Tarea 13 - Scripts y Módulos en `Python`

Curso de `Python`

Ejercicio 1

Crea un script llamado `vectors.py`. Ábrelo y crea una clase llamada `Vector2D`. El constructor debe guardar las coordenadas `x` e `y` del vector.

Ejercicio 2

Crea los siguientes métodos:

- La propiedad `.module` que devuelva el módulo del vector. Recuerda que el módulo de un vector 2D se calcula como $\sqrt{x^2 + y^2}$. Para calcular raíces cuadradas, dispones del método `math.sqrt()`.
- El método de instancia `.scalar_prod()` que multiplique el vector por el número real dado por parámetro, que por defecto vale 1.

Configura el método `.__str__` para que se nos muestre por pantalla el vector de la forma `(x, y)`.

Ejercicio 3

Crea los siguientes métodos:

- El método de clase `.sum()` que dados dos vectores los sume y devuelva un objeto de la clase `Vector2D`.
- El método de clase `.subtract()` que dadoas dos vectores los reste y devuelva un objeto de la clase `Vector2D`.

Ejercicio 4

Crea los siguientes métodos:

- El método estático `.dot_product()` que dados dos vectores calcule su producto escalar. Recuerda que el producto escalar de 2 vectores 2D u y v se calcula como $u \cdot v = u_x v_x + u_y v_y$
- El método de clase `.distance()` que dados dos vectores calcule la distancia entre ellos. Recuerda que la distancia entre 2 vectores 2D u y v se calcula como $\sqrt{(u_x - v_x)^2 + (u_y - v_y)^2}$

Ejercicio 5

Ahora crea la clase `Vector3D` que herede de la clase `Vector2D`. Empieza con el constructor para que además de las coordenadas `x` e `y`, también tome la coordenada `z`. Recuerda que puedes utilizar el método `.super()` para acceder a métodos de la clase padre.

Ejercicio 6

Crea en la clase `Vector3D` los métodos siguientes:

- el método `.__str__()` para que muestre el vector por pantalla de la forma `(x, y, z)`.
- la propiedad `.module`. Al tener vectores 3D, el módulo se calcula como $\sqrt{x^2 + y^2 + z^2}$
- el método de instancia `.scalar_prod()`
- los métodos de clase `.sum()` y `.subtract()` para que devuelvan objetos de la clase `Vector3D`.
- el método estático `.dot_product()`, pues ahora el producto escalar se calcula como $u \cdot v = u_x v_x + u_y v_y + u_z v_z$
- el método de clase `.distance()`, pues ahora la distancia se calcula como

$$\sqrt{(u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2}$$

Recuerda que dispones del método `.super()` para evitar repeticiones innecesarias de código.

Ejercicio 7

Crea en la clase `Vector3D` los métodos siguientes:

- el método de clase `.zero()` que devuelva un objeto `Vector3D` con todas sus componentes 0.
- el método de clase `.horizontal()` que devuelva un objeto `Vector3D` con todas sus componentes 0 salvo la primera que valdrá 1.
- el método de clase `.vertical()` que devuelva un objeto `Vector3D` con todas sus componentes 0 salvo la segunda que valdrá 1.
- el método de clase `.forward()` que devuelva un objeto `Vector3D` con todas sus componentes 0 salvo la tercera y última que valdrá 1.

Ejercicio 8

Crea en la clase `Vector2D` el método de instancia `.extend_to_3D()` que devuelva un objeto de la clase `Vector3D` siendo la componente `z` el valor indicado por parámetro, que por defecto valdrá 0.

Ejercicio 9

En un notebook de Google Colab, importa el script, crea dos objetos de la clase `Vector2D` y prueba que todos los métodos de la clase `Vector2D` funcionan correctamente.

Ejercicio 10

Ahora crea dos objetos de la clase `Vector3D` y prueba que todos los métodos de la clase `Vector3D` funcionan correctamente.