CSCI 243
Test 1 review. Chapters 1 & 2. Test to be held on Wed, Sept 24

Recall that this course uses *standards-based grading.* This means that your progress in the course, as well as your final grade, is measured by how many learning outcome standards you have met. The syllabus lists 21 standards. The first test is comprised of questions to assess you on Standards 1–6. This review sheet summarizes the concepts we have seen so far this semester and how those concepts align with standards that are assessed on this test.

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`.	**Standard 4.** 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.) The test will include a section with approximately 6 expressions to type-analyze.
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. (This is assessed indirectly with Standards 5 and 6.)
1.4. Denoting sets by listing elements and using set-builder notation. The `set` type in Python. Python set comprehensions. Python ranges.	**Standard 2.** Describe a set using set-builder notation. The test will include approximately 2 problems that ask you two denote a set using set-builder notation. **Standard 5.** Write Python functions that use set comprehensions and `range`. (See also below)
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. (This is assessed indirectly with Standards 5 and 6.)
1.6. Verifying propositions about the equality of set expressions using Venn diagrams.	**Standard 1.** Verify set equality propositions using Venn diagrams, shading, labeling, and accompanying verbal explanations. The test will include approximately 2 problems asking you verify a set equality proposition using Venn diagrams
1.7. Cardinality (modeled by `len` in Python), disjointedness, pairwise disjointedness, partitions, Cartesian products, tuples.	Write Python functions that use tuples. (This is assessed along with Standards 5 and 6
1.8. Powersets.	Write Python functions that use sets of sets (which requires frozen sets). (This is assessed along with Standard 5.)
2.1. Sequences, zero-based indexing. Python lists, subscripting/indexing (with `[]`). List comprehensions. List concatenation and multiplication. Negative indexing and slicing.	**Standard 3.** Write Python expressions using slicing, negative indexing, etc. The test will include approximately 4 problems that ask you to write an expression to index into a list or array (see also below). **Standard 5.** Write Python functions that use list comprehensions, list operations, and various forms of subscripting. (See also above.)
2.2. Recurrence relations. Recursive functions. Conditional expressions and statements.	**Standard 6.** Write recursive Python functions. The test will include approximately 2 problems that ask you to write a recursive function. (See also below.)
2.3. Recursive processing of lists, such as splitting a list as `xx[0]` and `xx[1:]`.	**Standard 6.**Write recursive Python functions that process lists. (See also above)
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.	**Standard 3.** Write Python expressions using multidimensional indexing. (See also above.)

(See other side for important tips and clarifications.)

Tips: Make sure you label all expressions in a type analysis, including functions. You may use abbreviations like *i* for `int`, *b* for `bool`, etc, as long as your abbreviation is unambiguous and legible—don't use *f* ambiguously for `func` and `float` or *s* ambiguously for `set` and `str`.

Make sure that Venn diagrams for verifying set propositions are as neat as reasonably possible, with the parts of the expressions clearly labeled with their shading and with verbal explanations to indicate how the entire set appears in the Venn diagram (for example, "only ▨," "double-shaded", "any shade", etc).

Make sure you read the instructions carefully in the programming problems. They specify *what algorithmic approach or Python features* you should use. Many problems could be solved either by a comprehension or recursion (or other ways), but the instructions will indicate which approach you should take. You won't *meet the standard* if you give code that is correct but doesn't conform to what the question is asking for. Likewise, most Python features that we haven't covered are out of bounds.

I intend to be lenient on minor syntactic mistakes in Python, but *lenient* and *minor* are defined at my discretion. To be on the safe side, get the syntax right.

Test content organized by standards:

Standard 1. Verify set equality propositions using Venn diagrams, shading, labeling, and accompanying verbal explanations. *The test will include a section with approximately 2 problems asking you verify a set equality proposition using Venn diagrams.*

Standard 2. Describe a set using set-builder notation. *The test will include a section with approximately 2 problems that ask you two denote a set using set-builder notation.*

Standard 3. Write Python expressions using slicing, negative indexing, and (for arrays) multidimensional indexing. *The test will include a section with approximately 4 problems that ask you to write an expression to index into a list or array.*

Standard 4. Analyze the type of a Python expression. *The test will include a section with approximately 6 expressions for you to type-analyze.*

Standard 5. Write Python functions that use set and list comprehensions. *The test will include a section with approximately 2 problems that ask you to write a function that uses a set or list comprehension.*

Standard 6. Write Python functions that use recursion. *The test will include a section with approximately 2 problems that ask you to write a recursive function.*