Software Engineering Bootcamp

Hyperiondev

Inheritance

Lecture – Housekeeping

- ❑ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- ❑ 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.
- ❑ You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- ❑ For all non-academic questions, please submit a query: www.hyperiondev.com/support
- ❑ Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- ❑ We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Objectives

1. Understand inheritance
2. Learn syntax for inheritance
3. Let's do some live coding

Github Repository - Lecture Examples

https://github.com/HyperionDevBootcamps/C4_SE_lecture_examples

Recap on Classes and Objects

Creating a Class

- __init__ function is called when class is instantiated.

```
class House():
    city = “London”
    garden = False
    def __init__(self, address, type, bedrooms, bathrooms, rent):
        self.address = address
        self.type = type
        self.bedrooms = bedrooms
        self.bathrooms = bathrooms
        self.rent = rent
```

Creating an object – Class Instantiation

- Objects are basically initialised versions of your blueprint
- They each have the properties you have defined in your constructor.

```
house1 = House("123 4t ave", "cluster", 2, 1, 1000)
```

- Class takes in three values: a address, type, bedrooms, bathrooms and rent.

What is Inheritance?

- Inheritance is the ability to define a new class that is a modified version of the existing class.
- The primary advantage of this feature is that you can add new methods to a class without modifying the existing class.
- It is called inheritance because the new class inherits all of the methods of the existing class.
- The existing class is the parent class or base class.
- The new class may be called the child class or subclass

Why Inheritance?

- Inheritance is a powerful feature

- Some programs that would be complicated without inheritance can be written concisely and simply with it.

- Also, inheritance can facilitate code reuse, since you can customise the behavior of the parent classes without having to modify them.

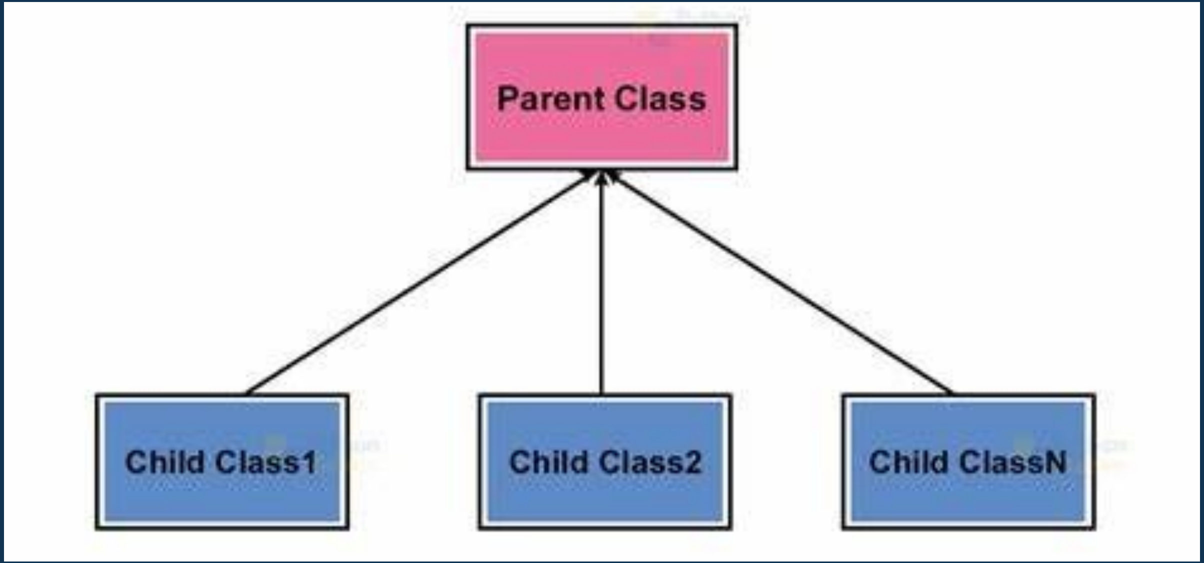
Apples and Fruit

- Is an apple a fruit? Yes.
- Is a fruit an apple? No.
- Let’s think of them as two classes - Fruit and Apple. Let’s also consider other classes like Banana, Mango and Kiwi.
- The Apple, Banana, Mango and Kiwi classes all share similar attributes.
- These attributes can be defined in the Fruit class. Apple, Banana, etc. then all inherit from that class.

Parents and Children

In our Apples and Fruit example, there are some points to note. The Fruit class is considered the **parent** class, and the Apples class is considered the **child** class.

Parents and Children



The super() Function

- To access an attribute in the current class, you can use self.
- However, if you need to access an attribute in the parent class, you can use super().

Example of super() Function

```
# Define parent class
class Computer():
    def __init__(self, computer, ram, ssd):
        self.computer = computer
        self.ram = ram
        self.ssd = ssd

# Define subclass
class Laptop(Computer):
    def __init__(self, computer, ram, ssd, model):
        super().__init__(computer, ram, ssd)
        self.model = model
```

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any

Hyperiondev

Thank you for joining us