Assessment RB109

Local Variable Scope

A variable's scope determines where in a program a variable is available for use. A variabl's scope is defined by where the variable is initialized or created. Variable scope is defined by a *block*. A block is a piece of code following a method invocation, usually delmited by either curly braces {} or do/end.

1. Inner scope can access variables initialized in an outer scope, but not vice versa.
2. Outer scope variables can be accessed by inner scope.
3. Inner scope variables cannot be accesed in outer scope.
4. Peer scopes do not conflict
5. Nested blocks create nested scopes
6. Variable shadowing prevents access to the outer scope local variable. It also prevents us from making changes to the outer scoped variable.

The only local variables a method definition has access to must be passed into the method definition. A method definition has no notion of "outer" or "inner" scope – you must explicitly pass in any parameters to a method definition.

A block is *part of* the method invocation. In fact, *method invocation* followed by curly braces or `do..end` is the way in which we *define* a block in Ruby. Essentially the blocks act as an argument to the method.

In the same way that a local variable can be passed as an argument to a method at invocation, when a method is called with a block it acts as an argument to that method.

Method definitions *cannot* directly access local variables initialized outside of the method definition, nor can local variables initialized outside of the method definition be reassigned from within it. A block *can* access local variables initialized outside of the block and can reassign those variables.

Methods can access local variables passed in as arguments, and methods can access local variables through interaction with blocks.

Method definition can be thought of as setting a certain scope for any local variables in terms of the parameters that the method definition has, what it does with those parameters, and also how it interacts (if at all) with a block.

Method invocation can be thought of as using the scopes set by the method definition. If the method definition is defined to use a block, then the scope of the block can provide additinoal flexibility in terms of how the method invocation ineracts with its surroundings.