PRINCIPLES OF PROGRAMMING LANGUAGES

1. **Abstraction:** Avoid requiring something to be stated more than once; factor out the recurring pattern.
2. **Automation:** Automate mechanical, tedious, or error-prone activities.
3. **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.
4. **Elegance:** Confine your attention to designs that *look* good because they *are* good.
5. **Impossible Error:** Making errors impossible to commit is preferable to detecting them after their commission.
6. **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.
7. **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.
8. **Localized Cost:** Users should pay only for what they use; avoid distributed costs.
9. **Manifest Interface:** All interfaces should be apparent (manifest) in the syntax.
10. **Orthogonality:** Independent functions should be controlled by independent mechanisms.
11. **Portability:** Avoid features or facilities that are dependent on a particular computer or a small class of computers.
12. **Preservation of Information:** The language should allow the representation of information that the user might know and that the compiler might need.
13. **Regularity:** Regular rules, without exceptions, are easier to learn, use, describe, and implement.
14. **Responsible Design:** Do not ask users what they want; find out what they need.
15. **Security:** No program that violates the definition of the language, or its own intended structure, should escape detection.
16. **Simplicity:** A language should be as simple as possible. There should be a minimum number of concepts, with simple rules for their combination.
17. **Structure:** The static structure of the program should correspond in a simple way to the dynamic structure of the corresponding computations.
18. **Syntactic Consistency:** Similar things should look similar, different things different.
19. **Zero-One-Infinity:** The only reasonable numbers are zero, one, and infinity.