Inheritance

- When we make a new class - we can reuse an existing class and inherit all the capabilities of an existing class and then add our own little bit to make our new class
- Another form of store and reuse
- Write once - reuse many times
- The new class (child) has all the capabilities of the old class (parent) - and then some more

Terminology: Inheritance

‘Subclasses’ are more specialized versions of a class, which inherit attributes and behaviors from their parent classes, and can introduce their own.

http://en.wikipedia.org/wiki/Object-oriented_programming

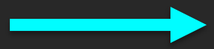
```
class PartyAnimal:

   def __init__(self, nam):
     self.x = 0
     self.name = nam
     print(self.name,"constructed")

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

class FootballFan(PartyAnimal):

   def __init__(self, nam) :
       super().__init__(nam)
       self.points = 0

   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"points",self.points)
```

```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```

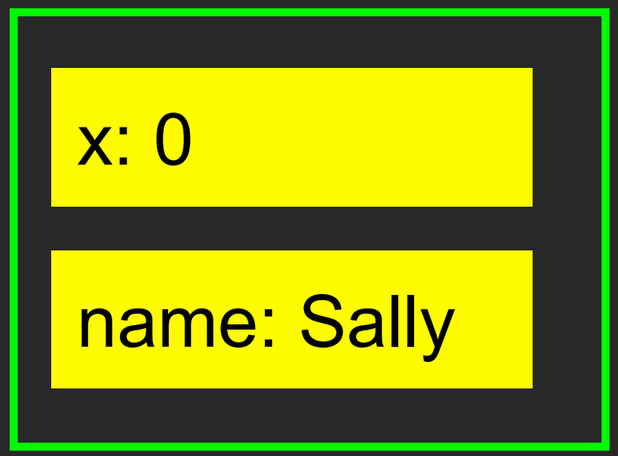
FootballFan is a class which extends PartyAnimal. It has all the capabilities of PartyAnimal and more.

party7.py

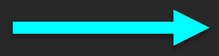
```
class PartyAnimal:

   def __init__(self, nam):
     self.x = 0
     self.name = nam
     print(self.name,"constructed")

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

class FootballFan(PartyAnimal):

   def __init__(self, nam) :
       super().__init__(nam)
       self.points = 0

   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"points",self.points)
```



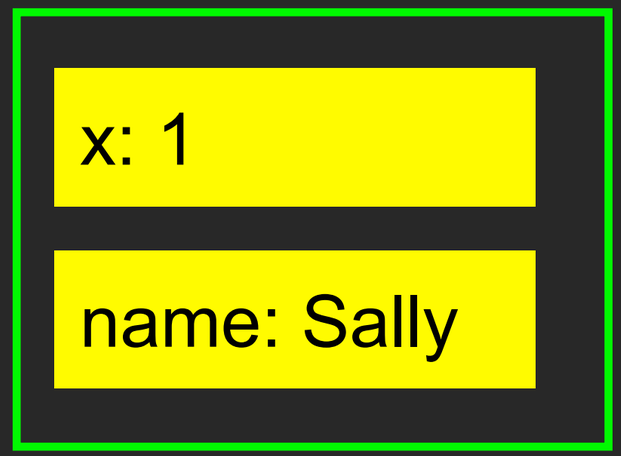
```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```



```
class PartyAnimal:

   def __init__(self, nam):
     self.x = 0
     self.name = nam
     print(self.name,"constructed")

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

class FootballFan(PartyAnimal):

   def __init__(self, nam) :
       super().__init__(nam)
       self.points = 0

   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"points",self.points)
```

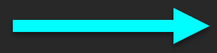


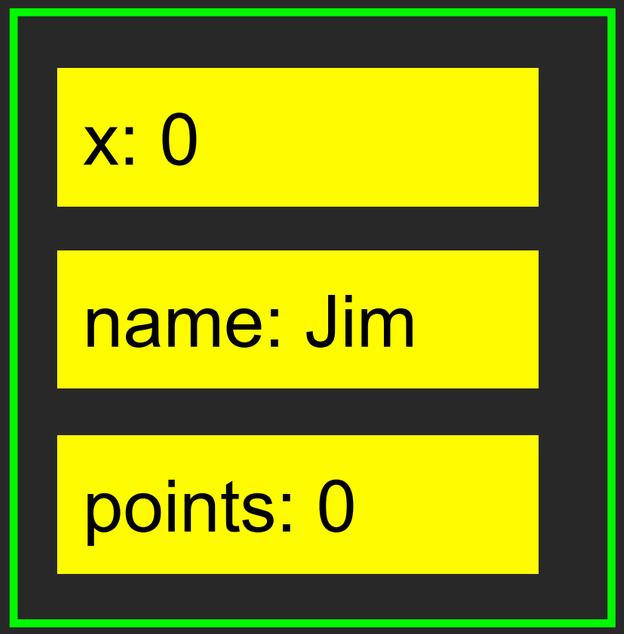
```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```



```
class PartyAnimal:

   def __init__(self, nam):
     self.x = 0
     self.name = nam
     print(self.name,"constructed")

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

class FootballFan(PartyAnimal):

   def __init__(self, nam) :
       super().__init__(nam)
       self.points = 0

   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"points",self.points)
```

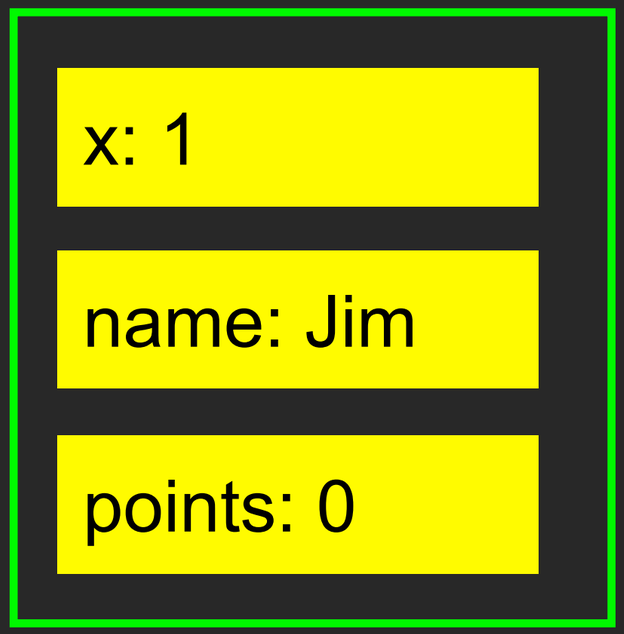
```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```





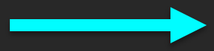
```
class PartyAnimal:

   def __init__(self, nam):
     self.x = 0
     self.name = nam
     print(self.name,"constructed")

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

class FootballFan(PartyAnimal):

   def __init__(self, nam) :
       super().__init__(nam)
       self.points = 0

   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"points",self.points)
```

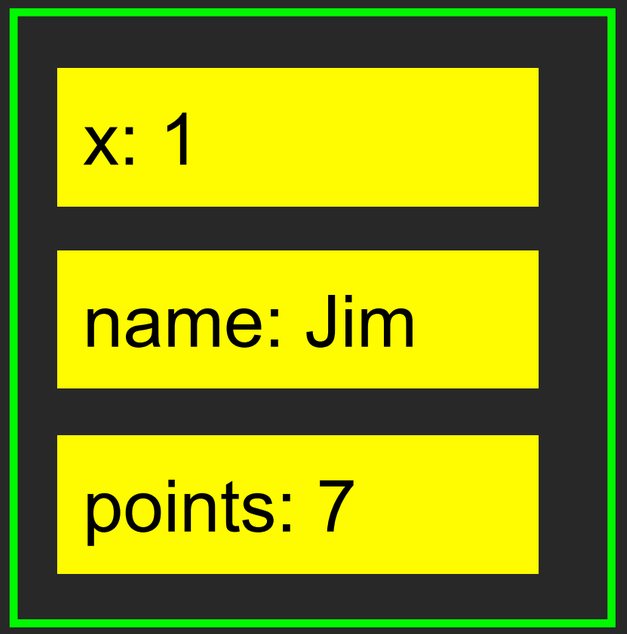
```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```



```
class PartyAnimal:

   def __init__(self, nam):
     self.x = 0
     self.name = nam
     print(self.name,"constructed")

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

class FootballFan(PartyAnimal):

   def __init__(self, nam) :
       super().__init__(nam)
       self.points = 0

   def touchdown(self):
      self.points = self.points + 7
      self.party()
      print(self.name,"points",self.points)
```

```
s = PartyAnimal("Sally")
s.party()

j = FootballFan("Jim")
j.party()
j.touchdown()
```





Definitions

Class - a template

Attribute – A variable within a class

Method - A function within a class

Object - A particular instance of a class

Constructor – Code that runs when an object is created

Inheritance - The ability to extend a class to make a new class.

Summary

- Object Oriented programming is a very structured approach to code reuse.
- We can group data and functionality together and create many independent instances of a class

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

… Insert new Contributors here

...

Additional Source Information