Inheritance (Inheritance) in Python

Inheriting in object-oriented programming allows us to declare new classes to re-use the parent class's functions and attributes and extra functions. In this article, we will learn how to use legacy in Python.

Inheriting in object-oriented programming allows us to declare new classes to re-use the parent class's functions and attributes and extra functions. In this article, we will learn how to use legacy in Python.

What is inheritance?

Inheritance allows a class (class) to inherit properties and methods from other defined classes. The class has been called a parent class (base class or parent class), the newly derived class is called a subclass (child class or derived class). The subclass inherits all elements of the superclass, which can extend inheritance elements and add new components.

Syntax of inheritance

```
 class BaseClass:
  Body of base class

 class DerivedClass(BaseClass):
  Body of derived class
```

For more clarity about using inheritance, let's take an example.

Polygon is a closed shape with 3 or more sides. We have a class called *DaGiac* defined as follows.

```
 class DaGiac:

  def __init__(self, socanh):
  self.n = socanh
  self.canh = [0 for i in range(socanh)]

  def nhapcanh(self):
  self.canh = [float(input("B?n hãy nh?p giá tr? c?
nh "+str(i+1)+" : ")) for i in range(self.n)]

  def hienthicanh(self):
  for i in range(self.n):
  print("Giá tr? c?nh",i+1,"là",self.canh[i])
```

Class *DaGiac* has the n attribute to define the number of edges and edges to store each edge value. The *nhapcanh ()* function takes the size of the edges and *hienthicanh ()* will display the list of polygons.

The triangle is polygon with three sides, so we will create a class *TamGiac* inherited from *DaGiac.* This new class will inherit all the properties available in the superclass, so you won't need to re-declare (reuse code). This *TamGiac* is declared as follows:

```
class TamGiac(DaGiac):

 def __init__(self):
 DaGiac.__init__(self,3)

 def dientich(self):
 a, b, c = self.canh
 # Tính n?a chu vi
 s = (a + b + c) / 2
 area = (s*(sa)*(sb)*(sc)) ** 0.5
 print('Di?n tích c?a hình tam giác là %0.2f' %area)
```

The *TamGiac* class not only inherits but also defines a new function as the *dientich* function .

```
>>> t = TamGiac()
>>> t.nhapcanh()
B?n hãy nh?p giá tr? c?nh 1 : 3
B?n hãy nh?p giá tr? c?nh 2 : 5
B?n hãy nh?p giá tr? c?nh 3 : 4

>>> t.hienthicanh()
Giá tr? c?nh 1 là 3.0
Giá tr? c?nh 2 là 5.0
Giá tr? c?nh 3 là 4.0

>>> t.dientich()
Di?n tích c?a hình tam giác là 6.00
```

We can see that the functions *nhapcanh (), hienthicanh ()* are not in TamGiac class, but we still use them

Overriding in Python

Python allows to override parent class methods. You can perform the override of the parent class method if you want to have special or special features in the subclass.

In the above example, we see that the *TamGiac* class uses the *__init __ ()* function from the *DaGiac* class , if you want to override the definition of *__init __ ()* function in the parent class, we use the super () function.

super () .*__ init __ (3)* is equivalent to *DaGiac .__ init __ (self, 3)*

Also Python has two *isinstance ()* and *issubclass () functions that* are used to test the relationship of two classes and instances.

The *issubclass* function *(classA, classB)* returns True if class A is a subclass of class B.

```
>>> issubclass(DaGiac,TamGiac)
False
```

```
>>> issubclass(TamGiac,DaGiac)
True

>>> issubclass(bool,int)
True
```

The *isinstance (obj, Class) function* returns True if obj is an instance of Class or is an instance of Class's subclass.

```
>>> isinstance(t,TamGiac)
True

>>> isinstance(t,DaGiac)
True

>>> isinstance(t,int)
False

>>> isinstance(t,object)
True
```

See more:

1. Object-oriented programming in Python
2. Manage files and folders in Python
3. Learn about Error and Exception

Previous lesson: Class and Object in Python

Previous article: Multiple Inheritance in Python