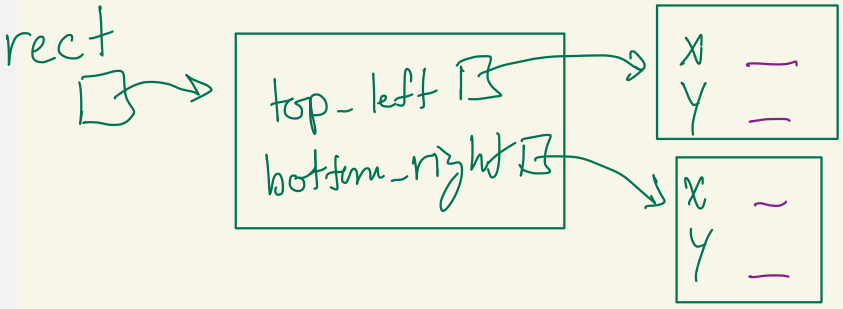
CSC 101

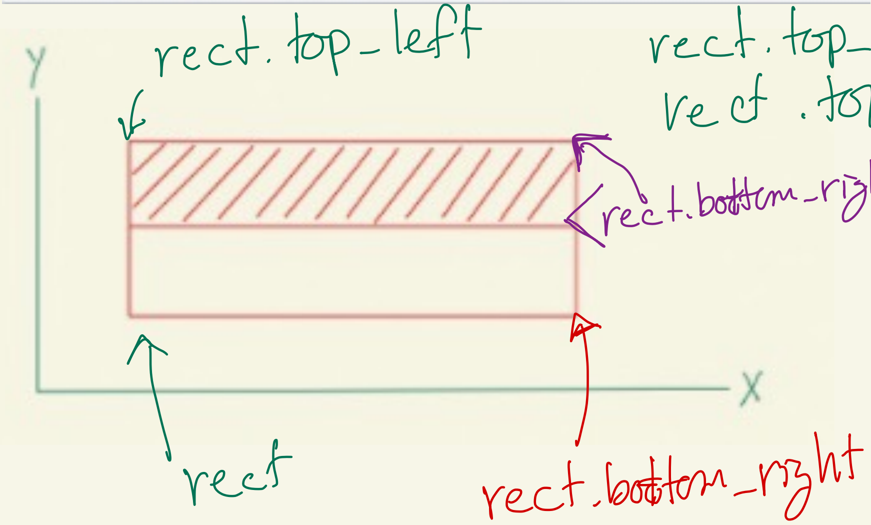
A New Day!

Today

- Quiz 2 Reflections
  - adapting
  - active learning
  - lab/assignment reflection
  - Growth Mindset

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

# an axis-aligned rectangle with top_left as the top-left point
# and bottom_right as the bottom right point
class Rectangle:
    def __init__(self, top_left: Point, bottom_right: Point):
        self.top_left = top_left
        self.bottom_right = bottom_right
```





rect.top_left.x
rect.top_left.y

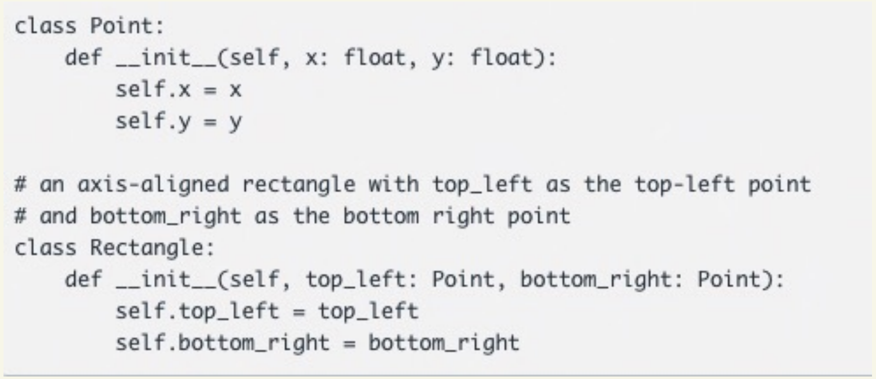
(rect.bottom_right.x, rect.top_left.y)

p1 = Point(2,3)
p2 = Point(4,1)

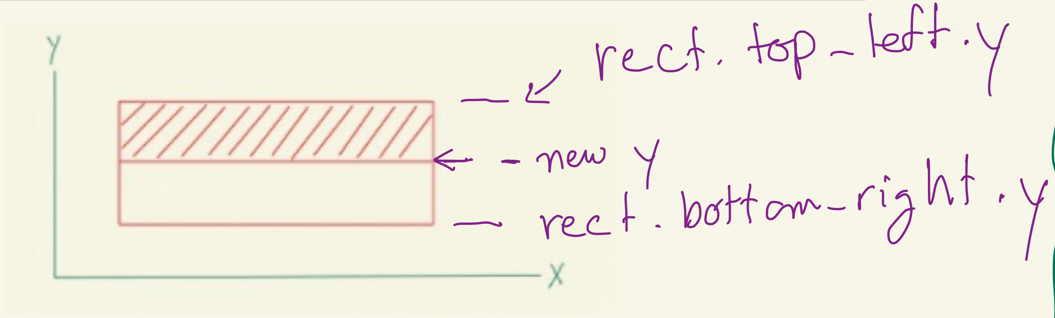
rect = Rectangle(p1, p2)

. -- -

def split(rect: Rectangle)
    -> Rectangle:

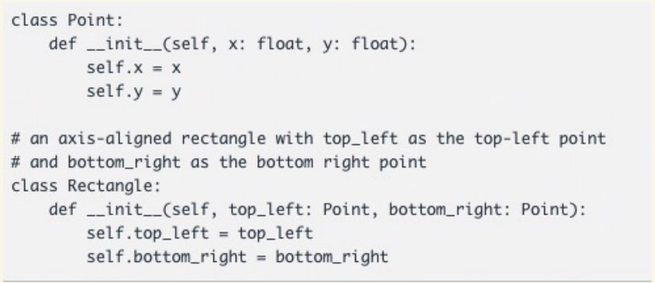


```
class Point:
    def __init__(self, x: float, y: float):
        self.x = x
        self.y = y

# an axis-aligned rectangle with top_left as the top-left point
# and bottom_right as the bottom right point
class Rectangle:
    def __init__(self, top_left: Point, bottom_right: Point):
        self.top_left = top_left
        self.bottom_right = bottom_right
```

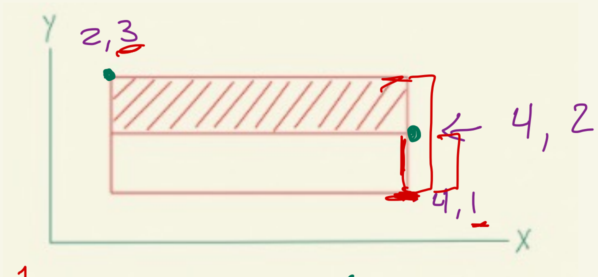


```
top_y = rect.top_left.y
bot_y = rect.bottom_right.y
mid_y = (bot_y +
         (top_y - bot_y)/2)

mid_pt = Point(
    rect.bottom_right.x,
    mid_y
)
```

```
new_y = rect.bottom_right.y + (rect.top_left.y - rect.bottom_right.y)/2
```



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

# an axis-aligned rectangle with top_left as the top-left point
# and bottom_right as the bottom right point
class Rectangle:
    def __init__(self, top_left: Point, bottom_right: Point):
        self.top_left = top_left
        self.bottom_right = bottom_right
```



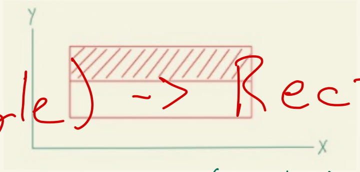
$(3-1)/2$

$\text{mid\_y} = \underline{\text{rect}}.\overset{1}{\text{bottom\_right}}.\text{y} + (\text{rect}.\overset{3}{\text{top\_left}}.\text{y} - \text{rect}.\overset{1}{\text{bottom\_right}}.\text{y})/2$

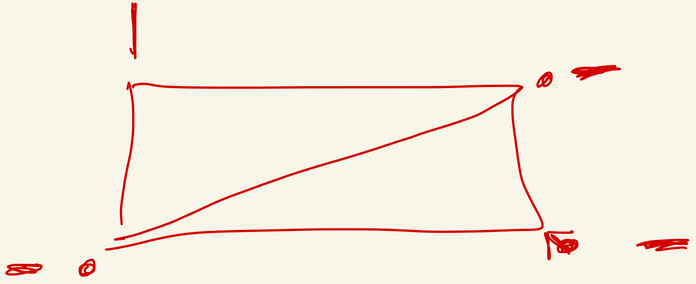
mid_pt = Point(rect.bottom_right.x, mid_y)

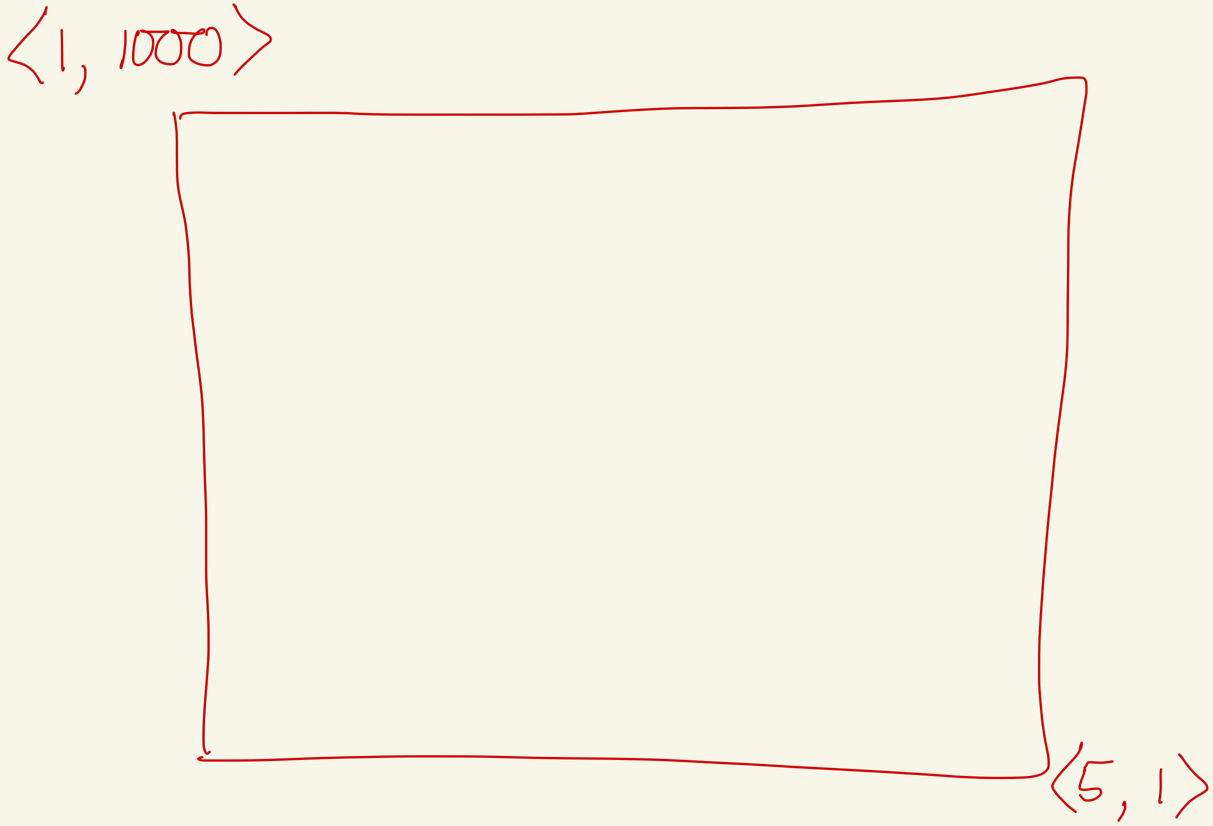
new_rect = Rectangle(rect.top_left, mid_pt)

```
def addOne(x : float) -> float :
    return x + 1
```

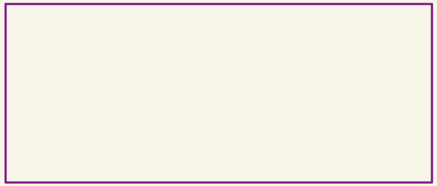


```
def split(rect : Rectangle) -> Rectangle :
    mid_y = rect.bottom_right.y + (rect.top_left.y
                                   - rect.bottom_right.y)/2
    mid_pt = Point(rect.bottom_right.x, mid_y)
    new_rect = Rectangle(rect.top_left, mid_pt)

    return new_rect
```




⟨1, 1000⟩
⟨5, 1⟩

Point (4, 1), Point (2, 9)



Rectangle(pt1, pt2)

```
def malformed(rect: Rectangle) -> bool:

    return (rect.top_left.x >= rect.bottom_right.x
                or
            rect.bottom_right.y >= rect.top_left.y)
```