PYTHON APPLICATION PROGRAMMING

Chitra G M
Computer Science and Engineering

PYTHON APPLICATION PROGRAMMING

Introduction

Chitra G M

Department of Computer Science and Engineering

What is a design pattern

- A design pattern in architecture and computer science is a formal way of documenting a solution to a design problem in a particular field of expertise.

- The idea was introduced by the architect Christopher Alexander in the field of architecture and has been adapted for various other disciplines, including computer science.

Design pattern

An organized collection of design patterns that relate to a particular field is called a pattern language.

(wiki definition)

Design Pattern Classifications

- **Creational Patterns** - They describe how best an object can be created.
- A simple example of such design pattern is a singleton class where only a single instance of a class can be created. This design pattern can be used in situations when we cannot have more than one instances of logger logging application messages in a file.

Design Pattern Classifications

- **Structural Patterns** – They describe how objects and classes can work together to achieve larger results.
- A classic example of this would be the façade pattern where a class acts as a façade in front of complex classes and objects to present a simple interface to the client.

Design Pattern Classifications

- **Behavioral Patterns** – They talk about interaction between objects. Mediator design pattern, where an object of mediator class, mediates the interaction of objects of different classes to get the desired process working, is a classical example of behavioral pattern

Singleton class

- Singleton pattern restricts the instantiation of a class to one object.

- It is a type of creational pattern and involves only one class to create methods and specified objects.

- It provides a global point of access to the instance created.

Singleton class Example

```
class single:
        instance=None
        def getinstance():
                if single.instance==None:
                        single()
                return single.instance
        def __init__(self):
                if single.instance!=None:
                        raise Exception("hey this is singleton")
                else:
                        single.instance=self

s1=single()
print(s1)
```

THANK YOU

Chitra G M
Department of Computer Science and Engineering

chitragm@pes.edu
+91 9900300411