1. The paradigm of the language
   1. Object-oriented
   2. Imperative
   3. Functional
   4. Procedural
   5. Reflective
2. Some historical account of the evolution of the language and its antecedents
   1. Founded by Guido can Rossum at Centrum Wiskunde & Informatica (CWI) in Netherlands. Successor to ABC language that would appeal to Unix/C hackers. Conceived in late 1980’s (find exact moment?), implementation began 12/1989. Python 2.0, released 10/16/2000, included many new features (cycle-detecting garbage collector and support for Unicode). Development became more transparent and community-backed. Python 3.0, released 12/3/2008, made changes to many aspects (print statement, libraries, syntax) and added many new ones (bin function, argument handling, chained exceptions).
   2. ABC
   3. ALGOL
   4. C
   5. C++
   6. CLU
   7. Dylan
   8. Haskell
   9. Icon
   10. Java
   11. Lisp
   12. Modula-3
   13. Perl
3. The elements of the language: reserved words, primitive data types, structured types
   1. **Reserved words:**
      * False
      * None
      * True
      * and
      * as
      * assert
      * break
      * class
      * continue
      * def
      * del
      * elif
      * else
      * except
      * finally
      * for
      * from
      * global
      * if
      * import
      * in
      * is
      * lambda
      * nonlocal
      * not
      * or
      * pass
      * raise
      * return
      * try
      * while
      * with
      * yield
   2. **Primitive data types:**

* Boolean
* Numeric: int, long, float, complex
  1. **Structured types:**
* list
* tuple
* dictionary
* set

1. A description (in some form) of the syntax of the language
   1. Indentation
2. The basic control abstractions of the language (loops, conditional controls, etc.)
   1. if/elif/else statements
   2. while/for loops
   3. iterators
   4. list comprehension expressions *(enhanced for loop?)*
   5. break/continue/pass statements
   6. try/except/finally statements
3. How the language handles abstraction (including functions, procedures, objects, modules, etc.)
4. An evaluation of the language's writability, readability, and reliability using the criteria discussed in chapter and of the textbook.
   1. Writability:
      1. Simplicity and Orthogonality (familiarity, utilization of language)
      2. Support for abstraction (hide details, quicker and cleaner)
         1. Process (call a function)
         2. Data (data structure)
      3. Expressivity (efficient convenient, short)
   2. Readability:
      1. Simplicity:
         1. Number of Basic Constructs (learning subsets):
         2. Feature Multiplicity (multiple ways to do one thing): too much is bad
         3. Operator Overloading (symbol has multiple uses):
      2. Orthogonality (small set of primitive constructs can be combined in a small number of ways to build the control and data structures of the language): limited use of orthogonality but still present
      3. Data types: comprehensive and understandable
      4. Syntax design
         1. Special words (reserved)
         2. Form and meaning (obvious and logical)
5. The major strengths and weaknesses of your language.
   1. 5 things better than others (explicit, readability, pretty)
6. An overview of the programs that you included and a discussion of what language features they highlight and how the language made the programs easy/hard to implement.