Object Oriented Programming in Python

- Objects - instance of a class

- Class - description of what its instances will know and do

How to write a OOP program in Python?

```
class Circle(object):
```
→ class

```
def __init__(self, circle, radius):
    self.circle = circle
    self.radius = radius
```
→ constructor

```
def draw(self, canvas):
    rad = self.radius
    :
    :
```
→ method

```
myCircle = Circle([10,30], 20)
```
→ creation of circle object

mycircle.draw → calling method

Note - If instance variable doesn't exist, python will create a new variable

Why use classes at all?

- Classes and objects allow you to define an interface to some object (it's operations) and then use them without know the internals.

- Defining classes helps modularize your program into multiple objects that work together, that each have a defined purpose

→ Getters - Setters -

```
def getName(self):
    return self.name
```

```
def setAge(self, age):
    self.age = age
```

↳ useful to provide data for encapsulation

→ Inheritance

Definition of a class extending student

```
Class Student:
  "A class representing a student."

  def __init__(self,n,a):
      self.full_name = n
      self.age = a

  def get_age(self):
      return self.age
```

```
Class Cs_student (student):
  "A class extending student."

  def __init__(self,n,a,s):
      student.__init__(self,n,a) #Call __init__ for student
      self.section_num = s

  def get_age():    #Redefines get_age method entirely
      print "Age: " + str(self.age)
```

Inheritance of a class

3 constructors