**Lisp**(Common Lisp, Scheme, Clojure)[11](http://dafoster.net/articles/2013/01/29/unique-features-of-various-programming-languages/#fn:11)

* **Homoiconic**
  + When you write a Lisp program, the notation you use (the *grammar*) is equivalent to what a compiler would see (an *abstract syntax tree* or *AST*).
  + Furthermore this Lisp code is represented as a nested structure of lists, symbols, and literals, all of which can be directly generated and manipulated in Lisp itself!
  + This allows Lisp code to generate list structures which can then be run as Lisp code directly.
    - Generation can be done at *compile* time with **macros**.
    - Generation can be done at *runtime* as well, and then invoked with **eval**.
  + However the highly uniform structure of Lisp code, devoid of operator and syntactic diversity, makes for lousy typography and thus low readability.
* **Macros**
  + A function that transforms the AST of its operands at *compile* time to new code.
  + Macros can be used to generate arbitrary new statements and control structures.
    - Domain specific languages, in particular, are very easy to implement in Lisp thanks to macros.
  + Macros can also be used to perform code optimizations at compile time[12](http://dafoster.net/articles/2013/01/29/unique-features-of-various-programming-languages/#fn:12) (similar to “template metaprogramming” in C++).
  + Fluent use of macros requires the host language to be homoiconic, which is rare. Thus Lisp remains the only well-known language family that has macros.
* **Lisp Conditions and Restarts**
  + Allows bidirectional communication between different parts of the call stack. More powerful than exceptions, since conditions can not only unwind the stack but also wind it back again via a restart.
* **Call-with-current-continuation**
  + Allows you to save the current execution state of the program in a variable and jump back to it later. Multiple times, even. It’s like a friggin' time machine. You can implement fairly complex control flow operators with this function.

<http://dafoster.net/articles/2013/01/29/unique-features-of-various-programming-languages/>

Acesso em: 06/04/2018

Lisp characteristics:

* Invented for symbolic computations
* Superficially inspired by mathematical function theory
* Is syntactically and uniformly based on parenthesized prefix notation
  + Parsing a Lisp program is trivial
* Programming goes hand in hand with language development
* It is easy to access and manipulate programs from programs
  + Calls for tool making in Lisp

Scheme characteristics:

* Supports functional programming - but not on an exclusive basis
* Functions are first class data objects
* Uses static binding of free names in procedures and functions
* Types are checked and handled at run time - no static type checking
* Parameters are evaluated before being passed - no lazyness

<http://people.cs.aau.dk/~normark/prog3-03/html/notes/fu-intr-1_themes-lisp-scheme.html>

Acesso em : 06/04/2018

https://arxiv.org/pdf/1008.3431.pdf

@book{Dybvig:1996:SPL:525334,

author = {Dybvig, R. Kent},

title = {The Scheme Programming Language: ANSI Scheme},

year = {1996},

isbn = {0134546466},

edition = {2nd},

publisher = {Prentice Hall PTR},

address = {Upper Saddle River, NJ, USA},

}