**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()

Points understood:

-> Here order is the base class

->Online order is the derived class- derives the properties from base class

It derives super().\_\_init\_\_(orderid) from parent class to give value for it.

Function defined is process\_payment

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->super.process\_payment() here process\_payment() method is overridden,to add extra feature that is to sent email. We are just adding the extra feature to parent’s super().process\_payment()

->online order is the object created, when this and process payment() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here employee is the base class

->manager is the derived class- derives the properties from base class

It derives super().\_\_init\_\_(salary) from parent class to give value for it.

Function defined is display

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->super.display() here display() method is overridden,to add extra feature that is add department name. We are just adding the extra feature to parent’s super().display()

->manager is the object created, when this and display() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here Vehicle is the base class

->Car  is the derived class- derives the properties from base class

Function defined is start

->super() is used by child class to call the parent class and to get their data.

->super.start() here start() method is not overridden,no  extra feature added .

->car is the object created, when this and start() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here User is the base class

->Admin is the derived class- derives the properties from base class

Function defined is login()

->super() is used by child class to call the parent class and to get their data.

->super.login() here login() method is overridden. No extra feature added, just username inherited

->admin is the object created, when this and login() is called, both the values and data from child and base class is executed.ata from child and base class is executed.

**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()

Points understood:

-> Here shape is the base class

->circle is the derived class- derives the properties from base class

It derives super().\_\_init\_\_() from parent class to give value for it.

Function defined is area

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->here radius is extra feature added by defining it in def \_\_init\_\_(radius)

super.area() method is overridden,extra feature added.

->circle is the object created, when this and area() is called, both the values and data from child and base class is executed and displayed

**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()

Points understood:

-> Here person is the base class

->Student is the derived class- derives the properties from base class

It derives super().\_\_init\_\_(name) from parent class to give value for it.

Function defined is show

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->super.show() here show() method is overridden.

 Extra feature is added that is along with name ,grade is added.

->student is the object created, when this and show() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here BankAccount is the base class

->SavingsAccount is the derived class- derives the properties from base class

It derives super().\_\_init\_\_(balance) from parent class to give value for it.

Function defined is show\_balance

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->super.show\_balance() here show\_balance() method is overridden.

 Extra feature is added that is along with balance , interest is added.

->Savings \_Account is the object created, when this and show\_balance() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here Product is the base class

->Electronic Product is the derived class- derives the properties from base class

It derives super().\_\_init\_\_(name) from parent class to give value for it.

Function defined is details

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->super.details() here details () method is overridden.

 Extra feature is added that is along with name ,warrenty is added.

->Electronic product is the details created, when this and details() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here Animal is the base class

->Dog  is the derived class- derives the properties from base class

Function defined is sound

->super() is used by child class to call the parent class and to get their data.

->super.sound() here start() method is not overridden,no  extra feature added, it inherits properties from animal class

->dog is the object created, when this and sound() is called, both the values and data from child and base class is executed.

**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()

Points understood:

-> Here book is the base class

->ebook is the derived class- derives the properties from base class

It derives super().\_\_init\_\_(title) from parent class to give value for it.

Function defined is show()

->super() is used by child class to call the parent class and to get their data.

Here child class call \_\_init\_\_() constructor ,so the parents data get initialized to base class

->super.show() here show() method is overridden.

 Extra feature is added that is along with title ,file\_size is added.

->ebook is the object created, when this and show() is called, both the values and data from child and base class is executed.