Programación para Física y Astronomía

Guía de estudios: Programación orientada a objetos

1^{ra} Actividad: crecimiento bacteriano

Inicialmente, hay una población de 100 bacterias. La población crece a una tasa del 20% cada hora. Su objetivo es simular el crecimiento de esta población durante 24 horas.

1. Inicializar la población de bacterias a 100. Esto se puede hacer asignando el valor 100 a una variable, por ejemplo, `poblacion_inicial`.
2. Establecer la tasa de crecimiento a 1.2. Esto se puede hacer asignando el valor 1.2 a una variable, por ejemplo, `tasa_crecimiento`.
3. Crear una lista vacía para almacenar la población total a cada hora. Esto se puede hacer con el código `poblaciones = []`.
4. Utilizar un bucle para simular el crecimiento durante 24 horas. Dentro del bucle, calcular la población total después de una hora utilizando la fórmula

$$\text{poblacion} = \text{poblacion\_inicial} \times \text{tasa\_crecimiento}^{i},$$

   donde i es la iteración actual del bucle (es decir, la hora actual). Añadir esta población a la lista de poblaciones.
5. Después del bucle, convertir la lista de poblaciones a un arreglo de NumPy para un análisis posterior. Esto se puede hacer con el código `poblaciones = np.array(poblaciones)`
6. Imprimir el arreglo de poblaciones para verificar los resultados.

2^{da} Actividad: Cara & Sello

En este problema, hay tres jugadores: Alice, Bob y Charlie. Cada uno comienza con 100 monedas. En cada ronda, cada jugador tira una moneda al aire (1 cara, 0 sello). Si sale cara, el jugador gana 10 monedas, si sale sello, pierde 10 monedas. El juego termina después de 100 rondas.

1. Inicializar las monedas de cada jugador a 100. Esto se puede hacer asignando el valor 100 a tres variables, por ejemplo, `alice`, `bob` y `charlie`.
2. Utilizar un bucle para simular las 100 rondas del juego. En cada ronda, cada jugador tira una moneda al aire.

3. Utilizar la función `random.randint(0, 1)` para simular el tiro de la moneda para cada jugador.

4. Si el resultado del tiro es 1 (cara), el jugador gana 10 monedas. Si el resultado es 0 (sello), el jugador pierde 10 monedas.

5. Después del bucle, crear un diccionario con los nombres de los jugadores y la cantidad de monedas que tienen.

6. Ordenar el diccionario en orden descendente por la cantidad de monedas.

7. Imprimir la cantidad de monedas que cada jugador tiene al final del juego.

3^{ra} Actividad: Aproximadamente π

El método de Monte Carlo también puede utilizarse para estimar el valor de π. Consideremos un círculo de radio 1 inscrito en un cuadrado de lado 2. Si generamos puntos aleatorios dentro del cuadrado, la proporción de puntos que caen dentro del círculo respecto al total de puntos generados será aproximadamente igual a la proporción entre el área del círculo y el área del cuadrado, lo cual nos permite estimar π.

Instrucciones:

- Utilice un ciclo for de N iteraciones.
- En cada iteración, genere dos números aleatorios x e y, ambos entre -1 y 1.
- Determine si el punto (x, y) está dentro del círculo. Para esto, calcule la distancia del punto al origen (0, 0). Si la distancia es menor o igual a 1, el punto está dentro del círculo.
- Lleve un conteo de las veces que el punto está dentro del círculo.
- Estime π como 4 veces la proporción de puntos que están dentro del círculo respecto al total de puntos generados.
- Haga un gráfico de scatter de todos los puntos generados, distinguiendo con colores diferentes los puntos dentro y fuera del círculo.
- Utilice como título de la gráfica el valor estimado de π.

Este ejercicio no solo te ayudará a comprender mejor el método de Monte Carlo, sino que también te dará una visión visual de cómo funciona.

4^{ta} Actividad: Aproximadamente π

Escriba un programa en Python3 que cree una clase llamada Esfera. Esta clase debe representar una figura tridimensional que puede estar centrada en cualquier punto del espacio. Teniendo en cuenta lo mencionado defina los atributos y métodos especificados a continuación:

- **Atributos:**
  - C: Tupla (cx, cy, cz) que representa la ubicación del centro de la esfera en el espacio tridimensional.
  - r: Variable que especifica el radio de la esfera.
- **Métodos:**
  - `Diametro`: retorna $2r$.
  - `AreaSuperficie`: retorna $4\pi r^2$.
  - `Volumen`: retorna $\frac{4}{3}\pi r^3$.
  - `Adentro`: Dependiendo si un punto en el espacio se encuentra dentro o fuera de la esfera, retorna verdadero o falso.
  - `IntersectaEjeX`: retorna verdadero o falso si el eje x intersecta o no a la esfera. Para esto considere la siguiente condición, si el centro de la esfera es $C = (cx, cy, cz)$, calcule $|cy| + |cz|$, si este valor es menor o igual al radio de la esfera, entonces el eje x intersecta la esfera. Si el valor es mayor, entonces el eje x no intersecta la esfera.
- **Nota:** Debe utilizar solo funciones vistas en clases.
- **Nota2:** No debe utilizar funciones integradas en numpy.

Una vez que la clase Esfera esté definida, pruebe cada uno de los métodos. Por ejemplo, cree una instancia de la esfera con centro en (0,0,0) y radio 5, para luego llamar a cada uno de los métodos y así verificar su funcionamiento.