This is the code being interpreted:

```
fun sayHi(first, last) {
      print "Hi, " + first + " " + last + "!";
}

sayHi("Dear", "Reader");
```

- In Interpreter, walk the collection of statements (syntax tree nodes) and execute each one
  - Stmt.Function is first
    - Run its accept()
    - That calls visitFunctionStmt(Stmt.Function instance) in Interpreter
    - That wraps the Stmt.Function with a LoxFunction – so it has call mechanics
    - That binds, in a new local environment, the function name to the LoxFunction instance **HERE WE ARE REMEMBERING THE DEFINITION OF THE FUNCTION, BINDING THE FUNCTION OBJECT (WITH PARAMETERS AND BODY) TO A NAME IN THE ENVIRONMENT FOR LATER RETRIEVAL**
  - Stmt.Expression … “Expression Statement” is next
    - It has an instance of Expr.Call
    - Call Stmt.Expression’s accept()
    - That calls Visitor’s (Interpreter’s) visitExpressionStmt(Stmt.Expression)
    - vES() calls evaluate() on the Stmt.Expression.expression … expression is an Expr.Call syntax tree node object
    - calls accept() on the Expr.Call instance
    - that calls visitCallExpr() on Visitor …. Interpreter, a reference to the Expr.Call is in scope within that function
    - evaluates the callee of the Expr.Call instance A) which assigns to Object call.
    - calls accept() on the callee, which is a variable, the function name,

- which calls visitVariableExpr(), which gets whatever is bound to the Expr.Variable.name in the environment, which per above will be the LoxFunction instance. Note that LoxFunction implements the LoxCallable interface, for use below.
- So, back to/in visitCallExpr(), callee is assigned the LoxFunction object retrieved from the environment
- Walk the arguments in Expr.Call instance
  - evaluate each argument and add it to a list of arguments
  - one argument is the literal “Dear”. evaluate() returns the string “Dear”
- the arguments list winds up containing “Dear”, “Reader”
- Cast the Object callee to a LoxCallable, see A) above. This can be done because callee is an instance of LoxFunction, which implements the LoxCallable interface. So the case to LoxCallable works and gives access to the call() method. **WE ARE PROCESSING A STMT.CALL BUT WE HAVE GRABBED CALLEE – CALLEE IS THE FUNCTION OBJECT THAT HAS PARAMETERS AND BODY NEEDED TO ALLOW THE FUNCTION TO BE CALLED.**
- Call the LoxCallable’s .. also a LoxFunction …. call() method
- call() creates a new environment//scope for the function call and remembers the old, parent environment
- walks the function object’s (LoxCallable) parameters and and the function call object’s arguments together, binding argument to parameter name in the new local environment
- executeBlock() on the function object’s body
- executes every statement in the block – which means print the string with the arguments inserted in the right places.
- resets the environment to the remembered “outer” environment since the block returned and it’s local environment went out of scope
-