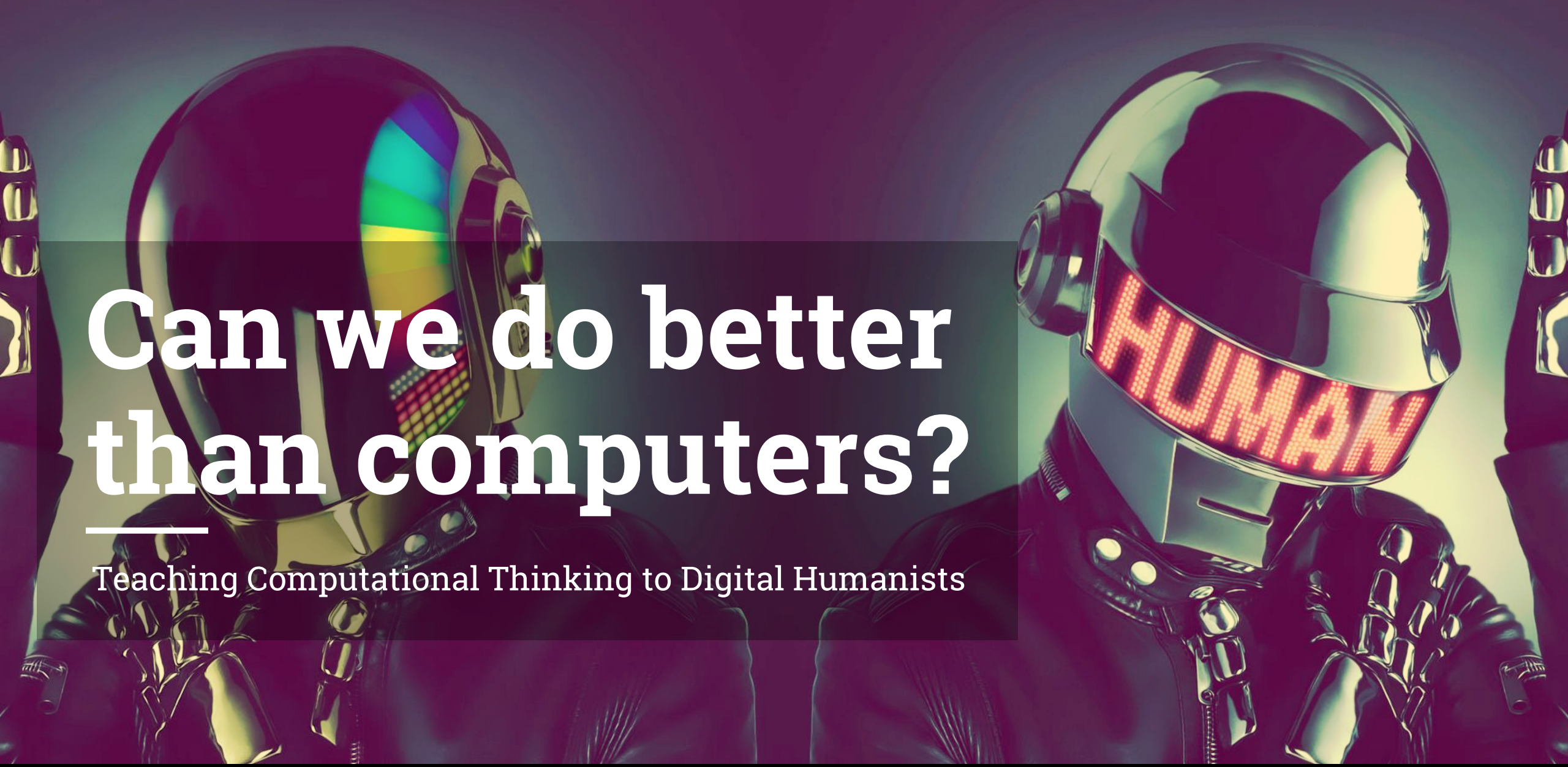


Can we do better than computers?

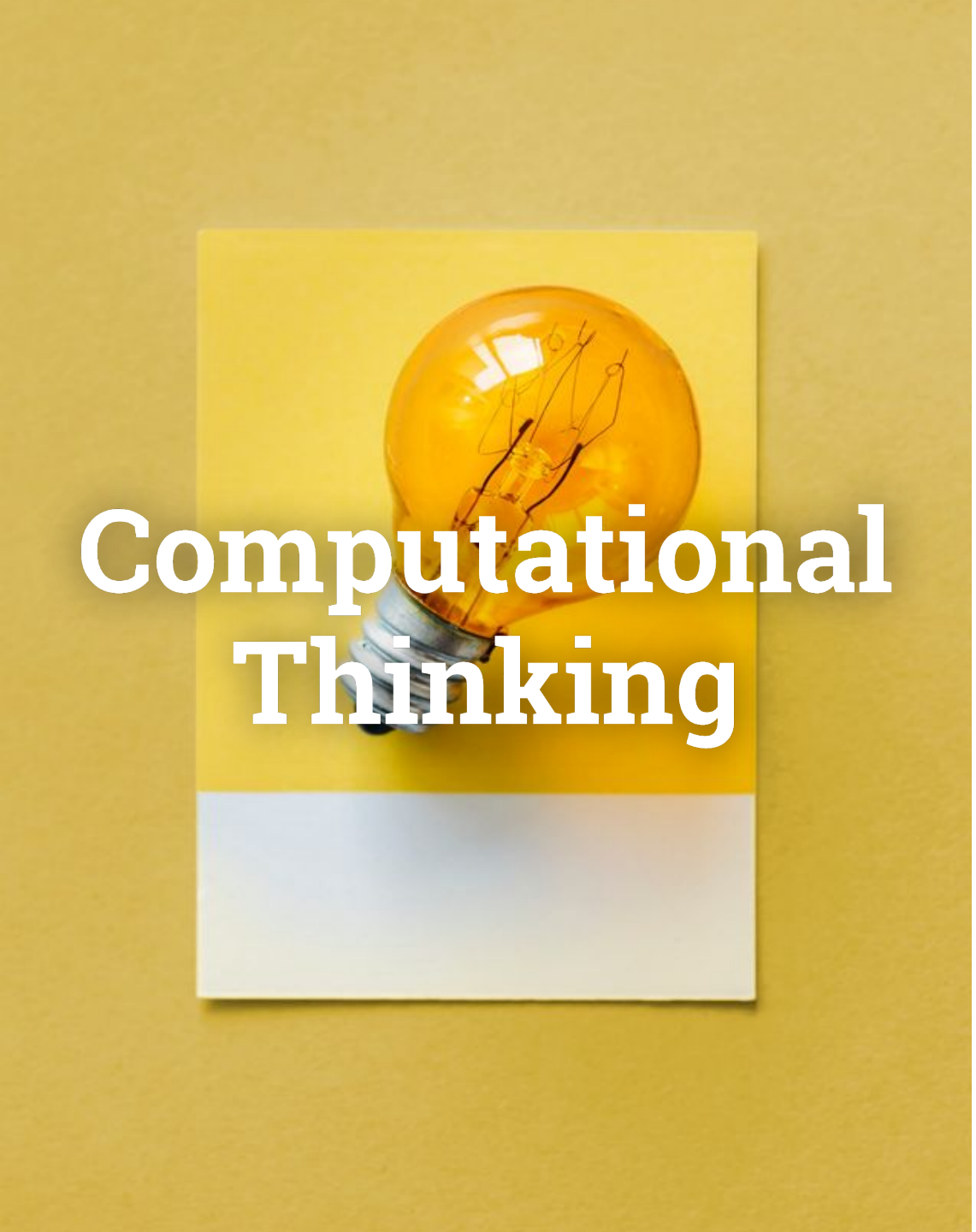
Teaching Computational Thinking to Digital Humanists

Marilena Daquino, Silvio Peroni, Francesca Tomasi
Digital Humanities Advanced Research Centre (/DH.arc), University of Bologna

What researchers usually do

When **designing innovative artefacts** - i.e., creating computational models, constructs, and methods - Digital Humanists try to (1) solve problems that cannot be addressed manually, and (2) learn about the real world.

The achievement of such objectives requires a number of **computational skills**.



What computational skills

"Computational thinking involves **solving problems**, **designing systems**, and **understanding human behaviour**, by drawing on the concepts fundamental to computer science." (Wing 2006, 33)

Attitude VS Artefact

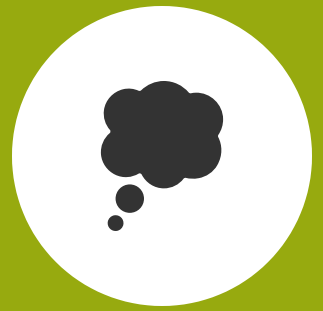
It includes the ability to **think** iteratively and recursively, **abstract** and decompose a complex problem, **design** large systems, use invariants to **describe** the systems, **apply** heuristic reasoning, and so on.

What makes the difference in Digital Humanities

beyond Humanities, Computer Science, and Engineering

provide domain expertise in Humanities fields

Understand mechanisms, obstacles, and requirements of a given domain other than Computer Science.



embrace Computational Thinking

Formulate research questions so as that solutions can be represented as computational steps and algorithms.

adopt Design-science methodology

Create artefacts that extend human capabilities and expand the reach and relevance of the Humanities.

Why learning Computational Thinking

Computational Thinking

- **A necessary background** for many disciplines
- **The red thread** between disciplines hence a tool for future researchers
- **A medium for teaching** other subjects than Computer Science

How to develop computational skills

Teaching Computational Thinking to Digital Humanists

- **Online courses**

  For educators

- **University courses**

  For students and researchers

  Open University (UK)

  Vrije + Huygens + KNAW (NL)

  DHDK (IT)

Computational Thinking and Programming @DHDK

Digital Humanities and Digital Knowledge (DHDK)

- **2-year International MA @unibo**

  started in 2017, around 55 students are currently enrolled

- **Knowledge hybridization**

  computer science, humanities, management, communication, law

- **Students from all around the word**

  Germany, Greece, Lithuania, Netherlands, Portugal, Argentina, Armenia, Bulgaria, China, Camerun, Ethiopia, Iran, Macedonia, Mali, Montenegro, Russia, Serbia and Turkey

- **A tailor-made profile**

Computational Thinking and Programming

People

DHDK Coordinator

Developed the **educational programme** at DHDK balancing Humanities and Technical courses

A Computer Scientist

Held theoretical classes on a number of **CT related topics**.

A Digital Humanist

Held hands-on classes to address **DH scenarios** and **implement algorithms** in Python.

Students

20-40 Master Students with **different backgrounds** (both in the Humanities and CS)

Objectives of the course

- historical background on topics of CT
- understand and use the main data structures
- develop algorithms to address computational problems
- implement algorithms in a programming language

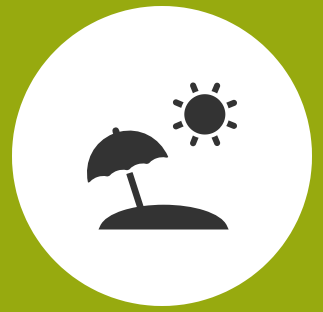
Course organization

- **30h theoretical classes**
  - historical background
  - biographical notes on notable scientists relevant to the topic at hand
  - definition of CT concepts supported by the usage of pseudo-code
- **16h hands-on classes**
  - leverage the knowledge acquired during theoretical classes
  - define common scenarios in DH
  - implement algorithms in Python to solve problems
- **Final exam**
  - partial examinations
  - written test (theory, understanding and practising)
  - group project tackling a real-world problem (e.g. a text-based search engine for querying bibliographic data)

Learning strategies

Learning by example

Show examples of algorithms and data structures in pseudo-code and in Python



Task-oriented learning

Definition of real-world scenarios wherein to apply shown algorithms and solve given tasks

Collaborative learning

Collaborative sense-making and problem solving

Topics

Theoretical classes

- **Flowcharts**

  Input, process, decision, output

- **Basics of programming**

  Variables, assignments, and statements

- **Organizing information**

  List, Stack, Queue, Set, Dictionary, Tree, Graph

- **Algorithmic methodologies**

  Brute-force, Divide and conquer, Dynamic programming, Backtracking, Greedy

- **Development methodologies**

  Test-driven development

- **Computational problems**

  Selection, Sorting, Mathematical computation, Abstract board game, Pagination

4 Scenarios

Hands-on classes

- **[LIS] Library and Information Science**
  - Manipulation of bibliographic data
- **[NLP] Natural Language Processing in Literature**
  - Content analysis of a corpus of literary texts
- **[WCH] Web scraping and fuzzy string matching**
  - Data reconciliation of authority files to existing datasets
- **[KO] Knowledge organization**
  - Extract a taxonomy of concepts from data

Scenarios, topics, and learning objectives

LIS

comp. problem: select, sort

data structures: list

LO: basics of programming; manipulate tabular data

NLP

comp. problem: math. comp.

data structures: set, tuple, dictionary

LO: use libraries; manipulate unstructured data

W-CH

comp. problem: select

data structure: tree

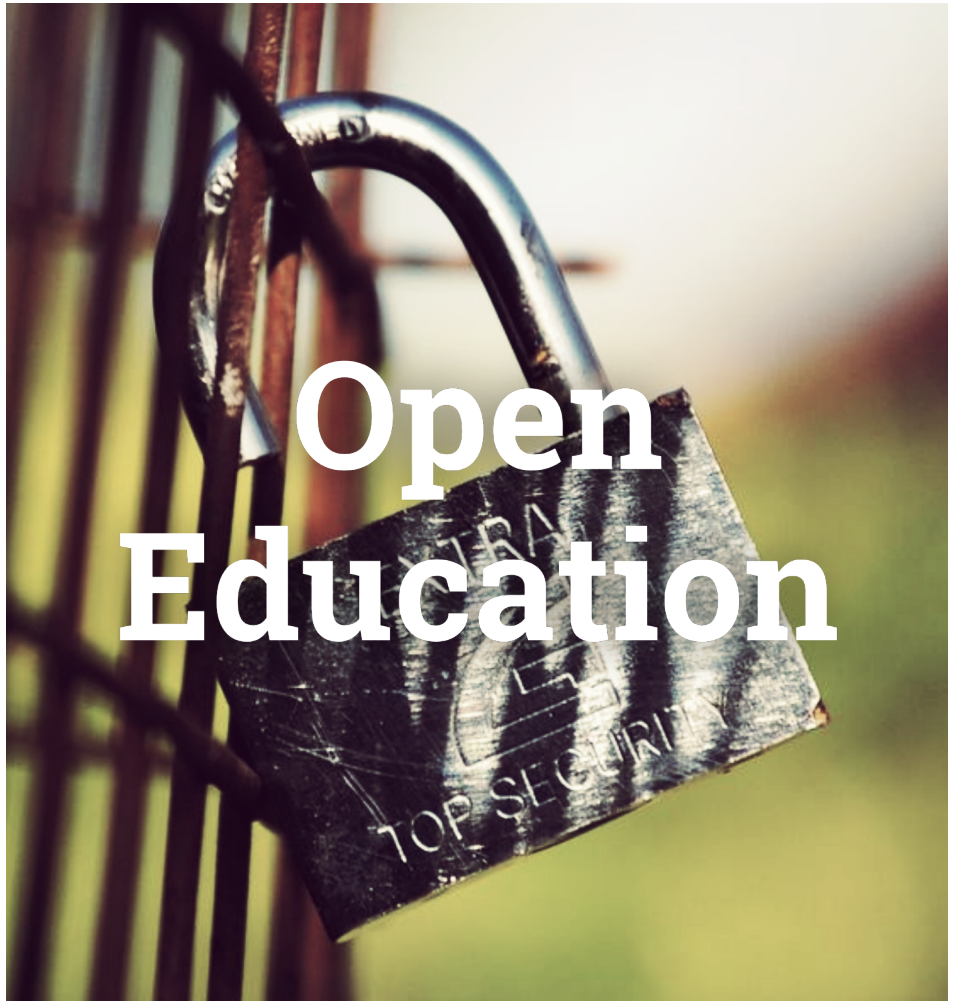
LO: rewrite somebody's code; query tree data

KO

comp. problem: select

LO: manipulate and create tree data

Materials and teaching tools



- **Materials**

  https://github.com/comp-think/

- **Tools**
  - RASH theoretical classes in machine-readable format
  - RAJE RASH editor
  - Google docs, HTML
  - GitHub
  - Jupyter notebook hands-on classes in python+markdown
  - Telegram

Preliminary evaluation and future work

- **Questionnaires**

  only first year students' questionnaires were analysed by using grounded theory techniques - used to **improve second year course** (e.g. retention period, mix theoretical and hands-on)

- **Future work**

  formalise a fixed **knowledge retention period** and measure its efficiency by means of a user-satisfaction evaluation and partial exams

Thanks!

QUESTIONS?