OOP: Inheritance

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**

- No question is daft or silly - **ask them!**

- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.

- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query:

  www.hyperiondev.com/support

- Report a **safeguarding** incident:

  www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: **Feedback on Lectures**
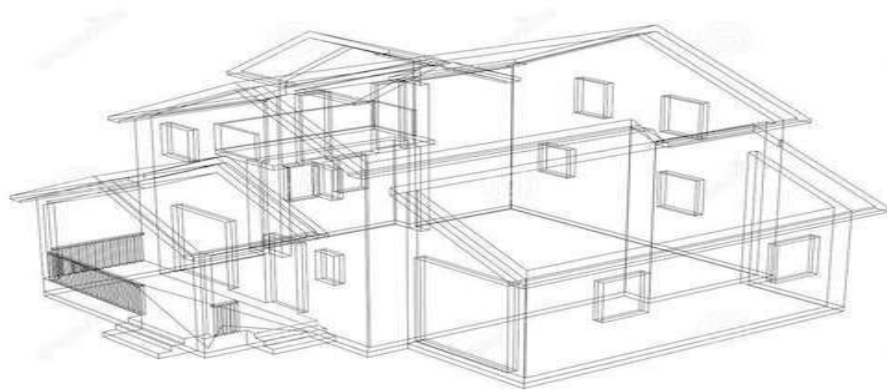
CoGrammar

Lecture Objectives

1. Define inheritance and its role in OOP.

2. Implement inheritance into your own classes.

CoGrammar

Recap on Previous Week

Classes in OOP

A class is a blueprint or template for creating objects. It defines the attributes and methods that all objects of that class will have.
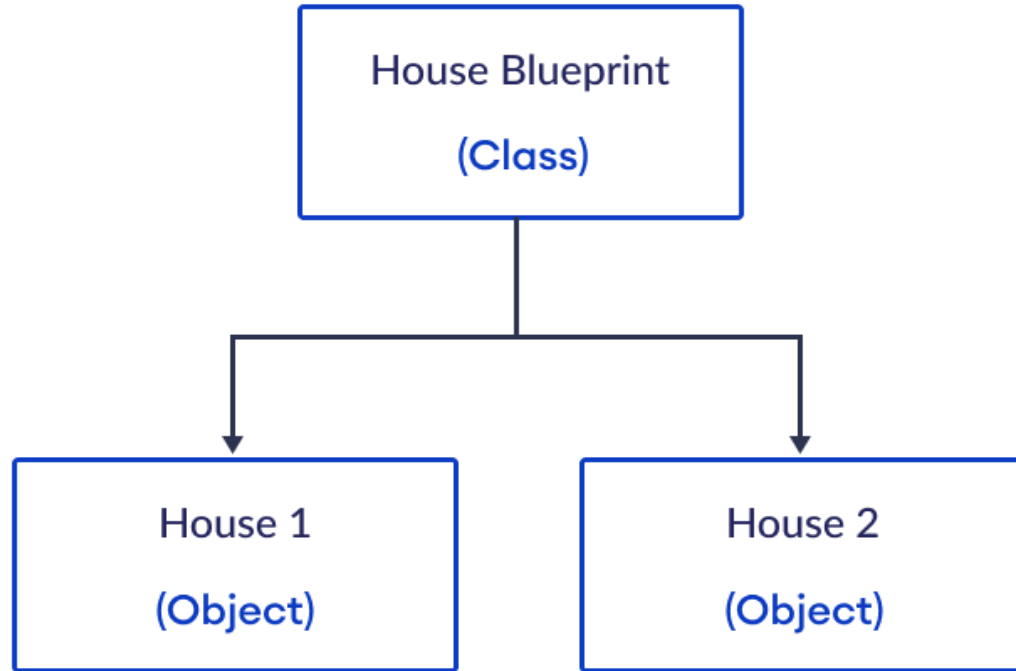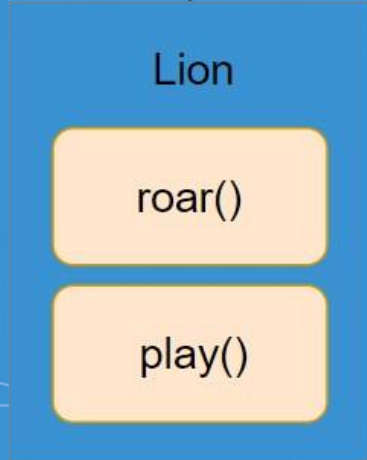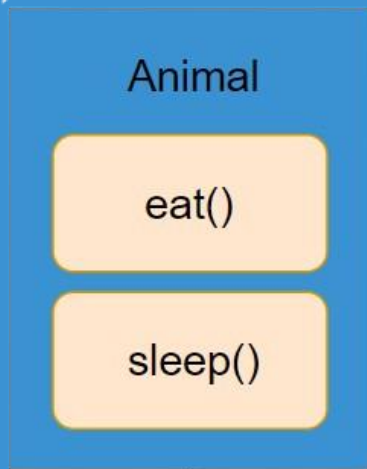
Attributes

- **Attributes are values that define the characteristics associated with an object.**

- **They define the state of an object and provide information about its current condition.**

- **For a class named 'House', some relevant attributes could be:**

  - **Number of bedrooms**

  - **Year built**

Methods (Behaviours)

- **Methods, also known as behaviours, define the actions or behaviours that objects can perform.**

- **They encapsulate the functionality of objects and allow them to interact with each other and the outside world.**

- **For a class named 'House', some relevant method could be:**
  - **set_location(): Allows updating the location of the house**
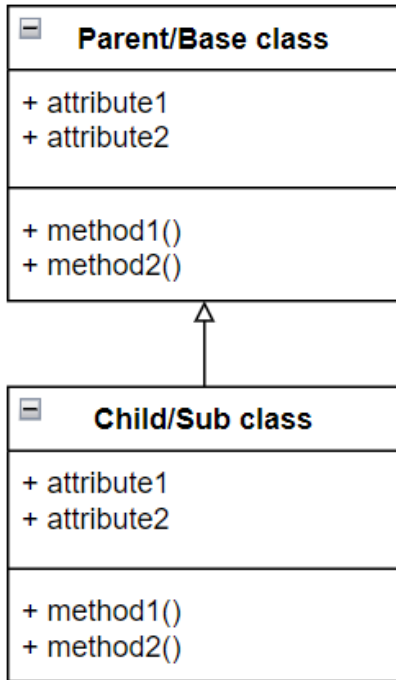
What is Inheritance?

- Sometimes we require a class with the same attributes and properties as another class but we want to extend some of the behaviour or add more attributes.

- Using inheritance we can create a new class with all the properties and attributes of a base class instead of having to redefine them.

What is Inheritance?

Inheritance

- Parent/Base class
  - The parent or base class contains all the attributes and properties we want to inherit.
- Child/Subclass
  - The sub class will inherit all of its attributes and properties from the parent class.

```python
class BaseClass:
    # Base class definition


class SubClass(BaseClass):
    # Derived class definition
```

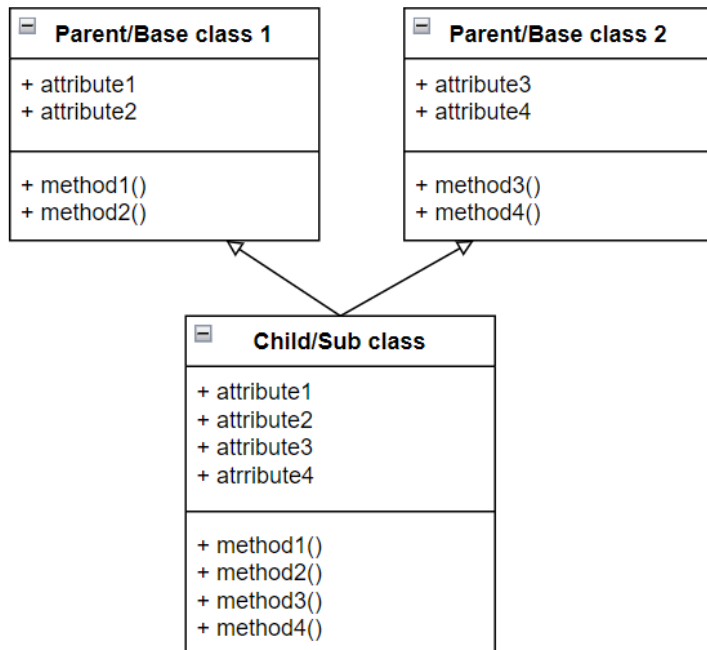# Multiple Inheritance

- Python allows multiple inheritance as well.
- This means we can have a subclass that inherits attributes and properties from more than one base class.

```python
class BaseClass:
    # Base class definition
    pass


class BaseClassA:
    # Base class definition
    pass


class SubClass(BaseClass, BaseClassA):
    # Subclass definition
    pass
```

Multiple Inheritance

Method Overriding

- We can override methods in our subclass to either extend or change the behaviour of a method.

- To apply method overriding you simply need to define a method with the same name as the method you would like to override.

- To extend functionality of a method instead of completely overriding we can use the super() function.

Super()

- The super() function allows us to access the attributes and properties of our Parent/Base class.

- Using super() followed by a dot "." we can call to the methods that reside inside our base class.

- When extending functionality of a method we would first want to call the base class method and then add the extended behaviour.

Methods overriding and Super()

- The method you will override the most will be __init__()

- When adding more instance attributes to the subclass we have to call to the base class __init__() to avoid having to redeclare all our base class attributes.

- We can use super().__init__() to call the constructor of the base class and set the values of the inherited attributes.

Methods overriding and Super()

Here we call __init__() from the Person class to set the values for the attributes "name" and "age".

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age


class Student(Person):
    def __init__(self, name, age):
        super().__init__(name, age)
        self.grades = []
```

Methods overriding and Super()

```python
class BaseClass:
    # Base class definition
    def print_name(self):
        print(self.name)


class SubClass(BaseClass):
    # Subclass definition
    def print_name(self):
        print("Code before base method call.")
        super().print_name()
        print("Code after base method call.")
```

isinstance() and issubclass()

- We can determine if an object is an instance of a particular class using isinstance()
  - E.g. isinstance(object, ClassType)

- We can determine if a class is a subclass of another class using issubclass()
  - E.g. issubclass(SubClass, BaseClass)

isinstance()

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age


person = Person("Peter", "Parker")
print(isinstance(person, Person)) # True
```

issubclass()

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age


class Student(Person):
    def __init__(self, name, age):
        super().__init__(name, age)
        self.grades = []


print(issubclass(Student, Person)) # True
```

Progression Criteria

✅ **Criterion 1: Initial Requirements**

- Complete 15 hours of Guided Learning Hours and the first four tasks within two weeks.

✅ **Criterion 2: Mid-Course Progress**

- Software Engineering: Finish 14 tasks by week 8.
- Data Science: Finish 13 tasks by week 8.

✅ **Criterion 3: Post-Course Progress**

- Complete all mandatory tasks by 24th March 2024.
- Record an Invitation to Interview within 4 weeks of course completion, or by 30th March 2024.
- Achieve 112 GLH by 24th March 2024.

✅ **Criterion 4: Employability**

- Record a Final Job Outcome within 12 weeks of graduation, or by 23rd September 2024.

Unlock Prestigious Co-Certification Opportunities

New Partnerships Unveiled!
- **University of Manchester & Imperial College London** join our circle along with The University of Nottingham Online.

Exclusive Opportunity:
- Co-certification spots awarded on a first-come basis.
- Meet the criteria early to gain eligibility for the co-certification.

Key Deadlines:
- **11 March 2024**: 112 Guided Learning Hours & 'Build Your Brand' tasks completion.
- **18 March 2024**: Record interview invitation or self-employment.
- **15 July 2024**: Submit verified job offer or new contract.

CoGrammar

CoGrammar

Questions around the Inheritance