Heinrich-Heine-University Düsseldorf
Computer Science Department
Software Engineering and Programming Languages
Philipp Körner

**Functional Programming – WT 2023 / 2024**
**Reading Guide 2: First Steps at Programming**

## 1 Material

- Learning Videos:
  - Lambda Calculus: `https://mediathek.hhu.de/watch/6cfddfc9-6d92-494f-90a0-8637b8b7f698`
  - Syntax: `https://mediathek.hhu.de/watch/4aa60c78-23e5-48be-a5f6-b676dd7f2013`
  - Coding Guidelines: `https://mediathek.hhu.de/watch/9b42cb61-e871-48c3-8ec8-0aca1f30a730`
- Rich Hickey: Clojure for Java Programmers `https://www.youtube.com/watch?v=P76Vbsk_3J0` (until 1:35:38 — introduction to the Clojure language and its motiviation — optional revision and introduction)
- Alternative material:
  - Clojure for the Brave and True, chapter 3 (covers all material)
  - 01_intro.clj (covers all material)
- Please be aware of `https://clojure.org/api/cheatsheet` and that there are many useful functions. Especially the blocks “Sequences”, “Primitives” and “Collections” are worth knowing. You do not have to learn everything by heart now, but go ahead and find out, what some functions do.

**Timeline:** This unit should be completed by 23.10.2023.

## 2 Learning Outcomes

After completing this unit you should be able to

- determine the evaluation order of elements in a function call.
- define functions with local bindings.
- use the fundamental control structures in small functions.
- recall fundamental sequence operations provided by the standard library.

## 3 Highlights

- $\beta$-reduction as evaluation-mechanism
- Lists and sequences
- Special forms: `def`, `fn`, `if`, `let`, `do`
- Sequence operations: `first`, `rest`, `range`, `cons`, `conj`, `assoc`, `dissoc`, `disj`
- Macros: `ns`, `for`, `cond`, `and`, `or`

## 4 Exercises

**Notes** Solving the exercises is voluntary. Admission to the exam is solely based on passing the tests in the ILIAS. Since programming is required in the exam it is highly recommended to work on the exercises.

**Exercise 2.1 (4clojure Exercise Unlocks — Recommended!)**
After completing this unit, you gained the knowledge to solve the following exercises (`https://4clojure.oxal.org/` — please bookmark for future reference):

- elementary: 4–13, 15–16, 35, 37, 57, 134, 145, 156, 161

**Exercise 2.2 (Sequences)**
In the following exercise simply define the described sequences. As Example: If you are to define the sequence of numbers between -100 and 100, a valid solution is: `(def example-seq (range -100 101))`

a) Define the sequence `seq-a`, containing all integers from 100 to -100 in descending order.

b) Define the sequence `seq-b`, containing all square integers between 0 and 1000.

c) Define the sequence `seq-c`, containing all integers between 0 and 1000, which are not evenly divisible by 3.

d) Define the sequence `seq-d`, containing all tuples [n, m] of integers for which the following holds: $0 < n < 1000 \wedge n^2 < m \wedge$ m is minimal. The sequence begins with tuples [1,2], [2,5], [3,10].

e) Define the sequence `seq-e`, containing all integers with 5 digits (represented as 5 tuple), which are palindromes. Additionally the digits should be strictly ascending up to the middle digit. [0 2 3 2 0] is part of the sequence, but [0 3 2 3 0] is not, neither is [1 1 1 1 1].

**Hint** `for` is a useful macro that is similar to list-comprehensions in Python.

## Questions

If you have any questions, please contact Philipp Körner (`p.koerner@hhu.de`) or post it to the Rocket.Chat group.