Further resources on Programming Languages

Online courses and notes

A lot of the course is based on the following two resources. Following these courses would be a great learning experience:

- Dan Grossman's PL Course[1] has a lot of material available, and occasionally is offered through Coursera. There are videos on the site, assignments and notes. The course covers SML (a language close to OCAML), Racket and Ruby and examines numerous topics along the way.
- Shriram Krishnamurthi's PL Course[2] using a version of Racket and building an interpreter along the way. There are extensive videos, assignments to build an interpreter for a version of Python, and a book[3].

Languages worth looking into

I have ordered the languages in the order that emphasizes the new ideas you will learn by studying these languages.

- Haskell[4] is a lazy functional programming language where all evaluations are delayed until actually needed. It has an immensely expressive type system, and is often a language where new ideas in PLs are implemented.
- Prolog[5] is a great way to learn about *Logic Programming*, a different paradigm. It has strong links to Artificial Intelligence and Linguistics. Or you can try Mercury[6].
- Erlang[7] was designed by the Ericsson telecommunications company to support distributed, fault-tolerant, soft real-time, highly available, non-stop applications. You can literally hot-swap your update to your application without stopping the system. Erlang is a great place to learn about concurrency and fault tolerance.
- Icon[8] is a very interesting high-level language emphasizing "goal-oriented execution".
- Forth[9] is a stack-based imperative programming language that feels quite different from other languages. See also Factor[10].

[1] http://homes.cs.washington.edu/~djg/teachingMaterials/spl/

[2] http://cs.brown.edu/courses/cs173/2012/Syllabus/

[3] http://cs.brown.edu/courses/cs173/2012/book/

[4] https://en.wikipedia.org/wiki/Haskell_(programming_language)

[5] https://en.wikipedia.org/wiki/Prolog

[6] https://en.wikipedia.org/wiki/Mercury_(programming_language)

[7] https://en.wikipedia.org/wiki/Erlang_(programming_language)

[8] https://en.wikipedia.org/wiki/Icon_(programming_language)

[9] https://en.wikipedia.org/wiki/Forth_(programming_language)

[10] http://factorcode.org/

- Idris[11] is a purely functional programming language with dependent types (types that depend on a value), something fairly rare. It is used in interactive theorem proving. Agda[12] and Coq[13] are other languages from the same category.
- Rust[14] is a systems programming language (i.e. serving similar goals to C) with strong functional influences and aiming for type safety.
- Oz[15] is a trully multi-paradigm programming language combining almost every programming paradigm.
- Scala[16] is emphasizing functional programming paradigms combined with objects and a strong type system. It runs on top of the Java Virtual Machine.
- Clojure[17] is a language based on the Lisp/Racket family, also running on top of the JVM.
- F#[18] is a functional programming language derived from the ML family (OCAML) and targeting the .NET platform.
- C#[19] is an object-oriented language with commonalities with C++ and targeting mostly the Windows Platform and .NET.

[11] https://en.wikipedia.org/wiki/Idris_(programming_language)
[12] https://en.wikipedia.org/wiki/Agda_(programming_language)
[13] https://en.wikipedia.org/wiki/Coq
[14] https://www.rust-lang.org/
[15] https://en.wikipedia.org/wiki/Oz_(programming_language)
[16] https://en.wikipedia.org/wiki/Scala_(programming_language)
[17] https://en.wikipedia.org/wiki/Clojure
[18] https://en.wikipedia.org/wiki/F_Sharp_(programming_language)
[19] https://en.wikipedia.org/wiki/C_Sharp_(programming_language)