Chapter 2 outline:

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

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	Testable skills
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`.	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	Testable skills
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.	Write simple Python functions. Write Python functions that call other functions and that take functions as parameters.
1.4. Denoting sets by listing elements and using set-builder notation. The set type in Python. Python set comprehensions. Python ranges.	Describe a set using set-builder notation. Write Python functions that use set comprehensions and `range`.
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).	Write Python functions that use set operators.

Concepts	Testable skills
1.6. Verifying propositions about the equality of set expressions using Venn diagrams.	Verify set equality propositions using Venn diagrams, shading, labeling, and accompanying verbal explanations.
1.7. Cardinality (modeled by `len` in Python), disjointedness, pairwise disjointedness, partitions, Cartesian products, tuples.	Write Python functions that use tuples.
1.8. Powersets.	Write Python functions that use sets of sets (which requires frozen sets).

Concepts	Testable skills
2.1. Sequences, zero-based indexing. Python lists, subscripting/indexing (with `[]`). List comprehensions. List concatenation and multiplication. Negative indexing and slicing.	Write Python expressions using slicing, negative indexing, etc. Write Python functions that use list comprehensions, list operations, and various forms of subscripting.
2.2. Recurrence relations. Recursive functions. Conditional expressions and statements.	Write recursive Python functions.
2.3. Recursive processing of lists, such as splitting a list as `xx[0]` and `xx[1:]`.	Write recursive Python functions that process lists.
2.4. Python arrays (ndarray) from the NumPy (`np`) package. Multidimensional subscripting/indexing. Arrays as models of various mathematical ideas, including vectors, matrices, and intervals.	Write Python expressions using multidimensional indexing.

For next time:

Study for test...

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