

SWE 621

FALL 2020

---

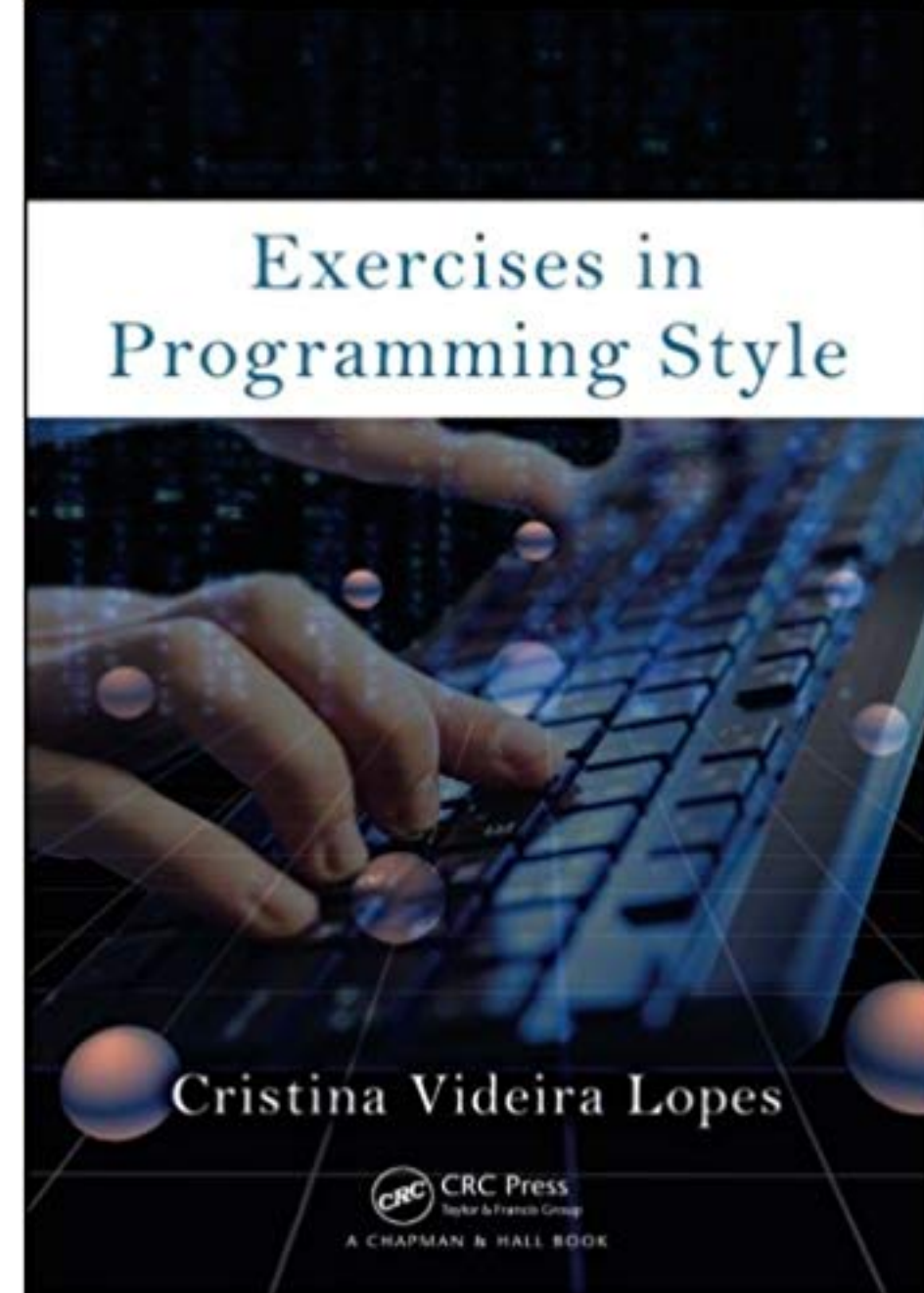
# PROGRAMMING STYLES

# LOGISTICS

- ▶ HW5 due on 4/28
- ▶ Project presentation on 5/5
  - ▶ Will summarize what you've learned about your systems over the course of the semester
  - ▶ More details next week

# PROGRAMMING STYLE

- ▶ A set of constraints on how code is written which help achieve specific requirements or quality attributes
- ▶ Describe alternative ways in which code might be written
  - ▶ make it object-oriented
  - ▶ make it functional
  - ▶ lazily load data from input source
  - ▶ give each element a separate thread
- ▶ Like architectural styles and design patterns, has consequences that adopting programming style help achieve
  - ▶ But not always as well-defined and enumerated



# EXERCISES IN PROGRAMMING STYLE

- ▶ Presentation is centered around an example problem
- ▶ Each program offers the same baseline behavior (sometimes adding an additional feature)
- ▶ Can directly compare and contrast how the same problem is solved each style
- ▶ Directly illustrates the diversity of ways of programming
  - ▶ Many different ways to solve the same problem
- ▶ Some are related to programming language features (e.g., OO, functional, reflection)
  - ▶ But many modern languages support a range of language features that support a diversity of styles
  - ▶ Can write something in a procedural style (i.e., ignoring OO features) even in Java
- ▶ Examples written in Python

# EXAMPLE PROBLEM: TERM FREQUENCY

- ▶ Given a text file, print the 25 most frequent words and corresponding frequencies
- ▶ Sort from most frequent to least frequent
- ▶ Normalize for capitalization and ignore "stop" words (e.g., the, for, ...)

## Input

Tigers live mostly in India

Wild lions live mostly in Africa

## Output

live - 2  
mostly - 2  
africa - 1  
india - 1  
lions - 1  
tigers - 1  
wild - 1

# SOME TYPES OF PROGRAMMING STYLES

- ▶ Basic styles
- ▶ Functional styles
- ▶ Reflection styles
- ▶ Data-centric styles
- ▶ Concurrency styles

# EXAMPLES OF PROGRAMMING STYLES

<https://github.com/crista/exercises-in-programming-style>

- 5-cookbook/procedural <https://github.com/crista/exercises-in-programming-style/tree/master/05-cookbook>
- 6-pipeline <https://github.com/crista/exercises-in-programming-style/tree/master/06-pipeline>
- 7-code golf <https://github.com/crista/exercises-in-programming-style/blob/master/07-code-golf/tf-07.py>
- 8-infinite mirror / recursive <https://github.com/crista/exercises-in-programming-style/tree/master/08-infinite-mirror>
- 10-things/OO <https://github.com/crista/exercises-in-programming-style/tree/master/11-things>
- 15-hollywood/inversion of control <https://github.com/crista/exercises-in-programming-style/tree/master/15-hollywood>
- 16-b board /publish subscribe <https://github.com/crista/exercises-in-programming-style/tree/master/16-bulletin-board>
- 19-aspects <https://github.com/crista/exercises-in-programming-style/tree/master/19-aspects>
- 20-plugins <https://github.com/crista/exercises-in-programming-style/tree/master/20-plugins>
- 25-persistent tables/relational <https://github.com/crista/exercises-in-programming-style/tree/master/26-persistent-tables>
- 28-lazy rivers/streams <https://github.com/crista/exercises-in-programming-style/blob/master/28-lazy-rivers/tf-28.py>
- 31-map reduce <https://github.com/crista/exercises-in-programming-style/tree/master/31-map-reduce>

# SUMMARY

- ▶ Many choices about how to implement a solution
- ▶ Programming styles offer a vocabulary for talking about alternative implementations
- ▶ Makes explicit the constraints which lead to a specific style of programming
  - ▶ Can consider explicitly the consequences of following these constraints



# IN CLASS ACTIVITY

# SKETCH IMPLEMENTATION IN LAZY-RIVER STYLE

- ▶ Work in groups of 2 or 3, pick an OO language (e.g., Java, Python, C#)
- ▶ Sketch an implementation of the following
  - ▶ Given a text file, output all words alphabetically, along with the page numbers on which they occur. Ignore all words that occur more than 100 times. Assume a page is a sequence of 45 lines.
  - ▶ abatement - 89  
abhorrence - 101, 145, 152, 241, 274, 281  
abhorrent - 253  
abide - 158, 292
- ▶ Does not need to compile and run, just looking for a sketch that illustrates following the programming style for this problem
- ▶ Hand in through Blackboard