Retos de Programación en Python para Seis Equipos

13 de junio de 2025

1. Introducción

Este documento presenta seis retos de programación en Python, diseñados para ser resueltos por seis equipos. Cada reto aplica conceptos de funciones y módulos de Python a problemas prácticos en geografía, química, petróleo, geofísica y reforestación. Los retos incluyen una descripción, ejemplos de entrada y la salida esperada. Cada equipo debe desarrollar un programa en Python que cumpla con los requisitos especificados.

2. Instrucciones

- Cada equipo debe resolver el reto asignado, escribiendo un programa en Python.
- Utilice los módulos `math` y `statistics` según sea necesario, como se indica en cada reto.
- Asegúrese de que la salida del programa coincida con el formato especificado en los ejemplos.
- Ejecute y pruebe el programa con las entradas proporcionadas y, opcionalmente, con otros casos.

3. Retos para los Equipos

3.1. Reto 1: Cálculo de Distancia Geográfica (Equipo 1)

Descripción: Desarrolle una función `distancia_haversine` que calcule la distancia entre dos puntos geográficos en la Tierra, dados sus coordenadas (latitud y longitud en grados). Use la fórmula de Haversine:

$$d = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

donde $r = 6371\,\text{km}$, ϕ_1, ϕ_2 son latitudes, y λ_1, λ_2 son longitudes. Importe el módulo `math` con alias `m` y use `m.radians()`, `m.sin()`, `m.cos()`, `m.asin()`, y `m.sqrt()`. Redondee el resultado a 2 decimales.

Ejemplo de Entrada:

```
lat1, lon1 = 19.43, -99.13  # Ciudad de M xico
lat2, lon2 = 4.71, -74.07   # Bogot
distancia_haversine(lat1, lon1, lat2, lon2)
```

Salida Esperada:

```
2523.55
```

3.2. Reto 2: Balance de Reacciones Químicas (Equipo 2)

Descripción: Cree una función `balancear_reaccion` que verifique si una lista de coeficientes para una reacción química está balanceada. La función recibe una lista de números (coeficientes de reactivos y productos) y usa `statistics.mean()` para comparar la suma de coeficientes de reactivos y productos. Si la diferencia absoluta entre las sumas es menor a 0.01, la reacción está balanceada. Importe `statistics` como `stats` y use `len()` para validar que la lista tenga al menos 4 elementos.

Ejemplo de Entrada:

```
coeficientes = [2, 1, 1, 2]  # Para 2H2 + O2 -> 2H2O
balancear_reaccion(coeficientes)
```

Salida Esperada:

```
Reacci n balanceada. Media: 1.50
```

3.3. Reto 3: Estimación de Viscosidad de Petróleo (Equipo 3)

Descripción: Defina una función `viscosidad_petroleo` que estime la viscosidad de petróleo crudo usando la ecuación empírica:

$$\mu = A \cdot e^{B/T}$$

donde μ es la viscosidad (en cP), T es la temperatura (en Kelvin), y $A = 0{,}01, B = 4000$. Importe `math` y use `math.exp()`. Devuelva la viscosidad redondeada a 2 decimales.

Ejemplo de Entrada:

```
viscosidad_petroleo(300)
```

Salida Esperada:

```
1.35
```

3.4. Reto 4: Análisis de Datos Sísmicos (Equipo 4)

Descripción: Escriba una función `analizar_sismico` que procese una lista de amplitudes de ondas sísmicas. La función debe:

- Validar que la lista no esté vacía con `len()`.
- Calcular la media y desviación estándar con `statistics.mean()` y `statistics.stdev()`.

- Identificar amplitudes anómalas (valores fuera de media $\pm 2 \cdot$ desviación estándar).

Importe `statistics` como `stats` y devuelva un diccionario con la media, desviación estándar (redondeadas a 2 decimales), y el número de anomalías.

Ejemplo de Entrada:

```
amplitudes = [1.2, 1.5, 1.3, 5.0, 1.4, 1.6]
analizar_sismico(amplitudes)
```

Salida Esperada:

```
{'media': 2.00, 'desviacion_estandar': 1.43, 'anomalias': 1}
```

3.5. Reto 5: Planificación de Reforestación (Equipo 5)

Descripción: Cree una función `plan_reforestacion` que calcule el área circular disponible para plantar árboles y estime el número de árboles según una densidad dada (árboles por metro cuadrado). Use `math.pi` para calcular el área (Área $= \pi \cdot r^2$) y `statistics.mean()` para promediar densidades de plantación. Importe `math` y `statistics` con alias `m` y `stats`. Devuelva una tupla con el área y el promedio de árboles estimados, redondeados a 2 decimales.

Ejemplo de Entrada:

```
radio = 10
densidades = [0.1, 0.2, 0.15]
plan_reforestacion(radio, densidades)
```

Salida Esperada:

```
(314.16, 47.12)
```

3.6. Reto 6: Evaluación de Yacimiento (Equipo 6)

Descripción: Desarrolle una función `evaluar_yacimiento` que combine datos geofísicos y petroleros. La función recibe una lista de porosidades (fracción) y un radio de pozo (en metros). Debe:

- Calcular la media de porosidad con `statistics.mean()`.
- Estimar el volumen de hidrocarburos usando $V = \pi \cdot r^2 \cdot h \cdot \phi$, donde $h = 100\,\text{m}$ y ϕ es la porosidad media.
- Validar que las porosidades estén entre 0 y 1.

Importe `math` como `m` y `statistics` como `stats`. Devuelva un diccionario con la porosidad media y el volumen, redondeados a 2 decimales.

Ejemplo de Entrada:

```
porosidades = [0.1, 0.15, 0.2]
radio = 5
evaluar_yacimiento(porosidades, radio)
```

Salida Esperada:

```
{'porosidad_media': 0.15, 'volumen': 1178.10}
```

4. Conclusión

Estos retos fomentan el aprendizaje práctico de Python aplicado a problemas reales en diversas disciplinas. Cada equipo debe colaborar para desarrollar una solución eficiente, utilizando funciones y módulos de Python. ¡Éxito en la resolución de los retos!