Introduction to Object-Oriented Programming in Python

Agenda

- Introduction to Object-Oriented Programming
- Key Terms
  - LLD
  - Procedural Programming
  - Object-Oriented Programming
  - Abstraction
  - Encapsulation
  - Classes
  - Object
- Why LLD
- What is Good Software?
  - Maintainability
  - Scalability
  - Extensibility
  - Extensibility vs Reusability
- Programming Paradigms
  - Procedural Programming
  - Object-Oriented Programming
- Reading List

Key Terms

LLD

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code, and ultimately, performance algorithms.

Procedural Programming

Procedural programming is a programming paradigm that uses a sequence of steps to solve a problem.

Object-Oriented Programming

Object-oriented programming (OOP) is a programming paradigm that uses objects to model real-world things and aims to implement state and behavior using objects.

Abstraction

Abstraction is the process of hiding the implementation details of a program from the user.

Encapsulation

Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized access.

Classes

A class in Python is a blueprint for creating objects.

Object

An object is an instance of a class.

Why LLD

LLD provides the internal logical design of the actual program code, which is crucial for maintainability, scalability, and extensibility.

What is Good Software?

Good software is maintainable, scalable, and extensible.

Maintainability

Maintainability is the ease with which a software system can evolve to meet new requirements or to correct deficiencies.

Scalability

Scalability is the capability of a system to handle a growing amount of work by adding resources to the system.

Extensibility

Extensibility is the ability to add new capabilities or functionality to a system without disrupting existing functionality.

Programming Paradigms

Procedural Programming

Procedural programming is based on the concept of calling procedures, functions, or routines.

Object-Oriented Programming

Object-oriented programming is based on the concept of "objects", which can contain data and code to manipulate the data.

Object-Oriented Programming in Python

In Python, classes and objects are used to implement OOP concepts.

```
class BankAccount:
    def __init__(self, number, balance):
        self._number = number
        self._balance = balance

    def deposit(self, amount):
        self._balance += amount

    def withdraw(self, amount):
        self._balance -= amount

    def transfer(self, destination, amount):
        self.withdraw(amount)
        destination.deposit(amount)
```

Abstraction in Python

Abstraction in Python can be achieved through the use of abstract classes and methods.

```
from abc import ABC, abstractmethod

class AbstractBankAccount(ABC):
    @abstractmethod
    def deposit(self, amount):
        pass

    @abstractmethod
    def withdraw(self, amount):
        pass
```

Encapsulation in Python

Encapsulation in Python

Encapsulation in Python is about hiding the internal state and requiring all interaction to be performed through an object's methods.

```
class EncapsulatedBankAccount:
    def __init__(self, number, balance):
        self._number = number
        self._balance = balance

    def deposit(self, amount):
        if amount > 0:
            self._balance += amount

    def withdraw(self, amount):
        if amount > 0 and amount <= self._balance:
            self._balance -= amount

    def get_balance(self):
        return self._balance
```