

Inheritance

Kalıtım, bir sınıfın özellikleri başka bir sınıftan türetme veya miras alma yeteneğidir.

Mirasın faydaları şunlardır:

Gerçek dünya ilişkilerini iyi temsil eder.

Bir kodun yeniden kullanılabilirliğini sağlar . Aynı kodu tekrar tekrar yazmak zorunda değiliz. Ayrıca, bir sınıfa onu değiştirmeden daha fazla özellik eklememize izin verir.

Bu, doğası gereği geçişlidir, yani B sınıfı başka bir A sınıfından miras alırsa, B'nin tüm alt sınıfları otomatik olarak A sınıfından miras alır.

Python Kalıtım Sözdizimi

Class BaseClass:

{Body}

Class DerivedClass(BaseClass):

{Body}

Ebeveyn Sınıfı Oluşturma

Creating a Person class with Display methods.

```
# A Python program to demonstrate inheritance

class Person(object):

    # Constructor
    def __init__(self, name, id):
        self.name = name
        self.id = id

    # To check if this person is an employee
    def Display(self):
        print(self.name, self.id)

# Driver code
emp = Person("Satyam", 102) # An Object of Person
emp.Display()
```

Output:

Satyam 102

Creating a Child Class

Here Emp is another class which is going to inherit the properties of the Person class(base class).(Burada Emp , Person sınıfının (temel sınıf) özelliklerini miras alacak başka bir sınıftır)

```

class Emp(Person):

    def Print(self):
        print("Emp class called")

Emp_details = Emp("Mayank", 103)

# calling parent class function
Emp_details.Display()

# Calling child class function
Emp_details.Print()

```

Output:

Mayank 103

Emp class called

Example of Inheritance in Python

```

# A Python program to demonstrate inheritance

# Base or Super class. Note object in bracket.
# (Generally, object is made ancestor of all classes)
# In Python 3.x "class Person" is
# equivalent to "class Person(object)"

class Person(object):

    # Constructor
    def __init__(self, name):
        self.name = name

    # To get name
    def getName(self):
        return self.name

    # To check if this person is an employee
    def isEmployee(self):
        return False

# Inherited or Subclass (Note Person in bracket)
class Employee(Person):

    # Here we return true
    def isEmployee(self):
        return True

# Driver code
emp = Person("Geek1") # An Object of Person
print(emp.getName(), emp.isEmployee())

emp = Employee("Geek2") # An Object of Employee
print(emp.getName(), emp.isEmployee())

```

Output:

Geek1 False

Geek2 True

Subclassing (Calling constructor of parent class)

Bir alt sınıf, hangi sınıfın üst sınıf olduğunu belirlemelidir. Bu, alt sınıfın tanımında üst sınıf adının belirtilmesiyle yapılabilir.

class subclass_name (superclass_name):

```
# Python code to demonstrate how parent constructors
# are called.

# parent class
class Person(object):

    # __init__ is known as the constructor
    def __init__(self, name, idnumber):
        self.name = name
        self.idnumber = idnumber

    def display(self):
        print(self.name)
        print(self.idnumber)

# child class

class Employee(Person):
    def __init__(self, name, idnumber, salary, post):
        self.salary = salary
        self.post = post

        # invoking the __init__ of the parent class
        Person.__init__(self, name, idnumber)

# creation of an object variable or an instance
a = Employee('Rahul', 886012, 200000, "Intern")

# calling a function of the class Person using its instance
a.display()
```

Output:

Rahul

886012

'a' is the instance created for the class Person. It invokes the `__init__()` of the referred class. You can see 'object' written in the declaration of the class Person. In Python, every class inherits from a built-in basic class called 'object'. The constructor i.e. the '`__init__`' function of a class is invoked when we create an object variable or an instance of the class.

The variables defined within `__init__()` are called the instance variables or objects. Hence, 'name' and 'idnumber' are the objects of the class Person. Similarly, 'salary' and 'post' are the objects of the class Employee. Since the class Employee inherits from class Person,

'a', Person sınıfı için oluşturulan örnektir. Başvurulan sınıfın `__init__()` ögesini çağırır. Person sınıfının bildiriminde yazılı 'nesne' görebilirsiniz. Python'da her sınıf, 'nesne' adı verilen yerleşik bir temel sınıftan miras alır. Yapıcı, yani bir sınıfın '`__init__`' işlevi, bir nesne değişkeni veya sınıfın bir örneğini oluşturduğumuzda çağrılır.

`__init__()` içinde tanımlanan değişkenlere örnek değişkenler veya nesneler denir. Bu nedenle, 'ad' ve 'idnumber', Person sınıfının nesneleridir. Benzer şekilde, 'maaş' ve 'görev', Çalışan sınıfının nesneleridir. Employee sınıfı Person sınıfından miras aldığından, 'name' ve 'idnumber' aynı zamanda Employee sınıfının nesneleridir.

'name' and 'idnumber' are also the objects of class Employee.	
---	--

Python program to demonstrate error if we forget to invoke `__init__()` of the parent

If you forget to invoke the `__init__()` of the parent class then its instance variables would not be available to the child class.

Üst sınıfın `__init__()` işlevini çağırılmayı unutursanız, örnek değişkenleri alt sınıf tarafından kullanılamaz.

```
class A:
    def __init__(self, n='Rahul'):
        self.name = n
```

```
class B(A):
    def __init__(self, roll):
        self.roll = roll
```

```
object = B(23)
print(object.name)
```

Output :

Traceback (most recent call last):

File `"/home/de4570cca20263ac2c4149f435dba22c.py"`, line 12, in

```
print (object.name)
```

AttributeError: 'B' object has no attribute 'name'

Python'da Kalıtım Türleri

Tekli kalıtım, türetilmiş bir sınıfın özellikleri tek bir üst sınıftan devralmasını sağlar, böylece kodun yeniden kullanılabilirliğini ve mevcut koda yeni özelliklerin eklenmesini sağlar.



Single Inheritance

```
# Python program to demonstrate
# single inheritance

# Base class
class Parent:
    def func1(self):
        print("This function is in parent class.")

# Derived class

class Child(Parent):
    def func2(self):
        print("This function is in child class.")

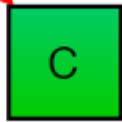
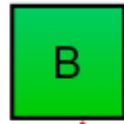
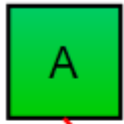
# Driver's code
object = Child()
object.func1()
object.func2()
```

Output:

This function is in parent class.

This function is in child class.

Multiple Inheritance:



Multiple Inheritance

Output:

Father : RAM
Mother : SITA

```
# Python program to demonstrate  
# multiple inheritance
```

```
# Base class1
```

```
class Mother:  
    mothername = ""  
  
    def mother(self):  
        print(self.mothername)
```

```
# Base class2
```

```
class Father:  
    fathername = ""  
  
    def father(self):  
        print(self.fathername)
```

```
# Derived class
```

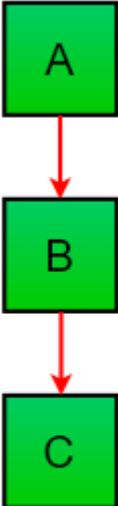
```
class Son(Mother, Father):  
    def parents(self):  
        print("Father :", self.fathername)  
        print("Mother :", self.mothername)
```

```
# Driver's code
```

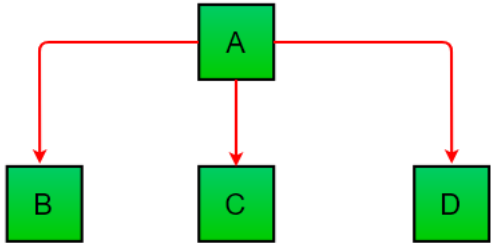
```
s1 = Son()  
s1.fathername = "RAM"  
s1.mothername = "SITA"  
s1.parents()
```

Multilevel Inheritance :

Çok düzeyli kalıtımda, temel sınıfın ve türetilmiş sınıfın özellikleri, yeni türetilmiş sınıfa ayrıca miras alınır. Bu, bir çocuğu ve bir büyükbabayı temsil eden bir ilişkiye benzer.--

<div data-bbox="119 129 542 645"> <p>Base Class</p> <p>Intermediary Class</p> <p>Derived Class</p>  <pre> graph TD A[A] --> B[B] B --> C[C] </pre> </div> <p>Multilevel Inheritance</p> <p>Output:</p> <p>Lal mani Grandfather name : Lal mani Father name : Rampal Son name : Prince</p>	<pre> # Python program to demonstrate # multilevel inheritance # Base class class Grandfather: def __init__(self, grandfathername): self.grandfathername = grandfathername # Intermediate class class Father(Grandfather): def __init__(self, fathername, grandfathername): self.fathername = fathername # invoking constructor of Grandfather class Grandfather.__init__(self, grandfathername) # Derived class class Son(Father): def __init__(self, sonname, fathername, grandfathername): self.sonname = sonname # invoking constructor of Father class Father.__init__(self, fathername, grandfathername) def print_name(self): print('Grandfather name :', self.grandfathername) print("Father name :", self.fathername) print("Son name :", self.sonname) # Driver code s1 = Son('Prince', 'Rampal', 'Lal mani') print(s1.grandfathername) s1.print_name() </pre>
--	---

Hierarchical Inheritance:

<p>Tek bir temelden birden fazla türetilmiş sınıf oluşturulduğunda, bu tür kalıtım hiyerarşik kalıtım olarak adlandırılır. Bu programda bir ebeveyn (temel) sınıfımız ve iki çocuk (türetilmiş) sınıfımız var.</p> <div data-bbox="223 1489 715 1736">  <pre> graph TD A[A] --> B[B] A --> C[C] A --> D[D] </pre> </div> <p>Hierarchical Inheritance</p>	<pre> # Python program to demonstrate # Hierarchical inheritance # Base class class Parent: def func1(self): print("This function is in parent class.") # Derived class1 class Child1(Parent): def func2(self): print("This function is in child 1.") # Derived class2 class Child2(Parent): def func3(self): print("This function is in child 2.") # Driver's code object1 = Child1() object2 = Child2() object1.func1() object1.func2() object2.func1() object2.func3() </pre>
--	--

