1 - MacLennan's Principles of Programming Languages

- From the inside front cover of the course text, MacLennan's "Principles of Programming Languages", 3rd edition;
  - ...and discussed, illustrated throughout the text;
- some are contradictory, or are at cross-purposes;
- but all are interesting to keep in mind in designing (or evaluating) programming languages

2 - Abstraction

- "Avoid requiring something to be stated more than once; factor out the recurring pattern."

3 - Automation

- "Automate mechanical, tedious, or error-prone activities."

4 - Defense in Depth

- "Have a series of defenses so that if an error is not caught by one, it will probably be caught by another."

5 - Elegance

- "Confine your attentions to designs that look good because they are good."

6 - Impossible Error

- "Making errors impossible to commit is preferable to detecting them after their commission."

7 - Information Hiding

- "The language should permit modules designed so that:
  - 1. the user has all of the information needed to use the module correctly, and nothing more; and
  - 2. the implementor has all of the information needed to implement the module correctly, and nothing more."

8 - Labeling

- "Avoid arbitrary sequences more than a few items long.
  - Do not require the user to know the absolute position of an item in a list.
  - Instead, associate a meaningful label with each item and allow the items to occur in any order."

9 - Localized Cost

- "Users should pay only for what they use; avoid distributed costs."

10 - Manifest Interface

- "All interfaces should be apparent (manifest) in the syntax."

11 - Orthogonality

- "Independent functions should be controlled by independent mechanisms."

12 - Portability

- "Avoid features or facilities that are dependent on a particular computer or a small class of computers."

13 - Preservation of Information

- "The language should allow the representation of information that the user might know and that the compiler might need."

14 - Regularity

- "Regular rules, without exceptions, are easier to learn, use, describe, and implement."

15 - Responsible Design

- "Do not ask users what they want; find out what they need."

16 - Security

- "No program that violates the definition of the language, or its own intended structure, should escape detection."

17 - Simplicity

- "A language should be as simple as possible.
  - There should be a minimum number of concepts, with simple rules for their combination."

18 - Structure

- "The static structure of a program should correspond in a simple way to the dynamic structure of the corresponding computations."

19 - Syntactic Consistency

- "Similar things should look similar, different things different."

20 - Zero-One-Infinity

- "The only reasonable numbers are zero, one, and infinity."