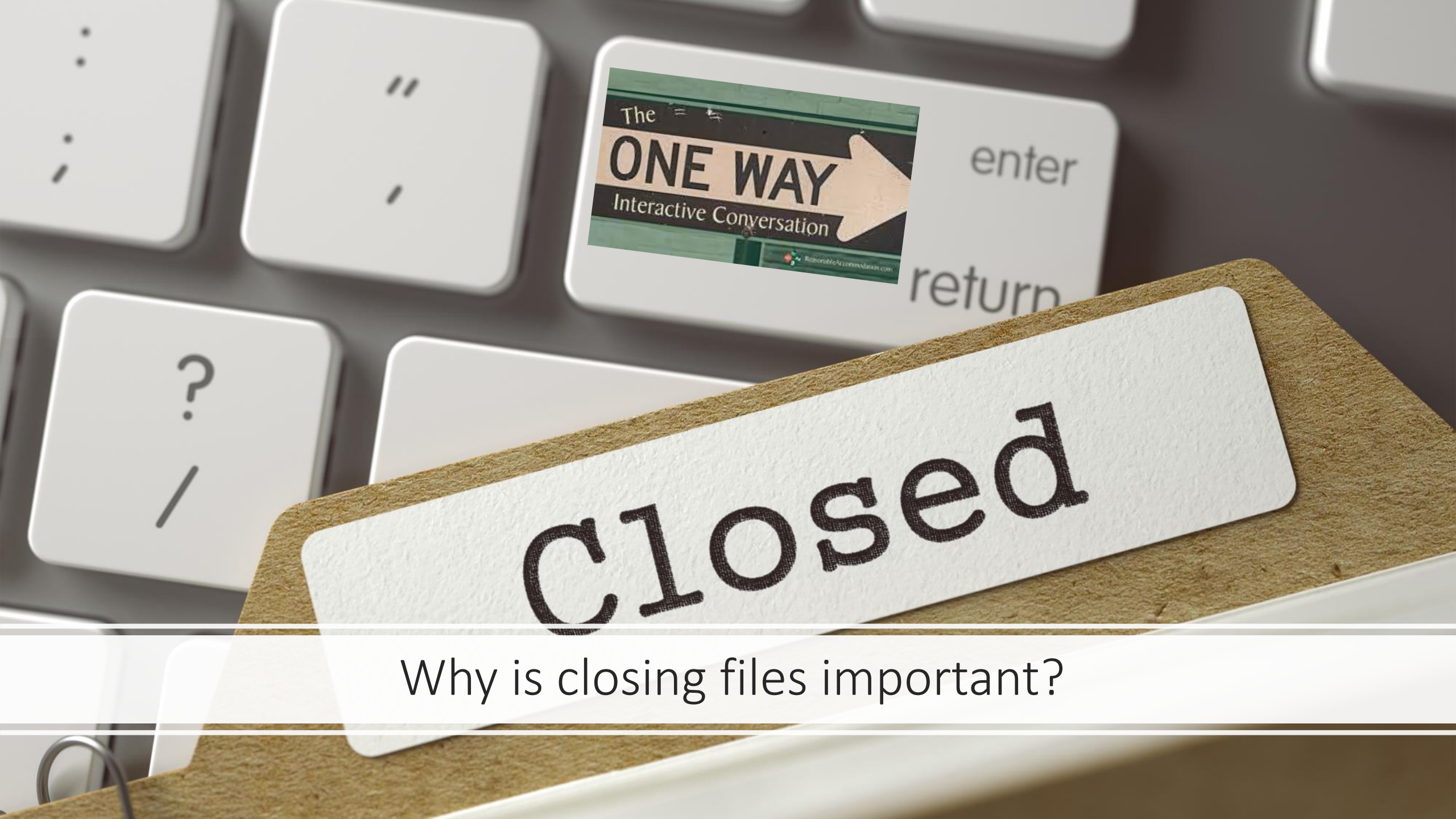
Open the file

Do some writing

```
newFile = open(”text.txt”,”a”)
myList =(”Woohooo \n”, ”we can write \n”)
newFile.writelines(myList)

newFile.close()
```

Close the file

Character	Function
r	Open file for reading only. Starts reading from beginning of file. This default mode.
rb	Open a file for reading only in binary format. Starts reading from beginning of file.
r+	Open file for reading and writing. File pointer placed at beginning of the file.
w	Open file for writing only. File pointer placed at beginning of the file. Overwrites existing file and creates a new one if it does not exists.
wb	Same as **w** but opens in binary mode.
w+	Same as **w** but also alows to read from file.
wb+	Same as **wb** but also alows to read from file.
a	Open a file for appending. Starts writing at the end of file. Creates a new file if file does not exist.
ab	Same as **a** but in binary format. Creates a new file if file does not exist.
a+	Same a **a** but also open for reading.
ab+	Same a **ab** but also open for reading.



Why is closing files important?

How do we write?

We can only write Strings

What do you do if you want to write down calculations?

How do you get a new line?

Good to Know

- newFile.open(”text.txt”,”x”)
  x = w: overwrite text.txt
  x = a: append to the already existing file
- newFile.write(myString)
  ”\n” not added automatically
  myString must be a string → convert everytin
  - myString = ”{} is the Answer”.format(42)
  - myString = str(42) + ” is the Answer”
- newFile.close()
  Very important! Otherwise maybe buffer problems → newFile.flush()
- OSError useful here as well (FileNotFoundError is a sublcass)
  Covers extra permission issues or shortage of HD space

Example 1

```
def main1():
    myFile = open("destination.txt","w")
    myFile.write("Hallo")
```

destination.txt

Line0
Line1

A: destination.txt

Hallo

B destination.txt

#emptyFile

C: destination.txt

Line0
Line1
Hallo

Output

A: File A

B: File B

C: File C

D: FileNotFoundError

E: TypeError

Does this ever make sense?

```
def main1():
    try:
        myFile = open("destination.txt","w")
        myFile.write("Hallo")

    except FileNotFoundError:
```

FileNotFoundError: [Errno 2] No such file or directory: "bla/text.txt"

Example 2

```
def main1():
    myFile = open("destination.txt","w")
    myFile.write("Hallo")
    myFile.close()
```

destination.txt

Line0
Line1

A: destination.txt

Hallo

B destination.txt

#emptyFile

C: destination.txt

Line0
Line1
Hallo

Output

- A: File A
- B: File B
- C: File C
- D: FileNotFoundError
- E: TypeError

Example 3

```
def main1():
    myFile = open("destination.txt","w")
    myFile.write("Hallo")
    myFile.write("Hallo")
    myFile.close()
```

destination.txt

Line0
Line1

A: destination.txt

HalloHallo

B destination.txt

#emptyFile

C: destination.txt

Hallo
Hallo

Output

- A: File A
- B: File B
- C: File C
- D: FileNotFoundError
- E: TypeError

Example 4

```
def main1():
    myFile = open("destinationWrong.txt","a")
    myFile.write("Hallo")
    myFile.close()
```

destination.txt

Line0
Line1

A: destinationWrong.txt

Hallo

B destinationWrong.txt

#emptyFile

C: destinationWrong.txt

Line0
Line1
Hallo

Output

- A: File A
- B: File B
- C: File C
- D: FileNotFoundError
- E: TypeError

Example 5

```
def main1():
    myFile = open("destination.txt","a")
    myFile.write("Hallo")
    myFile.write(42)
    myFile.close()
```

destination.txt

Line0
Line1

A: destination.txt

Hallo42

B destination.txt

#emptyFile

C: destination.txt

Line0
Line1
Hallo42

Output

A: File A
B: File B
C: File C
D: FileNotFoundError
E: TypeError

File Writing in Practice

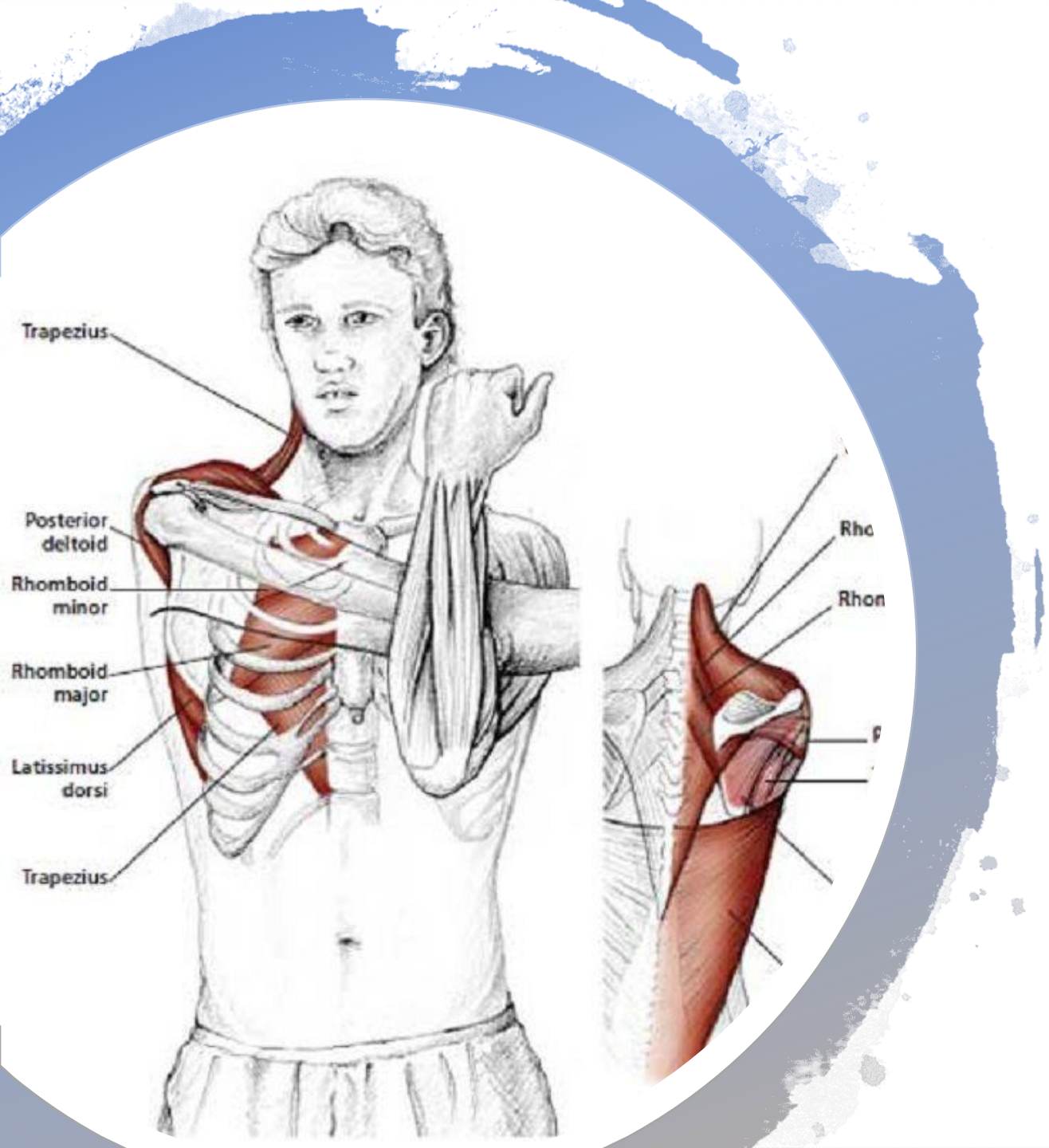
In general reading and writing from and to the same file is error-prone

Often you want to keep the source file unaltered

- In general: Keep source and destination files separate:
  1. read in source file (to appropriate data structure), close it
  2. do calculations
  3. write to your destination file, close it

This will make your life easier if the program crashes

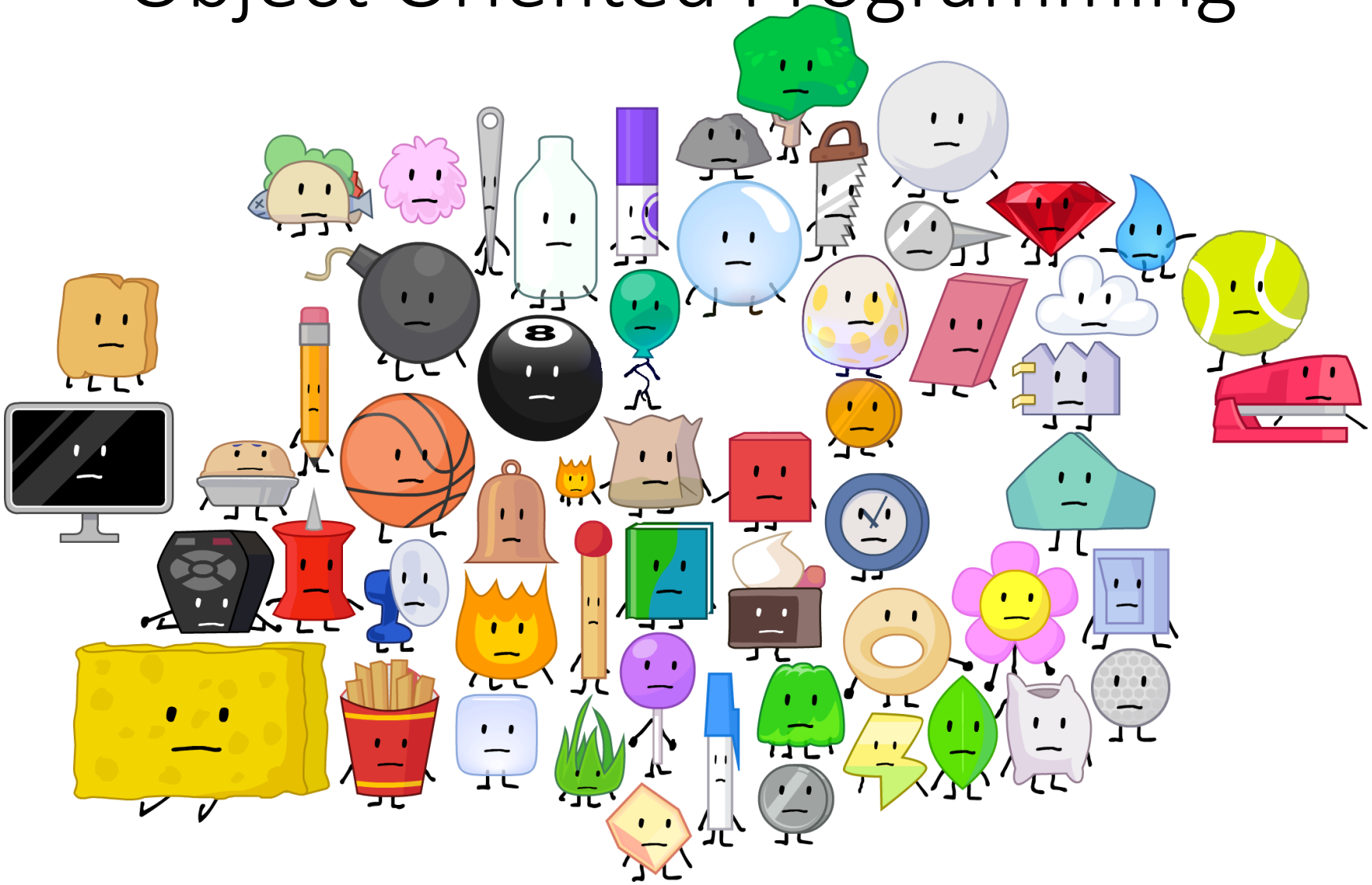
- If you have big data:
  - Do not read in all of it
  - save your progress every now and then (write progress to a file) in a way you can deduce the progress



Break:
Move your Shoulders


Functional
Programming

Object Oriented Programming



How do you implement a Registry for Students

A student has

- a name
- a student number
- courses he/she is enrolled in
- grades

How do you find students?
How do you change the grades?

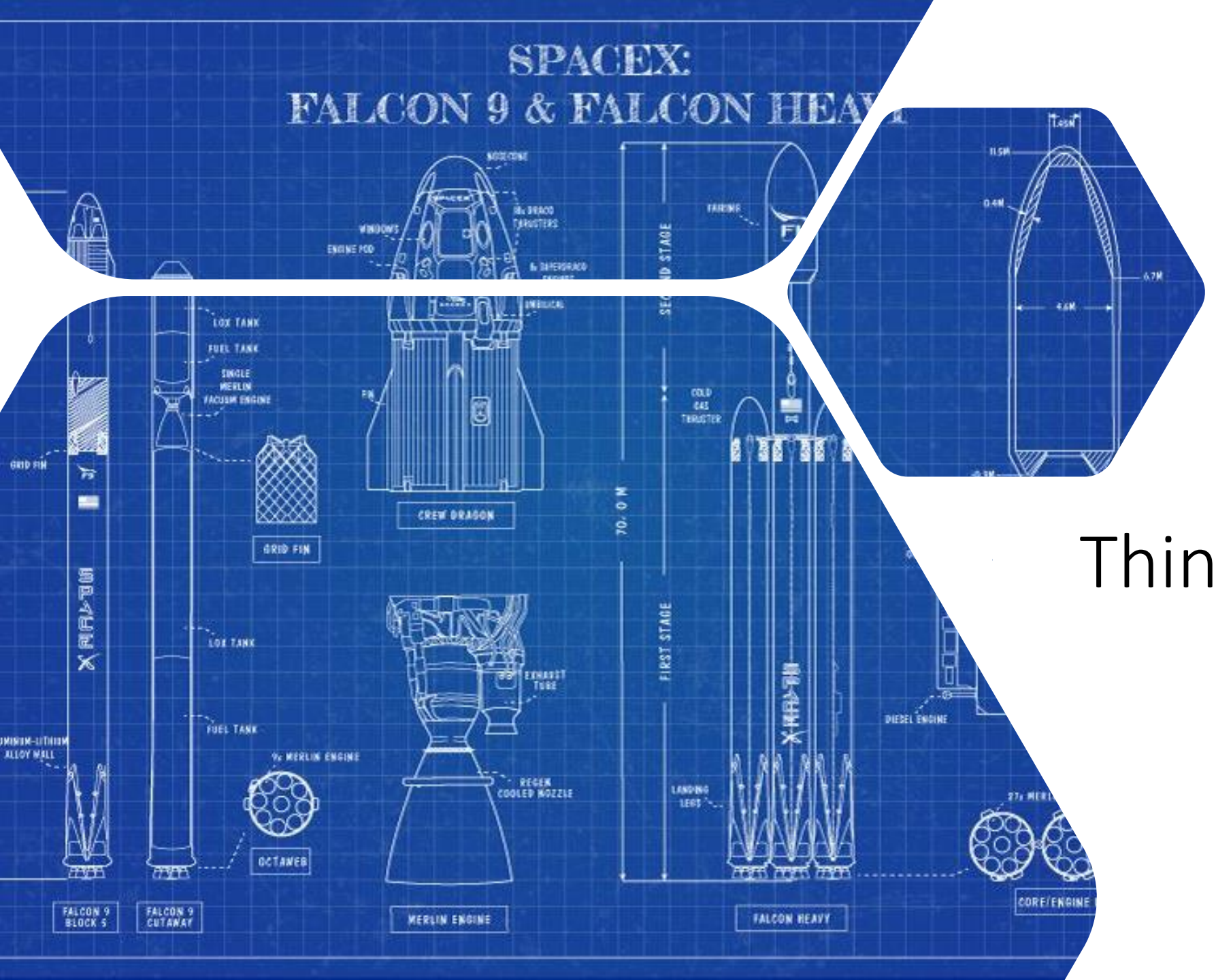
ZOOM BREAKOUT ROOMS

Group Work Task:

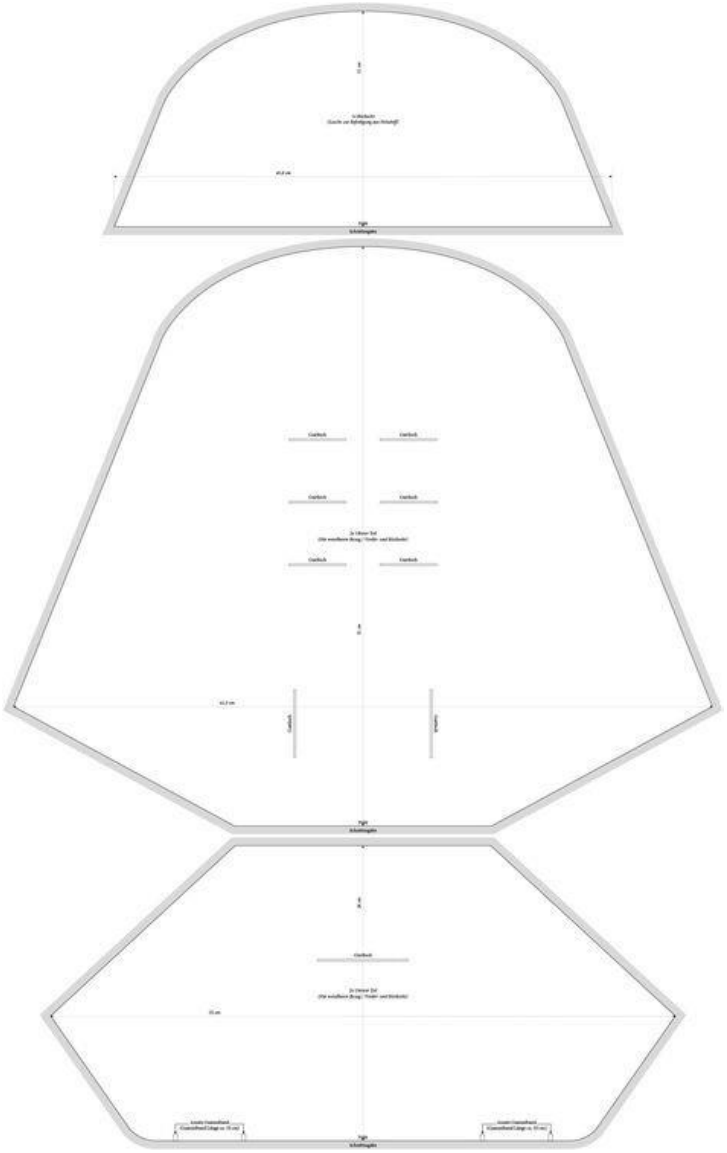
Groups of 4:
Discuss the structure you use for implementing such a student register

Each group writes one entry to Presemo how they structure their students

Go to:
http://presemo.aalto.fi/csa1113



Think of a Class as a blueprint



Sewing Pattern

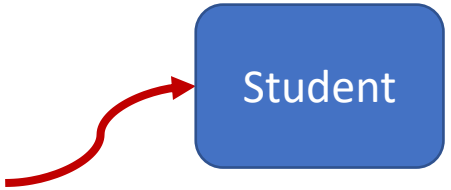
- You can build more than one object from it
- It describes the underlying structure
- It is not an object itself

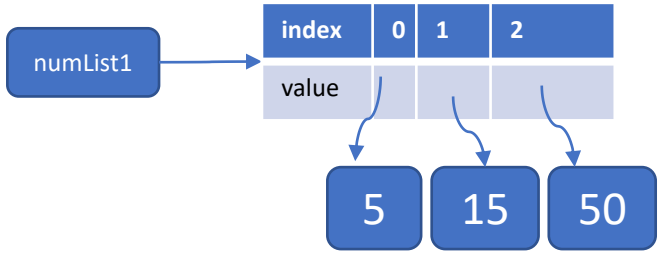
Class

```
class Student:

    def _ _ init_ _(self, myName, myNumber):
        self._ _ name = myName
        self._ _id = myNumber
        self._ _grades = [ ]
        self._ _courses = [ ]


main():
    student1 = Student(”Barbara”,”123”)
```

Class

```
class Student:

    def _ _ init_ _(self, myName, myNumber):
        self._ _ name = myName
        self._ _id = myNumber
        self._ _grades = [ ]
        self._ _courses = [ ]


main():
    student1 = Student("Barbara","123")
```





Class

```
class Student:

    def _ _ init_ _(self, myName, myNumber):
        self._ _ name = myName
        self._ _id = myNumber
        self._ _grades = [ ]
        self._ _courses = [ ]


main():
    student1 = Student(”Barbara”,”123”)
    student2 = Student(”Angelina”,564)
    student3 = Student(”Brad”, 897)
```

Class

```
class Student:

    def _ _ init_ _(self, myName, myNumber):
        self._ _ name = myName
        self._ _id = myNumber
        self._ _grades = [ ]
        self._ _courses = [ ]

main():
    student1 = Student(”Barbara”,123)
    student2 = Student(”Angelina”,564)
    student3 = Student(”Brad”, 897)
    studentRegistry = (student1,student2,student3)
```

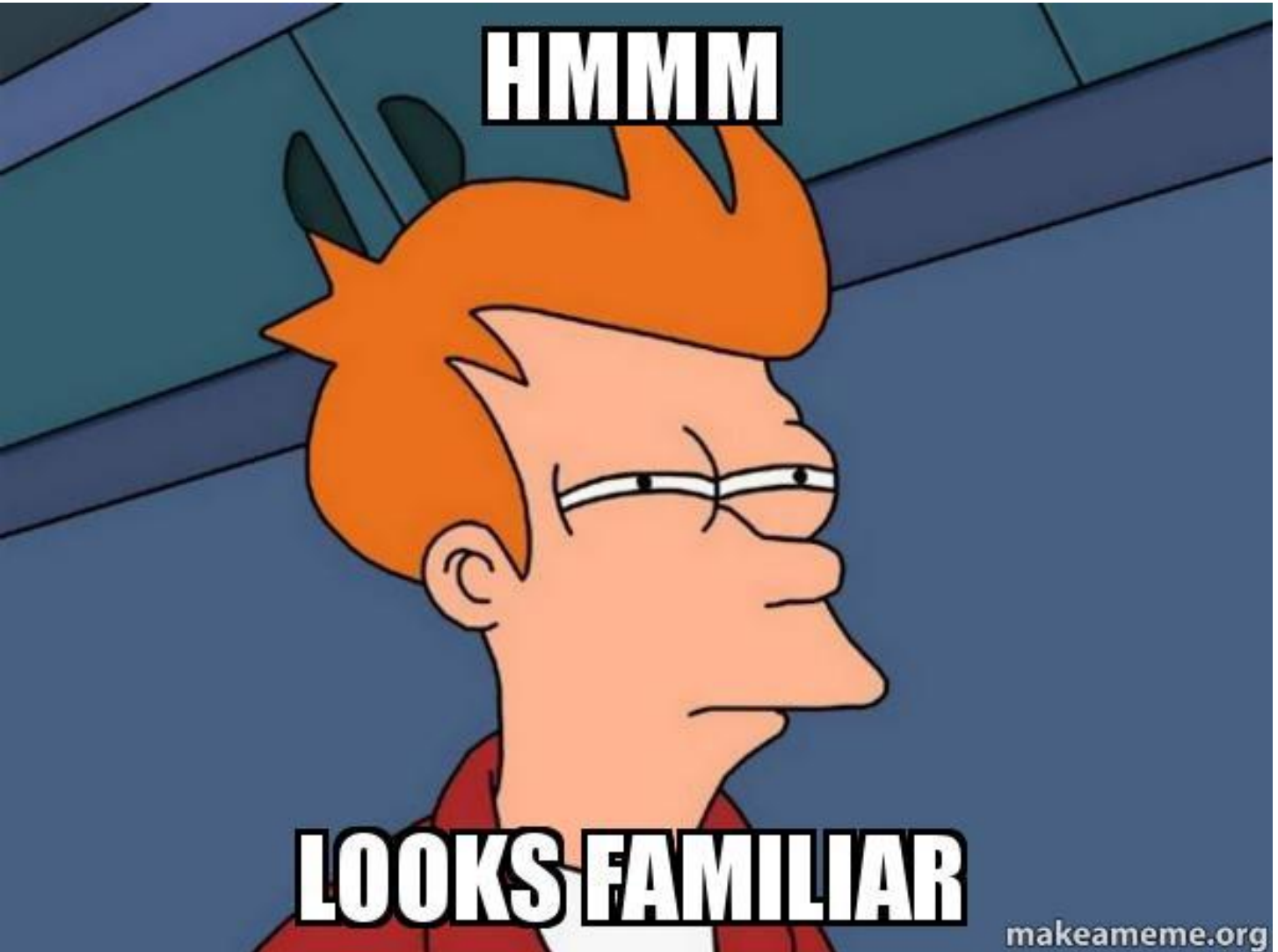
Class

```
class Student:

    def _ _ init_ _(self, myName, myNumber):
        self._ _ name = myName
        self._ _id = myNumber
        self._ _grades = [ ]
        self._ _courses = [ ]

    def add_course(self,course):
        self._ _courses.append(course)

main():
    student1 = Student(”Barbara”,123)
    student2 = Student(”Angelina”,564)
    student3 = Student(”Brad”, 897)
    studentRegistry = (student1,student2,student3)
    student1.add_course(”Basics in Programming”)
    student2.add_course(”Algorithms and Datastructures”)
```

Class

```
class Student:

    def add_course(self,course):
        self.__courses.append(course)

main():
    student1 = Student("Barbara",123)
    student1.add_course("Basics in Programming")
    student2.add_course("Algorithms and Datastructures")
```

Does this look familiar?



Where did you see something like this already?

myList.append(”x”)

myList.sort()

myDictionary.keys()

random.randint()


"That's all Folks!"
l s b e r g