Lecture 15

##By professor grimson

We are defining an abstract data type. Which are like the in built data types: ‘int’, ‘float’

Cartesian points as lists:

p1 = [1,2]

p2 = [2,pi/2] which is polar for [0,2]

We don’t know if the point is in Cartesian or in polar coordinates.

Class: template for creating instances of an object

instances are created are called as functions:

class Cartesian:

pass

cp = Cartesian() ##this is an instance

We can give some characteristics/attributes to the class.

cp.x = 15

cp.y = 16

this will correspond to the Cartesian point (15,16)

now if we print cp then it says that it is an instance of Cartesian

we can print and operate on cp.x and cp.y as if they were normal variables.

keyword ‘is’

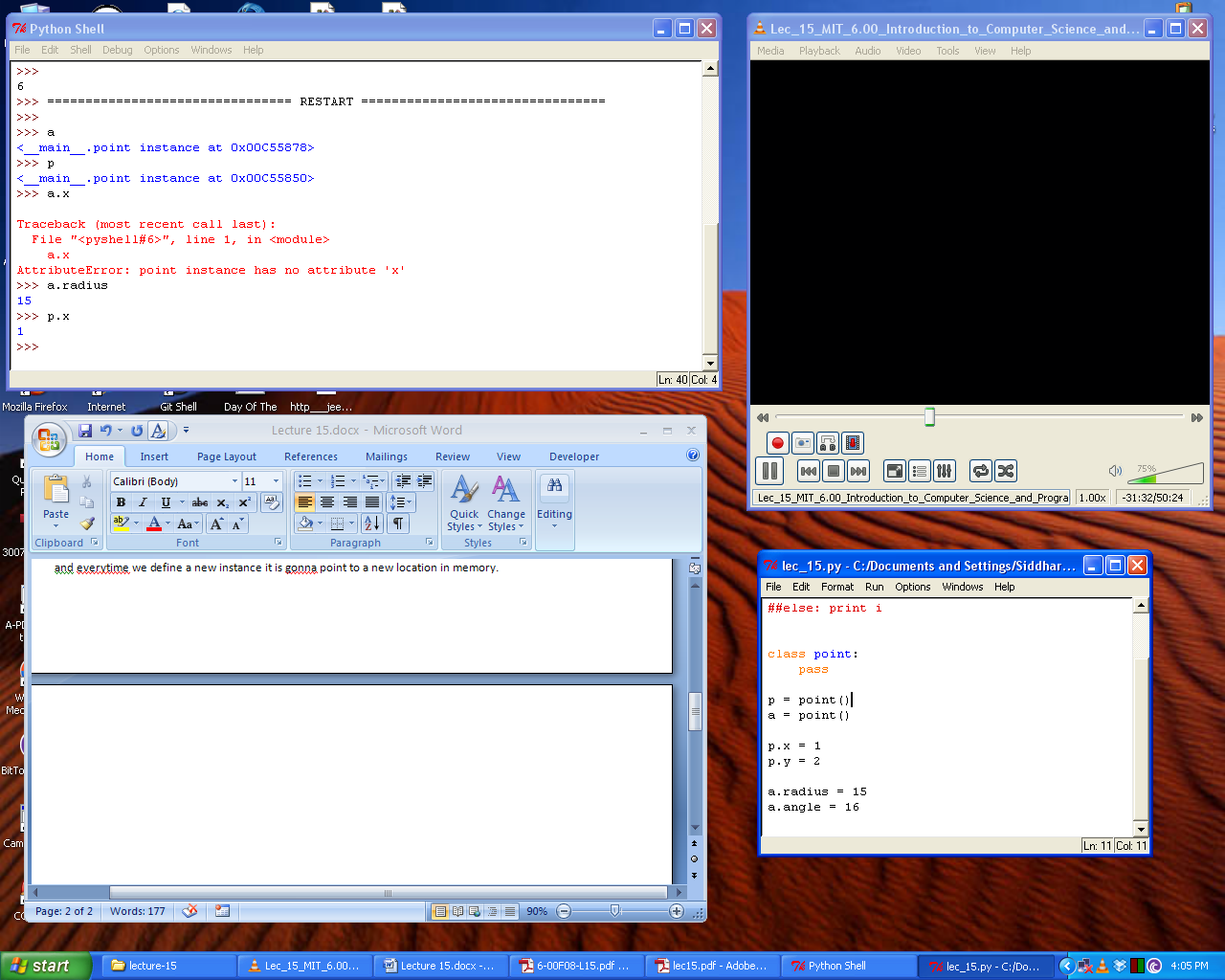
‘is’ does shallow/object equality.

a is b

will be true : if a and b point to the exact same location in memory.

whereas deep/value/set of values equality: we define it.

and everytime we define a new instance it is gonna point to a new location in memory.



here p has attributes x and y

and a has attributes has radius and angle attributes

when we call a class definition then the function \_\_init\_\_ is called.

\_\_init\_\_(self,x,y)

when we call \_\_init\_\_:

* it is going to create an instance
* and it is going to use self to refer to that instance

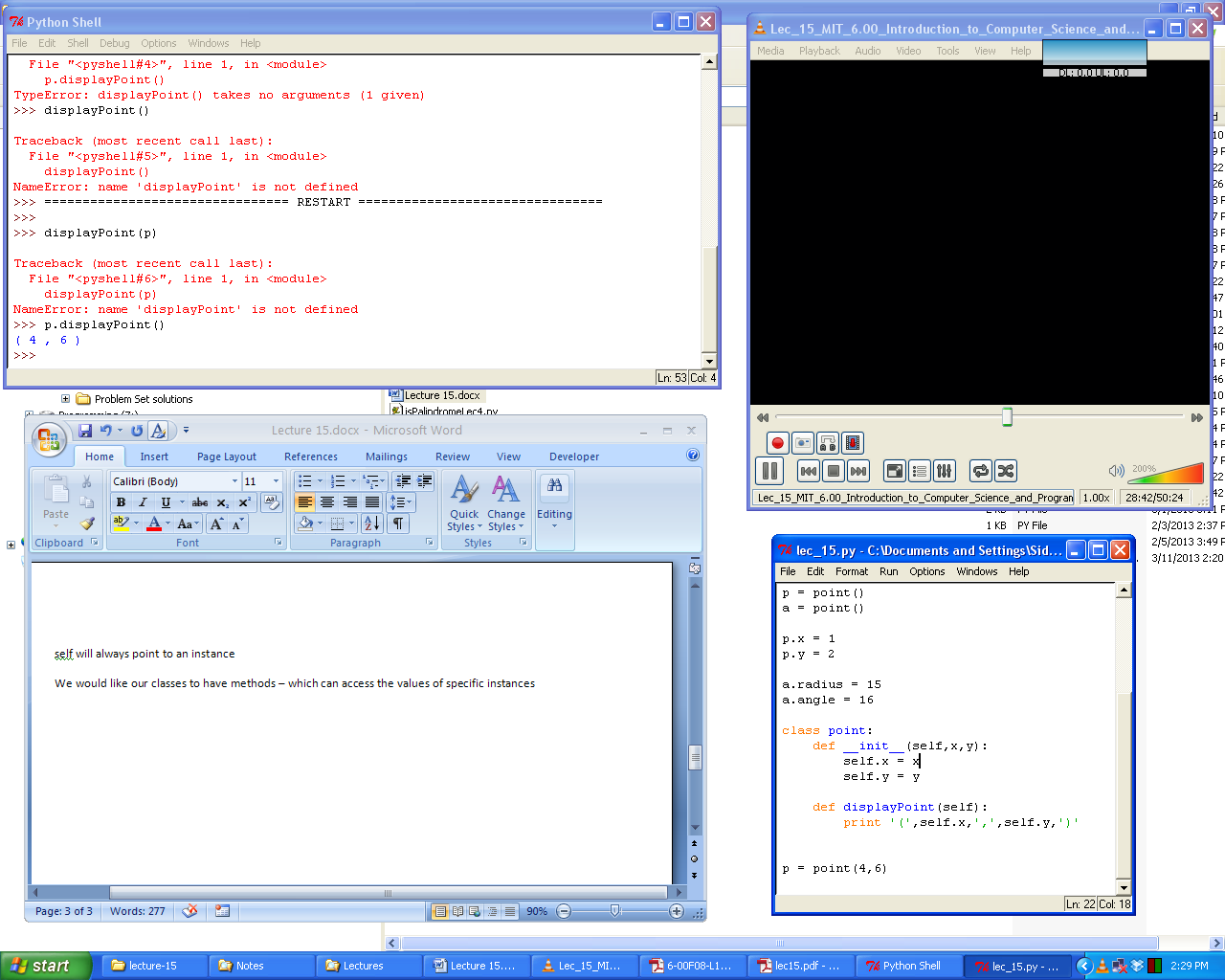
differently:

when a class definition is called

* init creates a pointer to the instance
* and uses init to refer to it

self will always point to an instance

We would like our classes to have methods – which can access the values of specific instances



here p.displayPoint() is an accessor.

the instances of classes are open to the world.

we can change the values of the x and y coordinates of a point to some strings and that really does not make any sense. This is because we don’t have data hiding.

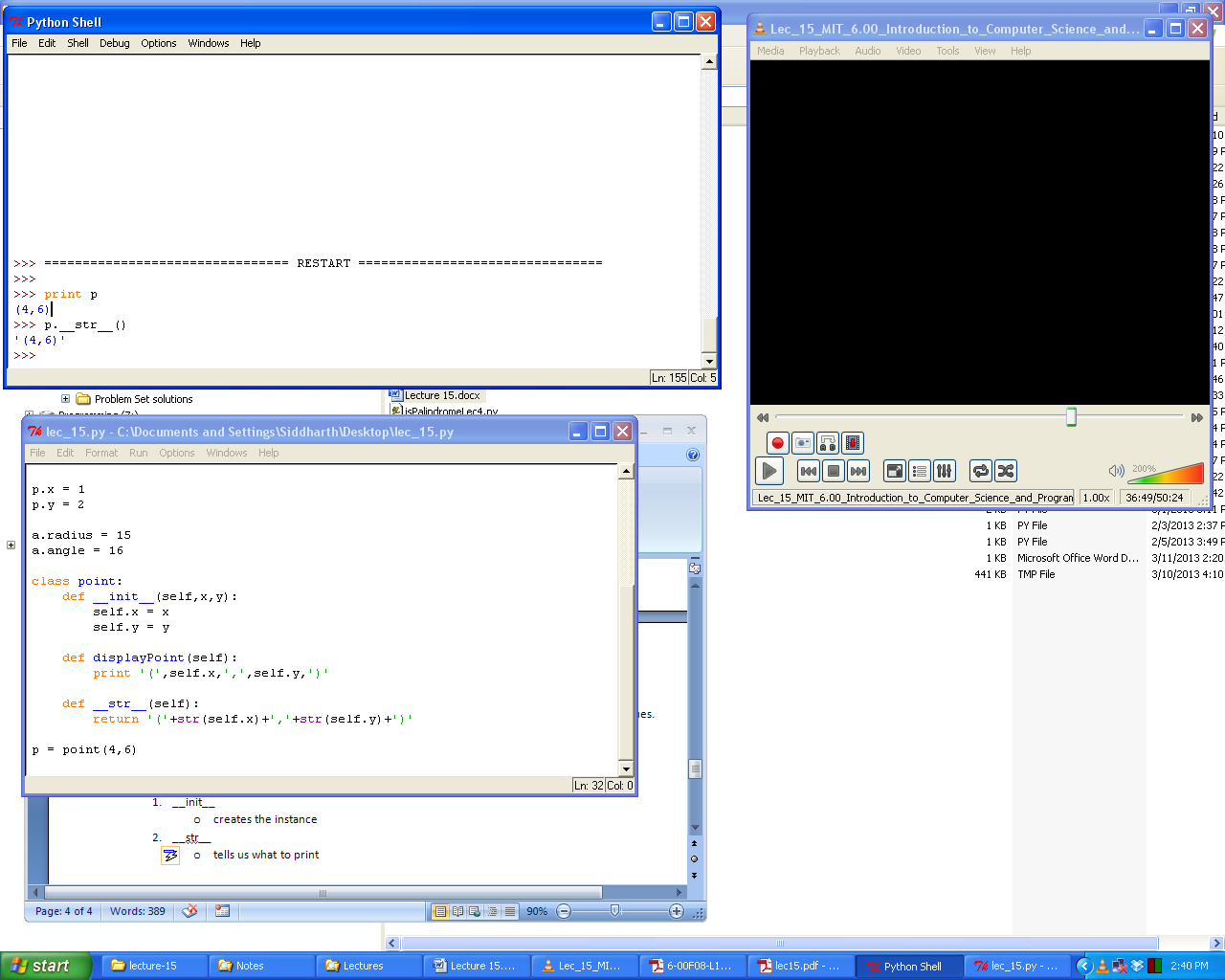
Data hiding: One can only access instance values through defined methods.

**Python does not have DATA HIDING**

So never change the values of the values of an instance using direct access. Always use defined values.

\_\_init\_\_ will define the internal variable or attributes of the instances

METHODS FOR CLASSES IN PYTHON

1. \_\_init\_\_
   * creates the instance
2. \_\_str\_\_
   * tells us what to print
   * 
3. \_\_cmp\_\_
   * does comparisons
4. \_\_eq\_\_
   * are two instances the same?
   * once \_\_eq\_\_ is defined in case that any two objects are compared with the ‘==’ sign then they will be pronounced equal or not based on the definition of this function

What all these methods are doing is OPERATOR OVERLOADING.

inside of an instance:

* methods
* fields

and together these are called attributes.

and when we do: dir(p) where p is an instance then we get a list of all the methods and the fields i.e. all the attributes.