

Read

Discuss

Courses

Practice

Video

Inheritance in Python

Difficulty Level: Easy • Last Updated: 21 Nov, 2022

One of the core concepts in object-oriented programming (OOP) languages is inheritance. It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class. Inheritance is the capability of one class to derive or inherit the properties from another class.

Benefits of inheritance are:

- It represents real-world relationships well.
- It provides the **reusability** of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.
- Inheritance offers a simple, understandable model structure.
- Less development and maintenance expenses result from an inheritance.

Python Inheritance Syntax

```
Class BaseClass:
    {Body}
Class DerivedClass(BaseClass):
    {Body}
```

Creating a Parent Class

Creating a Person class with Display methods.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our <u>Cookie Policy</u> & <u>Privacy Policy</u>

Got It!

A Python program to demonstrate inheritance

```
# 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)

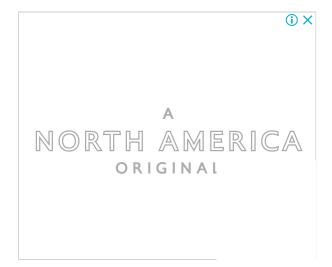
# Driver code
emp = Person("Satyam", 102) # An Object of Person
emp.Display()
```

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).



```
Data Structures and Algorithms Interview Preparation Data Science Topic-wise Practice C

Read Discuss Courses Practice Video

print("Emp Class called")

Emp_details = Emp("Mayank", 103)

# calling parent class function

Emp_details.Display()

# Calling child class function

Emp_details.Print()
```

```
Mayank 103
Emp class called
```

Example of Inheritance in Python

Python3

```
# 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 isEmplovee(self):
```

```
# Inherited or Subclass (Note Person in bracket)
   Read
           Discuss
                     Courses
                                Practice
                                          Video
   THELE WE LELULII LIVE
    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

What is object class?

Like the <u>Java Object class</u>, in <u>Python</u> (from version 3. x), the object is the root of all classes.

- In Python 3.x, "class Test(object)" and "class Test" are same.
- In Python 2. x, "class Test(object)" creates a class with the object as a parent (called a new-style class), and "class Test" creates an old-style class (without an objecting parent).

Subclassing (Calling constructor of parent class)

A child class needs to identify which class is its parent class. This can be done by mentioning the parent class name in the definition of the child class.

Eg: class subclass_name (superclass_name):

Python3

```
# parent class
   Read
           Discuss
                     Courses
                               Practice
                                          Video
   TILL 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()
```

Rahul

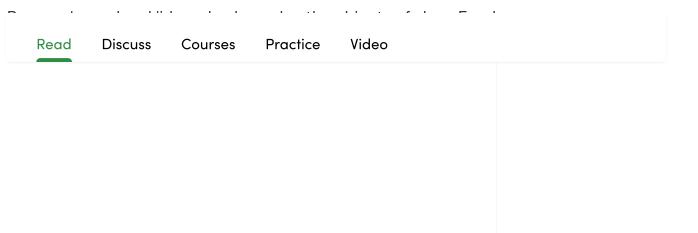
886012

'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.

Hanner Januari and Jidan make Jeneraka akinda akika alam Danana Cindiani. Jenjam Jend

'post' are the objects of the class Employee. Since the class Employee inherits from class



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.

The following code produces an error for the same reason.

Python3

```
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):
```

```
print (object.name)

AttributoEnnon: 'B' object has no attributo 'name'

Read Discuss Courses Practice Video
```

Different types of Inheritance:

- **Single inheritance**: When a child class inherits from only one parent class, it is called single inheritance. We saw an example above.
- **Multiple inheritances**: When a child class inherits from multiple parent classes, it is called multiple inheritances.

Unlike java, python shows multiple inheritances.

Python3

```
# Python example to show the working of multiple
# inheritance
class Base1(object):
    def init (self):
        self.str1 = "Geek1"
        print("Base1")
class Base2(object):
    def __init__(self):
        self.str2 = "Geek2"
        print("Base2")
class Derived(Base1, Base2):
    def __init__(self):
        # Calling constructors of Base1
        # and Base2 classes
        Base1.__init__(self)
        Base2.__init__(self)
        print("Derived")
    def printStrs(self):
        print(self.str1, self.str2)
```

ob.printStrs()

```
Read Discuss Courses Practice Video

Base1
Base2
Derived
Geek1 Geek2
```

• Multilevel inheritance: When we have a child and grandchild relationship.

Python3

```
# 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 Base(object):
    # Constructor
    def __init__(self, name):
        self.name = name
    # To get name
    def getName(self):
        return self.name
# Inherited or Sub class (Note Person in bracket)
class Child(Base):
    # Constructor
    def __init__(self, name, age):
        Base.__init__(self, name)
        self.age = age
    # To get name
    def getAge(self):
        raturn calf aga
```

```
Read Discuss Courses Practice Video

def __init__(self, name, age, address):
    Child.__init__(self, name, age)
    self.address = address

# To get address
def getAddress(self):
    return self.address

# Driver code
g = GrandChild("Geek1", 23, "Noida")
print(g.getName(), g.getAge(), g.getAddress())
```

Geek1 23 Noida

- **Hierarchical inheritance** More than one derived class are created from a single base.
- **Hybrid inheritance**: This form combines more than one form of inheritance. Basically, it is a blend of more than one type of inheritance.

For more details please read this article: Types of inheritance in Python

Private members of the parent class

We don't always want the instance variables of the parent class to be inherited by the child class i.e. we can make some of the instance variables of the parent class private, which won't be available to the child class.

We can make an instance variable private by adding double underscores before its name. For example,

Python3

Python program to demonstrate private members

```
class C(object):
   def __init__(self):
                                           Video
   Read
           Discuss
                     Courses
                                Practice
       # n To hi Trace Thorance rai Tante
        self._d = 42
class D(C):
   def __init__(self):
        self.e = 84
        C.__init__(self)
object1 = D()
# produces an error as d is private instance variable
print(object1.d)
```

```
File "/home/993bb61c3e76cda5bb67bd9ea05956a1.py", line 16, in
    print (object1.d)
AttributeError: type object 'D' has no attribute 'd'
```

Since 'd' is made private by those underscores, it is not available to the child class 'D' and hence the error.



Like 283

Previous

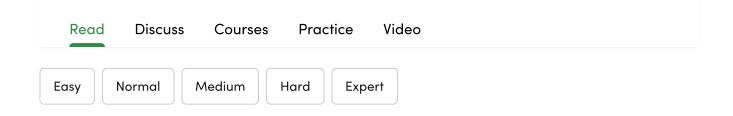
Read Discuss Courses Practice Video

Related Articles

- OOP in Python | Set 3 (Inheritance, examples of object, issubclass and super)
 Inheritance in Python | Set 2
- 3. Method resolution order in Python Inheritance
- 4. Python | super() in single inheritance
- 5. Data Classes in Python | Set 4 (Inheritance)
- 6. Python | super() function with multilevel inheritance
- 7. Multiple Inheritance in Python
- 8. Types of inheritance Python
- 9. Inheritance in Python Inner Class
- 10. Conditional Inheritance in Python

Article Contributed By:





Improved By: AyushShekhar, harshit_gokharu, shawnlouis2000, rohanasnani, droidbot,

meghanathj1, surajkr_gupta, kumar_satyam, uddhavpatil9560

Article Tags: python-inheritance, python-oop-concepts, Python

Practice Tags: python

Improve Article

Report Issue



A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company	Learn
About Us	DSA
Careers	Algorithms
In Media	Data Structures
Contact Us	SDE Cheat Sheet
Privacy Policy	Machine learning
Copyright Policy	CS Subjects

Courses

Read Discuss Courses Practice	Video
TOP NEWS	Python
Technology	Java
Work & Career	CPP
Business	Golang
Finance	C#
Lifestyle	SQL
Knowledge	Kotlin
Web Development	Contribute
Web Tutorials	Write an Article
Django Tutorial	Improve an Article
HTML	Pick Topics to Write
JavaScript	Write Interview Experience
Bootstrap	Internships
ReactJS	Video Internship
NodeJS	

@geeksforgeeks, Some rights reserved