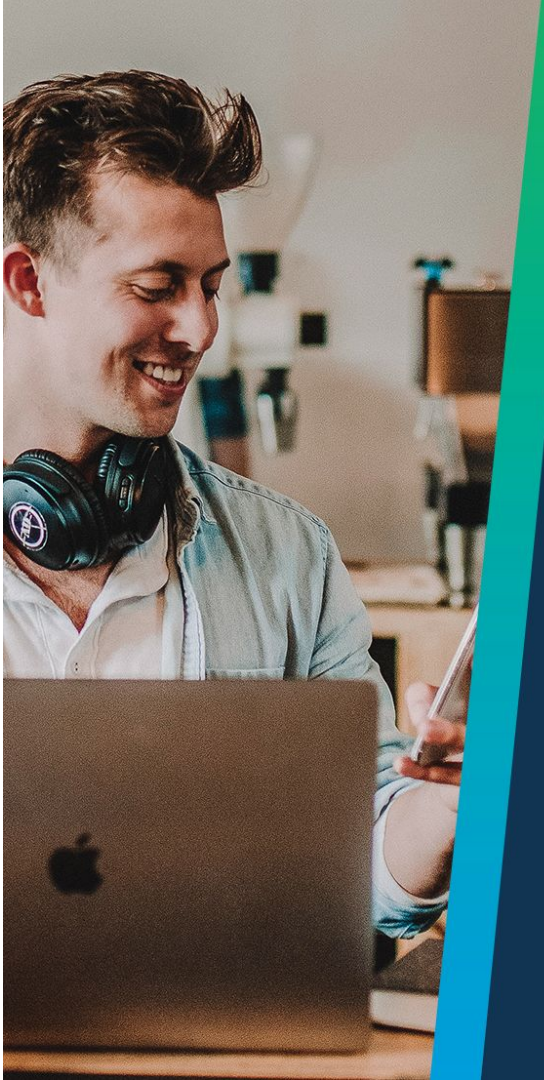


Software Engineering Bootcamp

Hyperiondev

Recap on Object-Oriented Programming

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. Recap on object-oriented programming
    a. What is OOP?
    b. OOP Components
    c. Creating classes & object
2. Recap on inheritance
    a. What is inheritance?
    b. Parent & child classes
3. Multiple inheritance
    a. Examples

Recap on Object-Oriented Programming

What is Object-Oriented Programming?

- A form of programming that models real-world interactions of physical objects.
- Relies on **classes** and **objects** over functions and logic.
- Powerful tool for abstraction.

OOP Components

- **Class**
  - Different to an object.
  - Think of an object as a house - the class is the blueprint.
- **Properties**
  - Data contained in classes.
  - For example, a student has a name, grade, ID, etc. These are properties of a student.
  - Comes in the form of variables that you can access (e.g. student.name).

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.

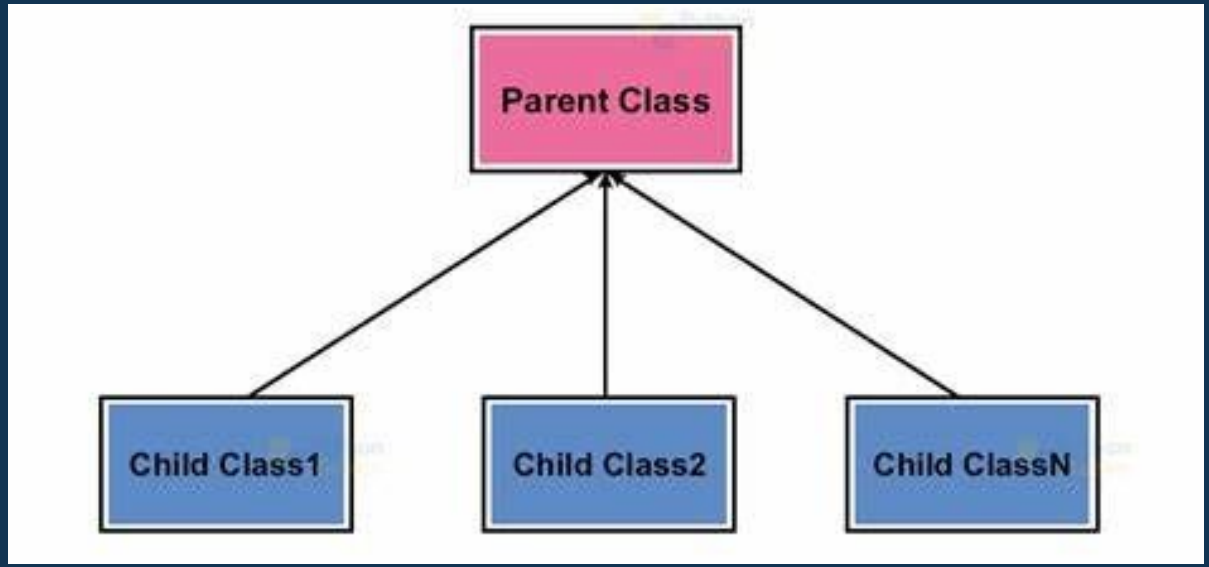
```
house1 = House("123 4t ave", "cluster", 2, 1, 1000)
```

- 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

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

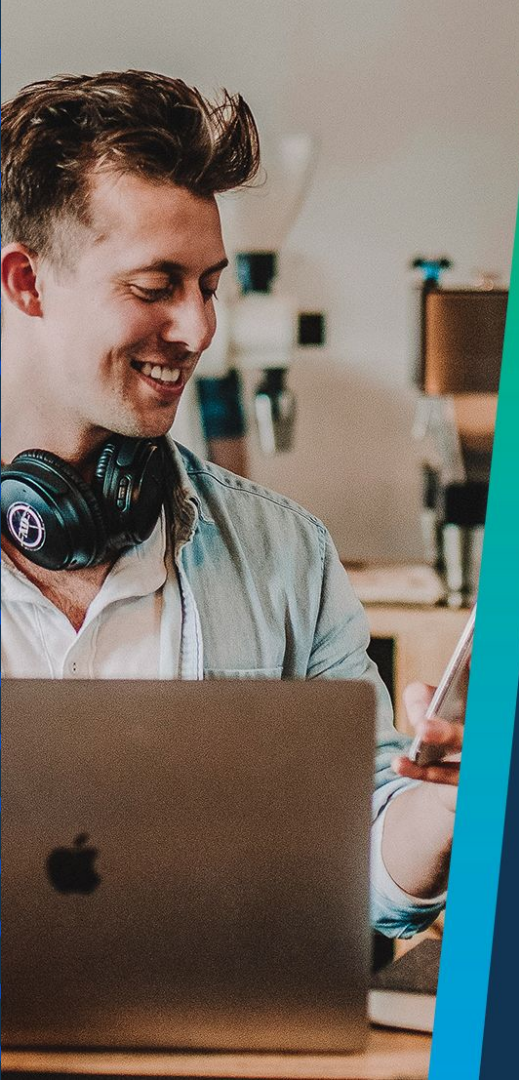
```
# Define parent class
class Computer():
    def __init__(self, computer, ram, ssd):
        self.computer = computer
        self.ram = ram
        self.ssd = ssd

# 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