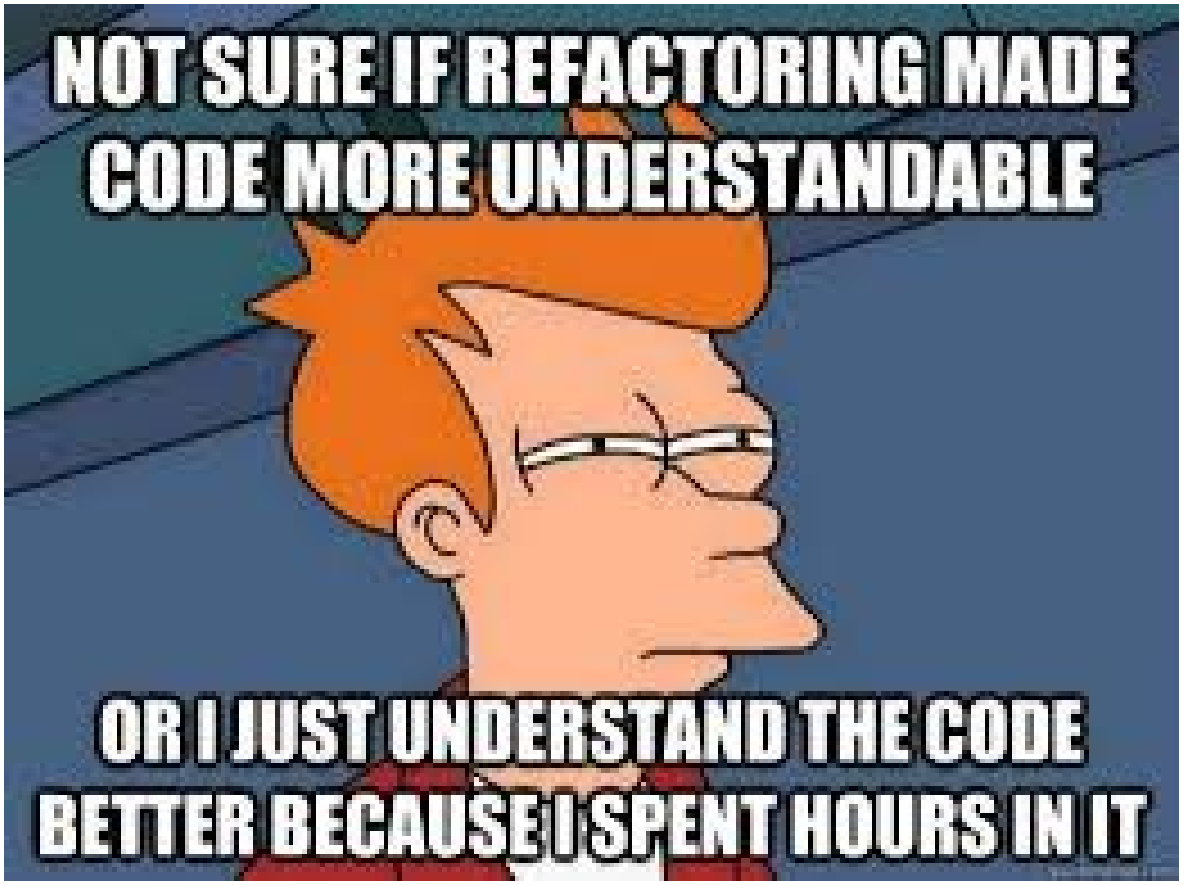
Refactoring patterns



Why talk about this?

- Creates clear concepts in a muddy space
- Gives us a common language to discuss how to change code architecture
- They are patterns, just like programming patterns

What is refactoring?

- Taking code from known state to known state
- Changing the *structure* of the code without changing functionality
- How do you know the state?
  - Test Cases
  - Good coverage

Challenges in Python

- Python is a very dynamic language
- MUST have test coverage to refactor well
- Metaprogramming makes things easier if you've designed clear semantics
- Metaprogramming makes things harder if you've focused on implementation

Refactoring Tools

- https://github.com/python-rope/rope/blob/master/docs/rope.rst
- bicyclerepair.sourceforge.net
- https://www.jetbrains.com/pycharm/
- sublimerope.readthedocs.org > En > Latest > Refactoring

The simple ones

- Rename class
- Rename variable
- Rename module
- Extract constant

More complex

- Extract method
- Inline method
- change property -> method
- change method -> property
- change method signature

Inheritance/Classes

- Insert class
- Extract superclass
- Inline Class

Inheritance/Methods & Properties

- Pull up
- Push down

A few antipatterns in code

- Fat Object
- "isinstance"