Digital World (2018)

Week 8, S1: Objects and Classes

Chris Poskitt

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Refresher: user-defined data types

b.socrative.com, POSKITT5665

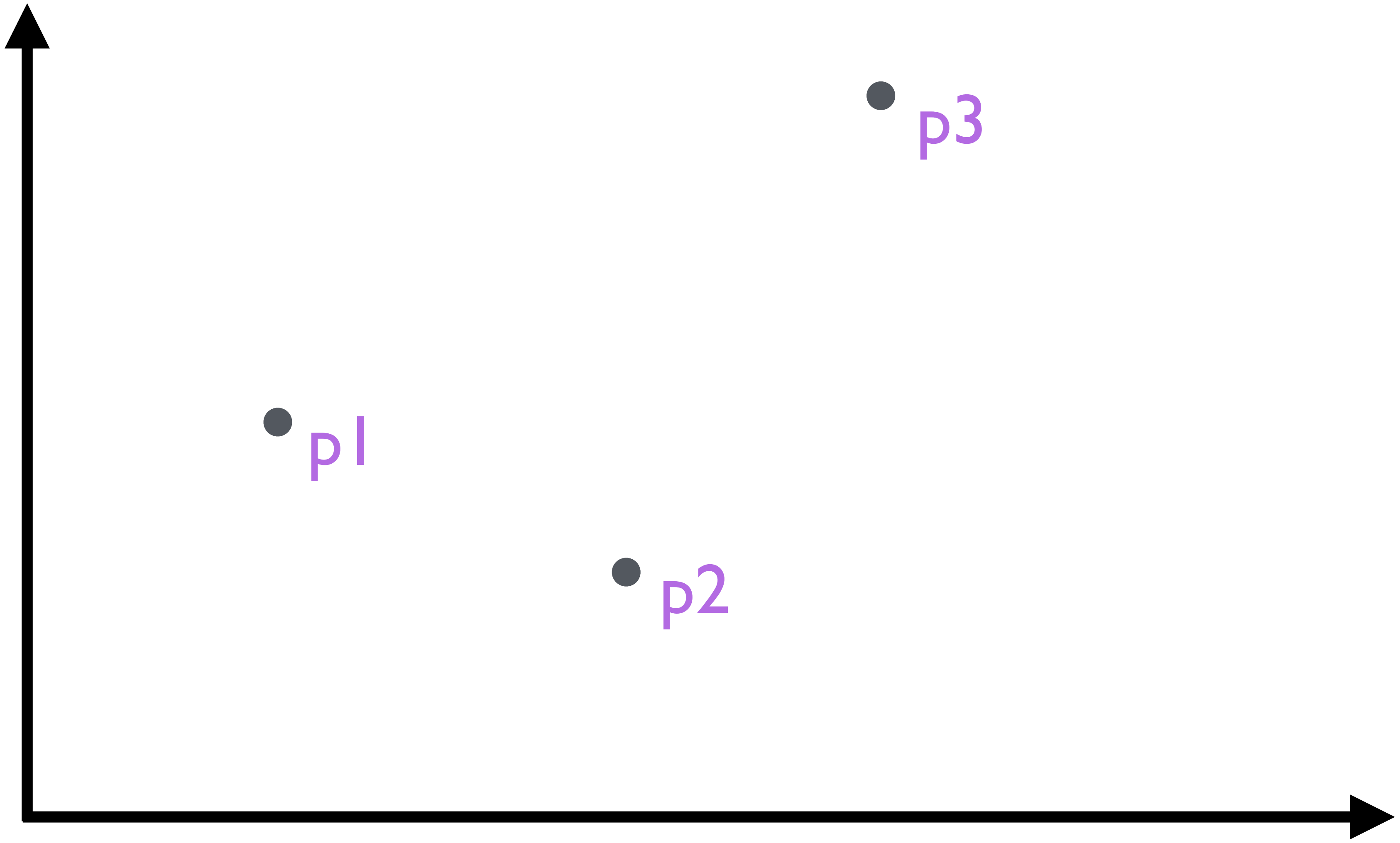
- recall how we structured our data using compound types:

```
class Coordinate:
    x = 3.2
    y = -1.5
```

```
p1 = Coordinate()
p1.x = 0.0
p1.y = p1.y * 2
```

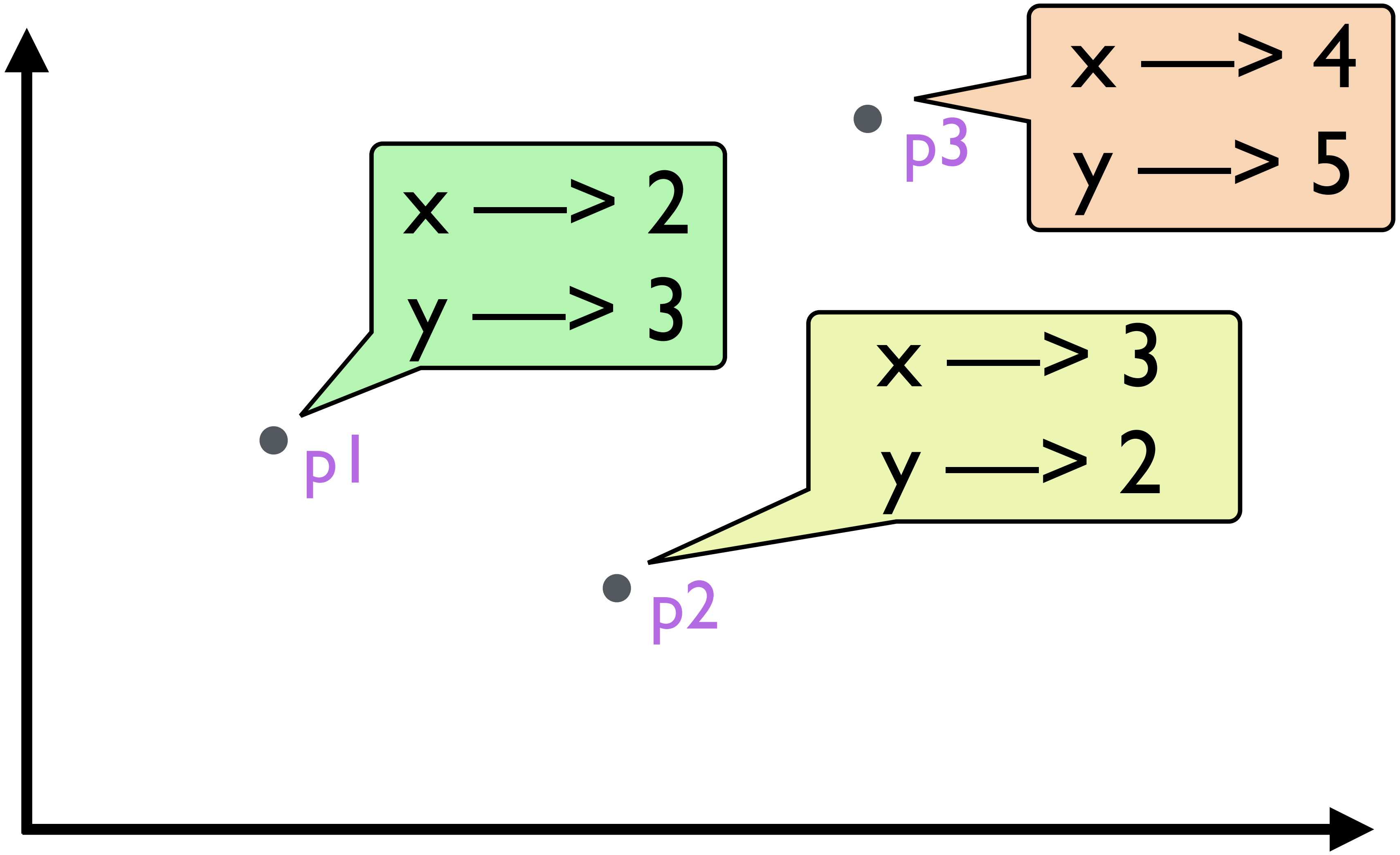
⚠ can we better structure the functions that operate on instances of these data types?

Coordinates — *structuring* *<u>data</u>*



modifying p1.x does not affect p2.x, p3.x, etc.

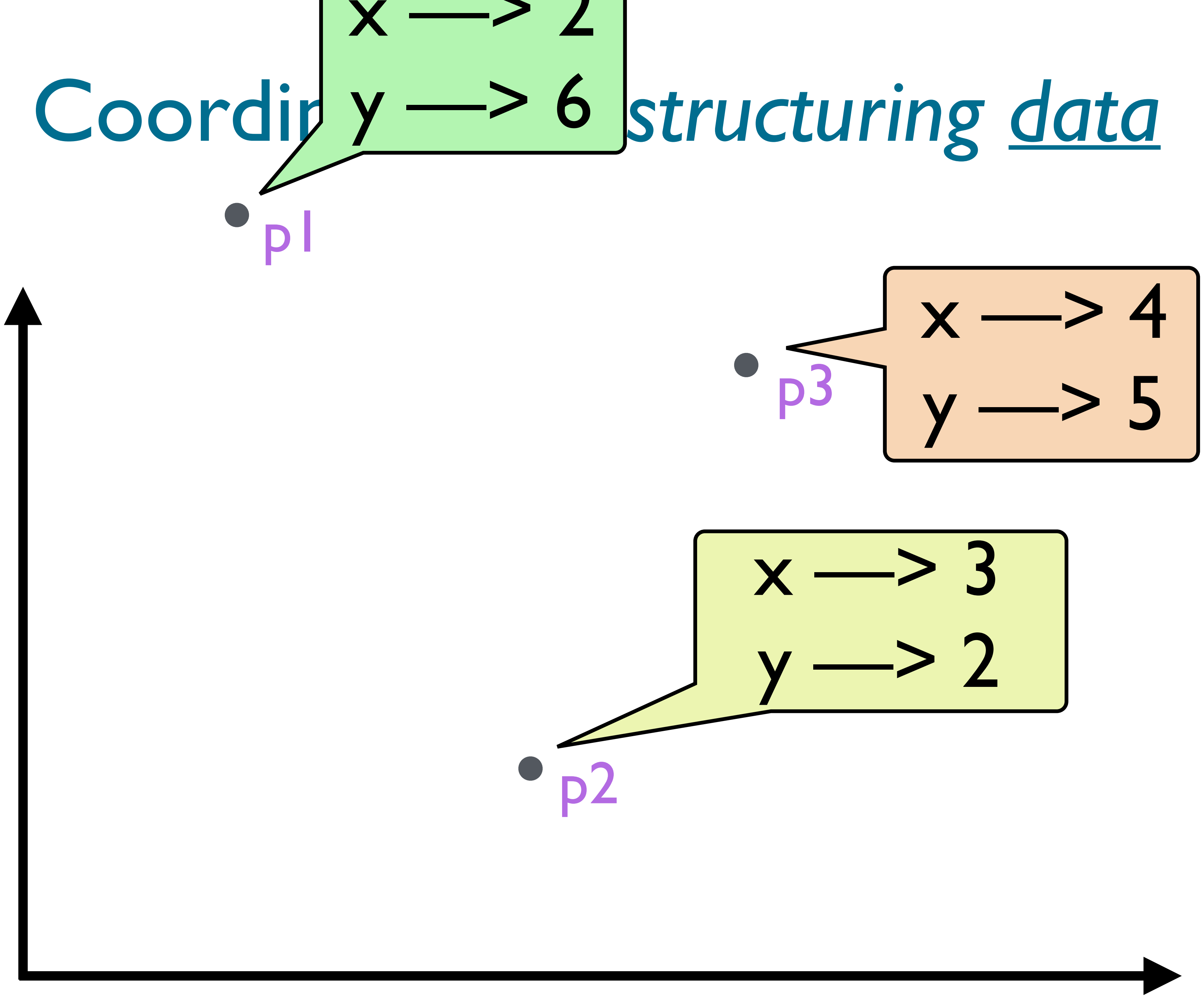
Coordinates — *structuring data*



modifying p1.x does not affect p2.x, p3.x, etc.

```
p1.y = p1.y * 2
```

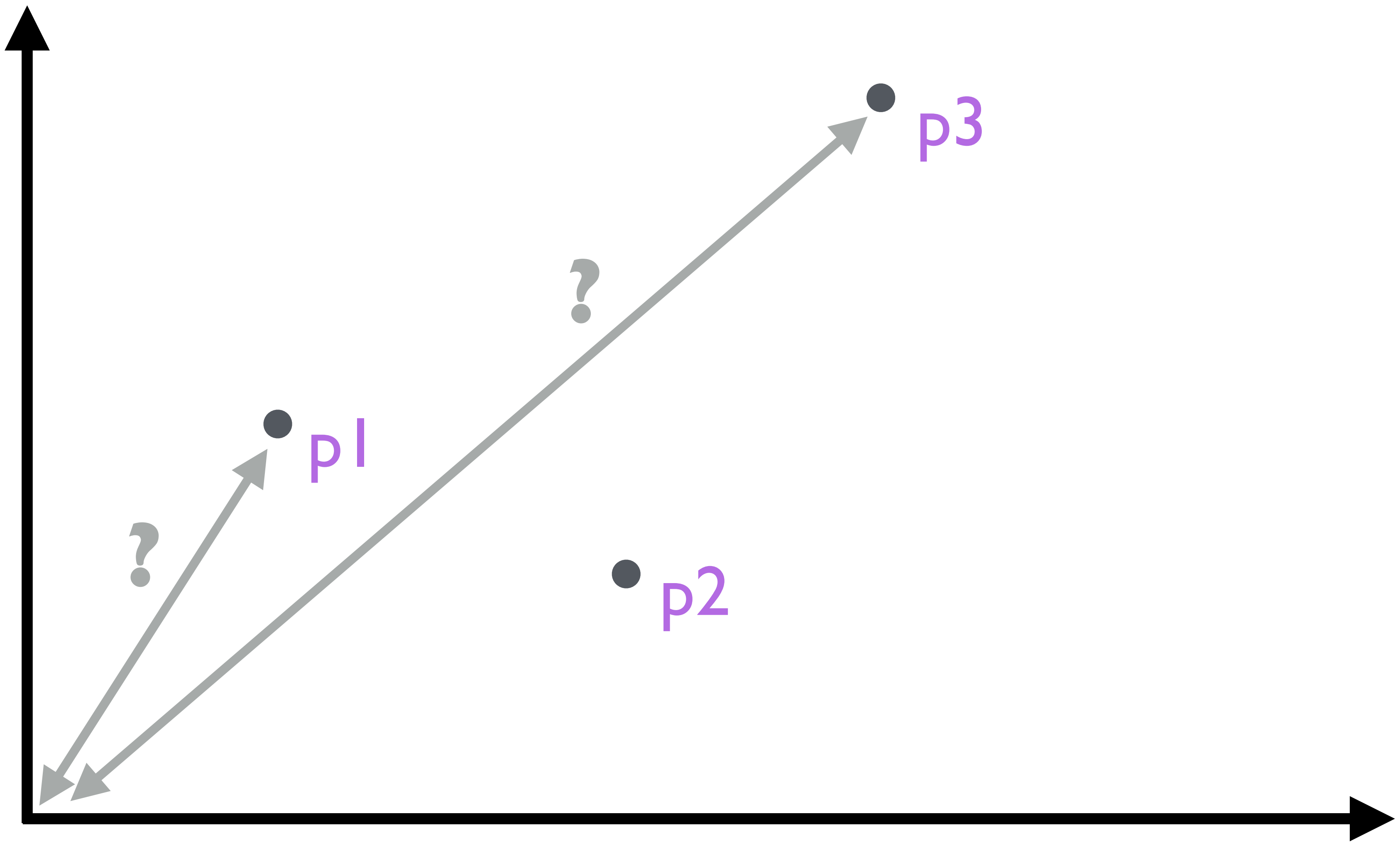
Coordi structuring <u>*data*</u>



modifying p1.x does not affect p2.x, p3.x, etc.

```
p1.y = p1.y * 2
```

Coordinates — *structuring functions…?*



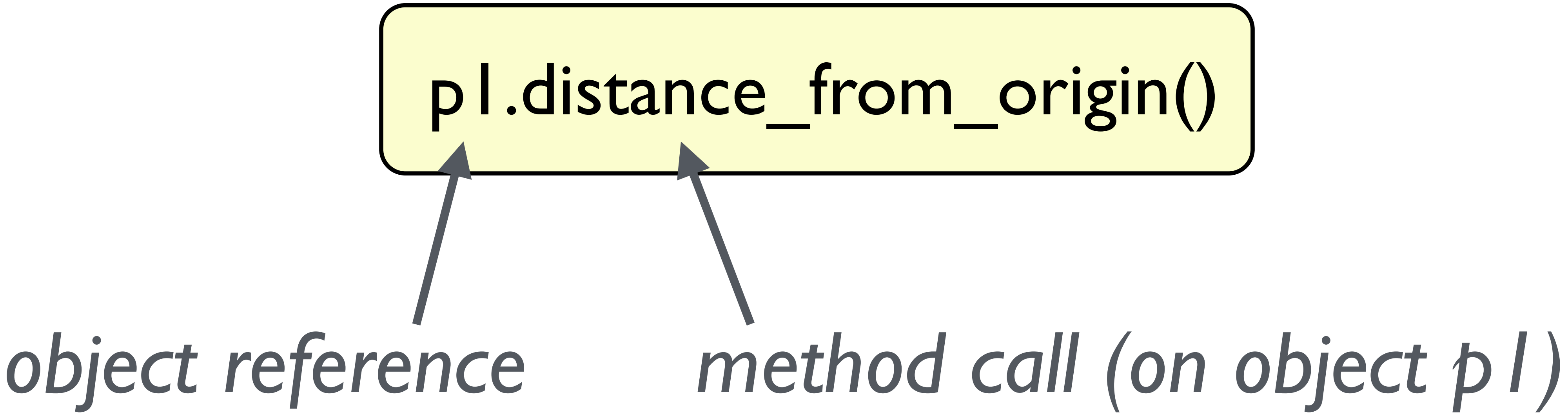
we can define built-in functions ("methods") for Coordinates

```
p1.distance_from_origin()
```

```
p3.distance_from_origin()
```

object = state + behaviour

- objects encapsulate state (data) and behaviour (methods)
- a class is a template from which objects are instantiated
- p1, p2, and p3 are objects of type Coordinate



```
p1.distance_from_origin()
```

object reference

method call (on object p1)

We’ve seen a few objects/methods already

```
“{} is a naughty {}”.format(“johnny”,“boy”)
```

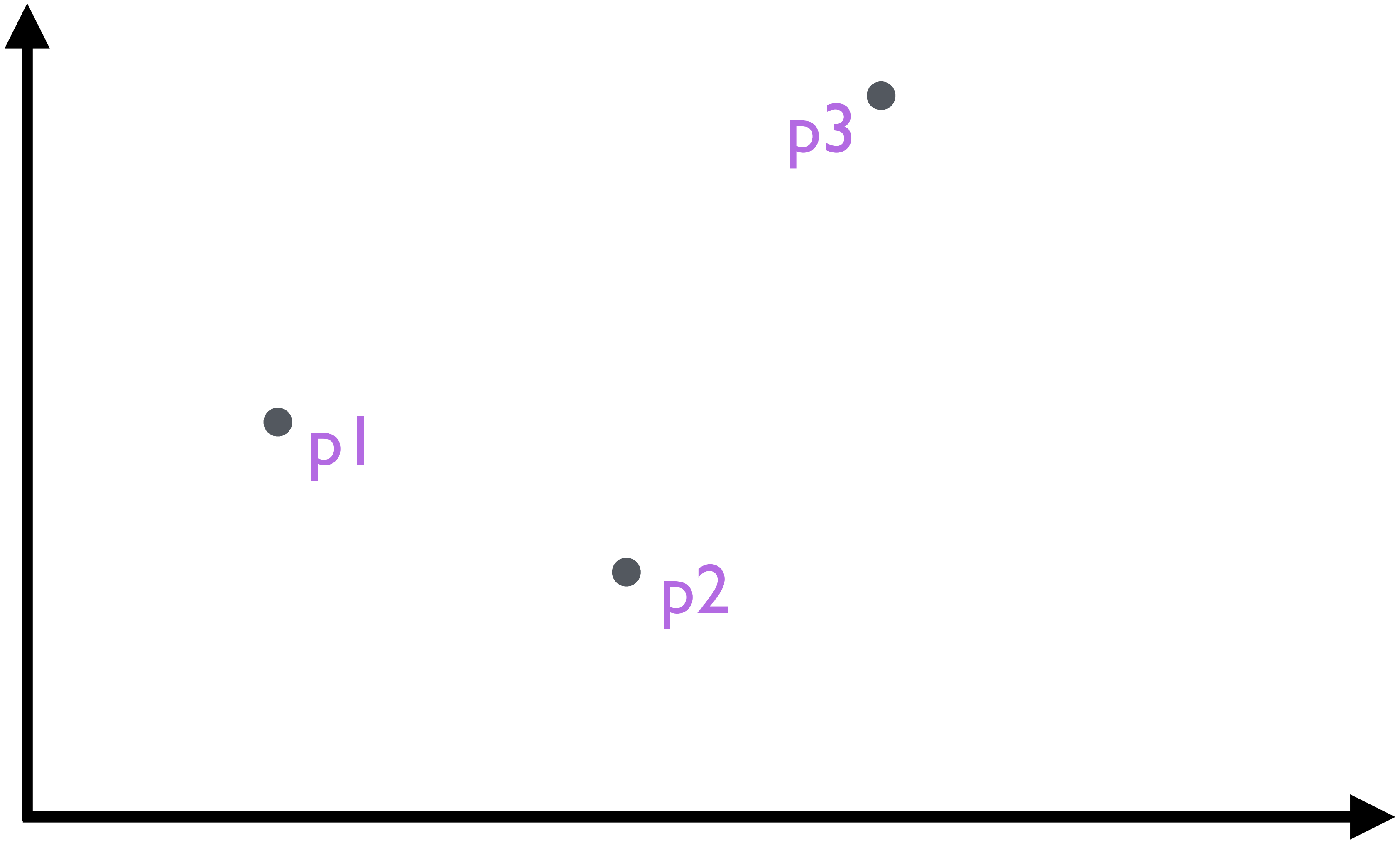
```
fav_dishes_list.append(“sambal tempoyak”)
```

```
robot = ThymioReal()
robot.wheels(100, 100)
```

```
f = open(“fav_food.txt”,“r”)
print(f.readline())
```

Object equality

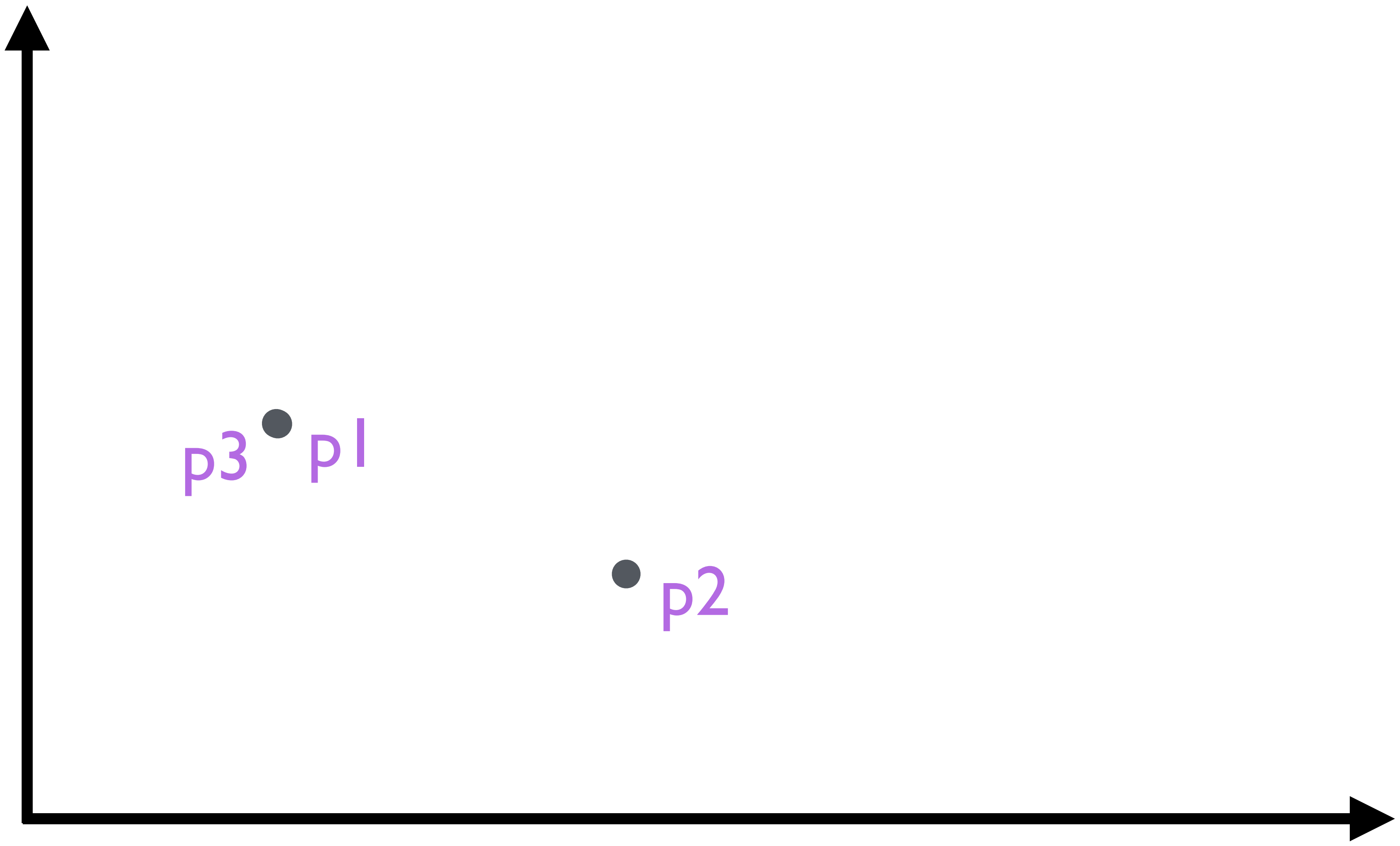
(same same but different)



```
p1 is p3
```

Object equality

(same same but different)



p1 is p3

We can define “equivalence” in our class

- the equivalence method must be named __eq__

- given two objects p1 and p2, Python translates:

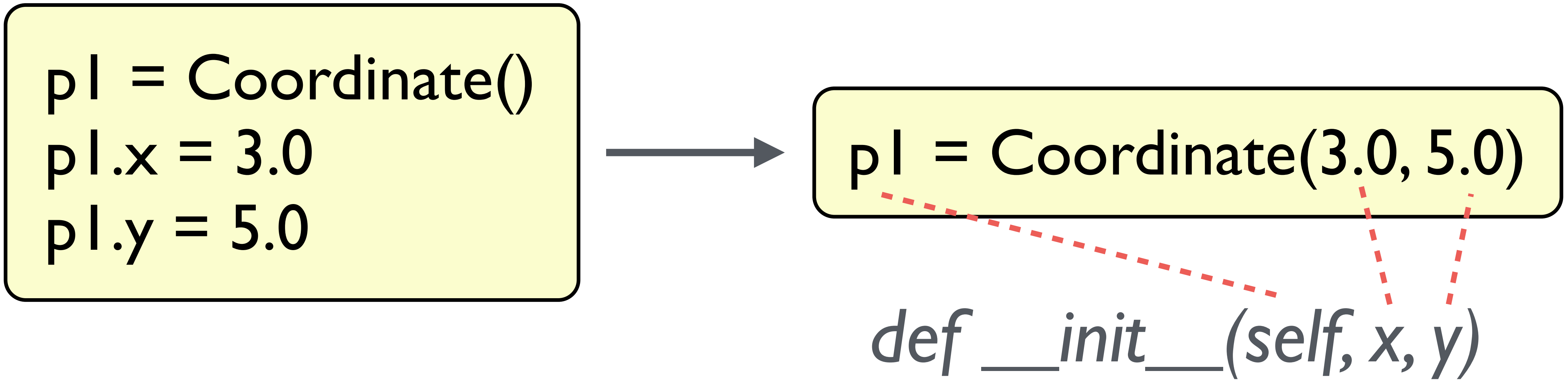
```
p1 == p2
```

into:

```
p1.__eq__(p2)
```

Activity: have a go at Question CS1

- note that the class has another special method, __init__
- this function is called a constructor
- it is called automatically to initialise new objects:



```
p1 = Coordinate()
p1.x = 3.0
p1.y = 5.0
```

```
p1 = Coordinate(3.0, 5.0)
```

def __init__(self, x, y)

Summary

- the object-oriented paradigm helps structure your program to better model some part of the world

- objects encapsulate both state (data) and behaviour (methods)

- objects are instantiated from “templates” called classes

- classes can define a number of special methods

  => *constructor methods (__init__)*
  => *comparison methods (__eq__)*
  => *type conversion methods (__str__)*