## Designing a Point

<table>
<thead>
<tr>
<th>Class Point</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes:</strong></td>
</tr>
<tr>
<td>x: float</td>
</tr>
<tr>
<td>y: float</td>
</tr>
<tr>
<td><strong>Functions:</strong></td>
</tr>
<tr>
<td>distance ():</td>
</tr>
<tr>
<td>find_midpoint ():</td>
</tr>
<tr>
<td>point_within ():</td>
</tr>
</tbody>
</table>
Designing a Point

In two dimensions, a point is two numbers (coordinates) that are treated collectively as a single object. Points are often written in parentheses with a comma separating the coordinates. For example, (0, 0) represents the origin, and (x, y) represents the point x units to the right and y units up from the origin. Some of the typical operations that one associates with points might be calculating the distance of a point from the origin, or from another point, or finding a midpoint of two points, or asking if a point falls within a given rectangle or circle.

To find the Midpoint between two points: Point A and Point B

The midpoint is (x, y) where:

\[ x = \frac{x_1 + x_2}{2} \quad \text{and} \quad y = \frac{y_1 + y_2}{2} \]

We use Pythagoras Theorem to work out AB

\[(AB)^2 = 4^2 + 8^2\]

\[(AB)^2 = 16 + 64\]

\[(AB)^2 = 80\]

\[AB = \sqrt{80} \text{ or } 8.94 \sqrt{\text{v}}\]
Designing the class in PyCharm

Step 2 – define a class API:

- Choose a class name and write a brief description in the class docstring
- Write some examples of client code that uses your class
- Put this code in the “main block”
- Decide what operations your class should provide as public methods, for each method declare an API (examples, type contract, header, description)
- Decide which attributes your class should provide without calling a method, list them in the class docstring
Examples

Built-in objects • int, string, Turtle, etc.

Using Turtle class to draw:

```python
>>> from turtle import Turtle
>>> t = Turtle()
>>> t.pos()
(0.00,0.00)
>>> t.forward(100)
>>> t.pos()
(100.00,0.00)
>>> t.right(90)
>>> t.forward(100)
>>> t.pos()
(100.00,-100.00)
```
Examples

Built-in objects • int, string, Turtle, etc.

Using Turtle class to draw:
>>> from turtle import Turtle
>>> t = Turtle()
>>> t.pos()
(0.00,0.00)
>>> t.forward(100)
>>> t.pos()
(100.00,0.00)
>>> t.right(90)
>>> t.forward(100)
>>> t.pos()
(100.00,-100.00)

Vandalizing the Turtle class (deeply wrong!)
>>> t.neck
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AttributeError: 'Turtle' object has no attribute 'neck'
>>> Turtle.neck = "very reptilian"
>>> t1.neck
'very reptilian'
Wrong way of designing a Point class

```python
>>> class Point:
... pass
...
>>> def initialize(point, x, y):
... point.x = x
... point.y = y
...
>>> def distance(point):
... return (point.x**2 + point.y**2) ** (1 / 2)
...
>>> Point.__init__ = initialize
>>> Point.distance = distance

>>> p2 = Point(12, 5)
>>> p2.distance()
13.0
```
More Functionalities to Point Class

• Implement the class:

• 1. Body of special methods: __init__, __eq__, __str__, __add__ (if the object should act like a numeric entity)

Note: Python provides special methods:

__init__, __str__,
__eq__, __ne__, __lt__, __gt__, __le__, __ge__,
__add__, __mul__, etc.
Interesting aspects of Python

Methods can be invoked in two equivalent ways:

- \( p = \text{Point}(3, 4) \)
- \( p.\text{distance\_to\_origin()} \)
- 5.0
- \( \text{Point}.\text{distance\_to\_origin}(p) \)

In both, the first parameter (self) refers to the instance named \( p \)
Interesting aspects of Python

What if I try these?

- print Point.x
- Point.y = 17

- Class namespaces vs object namespaces
  (using __dict__, check the example)
Practice more

Develop other methods yourselves

• Keep in mind the docstring contract!

• Practice coding!

• Simply understanding these examples is not enough!

• Did I mention practice?