# Scientific Software Development

Inga Ulusoy, Scientific Software Center, Interdisciplinary Center for Scientific Computing, Heidelberg University

March 2023

# Unit 3: Think before you code: Planning your programming project

- The programming paradigm
- Planning a piece of software: Demonstrations
- Planning a piece of software: Do’s and don’ts

A python module will be planned and you will start the development with your team.

# The programming paradigm

The programming paradigm is an approach of solving a programming problem.

**declarative** ⟺ **imperative**

Programming through defining goals but not on how to achieve these.
Focus on *what to do*.

- Logic programming paradigm
- Functional programming paradigm (Java script)
- Database processing approach

Programming through step-by-step instructions (explicit status updates of the program).
Focus on *how to do something.*

- *Procedural programming paradigm*
- *Object-oriented programming paradigm*

**Different languages support different paradigms.**
Python: Imperative, Procedural, Object-oriented, Functional

# The programming paradigm

- Procedural: Statements are structured into procedures (subroutines/`functions`) with main program calling the procedures.
  Allows reuse of procedures through modules/libraries
  Multitude of modules can lead to overhead, duplication and difficulty finding correct calls
- Object-oriented: Objects contain both data and methods (`classes`).
  Data hiding (security), code reusability, inheritance
  Programming becomes quite complex, harder to implement logic
- Functional: Statements are formulated as evaluations of mathematical functions using `lambda` calculus.
  Simple to understand and debug
  Low performance of code, hard to implement

# Unit 3: Think before you code: Planning your programming project

- *The programming paradigm*
- Planning a piece of software: Demonstrations
- Planning a piece of software: Do’s and don’ts

A python module will be planned and you will start the development with your team.

# Planning a piece of software

- I will demonstrate these three different paradigms on a small example.

# Planning a piece of software

Take a small example as in the video, for example the factorial of a given number (https://en.wikipedia.org/wiki/Factorial), and try to implement it using two different paradigms. Can you identify advantages and disadvantages of one or the other paradigm in your specific example?

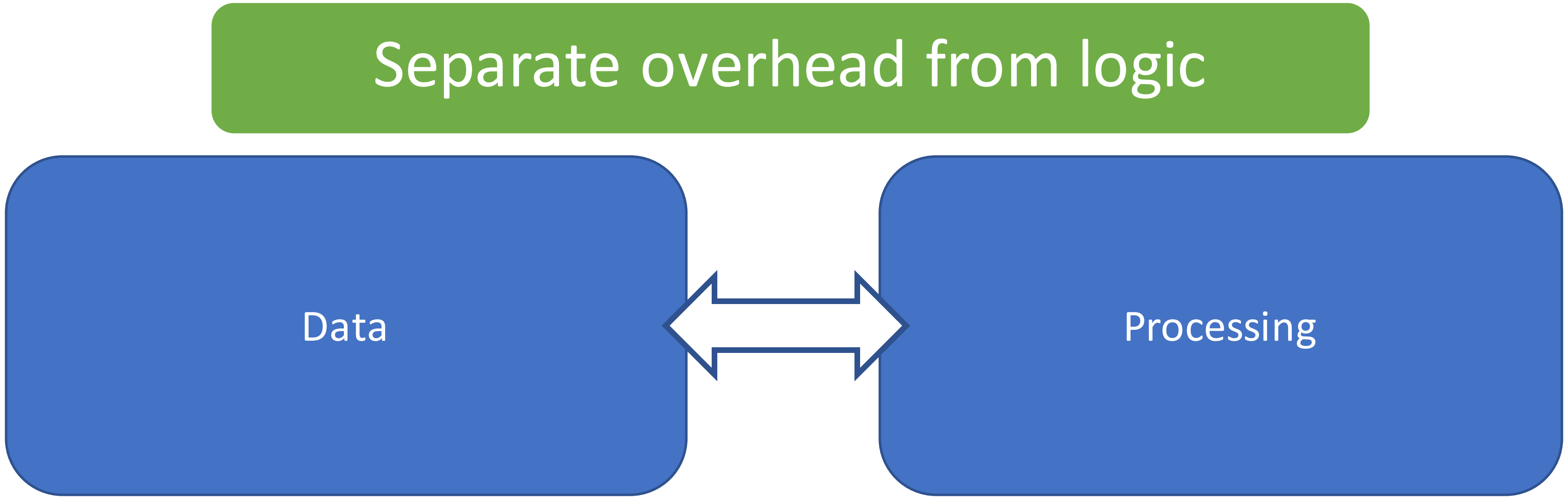
# Unit 3: Think before you code: Planning your programming project

- *The programming paradigm*
- *Planning a piece of software: Demonstrations*
- Planning a piece of software: Do’s and don’ts

A python module will be planned and you will start the development with your team.

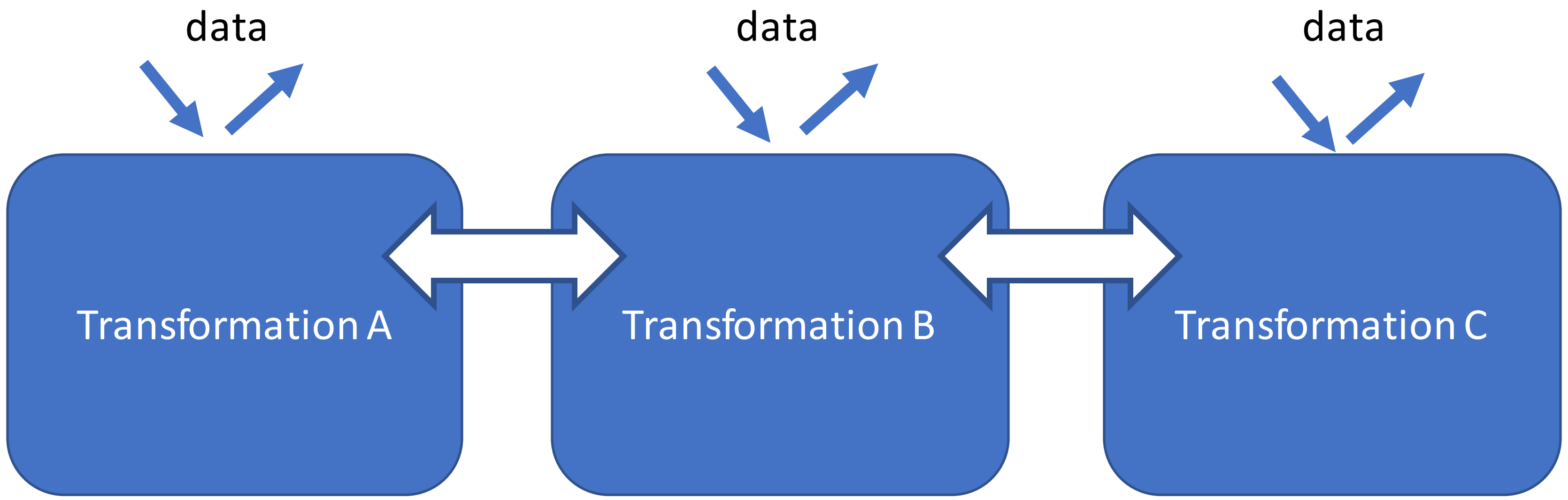
# Planning a piece of software: Do

- Define input
- Define output
- SEPARATE input and output from the main program logic



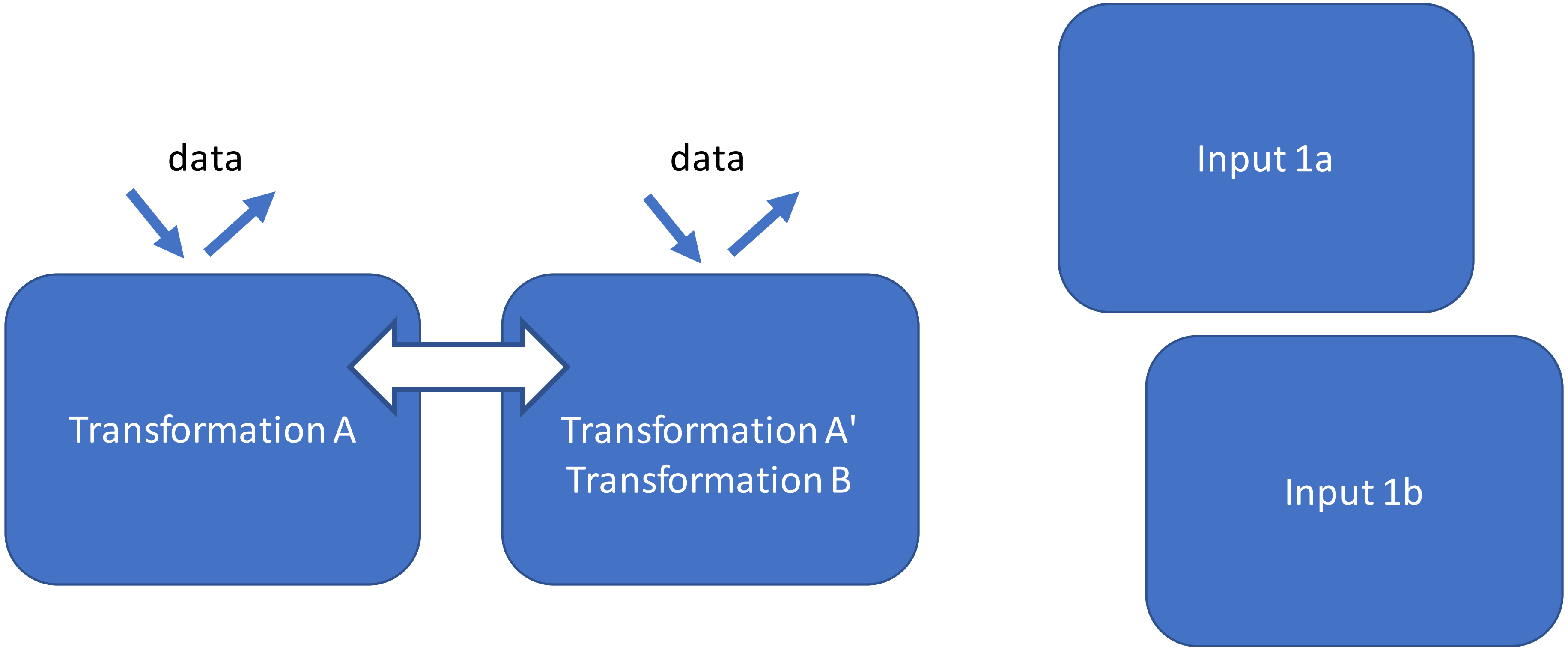
# Planning a piece of software: Do

- Define data flow
- Clarify processing steps from input to output (the logic) of the program



# Planning a piece of software: Don't

- Repeat yourself



# Planning a piece of software: Don't

- Mix logic with data transformation

```
if a then:
    if b then:
        mathematical operation
```

- can make the code hard to read and too complex

# Planning a piece of software: Styles

Top-down:
1. Write logic in pseudo code
2. Work out specifics

Bottom-up:
1. Define specifics
2. Design the logic

*Bottom-up approach better at detecting dependencies that influence top level*

# Unit 3: Think before you code: Planning your programming project

- *The programming paradigm*
- *Planning a piece of software: Demonstrations*
- *Planning a piece of software: Do’s and don’ts*

A python module will be planned and you will start the development with your team.

# Planning a piece of software

- Take a piece of paper and draft a program based on the parts of the Jupyter notebook that you wrote. Consider: programming paradigm (style), top-down or bottom-up? Do this without your ***team***.
- In the design, consider that your ***team*** will also have contributions to the program.

# Live lesson

- In the beginning of the live lesson, you will discuss with your ***team*** how to implement the Jupyter notebook as a Python module. Discuss the strategy that you came up with and your reasoning.
- Identify overlap in the logic/specifics and discuss the implementation considering reusability and clean code.

# Live lesson - Demonstrations

- The following demonstrations will take place later in the live session:
  - Design of the software for this specific example