Programación orientada a Objetos

Ejercicios

> Nota: (★☆☆, ★★☆, ★★★) Esta notación indica la dificultad (ascendente).

1. ★☆☆ Definir una clase `Coordinate` que se utilice para representar puntos (coordenadas). Los clase debe instanciarse utilizando 2 floats, que deben guardarse como estado de la misma. Definir, además, los siguiente métodos:

   - `__repr__()`
   - `distance()` que devuelve la distancia de la coordenada a otra coordenada pasada como argumento.
   - `get_x()` y `get_y()` que devuelven el estado de la coordenada, en coordenadas cartesianas.
   - `get_norm()` y `get_angle()` que devuelven el estado de la coordenada, en coordenadas polares.

   Verificar que se puede reproducir la siguiente sesión en la consola interactiva de python.

   ```
   In [1]: from coordinate import Coordinate

   In [2]: c1 = Coordinate(3, 1)

   In [3]: print(c1)
   Coordinate(3.0, 1.0)

   In [4]: print(c1.get_x())
   3.0

   In [4]: print(c1.get_y())
   1.0

   In [5]: c2 = Coordinate(7, 4)

   In [6]: c1.distance(c2)
   Out[6]: 5.0
   ```

2. ★☆☆ Definir una clase **Triangle**, usada para modelar un triángulo, cuyo estado sea modelado por una lista de coordenadas. Definir los siguientes métodos como parte del comportamiento:

   - `get_area()`, que devuelve el área del triángulo.

3. ★☆☆ Crear un objeto **Rectangle** que represente un rectángulo.

   - Debe tener un método para representarse de forma que la siguiente interacción sea posible:

     ```
     In [2]: rec = Rectangle(5, 3)

     In [3]: rec.draw()
     *---*
     |   |
     *---*
     ```

   - Dentro del comportamiento del objeto, debe ser posible solicitarle el área (mediante un método `get_area()`).

4. ★★☆ Agregar a las clases `Rectangle` y `Triangle` el método `get_perimeter()` que retorne el perímetro de la figura.

5. ★★☆ Definimos que dos figuras geométricas son iguales si son de la misma clase y tienen el mismo perímetro y área. Implementar el método `__eq__` que recibe otro objeto y retorna `True` si son iguales y `False` en caso contrario.

6. ★★☆ Se tiene una lista de figuras geométricas y, por la naturaleza del problema, se desean ordenadas por perímetro. Sabemos que las listas en el lenguaje Python contienen un método llamado `sort` que comanda a la lista a ordenar sus elementos. Para que ese método funcione, los elementos de la lista deben poder ser comparados. Particularmente, funciona si la relación `<` entre los objetos funciona, es decir, `A < B` devuelve `True` o `False`.

   Implemente el comportamiento necesario en las clases `Rectangle` y `Triangle` que permita ordenarlos.

   Verifique que la siguiente sesión ejecutada en la consola interactiva funciona y devuelve una lista ordenada.

```
In [5]: def U(a: float = 0, b: float = 10) -> float:
   ...:     return random.random() * (b - a) + a
   ...:

In [6]: lista = [Rectangle(U(), U()) for _ in range(5)] + [Triangle(U(), U()) for _ in range(5)]

In [7]: lista.sort()
```

7. ★★☆ Definir una clase `GeometryFigure` de la cual heredan **Rectangle** y **Triangle**. Revisar los ejercicios anteriores pensando en esta clase madre.

8. ★★★Definir una clase **Interval**, la cual debe representar un intervalo de valores reales.

   - La misma debe tener una forma de asignarle el paso (step si pensamos en `range` de python).
   - Debe implementar un método `get_next()` que retorne el próximo valor del intervalo tomando como base el inicio del intervalo y el paso preseteado, pero teniendo en cuenta el avance que se tuvo y no retornar siempre el mismo valor.
   - Implementar el método `has_next()` que retorne `True` si existe otro valor en el intervalo utilizando dicho paso.

   El siguiente fragmento de código debe ser funcional:

```
interval = Interval(0, 100)
interval.set_step(0.5)
while interval.has_next():
    print(interval.get_next())
```

9. ★★★ Escribir una versión del juego "Piedra Papel o Tijeras" basado en objetos. Utilice las clases `Scissors`, `Rock` y `Paper` para representar las opciones disponibles. Utilice una función principal `main` para ejecutar el juego, el oponente será la computadora. Cada instancia de `Scissors`, `Rock` y `Paper` debe tener un método `cmp` que reciba otra instancia y retorne un número siguiendo las siguientes reglas:

   - El número es mayor a cero en caso que el mismo objeto (self) sea ganador.
   - El número es menor a cero en caso que el mismo objeto (self) sea perdedor.
   - El número es igual a cero en caso de empate.

   Utilizar dicha funcionalidad para resolver el problema.

Link: https://pamoreno.github.io/ejercicios/guias/