Object Lifecycle

http://en.wikipedia.org/wiki/Constructor_(computer_science)

Object Lifecycle

- Objects are created, used and discarded
- We have special blocks of code (methods) that get called
    - At the moment of creation (constructor)
    - At the moment of destruction (destructor)
- Constructors are used a lot
- Destructors are seldom used

Constructor

- The primary purpose of the constructor is to set up some instance variables to have the proper initial values when the object is created

Objects – Part 3

```
class PartyAnimal:

   def __init__(self):
     self.x = 0
     print('I am constructed')

   def party(self) :
     self.x = self.x + 1
     print('So far',self.x)

   def __del__(self):
     print('I am destructed', self.x)

an = PartyAnimal()
an.party()
an.party()
an = 42
print('an contains',an)
```

```
$ python party4.py
I am constructed
So far 1
So far 2
I am destructed 2
an contains 42
```

The constructor and destructor are optional. The constructor is typically used to set up variables. The destructor is seldom used.

Constructor

- In object oriented programming, a constructor in a class is a special block of statements called when an object is created

http://en.wikipedia.org/wiki/Constructor_(computer_science)

Many Instances

- We can create lots of objects - the class is the template for the object
- We can store each distinct object in its own variable
- We call this having multiple instances of the same class
- Each instance has its own copy of the instance variables

```
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```

Constructors can have additional parameters. These can be used to set up instance variables for the particular instance of the class (i.e., for the particular object).

`party5.py`

Objects – Part 3

```
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```

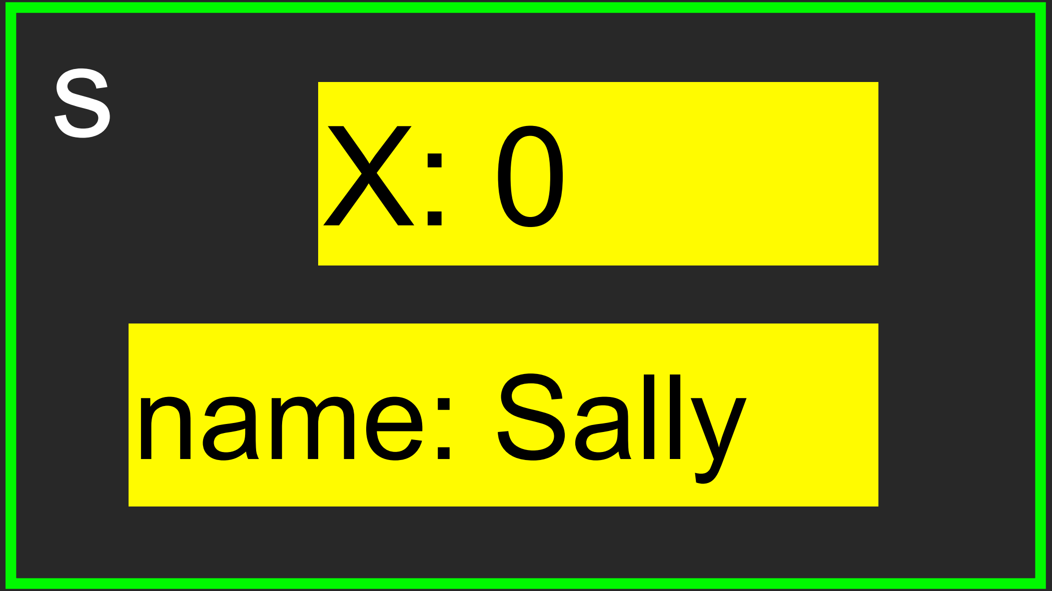
```
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```



Objects – Part 3

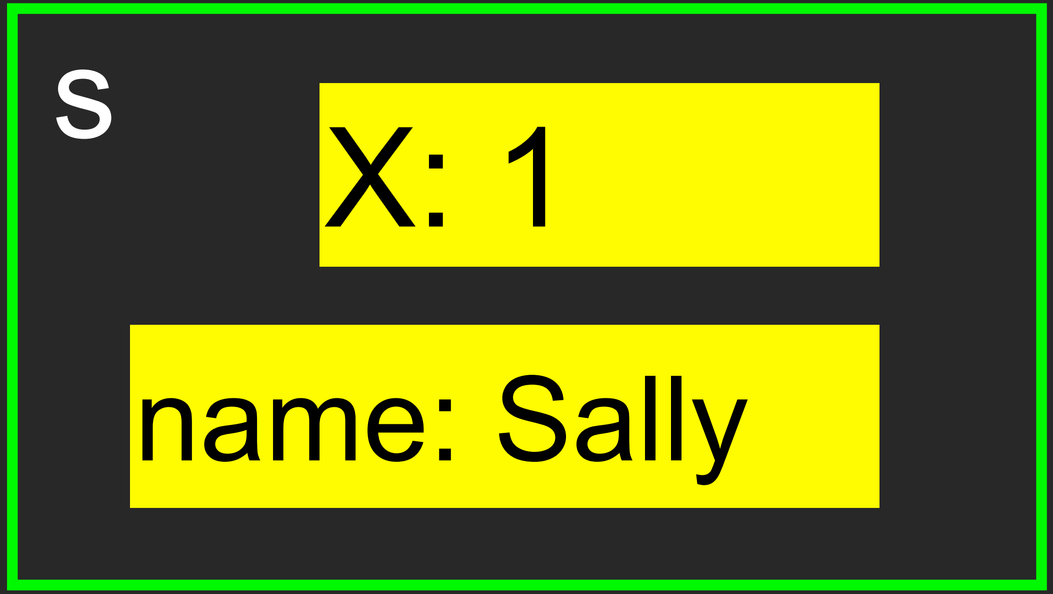
```
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```



Objects – Part 3

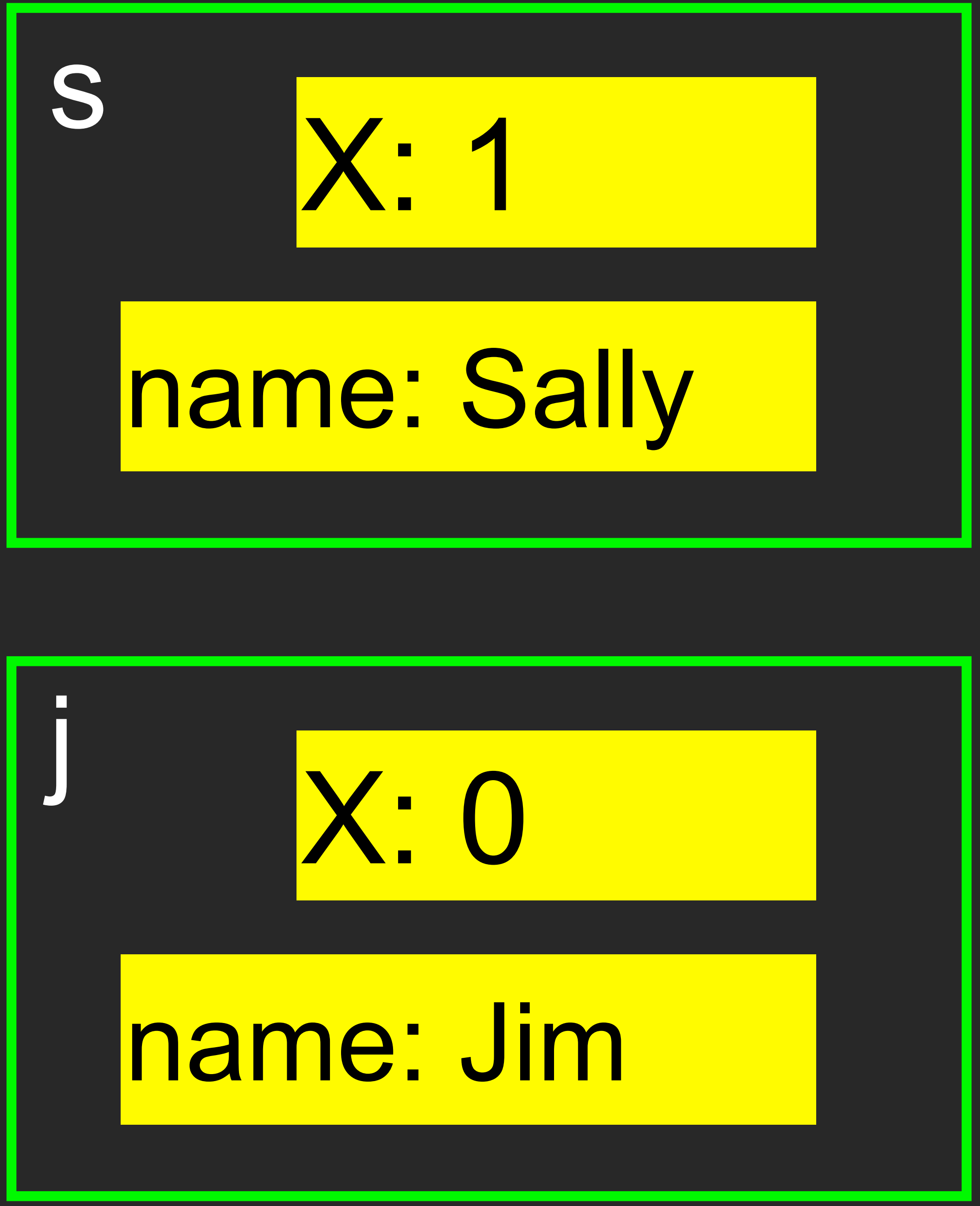
```
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```



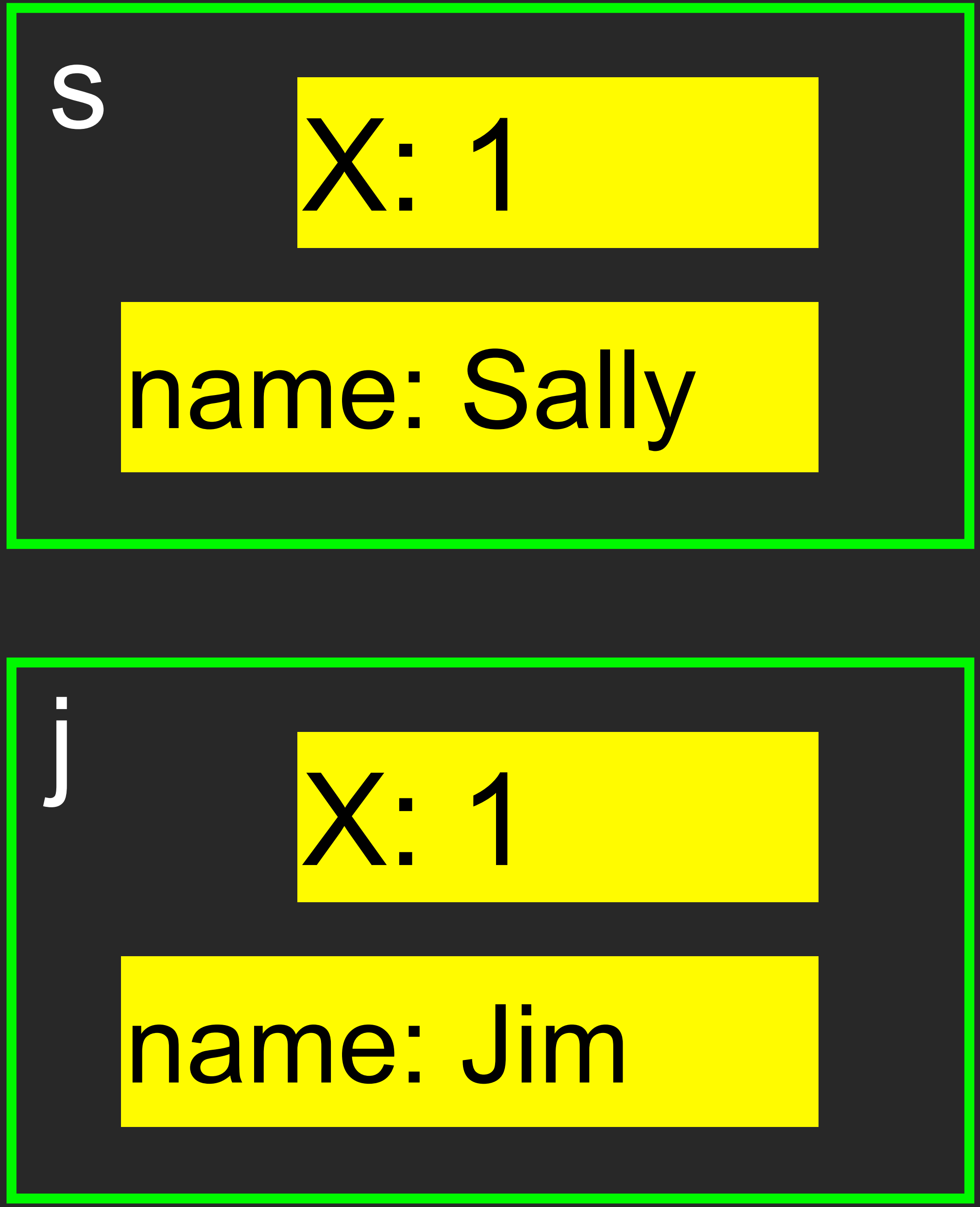
```python
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```



Objects – Part 3

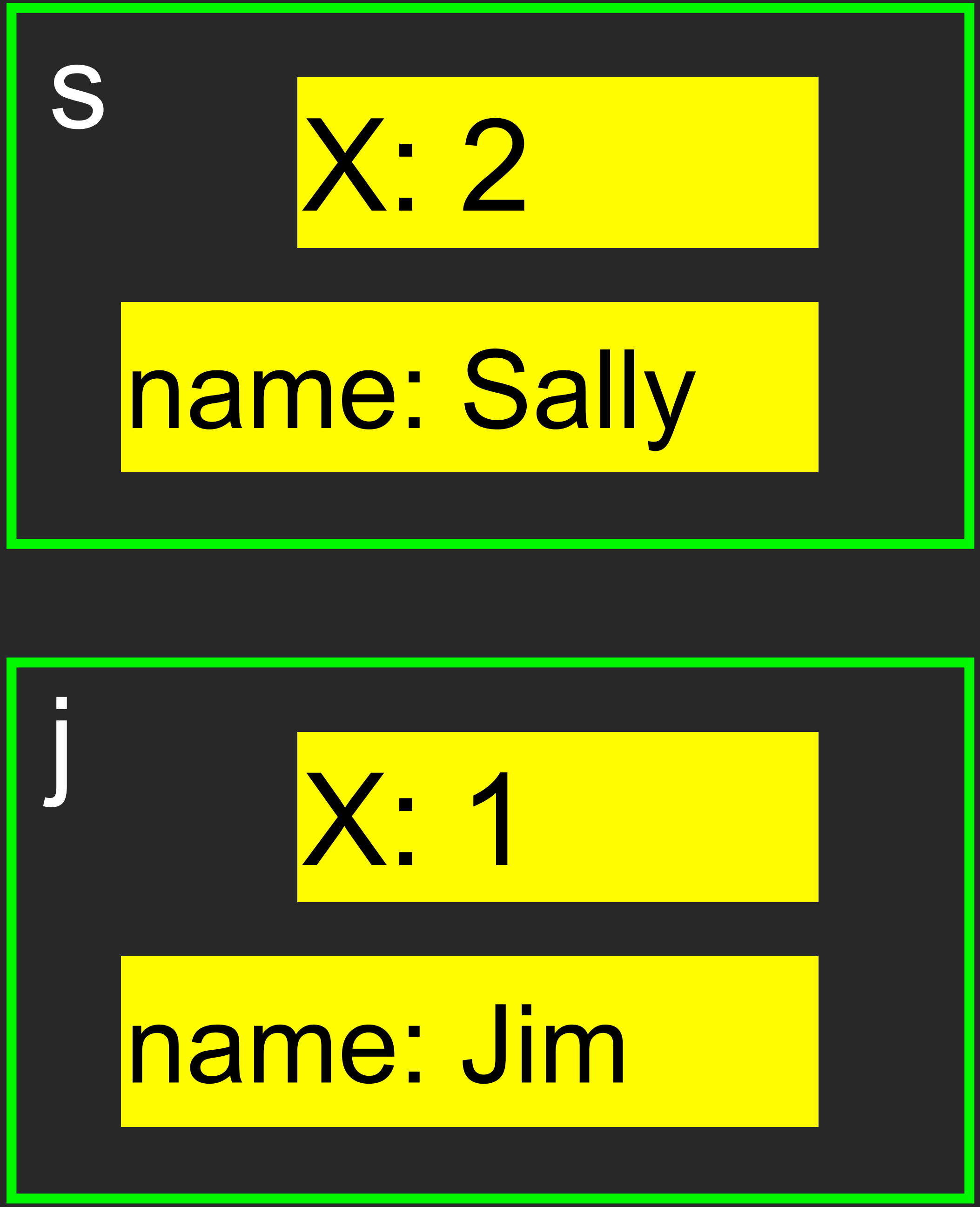
```
class PartyAnimal:

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

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

s = PartyAnimal("Sally")
s.party()
j = PartyAnimal("Jim")

j.party()
s.party()
```



Inheritance

http://www.ibiblio.org/g2swap/byteofpython/read/inheritance.html

Acknowledgements / Contributions

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

… Insert new Contributors here

...

Additional Source Information