Class Inheritance

What is inheritance?

Defining a new class that extends an existing class.

Existing class: parent class / superclass

New class: child class / subclass

Child class inherits attributes & methods from parent class

Benefits: Reuse code, reduce complexity of program

Class inheritance syntax


```
class Parent:
    def __init__():
    # parent methods and attributes are defined here


class Child(Parent):
    # parent methods and attributes are inherited
    # child methods and attributes are defined here
```

Inheritance example

We have a Human class

We want to add a Wizard class that can do everything Humans can, but more

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

    def walk(self, direction):
        print(self.name, "walks to the", direction)

    def talk(self, speech):
        print(self.name, "says:", speech)
```

Inheritance example

We have a Human class

We want to add a Wizard class that can do everything Humans can, but more

Wizard class inherits from Human

Wizard class is subclass of Human superclass

Wizard objects have name and age attributes

Wizard objects have walk(), talk(), and cast_spell methods()

```
class Wizard(Human):
    def cast_spell(self, spell):
        print(self.name, "casts", spell)
```

super() function

Child class can have an __init__ method

This overrides Parent's __init__ method

Child no longer has access to Parent's attributes

Resolve by using super() function to call Parent's __init__()

Now Child can have both its own attributes, and the Parent's

```
class Parent:
    def __init__():


class Child(Parent):
    def __init__():
    super().__init__()
    # calls __init__() method of parent
```

super() function example

Wizard once again extends Human class

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

    def walk(self, direction):
        print(self.name, "walks to the", direction)

    def talk(self, speech):
        print(self.name, "says:", speech)
```

super() function example

Wizard once again extends Human class

```
class Wizard(Human):
    def __init__(self, name, age, spell_points):
        super().__init__(name, age)
        self.spell_points = spell_points

    def cast_spell(self, spell):
        print(self.name, "casts", spell)
```

super() function example

Wizard once again extends Human class

Define __init__ in Wizard to add instance attribute spell_points

Now the name and age attributes from Human class are no longer initialized

To resolve this, use super() to call the superclass's init method

Now Wizard class has name, age, and spell_points attributes

```
class Wizard(Human):
    def __init__(self, name, age, spell_points):
        super().__init__(name, age)
        self.spell_points = spell_points

    def cast_spell(self, spell):
        print(self.name, "casts", spell)
```