SOLID Principles

-SOLID:

is a set of object oriented design principles aimed at making code more maintainable and flexible. They were coined by Robert "Uncle Bob" Martin in the year 2000 in his paper *Design Principles and Design Patterns*. The SOLID principles apply to any object oriented language, but I'm going to concentrate on what they mean in a Python application in this post.

-SOLID is an acronym that stands for the following:

- **S**ingle responsibility principle
- **O**pen/closed principle
- **L**iskov substitution principle
- **I**nterface segregation principle
- **D**ependency inversion principle

-Single Responsibility Principle

This states that a class should have a single responsibility, but more than that, a class should only have one reason to change.

-Open/Closed Principle

In the open/closed principle classes should be open for extension, but closed for modification**.** Essentially meaning that classes should be extended to change functionality, rather than being altered into something else.

-Liskov Substitution Principle

Created by Barbara Liskov in a 1987, this states that objects should be replaceable by their subtypes without altering how the program works. In other words, derived classes must be substitutable for their base classes without causing errors.

-Interface Segregation Principle

This states that many client-specific interfaces are better than one general-purpose interface**.** In other words, classes should not be forced to implement interfaces they do not use.

-Dependency Inversion Principle

Perhaps the simplest of the principles, this states that classes should depend upon abstractions, not concretions**.** Essentially, don't depend on concrete classes, depend upon interfaces.