OOP Practice

Let’s use some Point objects to make a Line!
Get ready to write code on paper / a tablet!

Announcements

CQ04: OOP Code-Writing due tonight at 11:59pm

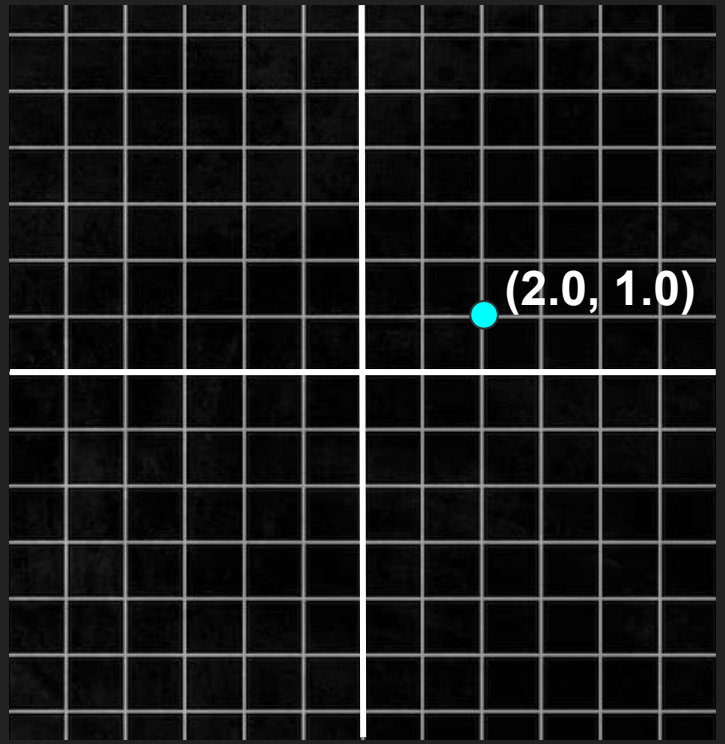
EX06: River Simulation due Thursday, 11/06 at 11:59pm

Consider the following Point class and objects:

```python
class Point:
    x: float
    y: float

    def __init__(self, x: float, y: float):
        self.x = x
        self.y = y

    def dist_from_origin(self) -> float:
        return (self.x**2 + self.y**2) ** 0.5

    def translate_x(self, dx: float) -> None:
        self.x += dx


p0: Point = Point(10.0, 0.0)
p0.translate_x(-5.0)
print(p0.dist_from_origin())
```

Consider this Point class

```
0 class Point:
1     x: float
2     y: float
3
4     def __init__(self, x: float, y: float):
5         self.x = x
6         self.y = y
7
8     def dist_from_origin(self) -> float:
9         return (self.x**2 + self.y**2) ** 0.5
10
11    def translate_x(self, dx: float) -> None:
12        self.x += dx
13
14    def translate_y(self, dy: float) -> None:
15        self.y += dy
16
17 pt: Point = Point(2.0, 1.0)
```



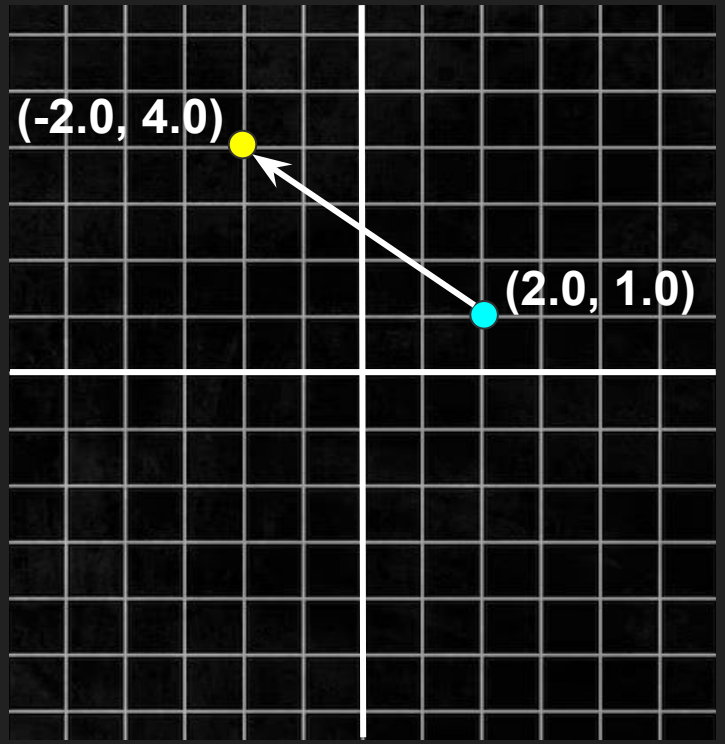
“Two points make a line”

Finding the length of a line:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Finding the slope of a line:

$$m = \frac{\text{Rise}}{\text{Run}} = \frac{y_2 - y_1}{x_2 - x_1}$$

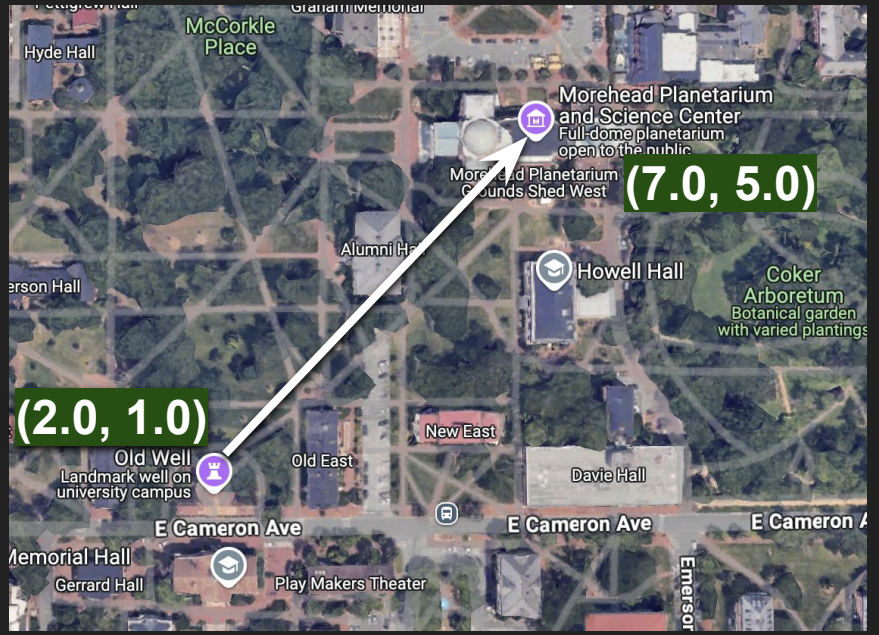


“Two points make a line”

Let’s define a Line class and use it to see the distance from the Old Well to the Planetarium!

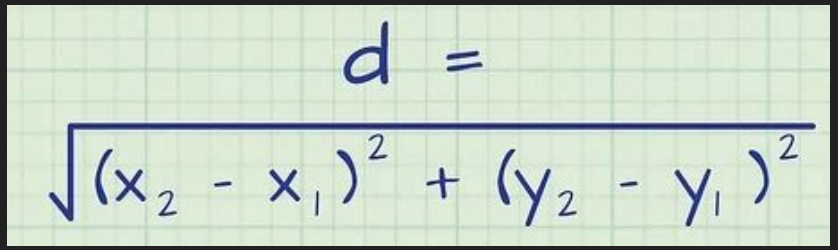
Create a **`Line`** class with two attributes: a starting point (**`start: Point`**) and an ending point (**`end: Point`**).
The **`Line`** class should have the following method definitions:

- `def __init__(self, start: Point, end: Point):`
- `def get_length(self) -> float:`
  - Calculates the length of the line
- `def get_slope(self) -> float:`
  - Calculates the slope (from **`start`** to **`end`**)

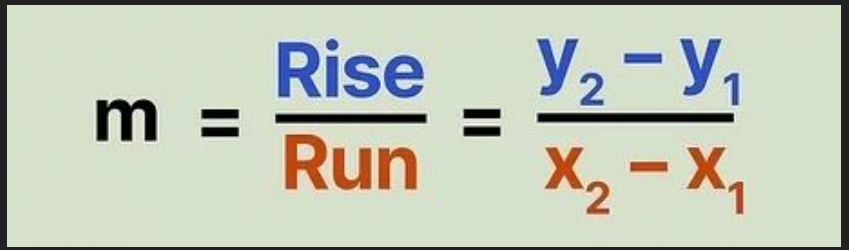


“Two points make a line” – Let’s make a Line class!

Finding the length of a line:



Finding the slope of a line:

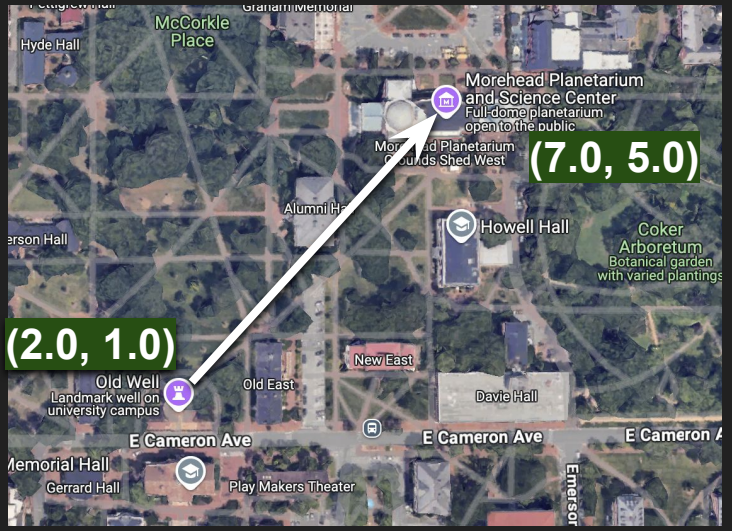


Step 1: Create a `Line` class with two attributes: a starting point (`start: Point`) and an ending point (`end: Point`).
The `Line` class should have the following method definitions:

- **Step 2:** `def __init__(self, start: Point, end: Point):`
- **Step 3:** `def get_length(self) -> float:` calculates the length of the line
- **Step 4:** `def get_slope(self) -> float:` calculates the slope (from `start` to `end`)

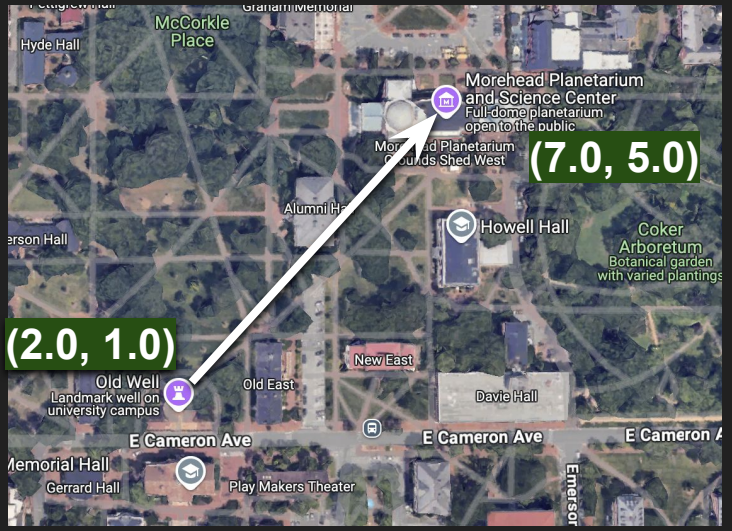
Let’s go over it together! →

```
1-16 class Point: … # collapsed for space
  17
  18 class Line:
  19     start: Point
  20     end: Point
  21
  22     def __init__(self, start: Point, end: Point):
  23         self.start = start
  24         self.end = end
  25
  26     def get_length(self) -> float:
  27         x_diffs: float = self.end.x - self.start.x
  28         y_diffs: float = self.end.y - self.start.y
  29         return (x_diffs**2 + y_diffs**2) ** 0.5
  30
  31     def get_slope(self) -> float:
  32         x_diffs: float = self.end.x - self.start.x
  33         y_diffs: float = self.end.y - self.start.y
  34         return y_diffs / x_diffs
```



Create a Line object and find the distance from the Old Well to the Planetarium:

```python
class Point: … # collapsed for space

class Line:
    start: Point
    end: Point

    def __init__(self, start: Point, end: Point):
        self.start = start
        self.end = end

    def get_length(self) -> float:
        x_diffs: float = self.end.x - self.start.x
        y_diffs: float = self.end.y - self.start.y
        return (x_diffs**2 + y_diffs**2) ** 0.5

    def get_slope(self) -> float:
        x_diffs: float = self.end.x - self.start.x
        y_diffs: float = self.end.y - self.start.y
        return y_diffs / x_diffs
```



Create a Line object and find the distance from the Old Well to the Planetarium:

```python
me: Point = Point(2.0, 1.0)
planet_y: float = 5.0
planet_loc: Point = Point(me.x + 5, planet_y) # Example of accessing an attribute's value
path: Line = Line(me, planet_loc)
print(path.get_slope())
```

Want extra practice? Try diagramming this!

```
1-16 class Point: … # collapsed for space
  17
  18 class Line:
  19     start: Point
  20     end: Point
  21
  22     def __init__(self, start: Point, end: Point):
  23         self.start = start
  24         self.end = end
  25
  26     def get_length(self) -> float:
  27         x_diffs: float = self.end.x - self.start.x
  28         y_diffs: float = self.end.y - self.start.y
  29         return (x_diffs**2 + y_diffs**2) ** 0.5
  30
  31     def get_slope(self) -> float:
  32         x_diffs: float = self.end.x - self.start.x
  33         y_diffs: float = self.end.y - self.start.y
  34         return y_diffs / x_diffs
  35
  36 me: Point = Point(2.0, 1.0)
  37 planet_y: float = 5.0
  38 planet_loc: Point = Point(me.x + 5, planet_y)
  39 path: Line = Line(me, planet_loc)
  40 print(path.get_slope())
```