Python Design Patterns Playbook

Introduction



Gerald BrittonIT Specialist

@GeraldBritton www.linkedin.com/in/geraldbritton

Overview



What are design patterns?

Why do we need them?

Classification of design patterns

Principles of object-oriented design

SOLID

Tools you will need

Defining interfaces in Python

Design Pattern

A design pattern is a model solution to a common design problem. It describes the problem and a general approach to solving it.

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in a way that you can use this solution a million times over, without ever doing it the same way twice."

Christopher Alexander (1977), The Timeless Way of Building, Oxford University Press



Examples of Design Patterns

Building architecture

Electrical and plumbing codes

Automobile design

Mobile phone interfaces



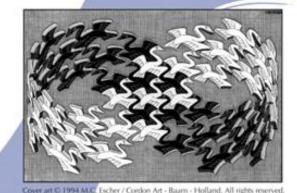
We need design patterns to ensure that our work is consistent, reliable, and understandable.





Elements of Reusable Object-Oriented Software

Erich Gamma Richard Helm Ralph Johnson John Vlissides



Foreword by Grady Booch



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

First published in 1995

"Gang of Four"

Gamma, Helm, Johnson, and Vlissides

First comprehensive work on the topic

Remains the authoritative reference

This course would look very different without this book



Classification

Creational
Object creation

Structural

Object composition

Behavioral

Object interaction and responsibility



SOLID Principles of Object-oriented Design

Single responsibility

Open-closed

Liskov substitution

Interface segregation

Dependency inversion



Tools You Will Need

Python language, 3.x series

https://www.python.org/downloads/

Visual Studio Code

https://code.visualstudio.com/download

Python extension (ms-python.python)

- Install within VS Code

Interfaces in Python

The "I" in SOLID

Supported in Java, C#, Visual Basic with Interface definitions

Supported in C++ with Abstract Classes

Introduced by PEP 3119

Previously no provision in Python

First appeared in Python versions 2.6 and 3.0



Abstract Base Class Definition

```
abc
                  import abc
 module
                  class MyABC(abc.ABC):
Make class
                        "Abstract Base Class definition"""
abstract
                      @abc.abstractmethod
Abstract
                      def do_something(self, value):
 method
                          """Required method"""
                      @abc.abstractproperty
Abstract
                      def some_property(self):
property
                          """Required property"""
```

Concrete Class Implementation

```
Inherit from
   ABC
 Standard
constructor
Implement
 abstract
  method
Implement
 abstract
 property
```

```
class MyClass(MyABC):
     ""Implementation of abstract base class"""
    def __init__(self, value=None):
        self._myprop = value
    def do_something(self, value):
        """Implementation of abstract method"
        self._myprop *= value
    @property
    def some_property(self):
        """Implementation of abstract property"
        return self._myprop
```

Python Catches Missing Implementations

```
>>> class BadClass(MyABC):
... pass
...
>>> bad = BadClass()
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
TypeError: Can't instantiate abstract class BadClass with abstract methods do_something, some_property
>>> _
```

Summary



What design patterns are

Why we need them

Object-oriented design principles (SOLID)

Tools you will need

Interfaces in Python

"Gentlemen's agreement"