Triângulos

Neste exercício, definiremos uma classe chamada `Triangle` para representar triângulos no plano.

Observe que fornecemos um "esqueleto" abaixo, que inclui uma definição da classe `Point` a partir das leituras. Você deve fazer seu trabalho para este exercício a partir desse material fornecido (quer dizer, você deve usar a classe `Point` como definida abaixo). Copie-o no começo do seu arquivo!

```
import math

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

    def distance_to_origin(self):
        return (self.x**2 + self.y**2)**0.5

    def euclidean_distance(self, other):
        return ((self.x - other.x)**2 + (self.y - other.y)**2)**0.5

    def manhattan_distance(self, other):
        return abs(self.x - other.x) + abs(self.y - other.y)

    def angle_between(self, other):
        vert = other.y - self.y
        horiz = other.x - self.x
        return math.atan2(vert, horiz)
```

O método `__init__` para triângulos deve receber três instâncias de `Point`, onde cada instância representa um vértice do triângulo. Por exemplo:

```
t = Triangle(Point(2,3), Point(5,6), Point(3,5))
```

Sua classe deve fornecer os seguintes métodos:

- Um método `side_lengths()`, que retorna uma tupla contendo os comprimentos dos lados (nas mesmas unidades das coordenadas), em qualquer ordem.
- Um método `angles()`, que retorna uma lista ou tupla contendo os ângulos presentes no triângulo (em radianos), em qualquer ordem.
- Um método `side_classification()`, que retorna uma string: `"scalene"` (escaleno), `"isosceles"` (isósceles) ou `"equilateral"` (equilátero), dependendo do tipo de triângulo representado.
- Um método `angle_classification()`, que retorna uma string: `"acute"` (acutângulo), `"right"` (retângulo), `"obtuse"` (obtusângulo) ou `"equiangular"`; o que melhor descreve o triângulo.
- Um método `is_right()`, que retorna um booleano: `True` se este for um triângulo retângulo e `False` caso contrário.

- Um método `area()`, que retorna a área do triângulo.
- Um método `perimeter()`, que retorna o perímetro do triângulo.

Observe que aqui faremos comparações em floats, mas lembre-se de que floats não podem representar exatamente todos os números reais. Como tal, o teste de igualdade com `==` não é confiável. Dessa forma, se você precisar verificar se dois floats são iguais, escreva uma função que receba dois argumentos e retorne `True` se eles estiverem dentro de 10^{-6} um do outro, e `False` caso contrário, e use essa função para testar a "quase igualdade", em vez de testar a igualdade exata de floats.

Submissão

Quando estiver pronto (depois de ter simulado manualmente e testado em sua própria máquina e estiver convencido de que seu programa fará a coisa certa), faça upload do seu arquivo Python no **Problema 4.2** no Gradescope. Lembre de nomear seu arquivo `p4_2.py`.