أكاديمية كاوست
KAUST ACADEMY

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

# OOP in Python Workshop

Day 2: February 10

# Agenda

- Morning Session (9:00 – 12:00)
  - OOP Concepts
  - Classes and Objects
  - Inheritance
  - Encapsulation
  - Advanced OOP:
    - Polymorphism
    - Abstract Classes
    - Design Patterns
- Lunch Break (12:00 – 13:30)

# Agenda

- Afternoon Session (13:30 – 15:30)
  - Practical OOP Exercises
  - Building Class Hierarchies
  - Implementing Interfaces
  - Real-World Applications

# OOP Concepts

- What is OOP?
  - A programming paradigm centered around objects that bundle data and behavior.
- Benefits:
  - Reusability, scalability, and improved code maintenance.
- Core Principles:
  - Encapsulation, Inheritance, Abstraction, and Polymorphism.

# Classes and Objects

- Classes:
  - Blueprints for creating objects.
  - Define attributes (data) and methods (functions).
- Objects:
  - Instances of classes.

# Inheritance

- Definition:
  - A mechanism where a new class (child) inherits attributes and methods from an existing class (parent).
- Benefits:
  - Code reuse and method overriding.

# Encapsulation

- Definition:
  - Bundling data and methods within a single unit (class) and restricting access to some of the object's components.
- Why Encapsulate?
  - To protect the internal state of an object.
- Techniques:
  - Using private attributes (e.g., prefixing with _ or __).

# Advanced OOP – Polymorphism

- Definition:
  - The ability for different classes to be treated as instances of the same class through a common interface.
- How It Works:
  - Method overriding and duck typing in Python.

# Advanced OOP – Abstract Classes

- Purpose:
  - To provide a blueprint for other classes and enforce method implementation in subclasses.
- Using the abc Module:
  - Helps create abstract base classes in Python.

# Advanced OOP – Design Patterns

- What Are Design Patterns?
  - Reusable solutions to common software design problems.
- Common Patterns:
  - Factory Pattern: For creating objects.
  - Singleton Pattern: Ensuring a class has only one instance.
  - Observer Pattern: Defining a subscription mechanism for notifying multiple objects.
- Benefits:
  - Enhanced code organization, flexibility, and scalability.

# Break

- Diner is in building 13
- Come back by 13:30(1:30PM)

# Building Class Hierarchies

- Structuring classes to represent “is-a” relationships.
- Differentiating base classes from derived classes.
- Overriding methods for specialized behavior.
- Example a worker works on a task, a manager manages team but both are employees

# Implementing Interfaces

- Defining a common set of methods that different classes must implement.
- Use abstract base classes or duck typing.
  - If it moves like a duck, quacks like a duck, swims lick a duck, it is probably a duck 🦆🦆🦆🦆🦆🦆

# Real world applications of OOP

- English to Arabic Translator
  - By Ali ☺
- Coding Patterns in python
- Literally any decent package in python has OOP:
  - https://github.com/bokeh/bokeh
  - https://github.com/pytorch/pytorch/tree/main
  - https://github.com/numpy/numpy
  - https://github.com/numpy/numpy/tree/main