CSE 20 11-18-21

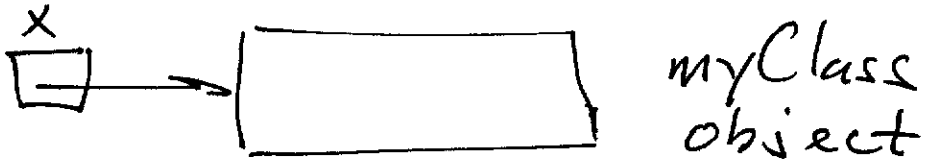
---

classes:

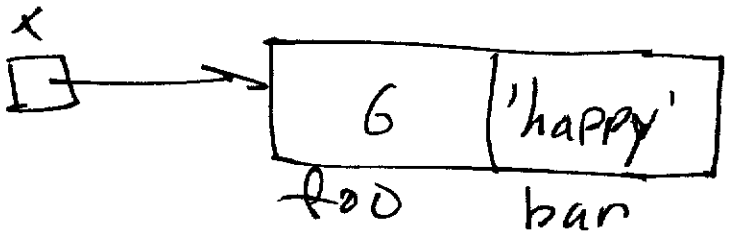
smallest possible

```
class myClass:
    pass
#end class myClass

x = myClass()
```




```
x.foo = 5
x.bar = 'happy
```



- instance methods:

defined:

```
def fcn(self, . , . ):
    :
#end
```

called

```
x = myClass(...)
x.fcn( . , . )
```

- also have class methods (or static methods), later...

- overridden built-in methods

```
__init__()
__str__() ← pretty printing string
__repr__() ← official string representation
__eq__() ← overrides the ==
```

Examples: Point1.py

2

3

4

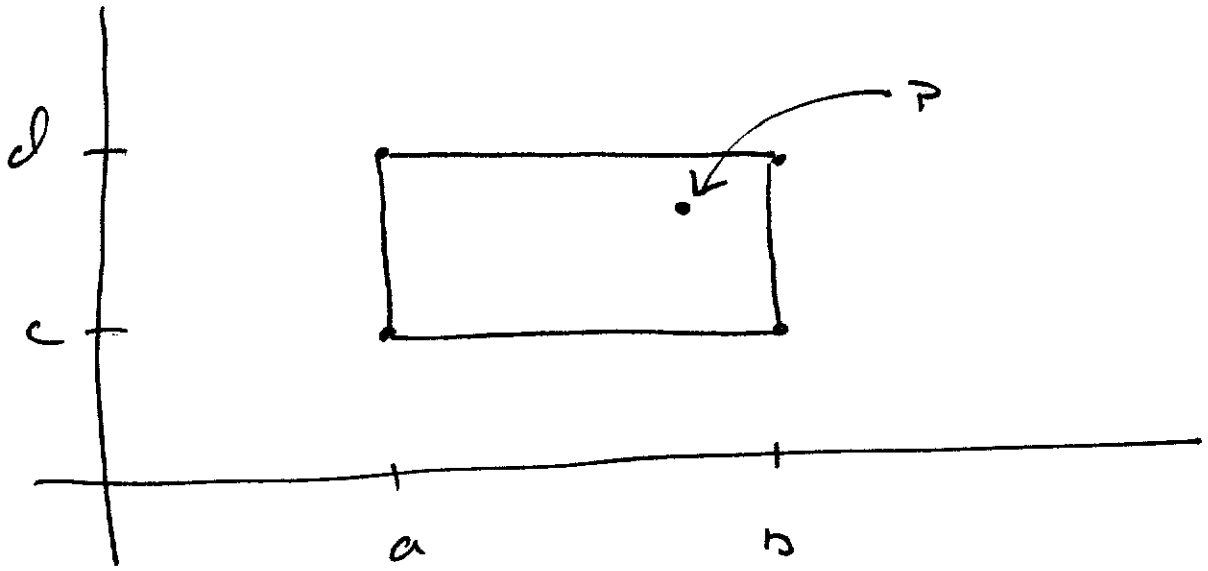
5

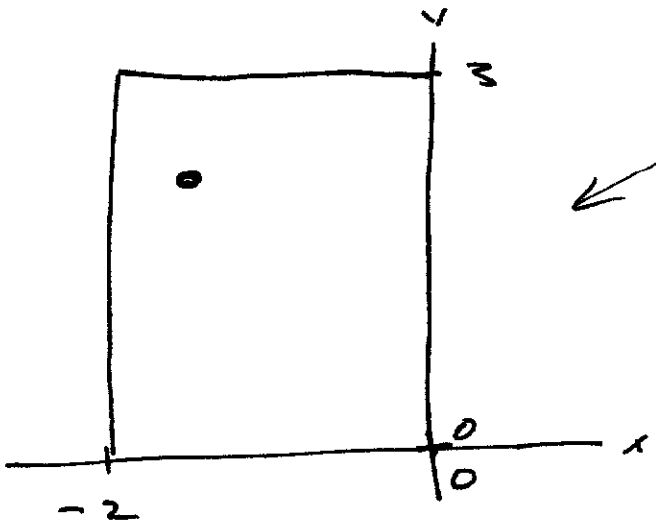
Point6.py

Ex. Point7.py

contains: `__str__()` } override built-in
`__eq__()` }
`sqNorm()` } instance methods
`getCoord()` }
`randPoint()` } class method.

randPoint(a, b, c, d) returns a uniform Point in the rectangle





in Point7.py
main()

Exercise: create Point8.py with a Point class having more functions.

Object Oriented Programming (OOP)

Two points of view concerning functions & data.

- functions as active agents (Procedural Programming)

  Ex. forward(tess, 100)

  apply forward operation to a turtle.

- data as active agent

  Ex. tess.forward(100)

  turtle objects can move forward.

2nd is sometimes more natural

  — cars drive ← OOP

  or

  — drive is applied to car

- Python: class
- Java, C++: class
- C: not directly supported

  can do it with

  struct, and some effort.

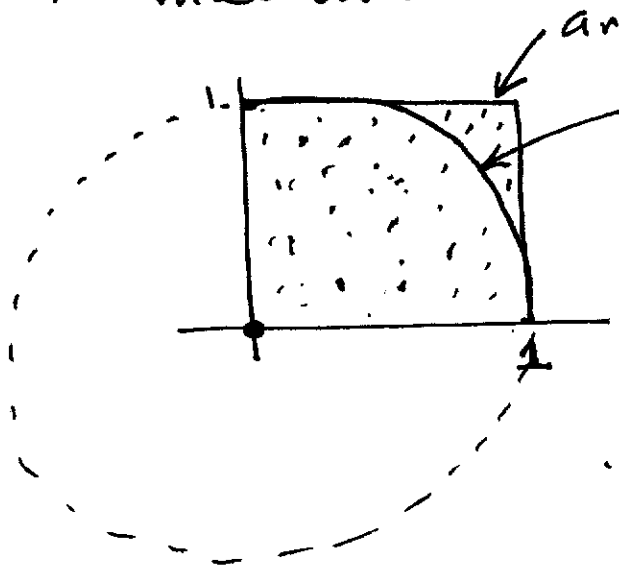
  Take cse13, cse101

Ex. PointTest.py

use Point class in Point7.py

Compute π using monte-carlo.

1st method:



$$\frac{\text{count}}{\text{total}} \approx \frac{\pi/4}{1} = \frac{\pi}{4}$$

$$\therefore \pi \approx 4\left(\frac{\text{count}}{\text{total}}\right)$$