

Language Evaluation Criteria:

Some of the language criterias to evaluate a programming language are:

1. Readability
2. Writability
3. Reliability
4. Cost
5. Generality
6. Extensibility
7. Standardability
8. Support for internationalization

1. **Readability:** Coding should be simple and clear to understand.

1. **Simplicity:** Should not involve complex syntax, many ways to perform a single task, overloading of methods and operator etc.
2. **Orthogonality:** This means relatively small set of primitive constructs can be combine.
 - For ex., `int *count;` Here pointer and integer is combined.
 - Another ex., `int count[5];` Here array and pointer is combine.
3. **Control Statements:** There should be adequate control statements.
 - Use of `for` loop, `while` loop, `do while` loop is adequate.
 - Using of `go to` statements causes poor readability.
4. **Data Types and Structures:** Language should involve adequate facilities for defining data types and data structure.
 - For ex., `timeout = 1;` is unclear as compare to `timeout = true;`.
5. **Syntax Design:** Syntax design affects the readability in the following way.
 1. *Identifier forms:* Restriction to very short length of identifier is a barrier to readability.
 2. *Special words:* Special words like `while`, `for`, `class`, `int` affects the

readability of any language. If special words are allowed to be variable names than it will become confusing.

2. **Writability:** Writability is a measure of how easily language can be used to code. Most of the language characteristics that affect readability also affect writability.
 1. **Simplicity:** Should not involve complex syntax, many ways to perform a single task, overloading of methods and operator etc.
 2. **Orthogonality:** This means relatively small set of primitive constructs can be combine.
 - For ex., `int *count;` Here pointer and integer is combined.
 - Another ex., `int count[5];` Here array and pointer is combine.
 3. **Support for Abstraction:** Language should support process and data abstraction.
 4. **Expressivity:** In less lines of code program should be writable.
 - For ex., `for` statements makes counting loops easier than `while`.
 - Another ex., `i++` is more expressive than `i=i+1`.
3. **Reliability:** A program is said to be reliable if it performs to its specifications under all conditions.
 1. **Type Checking:** It is testing for type error, either at compile or run time.
 - For ex., `float percentage;` is more desirable as compare to `int percentage`.
 2. **Exception Handling:** It is the ability of program to handle run time error. Remember, handling runtime error are more expensive than compile errors.
 3. **Aliasing:** It is same memory location (variable) having more than one name. Which is causes confusion.
 4. **Readability:** Readability influences reliability.
 5. **Writability:** Writability also influence reliability.
4. **Cost:** Total cost of programming should be minimum.
 - For ex., cost of trainer.
 - Cost of writing algorithm.

- Cost of compiling program in the language.
 - Cost of hardware required for program.
 - Cost of maintenance.
5. **Generality:** Language should not be limited to specific application only.
 6. **Extensibility:** Should be flexible, must be able to add new constructs.
 7. **Standardability:** Language should be platform independent.
 8. **Support for Internationalisation:** Various formats like time, date, currency etc should be supportable.

A lecture video on language evaluation criteria

Click here to view on YouTube.

Previous years solved papers:

- [PPL|RGPV|May 2018](#)
- [PPL|RGPV|June 2017](#)

A list of Video lectures

- [Click here](#)

References:

1. Sebasta, "Concept of programming Language", Pearson Edu
2. Loudon, "Programming Languages: Principles & Practices" , Cengage Learning
3. Tucker, "Programming Languages: Principles and paradigms ", Tata McGraw -Hill.
4. E Horowitz, "Programming Languages", 2nd Edition, Addison Wesley

Related posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants
3. Parse Tree | PPL | Prof. Jayesh Umre
4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria
16. OOP in C++ | PPL
17. OOP in C# | PPL
18. OOP in Java | PPL
19. PPL: Abstraction & Encapsulation
20. PPL: Semaphores
21. PPL: Introduction to 4GL
22. PPL: Variable Initialization
23. PPL: Conditional Statements
24. PPL: Array
25. PPL: Strong Typing

26. PPL: Coroutines
27. PPL: Exception Handler in C++
28. PPL: OOP in PHP
29. PPL: Character Data Type
30. PPL: Exceptions
31. PPL: Heap based storage management
32. PPL: Primitive Data Type
33. PPL: Data types
34. Programming Environments | PPL
35. Virtual Machine | PPL
36. PPL: Local referencing environments
37. Generic Subprograms
38. Local referencing environments | PPL | Prof. Jayesh Umre
39. Generic Subprograms | PPL | Prof. Jayesh Umre
40. PPL: Java Threads
41. PPL: Loops
42. PPL: Exception Handling
43. PPL: C# Threads
44. Pointer & Reference Type | PPL
45. Scope and lifetime of variable
46. Design issues for functions
47. Parameter passing methods
48. Fundamentals of sub-programs
49. Subprograms
50. Design issues of subprogram
51. Garbage Collection
52. Issues in Language Translation

53. PPL Previous years solved papers
54. Type Checking | PPL | Prof. Jayesh Umre
55. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
56. PPL Viva Voce
57. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre
58. Concurrency
59. Basic elements of Prolog
60. Introduction and overview of Logic programming
61. Application of Logic programming
62. PPL: Influences on Language Design
63. PPL: Sequence Control & Expression
64. PPL: Programming Environments
65. PPL: Virtual Machine
66. PPL: Programming Paradigm
67. PPL: Pointer & Reference Type
68. try-catch block in C++