Rust Wishlist

* Automatic conversion (explicitly defined), not just conversion of references.
* Function overloading!!!
* Accessor functions – putters and getters (methods) without parenthesis.
* Create new methods and traits for outside objects without requiring a wrapper.
* While lifetime is explicit, boxing, pinning are awkward.
* Sometimes all borrows should be explicit. RefCell helps, would like something similar with compile time checks.
* Overloaded “apply” methods on structs so “new” isn’t needed.
* Function calls can use {} for named arguments.
* Default values in function calls.
* Extend “Try” and try-blocks to be general monad comprehensions.
* Encourage it as a functional language with emphasis on unique updates,  
  where unique updates are essentially optimized lenses.
* Allow macros to look like functions and control structures.
* Promoted as a “systems” language with few hidden constructs, Instead, allow the language to grow and do the work. Keep the “systems” aspect in the libraries, not the language.
* Strongly typed error handling is a boilerplate mess unless using macros. Language should allow more natural syntax for “all that stuff”. (Maybe more natural macros?)
* Async and generator functions are nice.
* Futures are too “internal” and their use is somfewhat inconsistent. What does “await” actually require? Read<>, Write<>, Future<>? Is there an Await trait? A Future trait object?
* Get better support for generator functions and stack-less coroutines.
* References which cause compiler errors if held over a yield or await. (may already be so, but how to declare? )
* Would like [derive()] to apply to entire file/crate. Warnings if it is unable to derive.
* Want a trait method which consumes the object. Can’t specify. (dyn dispatch is awkward … maybe trait macro?)
* Question: How to create a trait object from an object which implements the trait?
* Question: How to create a “service” function, generalizing across inputs and output?   
  eg. f1(trait x) -> trait y -- Maybe needs trait objects. Boxed?
* Continuing, want to create such a function pointer and add to a table. My