CSCE 2100: Foundations of Computing - Spring 2021

Instructor: Paul Tarau, Professor - see my home page for contact info and office hours.

course directory

Grader(s):

contact via Canvas

Midterm exam: Thursday, March 11, open net

Final exam: TBD, cumulative, open net

Description and Objectives:

The course covers basic mathematical concepts and their computational equivalents, including propositional and predicate logic, induction, set theory, functions, and relations. It introduces key concepts of discrete mathematics, probabilities, computability theory and complexity theory, with emphasis on computational instantiations in terms of data structures and executable algorithms.

Syllabus

Intro to Python: sets, sequences, dictionaries, functions and generators

Propositional Logic, truth tables, satisfiability

Predicate logic, quantifiers, equivalent formulas

Peano's axioms for Natural Numbers

Intro to Computational Number Theory

Mathematical induction, big-Oh notation, recursion

Set theory, finite and infinite cardinals, ordinals, cartesian product, power set, bitstring representations

Functions: injections, surjections, and bijections
Relations: equivalence, partial order, total orders, similarities, distances

Elements of Probability Theory and Combinatorics

Trees, Graphs, matrix representations, algorithms

Transitive closure and matrix products

Intro to complexity classes, P, NP, Co-NP, PSPACE, EXP

Finite automata, regular expressions, context-free grammars

Decidable and recognizable languages, self-reference and undecidability, Universal Turing machines, Goedel's Theorem

Prerequisites: CSCE 1030

Evaluation:

- 2 Individual Exams: 30%+30%
- Assignments (teams of up to 3): 40%

Recommended online resources:

Foundations of Computing Textbooks:

- online textbook
- other online textbook

Python3 resources

- Advanced Programming in Python
- Python 3 tutorial
- Functional Python 3
- Python 3 generators and list-comprehensions
- Python 3 itertools library

Outcomes:

1. Demonstrate a solid foundation in conceptual and formal models by describing loop structures in summation and/or product notation.
2. Use abstraction in the design and implementation of algorithms.
3. Design programming solutions to “simple” problems and implement those designs Python.
4. Apply big-Oh notation to evaluate and compare algorithms and programs.
5. Use combinatorics and conditional probability in solving real-world problems.
6. Define the basic operations of sets, functions, relations, trees and graphs.
7. Demonstrate an introductory knowledge of formal languages by using regular expressions, deterministic finite automata and non-deterministic finite automata to describe patterns in strings.