# 14.9: INHERITANCE

Contributed by Chuck Severance
Clinical Associate Professor (School of Information) at University of Michigan

Another powerful feature of object oriented programming is the ability to create a new class by extending an existing class. When extending a class, we call the original class the 'parent class' and the new class as the 'child class'.

For this example, we will move our `PartyAnimal` class into its own file:

```
class PartyAnimal:
   x = 0
   name = ''
   def __init__(self, nam):
     self.name = nam
     print(self.name,'constructed')

   def party(self) :
     self.x = self.x + 1
     print(self.name,'party count',self.x)

# Code: http://www.py4e.com/code3/party.py
```

Then, we can 'import' the `PartyAnimal` class in a new file and extend it as follows:

```
from party import PartyAnimal

class CricketFan(PartyAnimal):
   points = 0
   def six(self):
      self.points = self.points + 6
      self.party()
      print(self.name,"points",self.points)

s = PartyAnimal("Sally")
s.party()
j = CricketFan("Jim")
j.party()
j.six()
print(dir(j))

# Code: http://www.py4e.com/code3/party6.py
```

When we are defining the `CricketFan` object, we indicate that we are extending the `PartyAnimal` class. This means that all of the variables ( `x` ) and methods ( `party` ) from the `PartyAnimal` class are inherited by the `CricketFan` class.

You can see that within the `six` method in the `CricketFan` class, we can call the `party` method from the `PartyAnimal` class. The variables and methods from the parent class are *merged* into the child class.

As the program executes, we can see that the `s` and `j` are independent instances of `PartyAnimal` and `CricketFan` . The `j` object has additional capabilities beyond the `s` object.

```
Sally constructed
Sally party count 1
Jim constructed
Jim party count 1
Jim party count 2
Jim points 6
['__class__', '__delattr__', ... '__weakref__',
'name', 'party', 'points', 'six', 'x']
```

In the `dir` output for the `j` object (instance of the `CricketFan` class) you can see that it both has the attributes and methods of the parent class as well as the attributes and methods that were added when the class was extended to create the `CricketFan` class.