# Refactoring Steps and Evaluation

1. Making tests more effective.

Sourcecode tested = Solution2/DataInterpreter/di

Look at TestCoverageResults.jpg for a screenshot - 90% coverage

## Bad Smell 1 - Long Method

1. Push to GitHub
2. Replace method with method object - DataInterpreter.\_\_validated() code removed to separate validate.Validator class
3. Run Tests, tests pass
4. Push to GitHub
5. extract method - remove code to wash data from validated function and put in wash function
6. Run Tests, tests pass
7. Push to GitHub
8. Made wash code more easy to read by creating a wash\_field function.
9. Run Tests, tests pass
10. Push to GitHub
11. ExtractMethod making validate easier to read - creating validate\_id(some\_id)... and so on functions
12. Run Tests, tests pass
13. Push to GitHub
14. Removed duplicated code by putting in a validate\_field method
15. Run Tests, tests pass
16. Push to GitHub
17. I've made the validated function smaller. Removed duplicated function call (function was being called twice - once to check if it was valid, and again to get the validated list).
18. Run Tests, tests pass
19. Push to GitHub

At # 17. Identified new, smaller bad smell injected (uncovered?) – Primitive Obsession - (relied on indexes to find values) The code was not easy to read.

1. Added a FieldValidator object, and loop over multiple lists using the zip function.
2. Run Tests, tests pass
3. Push to GitHub

At # 20 possibly added a Lazy Class, but it is doing One Thing well (good separation of concerns)

### Evaluation:

* Has the bad smells successfully been removed at the end?
  + Yes. Function is now only doing one job - delegates job to Validator class.
  + Function is smaller, easier to read
* How well it is?
  + Validator code is beautiful and elegant and Pythonic
* Do you bring new bad smells into the program
  + Possible Lazy Class – Field Validator, BUT it is doing One Thing well (good separation of concerns)

## Bad Smell 2 - Primitive Obsession

\*Already had test coverage, so didn’t need to write additional tests.

\*Strategy is to replace array with object

1. Create a Record class. Wrap data array in record object
2. Run Tests, tests pass
3. Push to GitHub
4. Replace array with ordered dict
5. Run Tests, tests pass
6. Push to GitHub

At # 4 – creating properties for the Record would have added a switch statement/ if/elif bad smell in the get data code.

1. Removed RECORD\_COLUMNS array from di
2. Run Tests, tests pass
3. Push to GitHub
4. Rewrite DataInterpreter get\_valid\_data(data\_name) and get\_all\_valid\_records()to use list Comprehensions. Smaller, more readable code
5. Run Tests, tests pass
6. Push to GitHub

\*Next – Move Field so that the rules for validating data are stored in the Record Class and given to the Validator (Single Responsibility Principle, Separation of Concerns)

1. Removed duplicate data - input order field in FieldValidator (relies on data input in the right order, storing this number here will cause duplicated data)
2. Run Tests, tests pass
3. Push to GitHub
4. Add Field- RULES - private class member of record. Now validate does the checking, based on whatever rules it is given by DataInterpreter class, who gets rules from Record.
5. Run Tests, tests pass
6. Push to GitHub

# Potentially added another really small class/ Lazy Class, but really improves separation of concerns

* DataInterpreter class is now only acting as coordinator, doesn’t need to know about rules or validation
* Validator only does validation
* Record knows about rules and fields

### Evaluation:

* Has the bad smells successfully been removed at the end?
  + Yes. Data Array replaced with object.
  + Class arrays used to get indexes by name are gone
* How well it is?
  + Separation of concerns, Single responsibility principle
  + Record Knows fields, rules
  + Validator can validate based on rules given to it by DataInterpreter, who gets the rules from Record.
  + Data Interpreter is just a coordinator, doesn’t need to know rules
* Do you bring new bad smells into the program
  + Possible Lazy Class – Record? But I like the encapsulation of the rules away from the DataInterpreter
  + Possibly this is over refactored? In real life , I don’t think anyone does it so much/ well….

## Bad Smell 3 - Long Method \_\_add\_data()

1. Extract method - \_\_add\_valid\_record(valid\_data) to makecode more readable
2. Run Tests, tests pass
3. Push to GitHub
4. Calculate count\_invalid from length of invalid data ids
5. Run Tests, tests pass
6. Push to GitHub
7. Extract method - invalid\_data\_mesg() returns string message for invalid data
8. Run Tests, tests pass
9. Push to GitHub
10. Changed local variable from an array to a string - readability
11. Run Tests, tests pass
12. Push to GitHub

# Code is much more readable, helper functions doing individual jobs

### Evaluation:

* Has the bad smells successfully been removed at the end?
  + Yes. Function is now much more readable
* How well it is?
  + Good
* Do you bring new bad smells into the program
  + No