1. **Course number and name**
   CptS 355: Programming Language Design

2. **Credits and contact hours**
   3 credits, 3 lecture hours

3. **Instructor's or course coordinator's name**
   Sakire Arslan Ay

4. **Textbook, title, author, and year**
   Recommended Text:
   R. Harper. 2011. *Programming in Standard ML*, Carnegie Mellon University. <http://www.cs.cmu.edu/~rwh/isml/book.pdf>.
   R Harper. 2012. *Practical Foundations for Programming Languages*, Cambridge University Press. ISBN-10: 1107029570.
   J.C. Mitchell. 2003. *Concepts in Programming Languages*, Cambridge Univ. Press.
   Composing Programs. <http://composingprograms.com/>

5. **Specific course information**
   a. *Catalog description:* Design concepts of high-level programming languages; survey of existing languages, experience using some languages.
   b. *Prerequisites or corequisites:* CPT S 223 or 233, with a C or better; Certified major in CptS, CE, EE, or SE.

6. **Specific goals for the course**
   By the end of the course, students will be able to:
   - Understand components of programming languages including control structures, names, types, objects, exceptions, etc. (6a)
   - Understand different kinds of programming language paradigms such as imperative, functional, and object oriented languages (2a, 2b, 2c, 2d, 2e, 2g).
   - Demonstrate skills in using several programming languages (ML, Python, Java) (2a, 2e, 2g, 6a, 6b).
   - Master specific language concepts such as, scoping, parameter passing, function closures, garbage collection, etc. (2b, 2c, 2d).
   - Develop a basic understanding of programming language implementation, especially insofar as the implementation impacts the design (2b, 2c, 2d).
   - Develop the skills necessary to learn new programming languages quickly (7b, 7c, 7g)

7. **Brief list of topics to be covered**
   - Design influences and evaluation of programming languages
   - Classification of programming languages
   - Language translation: interpretation, compilation
   - Partial functions and computability
   - Functional programming

- ML(variable bindings, functions, tuples, lists, options, pattern matching, tail recursion, high-order functions, data types , recursive types, trees)
- Python (classes, iterators/generators, streams)
- Java (classes, inheritance, memory management) ; Java systems architecture
- Postscript
- Type systems in programming languages (type equivalence, type checking, and type
  inference)
- Scope and scoping
- Parameter passing, function closures
- Garbage collection