

## Chapter 2 outline:

- ▶ Mathematical sequences and Python lists (last week Monday)
- ▶ Recurrence relations and recursive functions (last week Wednesday)
- ▶ Functions on lists (last week Friday)
- ▶ More functions on lists (Monday)
- ▶ Review Chapter 1 & 2 (**today**)
- ▶ Test on Chapters 1 & 2 (Friday)
- ▶ (Begin Chapter 3 next week Monday)

## Today:

- ▶ General test comments
- ▶ Review of topics so far
- ▶ Specific test coaching
- ▶ How can I help you?

## Goals of this course

- ▶ Write programs in the functional style
- ▶ Think recursively
- ▶ Understand sets, relations, and functions so that they can model real-world (and abstract) information
- ▶ Use formal logic to prove mathematical propositions.

## Concepts of the first two chapters

- ▶ Sets and their operations
- ▶ Sequences
- ▶ Python expressions, types, and functions
- ▶ Python sets and lists
- ▶ Recursive algorithms

## Concepts

**1.1.** Sets and elements;  $\mathbb{Z}$ ,  $\mathbb{W}$ ,  $\mathbb{N}$ ,  $\mathbb{Q}$ , and  $\mathbb{R}$  as standard examples

**1.2.** Values, expressions, literals, types, operators. The idea of a value in Python (or in computer memory) representing or modeling some real-world or abstract/mathematical information. The types `int`, `float`, `bool`, `str`, and `type`. Integer division and mod (`//` and `%`). String operations—concatenation and multiplication (`+` and `*`), `len`, and `in`.

## Testable skills

Analyze the type of a Python expression. (Similarly, be able to do a type analysis that involves the kinds of Python expressions and the Python types that occur in the subsequent sections.)

## Concepts

**1.3.** Variables, identifiers, functions, parameters (actual and formal), return value/statement, function application. Functions as values that can be stored in variables and passed to other functions.

**1.4.** Denoting sets by listing elements and using set-builder notation. The set type in Python. Python set comprehensions. Python ranges.

**1.5.** Set operations: union, intersection, difference, symmetric difference, complement; subset, set equality. The universal set and the empty set. Python set operators. The analogy between set operations and arithmetic operations (on numbers).

## Testable skills

Write simple Python functions. Write Python functions that call other functions and that take functions as parameters.

Describe a set using set-builder notation. Write Python functions that use set comprehensions and `range`.

Write Python functions that use set operators.

## Concepts

**1.6.** Verifying propositions about the equality of set expressions using Venn diagrams.

**1.7.** Cardinality (modeled by `len` in Python), disjointedness, pairwise disjointedness, partitions, Cartesian products, tuples.

**1.8.** Powersets.

## Testable skills

Verify set equality propositions using Venn diagrams, shading, labeling, and accompanying verbal explanations.

Write Python functions that use tuples.

Write Python functions that use sets of sets (which requires frozen sets).

## Concepts

**2.1.** Sequences, zero-based indexing. Python lists, subscripting/indexing (with []). List comprehensions. List concatenation and multiplication. Negative indexing and slicing.

**2.2.** Recurrence relations. Recursive functions. Conditional expressions and statements.

**2.3.** Recursive processing of lists, such as splitting a list as `xx[0]` and `xx[1:]`.

## Testable skills

Write Python expressions using slicing, negative indexing, etc.

Write Python functions that use list comprehensions, list operations, and various forms of subscripting.

Write recursive Python functions.

Write recursive Python functions that process lists.

**For next time:**

*Study for test. . .*

*Read Sections 3.(1-3) for Friday (once you have Chapter 3. . .)*