shapes_driver

Rectangle and Triangle classes

Here's a bird's eye view of the Rectangle and Triangle classes which are defined in the shapes.py script file.

Discuss:

- Blueprint versus instance
- "underscore stuff"
- attributes
- properties
- object level vs. class level methods

```
+-----------------------------------------------------------+
|            Rectangle                                      |
+-----------------------------------------------------------+
| - width: float                                            |
| - height: float                                           |
+-----------------------------------------------------------+
| + __init__(width: float, height: float)                   |
| + area: float {property}                                  |
| + perimeter: float {property}                             |
| + is_square() -> bool                                     |
| + square(side_length: float) -> Rectangle {classmethod}   |
| + unit_square() -> Rectangle {staticmethod}               |
| + __str__() -> str                                        |
| + __repr__() -> str                                       |
+-----------------------------------------------------------+
```

```
+-------------------------------------------------------------------------+
|                   Triangle                                              |
+-------------------------------------------------------------------------+
| - base: float                                                           |
| - height: float                                                         |
+-------------------------------------------------------------------------+
| + __init__(base: float, height: float)                                  |
| + area: float {property}                                                |
| + area_from_sides(a: float, b: float, c: float) -> Triangle {classmethod} |
| + __str__() -> str                                                      |
| + __repr__() -> str                                                     |
+-------------------------------------------------------------------------+
```

```
#importing the code
from shapes import Rectangle  #we see shapes printed because of
                              #the print line demo'd in shapes.py
from shapes import Triangle
```

```
shapes
```

Creating an r1 object

```
r1 = Rectangle(3,4)
r1
```

```
Rectangle(width=3, height=4)
```

But what is r1 really?

r1 is an object. So it has encapsulated/"boxed up" all the needed Rectangle info.

Visualize it like below.

```
       ------------------
r1 -> | width  = 3       |
      | height = 4       |
      |                  |
      |                  |
       ------------------
```

To get the info out of `r1`, we need to use those periods...

```
r1.height, r1.width
```

```
(4, 3)
```

Attributes vs. Properties Quickie Demo

```
#we can change the attribute values
r1.height = 10
r1.height, r1.width
```

```
(10, 3)
```

Now, area is a property of r1, not a field/attribute on r1.

We know this because it is marked as `@property` in the shapes file.

```
r1.area  #we don't use parentheses with properties
```

```
30
```

Why would area be a property instead of an attribute of a Rectangle object?

Well, it's a design decision, but here's one reason for this choice.

It makes sense we might let a user change the width/height of a rectangle, like if we stepped on a tile doing a DIY bathroom project...

But letting someone change the area of a tile seems "wonky" since then the height and width wouldn't match up with area.

```
#Uncomment out this line and it will crash and give you an AttibuteError.
#AttributeError: can't set attribute 'area'

#r1.area = 44
```

Object Level vs. Class Level Methods

The `is_square method` needs to know about the "state info" (the height/width) of the `r1` object to know if `r1` is a square.

So the `is_square method` is an object-level method, ie, we use `variable_name.is_square()` to call it.

```
r1.is_square()
```

```
False
```

The `unit_square` method just creates a 1 by 1 Rectangle.

We call `unit_square` a static function –> This means that it does not need object level info. So it can live at the class level.

That's why we call `Rectangle.unit_square` instead of something like `r1.unit_square`.

```
Rectangle.unit_square()
```

```
Rectangle(width=1, height=1)
```

Triangle Example

So here's some triangle code examples.

```
# A triangle object needs a base and height attribute
t  = Triangle(3,4)
t
```

```
Triangle(base=3, height=4)
```

And we can get its area which is a property of a Triangle as before...

```
t.area # (1/2) b * h
```

```
6.0
```

The `area_from_sides` method just calculates area when it is given 3 side lengthsusing something called Heron's formula.

The method doesn't need any state info from a specific Triangle object.

So again, we see that it's a static method and we thus call it using the class name: `Triangle.XXX` instead of `t.XXX`

```
Triangle.area_from_sides(1,4,4)
```

```
1.984313483298443
```