# Python Examples: super() in Single Inheritance

## 1. Order and OnlineOrder

class Order:  
 def \_\_init\_\_(self, order\_id):  
 self.order\_id = order\_id  
  
 def process\_payment(self):  
 print(f"Processing payment for Order #{self.order\_id}")  
  
class OnlineOrder(Order):  
 def \_\_init\_\_(self, order\_id, email):  
 super().\_\_init\_\_(order\_id)  
 self.email = email  
  
 def process\_payment(self):  
 super().process\_payment()  
 print(f"Sending confirmation email to {self.email}")  
  
order = OnlineOrder(101, "customer@example.com")  
order.process\_payment()

## 2. Employee and Manager

class Employee:  
 def \_\_init\_\_(self, name, salary):  
 self.name = name  
 self.salary = salary  
  
 def display(self):  
 print(f"Name: {self.name}, Salary: ₹{self.salary}")  
  
class Manager(Employee):  
 def \_\_init\_\_(self, name, salary, department):  
 super().\_\_init\_\_(name, salary)  
 self.department = department  
  
 def display(self):  
 super().display()  
 print(f"Department: {self.department}")  
  
m = Manager("Shaik", 90000, "IT")  
m.display()

1.Here, Employee class is the parent class with attributes with name and salary.

2.Manager class is the child class with attributes same of employee and additional attribute ‘department’.

3.so, parent class is inherited into the child class.

4.Using super(), we have inherited the attributes and methods from the parent class.

5.initialized the additional attributes and method information in the child class.

## 3. Vehicle and Car

class Vehicle:  
 def start(self):  
 print("Vehicle started")  
  
class Car(Vehicle):  
 def start(self):  
 super().start()  
 print("Car is ready to go")  
  
c = Car()  
c.start()

1.Here, Vehicle is the parent class and it has only one method.

2.Car is the child class and has implemented the start method from the vehicle class using super()

3.Added additional line as ‘car is ready to go’ in the child’s start method.

4.creat3d an object for child class.

5.As the child class has inherited the attributes and methods from the parent class, the object has access to attributes and methods.

## 4. User Login System

class User:  
 def \_\_init\_\_(self, username):  
 self.username = username  
  
 def login(self):  
 print(f"{self.username} logged in")  
  
class Admin(User):  
 def login(self):  
 super().login()  
 print(f"{self.username} has admin privileges")  
  
a = Admin("admin\_user")  
a.login()

1.Here, User is the parent class with attributes username and method login.

2.Admin is the child class with inherited attribute and login method.

3.Since the user is admin in the child class, he has special privileges. So,this is additionally added in the child’s login method.

4.created an object for the child class

5.accessed all the attributes and methods through the object

## 5. Shape and Circle

class Shape:  
 def \_\_init\_\_(self):  
 print("This is a shape")  
  
 def area(self):  
 print("Area formula not defined")  
  
class Circle(Shape):  
 def \_\_init\_\_(self, radius):  
 super().\_\_init\_\_()  
 self.radius = radius  
  
 def area(self):  
 super().area()  
 print("Circle Area:", 3.14 \* self.radius \* self.radius)  
  
c = Circle(5)  
c.area()

1.Here, the shape is the parent class and circle is the child class.

2.Shape has a method called area and circle class inherited this method from the parent class.

3.In child class, the area for the circle is defined and added as a additional line to the inherited data.

4.object is created for the child class

5. All the attributes and methods are accessed using this object.

## 6. Person and Student

class Person:  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
 def show(self):  
 print(f"Name: {self.name}")  
  
class Student(Person):  
 def \_\_init\_\_(self, name, grade):  
 super().\_\_init\_\_(name)  
 self.grade = grade  
  
 def show(self):  
 super().show()  
 print(f"Grade: {self.grade}")  
  
s = Student("Ali", "A")  
s.show()

1.Here, Person is the parent class with attribute name and method show.

2. Student is the child class with attributes grade and inherited attribute name.

3.Child class has inherited method show and added few more lines to it.

4. Created a object for the child class

5.Accessed all the attributes and methods through the object created.

## 7. BankAccount and SavingsAccount

class BankAccount:  
 def \_\_init\_\_(self, balance):  
 self.balance = balance  
  
 def show\_balance(self):  
 print(f"Balance: ₹{self.balance}")  
  
class SavingsAccount(BankAccount):  
 def \_\_init\_\_(self, balance, interest):  
 super().\_\_init\_\_(balance)  
 self.interest = interest  
  
 def show\_balance(self):  
 super().show\_balance()  
 print(f"Interest Rate: {self.interest}%")  
  
acc = SavingsAccount(10000, 5)  
acc.show\_balance()

1.Here, bankaccount is the parent class with the attributes balance and method show\_balance().

2. Savings account is the child class with inherited attributes balance and inherited method show\_balance.

3.Additionally child class has one more attribute as interest and adds more details to the show\_balance method.

4.These attributes and methods are inherited using super().

5.created an object for the child class and accessed all the attributes and methods through it.

## 8. Product and ElectronicProduct

class Product:  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
 def details(self):  
 print(f"Product: {self.name}")  
  
class ElectronicProduct(Product):  
 def \_\_init\_\_(self, name, warranty):  
 super().\_\_init\_\_(name)  
 self.warranty = warranty  
  
 def details(self):  
 super().details()  
 print(f"Warranty: {self.warranty} years")  
  
p = ElectronicProduct("Laptop", 2)  
p.details()

1.Here, product is the super class and electronic product is the child class.

2.super class has attributes and methods in it.

3. Child class inherited all the attributes and methods from the parent class using super().

4. Child class has additional attributes and additional data of the method.

5.created an object for the child class and accessed all the methods and attributes through it.

## 9. Animal and Dog

class Animal:  
 def sound(self):  
 print("Animal sound")  
  
class Dog(Animal):  
 def sound(self):  
 super().sound()  
 print("Dog barks")  
  
d = Dog()  
d.sound()

1.Here, animal is the super class with an attribute.

2.Dog is the child class with inherited method from the parent class

3.since, dog is one of the animal, it has inherited the properties of animal.

4.As it has an additional feature/sound than other animals, it is added extra in this class.

5.created an object for the child class and accessed all the properties through it.

## 10. Book and EBook

class Book:  
 def \_\_init\_\_(self, title):  
 self.title = title  
  
 def show(self):  
 print(f"Title: {self.title}")  
  
class EBook(Book):  
 def \_\_init\_\_(self, title, file\_size):  
 super().\_\_init\_\_(title)  
 self.file\_size = file\_size  
  
 def show(self):  
 super().show()  
 print(f"File Size: {self.file\_size} MB")  
  
eb = EBook("Python Guide", 5)  
eb.show()

1.Here, book is the parent class and ebook is the child class.

2.As ebook is a different form of book, it inherits all the properties from book class

3.since, it is an ebook it has additional features compared to book, so these features are added additionally in the child class

4.We create an object for the child class to access all these properties.

5.using super() child class has inherited the properties from the parent class.