# Movimientos en Física utilizando Python

Autor: Luciano Suarez

## Resumen:

Este trabajo presenta un programa echo en Python para la simulación y análisis de diferentes tipos de movimientos físicos, incluyendo; Movimiento Rectilíneo Uniforme (MRU), Movimiento Rectilíneo Uniformemente Variado (MRUV), Tiro Oblicuo y Caída Libre. El programa permite calcular y graficar las trayectorias, velocidades y aceleraciones de los cuerpos en movimiento, además se pueden resolver incógnitas relacionadas con estos movimientos.

## Palabras Claves:

Movimiento Rectilíneo Uniforme, Movimiento Rectilíneo Uniformemente Variado, Tiro Oblicuo, Caída Libre, Python, Simulación

## Introducción:

En este trabajo se exploran los fundamentos de la física del movimiento, importantes para entender cómo se comportan los objetos en el espacio y el tiempo. Estos movimientos pueden ser descritos mediante ecuaciones matemáticas y representaciones gráficas. Se presentarán algunas clases de movimientos posibles junto con las funciones y las gráficas que las describen, utilizando Python para automatizar procesos como graficar y resolver incógnitas. El usuario introducirá valores y podrá visualizar los resultados.

## Desarrollo:

Para este trabajo se considerará un sistema ideal sin fricción del aire ni rozamientos y los movimientos se analizarán como una partícula.

El programa desarrollado en Python utiliza las librerías numpy y matplotlib para realizar los cálculos y generar gráficos que ilustran los movimientos.

Al iniciar el programa, este te muestra un Menú:

---

Seleccione el tipo de movimiento:

---

1. Graficar Movimiento Rectilíneo Uniforme (MRU)
2. Graficar Movimiento Rectilíneo Uniformemente Variado (MRUV)
3. Graficar Tiro Oblicuo
4. Graficar Caída Libre

5. Resolver incógnitas de MRU

6. Resolver incógnitas de MRUV

7. Resolver incógnitas de Tiro Oblicuo

8. Resolver incógnitas de Caída Libre

9. Salir

---

Debemos de elegir una de estas opciones enumeradas.

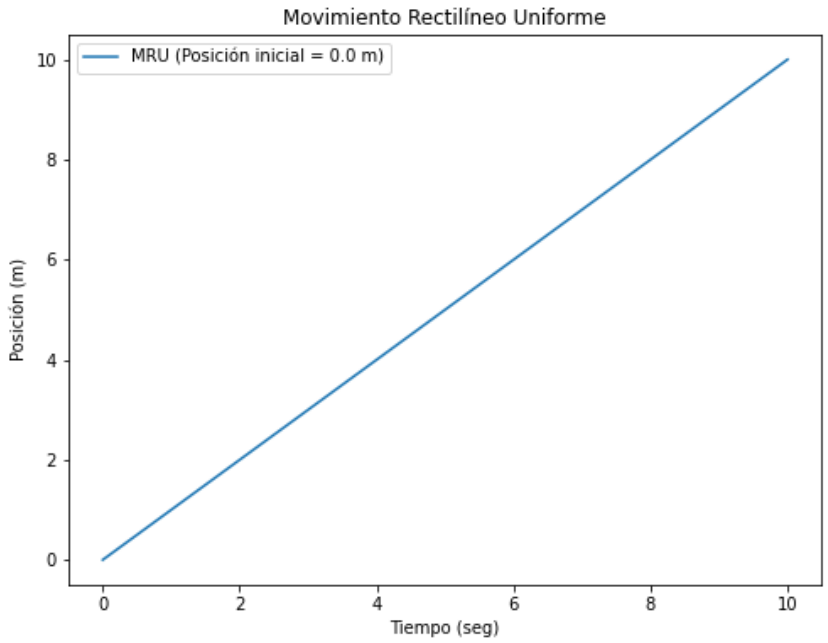
En las primeras cuatro opciones permiten graficar el movimiento seleccionado donde el usuario debe proporcionar valores para cada variable, y se muestran los gráficos correspondientes. A continuación, se describen los movimientos y sus ecuaciones.

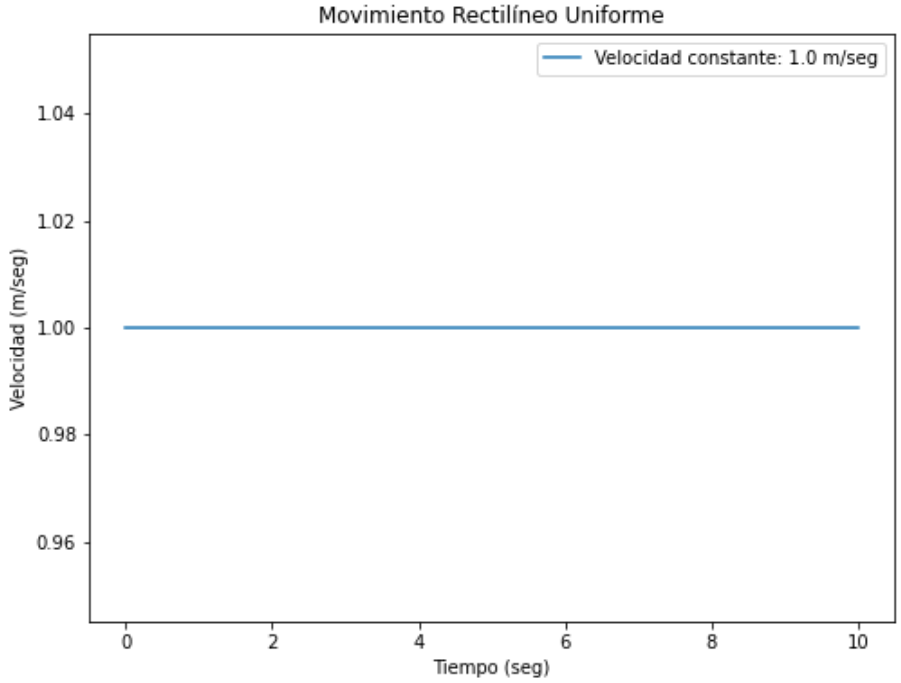
1. **Movimiento Rectilíneo Uniforme (MRU)**:
   El MRU describe el movimiento de un objeto que se desplaza en línea recta con velocidad constante. La ecuación que describe este movimiento es:

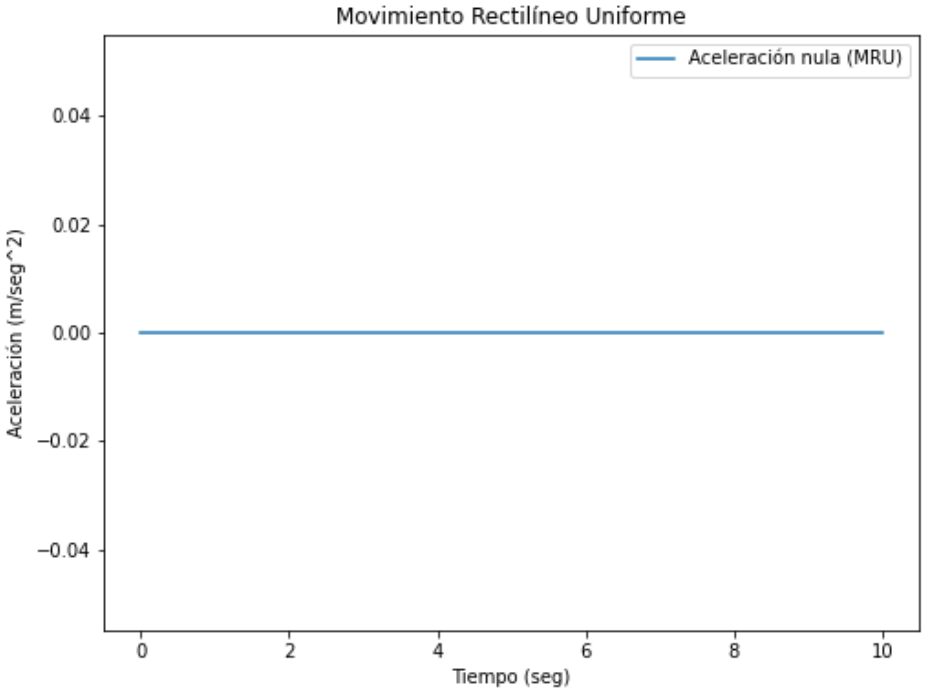
$$x(t) = xi + v \cdot t$$

   donde **x(t)** es la posición del objeto en el tiempo **t**, **xi** es la posición inicial y **v** es la velocidad constante.

   Graficas:






Ecuación usada: x = xi + v * t

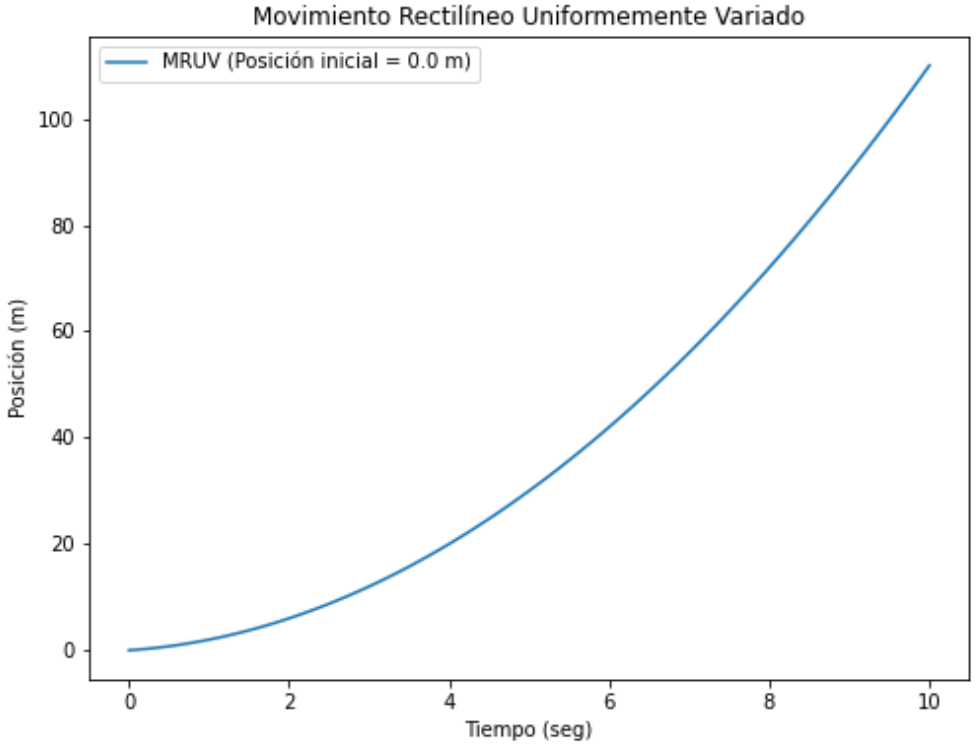
Datos Ingresados:
Velocidad en m/seg: 1
Tiempo total en seg: 10
Posición inicial en m: 0

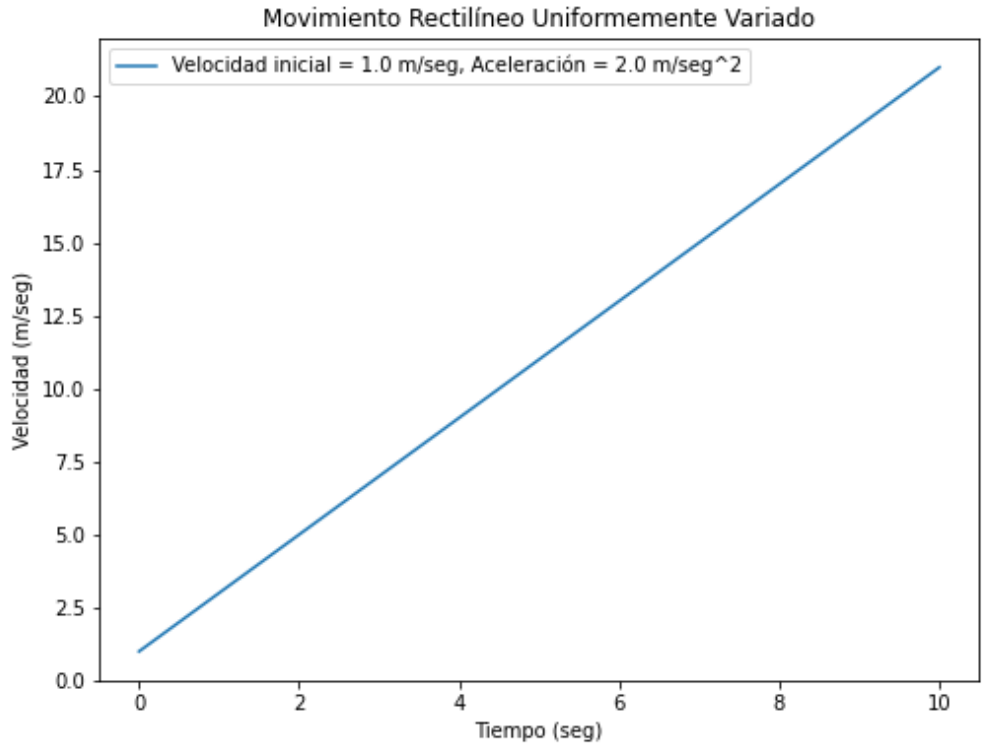
2. **Movimiento Rectilíneo Uniformemente Variado (MRUV)**:
   En el MRUV, la aceleración es constante, la velocidad lineal y la posición en función del tiempo cuadrática, estas son sus representaciones matemáticas:

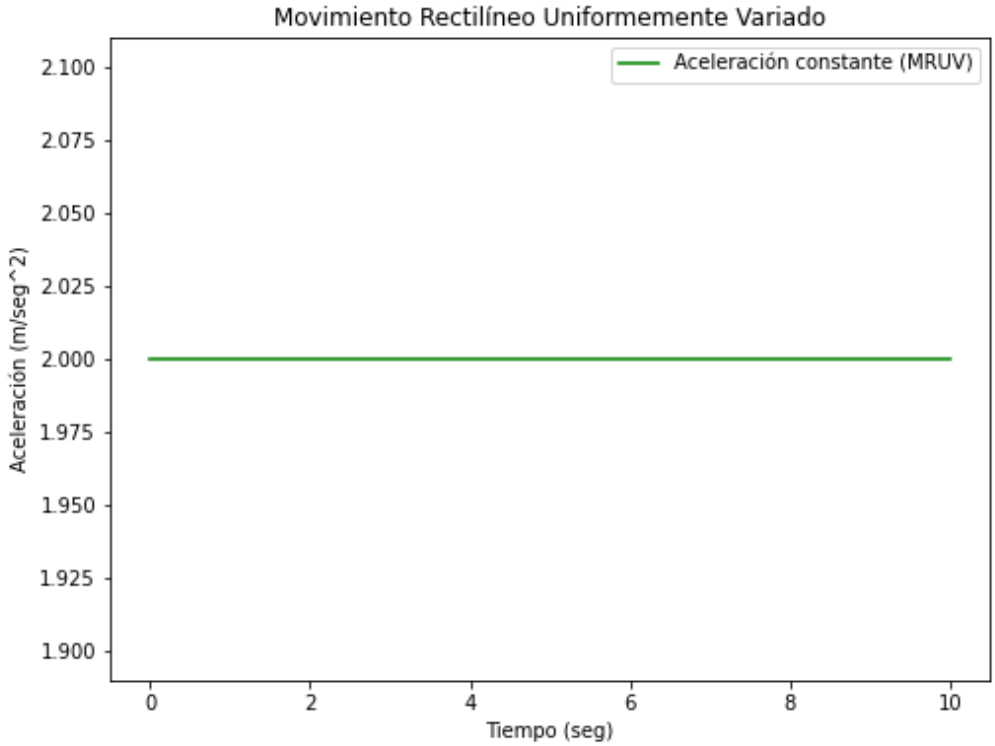
$$x(t) = xi + vi \cdot t + 0.5 \cdot a \cdot t^2$$

Donde **x(t)** es la posición del objeto en el tiempo **t**, xi es la posición inicial y **vi** es la velocidad inicial y **a** es la aceleración

Gráficos:






Ecuación usada: x = xi + vi * tiempo + 0.5 * a * tiempo^2

Datos Ingresados:

Velocidad inicial en m/seg: 1

Aceleración en m/seg^2: 2

Tiempo total en seg: 10

Posición inicial m: 0

Distancia recorrida: 110.00 m

3. **Tiro Oblicuo**:

El tiro oblicuo combina movimiento vertical (que será MRUV) y horizontal (MRU). Las ecuaciones para las componentes *x* e *y* del movimiento son:

$$v_x = vi \cdot cos(\theta)$$
$$v_y = vi \cdot sin(\theta)$$

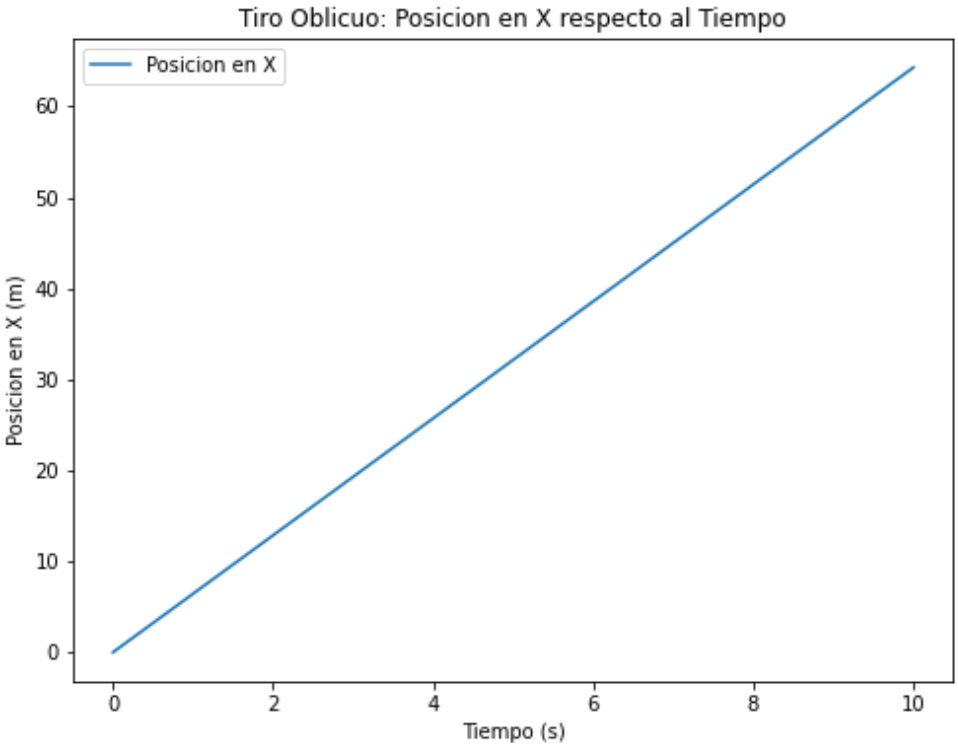
$$x(t) = v_x \cdot t$$

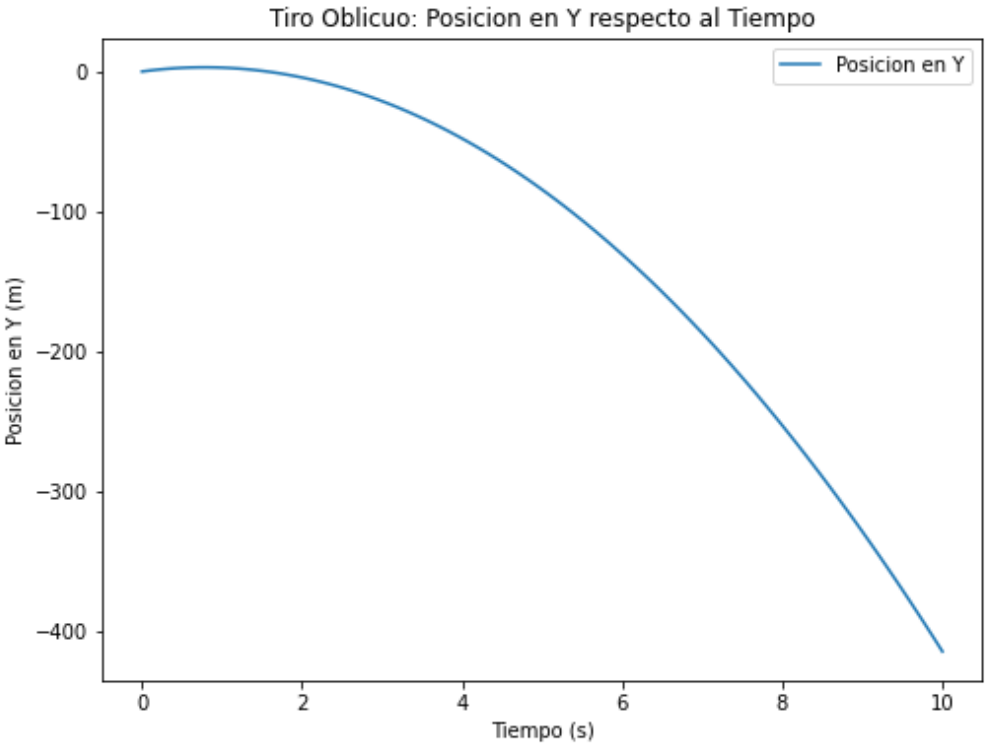
$$y(t) = v_y \cdot t + 0.5 \cdot g \cdot t^2$$

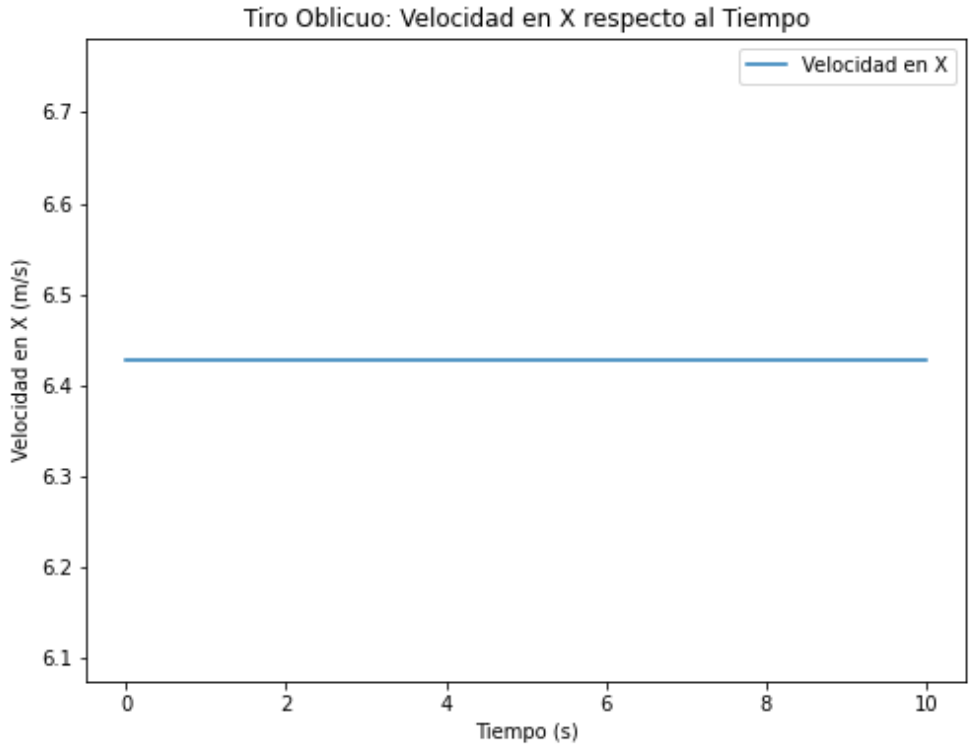
<u>*Movimiento en x:*</u> Donde **x(t)** es la posición del objeto en el tiempo (**t**), $\boldsymbol{v_x}$ es la velocidad en x que se obtiene a partir de $\boldsymbol{v_x = vi \cdot cos(\theta)}$, y θ es el ángulo de lanzamiento.

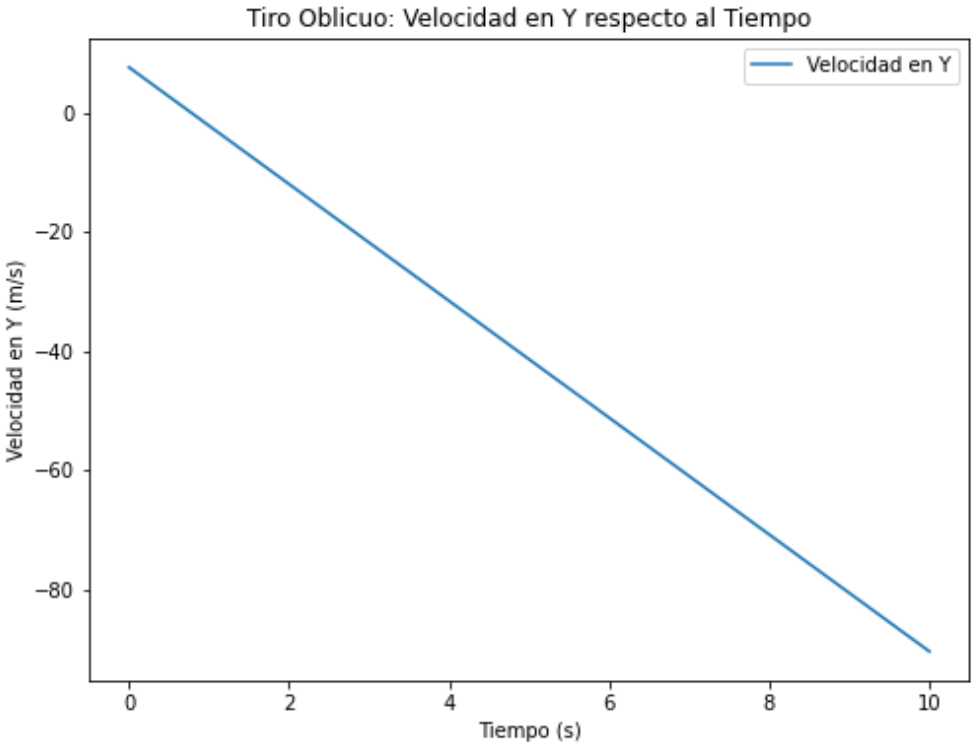
<u>*Movimiento en y:*</u> Donde **y(t)** es la posición del objeto en el tiempo (**t**), $\boldsymbol{v_y}$ es la velocidad en y que se obtiene a partir de $\boldsymbol{v_y = vi \cdot sin(\theta)}$, **g** es la gravedad, y **θ** es el ángulo de lanzamiento.

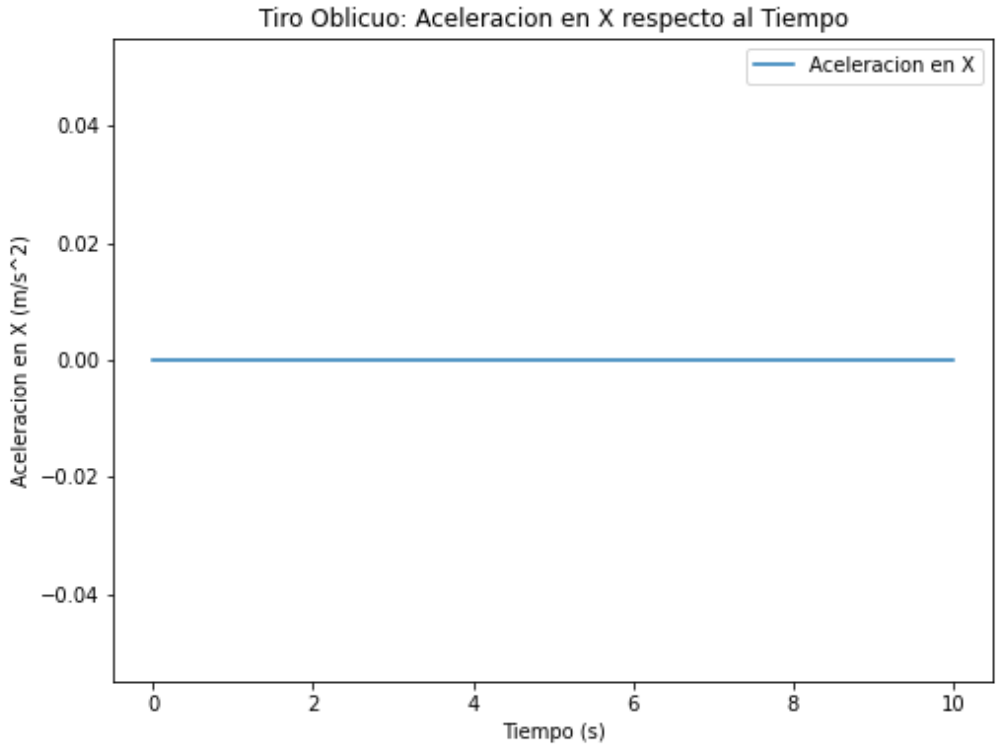
Gráficos:

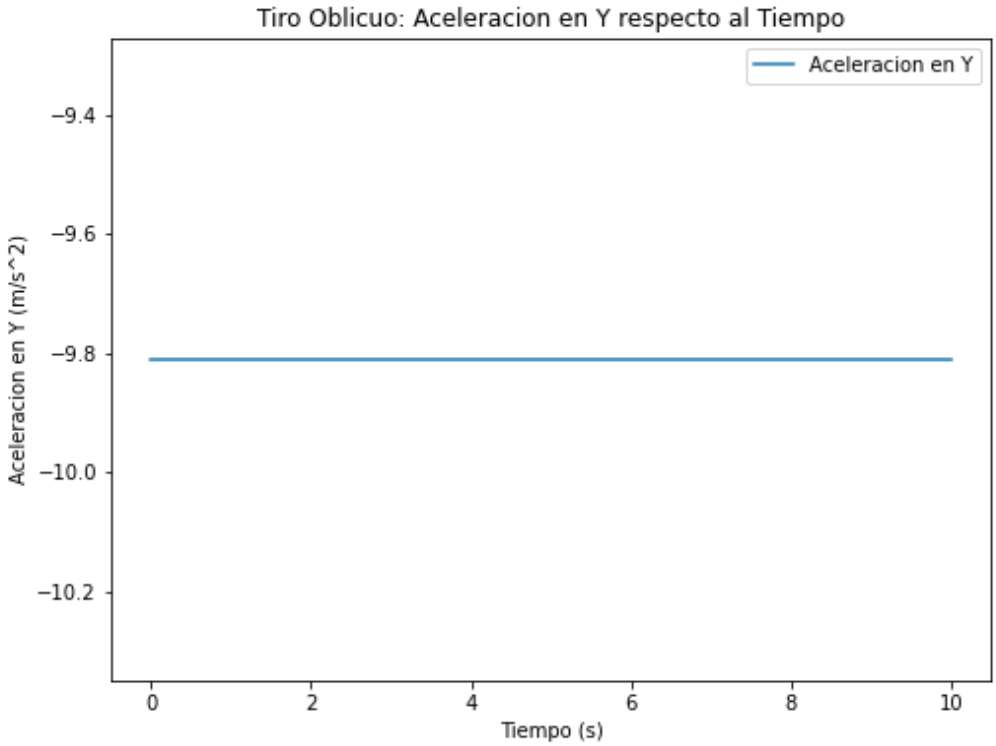

Tiro Oblicuo: Posicion en X respecto al Tiempo
Posicion en X
Posicion en X (m)
Tiempo (s)
0
10
20
30
40
50
60
0
2
4
6
8
10


Tiro Oblicuo: Posicion en Y respecto al Tiempo
Posicion en Y
0
−100
−200
−300
−400
Posicion en Y (m)
0
2
4
6
8
10
Tiempo (s)


Tiro Oblicuo: Velocidad en X respecto al Tiempo
Velocidad en X
6.7
6.6
6.5
6.4
6.3
6.2
6.1
Velocidad en X (m/s)
0
2
4
6
8
10
Tiempo (s)


Tiro Oblicuo: Velocidad en Y respecto al Tiempo
Velocidad en Y
0
−20
−40
−60
−80
Velocidad en Y (m/s)
0
2
4
6
8
10
Tiempo (s)


Tiro Oblicuo: Aceleracion en X respecto al Tiempo
Aceleracion en X
0.04
0.02
0.00
−0.02
−0.04
Aceleracion en X (m/s^2)
0
2
4
6
8
10
Tiempo (s)




Datos ingresados:

Velocidad inicial en m/seg: 10

Angulo de lanzamiento en grados: 50

Tiempo total en seg: 10

Distancia horizontal recorrida: 10.04 m

Altura máxima alcanzada: 2.99 m

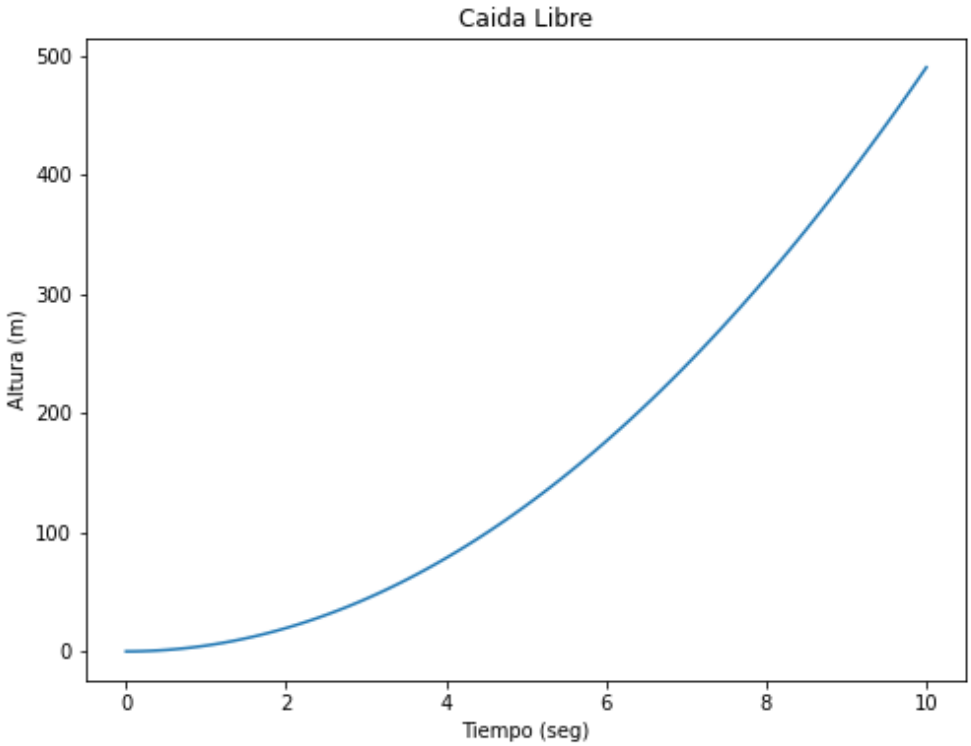
4. **Caída Libre**:

En la caída libre, un objeto se mueve bajo la influencia exclusiva de la gravedad, La ecuación de la posición vertical es:

$$y = 0.5 \cdot g \cdot t^2$$

Si el usuario elige las opciones 5 a 8 el programa resuelve la incógnita necesaria a partir de los valores de las demás variables conocidas.

Gráficos:




Ecuacion usada: y = 0.5 * g * t**2

Datos Ingresados:

Tiempo total en seg: 10

Altura alcanzada: 490.50 m

Y por último si el usuario elige la opción 9, el programa finaliza.

## Referencias:


- SERWAY, R., Física, Tomo II, México, Ed. Mc Graw-I Iill, (1994) HALLIDAY, D., RESNICK. R. y KRANE
- https://docs.python.org/es/3/tutorial/


## Anexos:

```
import numpy as np
import matplotlib.pyplot as plt

print("")

print("////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////")
```

```
print("Programa que permite resolver incognitas, calcular y graficar movimientos, pertenecientes a la rama de fisica 'Cinematica'")
print("////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////")

print("")

# Def`s ---------------------------------------------------------------------

# Ecuaciones de MRU, MRUV, Tiro Oblicuo y Caida Libre

def MRU(v, t, xi):

    x = xi + v * t

    return x

def MRUV(vi, a, t, xi):

    x = xi + vi * t + 0.5 * a * t**2

    return x

def tiro_oblicuo(vi, tita, t):

    g = 9.81

    tita_rad=np.radians(tita)

    vx = vi * np.cos(tita_rad)
    vy = vi * np.sin(tita_rad)

    x = vx * t

    y = vy * t - 0.5 * g * t**2

    return x, y

def caida_libre(t):

    g = 9.81

    y = 0.5 * g * t**2

    return y

# Graficacion de MRU, MRUV, Tiro Oblicuo y Caida Libre

def graficar_MRU(v, t_total, xi):

    tiempo=np.linspace(0, t_total,  100)

    # Gráfico de posición respecto al tiempo (MRU)

    x = xi + v * tiempo

    fig_MRU_1, ax1 = plt.subplots(figsize=(8, 6))
```

```
    ax1.plot(tiempo, x, label=f"MRU (Posición inicial = {xi} m)")
    ax1.set_title("Movimiento Rectilíneo Uniforme")
    ax1.set_xlabel("Tiempo (seg)")
    ax1.set_ylabel("Posición (m)")
    ax1.legend()

    fig_MRU_1.savefig("A Grafico de Posicion Respecto al Tiempo (MRU) .png")

    plt.tight_layout()

    plt.show()

    # Gráfico de velocidad respecto al tiempo (MRU)

    velocidad = [v] * len(tiempo)

    fig2, ax2 = plt.subplots(figsize=(8, 6))

    ax2.plot(tiempo, velocidad, label=f"Velocidad constante: {v} m/seg")
    ax2.set_title("Movimiento Rectilíneo Uniforme")
    ax2.set_xlabel("Tiempo (seg)")
    ax2.set_ylabel("Velocidad (m/seg)")
    ax2.legend()

    fig2.savefig("B Grafico de Velocidad Respecto al Tiempo (MRU) .png")

    plt.tight_layout()

    plt.show()

    # Gráfico de aceleración respecto al tiempo (MRU)

    aceleracion = [0] * len(tiempo)

    fig3, ax3 = plt.subplots(figsize=(8, 6))

    ax3.plot(tiempo, aceleracion, label="Aceleración nula (MRU)")
    ax3.set_title("Movimiento Rectilíneo Uniforme")
    ax3.set_xlabel("Tiempo (seg)")
    ax3.set_ylabel("Aceleración (m/seg^2)")
    ax3.legend()

    fig3.savefig("C Grafico de Aceleracion Respecto al Tiempo (MRU) .png")

    plt.tight_layout()

    plt.show()

def graficar_MRUV(vi, a, t_total, xi):

    tiempo = np.linspace(0, t_total, 100)

    # Gráfico de posición respecto al tiempo (MRUV)

    x = xi + vi * tiempo + 0.5 * a * tiempo**2
```

```
    fig1, ax1 = plt.subplots(figsize=(8, 6))

    ax1.plot(tiempo, x, label=f"MRUV (Posición inicial = {xi} m)")
    ax1.set_title("Movimiento Rectilíneo Uniformemente Variado")
    ax1.set_xlabel("Tiempo (seg)")
    ax1.set_ylabel("Posición (m)")
    ax1.legend()

    fig1.savefig("D Grafico de Posicion Respecto al Tiempo (MRUV) .png")

    plt.tight_layout()

    plt.show()

    # Gráfico de velocidad respecto al tiempo (MRUV)

    velocidad = vi + a * tiempo

    fig2, ax2 = plt.subplots(figsize=(8, 6))

    ax2.plot(tiempo, velocidad, label=f"Velocidad inicial = {vi} m/seg, Aceleración = {a}
m/seg^2")
    ax2.set_title("Movimiento Rectilíneo Uniformemente Variado")
    ax2.set_xlabel("Tiempo (seg)")
    ax2.set_ylabel("Velocidad (m/seg)")
    ax2.legend()

    fig2.savefig("E Grafico de Velocidad Respecto al Tiempo (MRUV) .png")

    plt.tight_layout()

    plt.show()

    # Gráfico de aceleración respecto al tiempo (MRUV)

    aceleracion = [a] * len(tiempo)

    fig3, ax3 = plt.subplots(figsize=(8, 6))

    ax3.plot(tiempo, aceleracion, 'g-', label="Aceleración constante (MRUV)")
    ax3.set_title("Movimiento Rectilíneo Uniformemente Variado")
    ax3.set_xlabel("Tiempo (seg)")
    ax3.set_ylabel("Aceleración (m/seg^2)")
    ax3.legend()

    fig3.savefig("F Grafico de Aceleracion Respecto al Tiempo (MRUV) .png")

    plt.tight_layout()

    plt.show()

def graficar_tiro_oblicuo(vi, tita, t_total):

    tiempo = np.linspace(0, t_total, 100)
```

```
x, y = tiro_oblicuo(vi, tita, tiempo)

g = 9.81

tita_rad = np.radians(tita)
vx = vi * np.cos(tita_rad)
vy = vi * np.sin(tita_rad)

# Grafico de posicion respecto al tiempo (eje X)

fig1, ax1 = plt.subplots(figsize=(8, 6))

ax1.plot(tiempo, x, label="Posicion en X")
ax1.set_title("Tiro Oblicuo: Posicion en X respecto al Tiempo")
ax1.set_xlabel("Tiempo (s)")
ax1.set_ylabel("Posicion en X (m)")
ax1.legend()

fig1.savefig("G Grafico de posicion respecto al tiempo (eje X)")

plt.tight_layout()

plt.show()

# Grafico de posicion respecto al tiempo (eje Y)

fig2, ax2 = plt.subplots(figsize=(8, 6))

ax2.plot(tiempo, y, label="Posicion en Y")
ax2.set_title("Tiro Oblicuo: Posicion en Y respecto al Tiempo")
ax2.set_xlabel("Tiempo (s)")
ax2.set_ylabel("Posicion en Y (m)")
ax2.legend()

fig2.savefig("H Grafico de posicion respecto al tiempo (eje Y)")

plt.tight_layout()

plt.show()

# Grafico de velocidad respecto al tiempo (eje X)

velocidad_x = [vx] * len(tiempo)

fig3, ax3 = plt.subplots(figsize=(8, 6))

ax3.plot(tiempo, velocidad_x, label="Velocidad en X")
ax3.set_title("Tiro Oblicuo: Velocidad en X respecto al Tiempo")
ax3.set_xlabel("Tiempo (s)")
ax3.set_ylabel("Velocidad en X (m/s)")
ax3.legend()

fig3.savefig("I Grafico de velocidad respecto al tiempo (eje X)")

plt.tight_layout()
```

```
plt.show()

# Grafico de velocidad respecto al tiempo (eje Y)

velocidad_y = vy - g * tiempo

fig4, ax4 = plt.subplots(figsize=(8, 6))

ax4.plot(tiempo, velocidad_y, label="Velocidad en Y")
ax4.set_title("Tiro Oblicuo: Velocidad en Y respecto al Tiempo")
ax4.set_xlabel("Tiempo (s)")
ax4.set_ylabel("Velocidad en Y (m/s)")
ax4.legend()

fig4.savefig("J Grafico de velocidad respecto al tiempo (eje Y)")

plt.tight_layout()

plt.show()

# Gráfico de aceleracion respecto al tiempo (eje X)

aceleracion_x = [0] * len(tiempo)

fig5, ax5 = plt.subplots(figsize=(8, 6))

ax5.plot(tiempo, aceleracion_x, label="Aceleracion en X")
ax5.set_title("Tiro Oblicuo: Aceleracion en X respecto al Tiempo")
ax5.set_xlabel("Tiempo (s)")
ax5.set_ylabel("Aceleracion en X (m/s^2)")
ax5.legend()

fig5.savefig("K Grafico de aceleracion respecto al tiempo (eje X)")

plt.tight_layout()

plt.show()

# Gráfico de aceleracion respecto al tiempo (eje Y)

aceleracion_y = [-g] * len(tiempo)

fig6, ax6 = plt.subplots(figsize=(8, 6))

ax6.plot(tiempo, aceleracion_y, label="Aceleracion en Y")
ax6.set_title("Tiro Oblicuo: Aceleracion en Y respecto al Tiempo")
ax6.set_xlabel("Tiempo (s)")
ax6.set_ylabel("Aceleracion en Y (m/s^2)")
ax6.legend()

fig6.savefig("L Grafico de aceleracion respecto al tiempo (eje Y)")

plt.tight_layout()

plt.show()
```

```
def calcular_altura_maxima_y_alcance(vi, tita):
    g = 9.81
    tita_rad = np.radians(tita)

    vy = vi * np.sin(tita_rad)

    # Tiempo para alcanzar la altura máxima
    t_max = vy / g

    # Altura máxima
    h_max = vy * t_max - 0.5 * g * t_max**2

    # Tiempo total de vuelo
    t_total = 2 * t_max

    # Alcance máximo
    vx = vi * np.cos(tita_rad)
    d_total = vx * t_total

    return d_total, h_max

def graficar_caida_libre(t_total):

    tiempo = np.linspace(0, t_total, 100)
    y = caida_libre(tiempo)

    fig_caida_libre = plt.figure(figsize=(8, 6))
    plt.plot(tiempo, y, label="Caida Libre")
    plt.title("Caida Libre")
    plt.xlabel("Tiempo (seg)")
    plt.ylabel("Altura (m)")

    fig_caida_libre.savefig("N Grafico de Caida Libre .png")

    plt.legend()

    plt.tight_layout()

    plt.show()

# Resolver Incognitas de MRU, MRUV, Tiro Oblicuo y Caida Libre

def resolver_incongnitas_MRU():

    print("¿Que incognita quiere calcular o encontrar?")
    print("")
    print("1. Velocidad (v)")
    print("2. Tiempo (t)")
    print("3. Posicion inicial (xi)")

    print("")

    print("Ingrese la operacion deseada: ")
    opcion = int(input())

    print("")
```

```
if opcion == 1:

    print("Ecuacion usada: v = (x - xi) / t")

    print("")

    print("Ingrese la posición inicial en metros (xi): ")
    xi = float(input())

    print("")

    print("Ingrese el tiempo en segundos (t): ")
    t = float(input())

    print("")

    print("Ingrese la posición final en metros (x): ")
    x = float(input())

    v = (x - xi) / t

    print("")

    print(f"La velocidad es: {v:.2f} m/seg")

elif opcion == 2:

    print("Ecuacion usada: t = (x - xi) / v")

    print("")

    print("Ingrese la posición inicial en metros (xi): ")
    xi = float(input())

    print("")

    print("Ingrese la velocidad en m/seg (v): ")
    v = float(input())

    print("")

    print("Ingrese la posición final en metros (x): ")
    x = float(input())

    print("")

    t = (x - xi) / v

    print("")

    print(f"El tiempo es: {t:.2f} seg")

elif opcion == 3:

    print("Ecuacion usada: xi = x - v * t")
```

```
        print("")

        print("Ingrese la velocidad en m/seg (v): ")
        v = float(input())

        print("")

        print("Ingrese el tiempo en segundos (t): ")
        t = float(input())

        print("")

        print("Ingrese la posicion final en metros (x): ")
        x = float(input())

        xi = x - v * t

        print("")

        print(f"La posicion inicial es: {xi:.2f} m")

    else:
        print("Opción inválida.")

def resolver_incongnitas_MRUV():

    print("¿Que incognita quiere calcular o encontrar?")
    print("")
    print("1. Velocidad final (vf)")
    print("2. Tiempo (t)")
    print("3. Aceleración (a)")
    print("4. Posición inicial (xi)")

    print("")

    print("Ingrese la opcion deseada: ")
    opcion = int(input())

    print("")

    if opcion == 1:

        print("Ecuacion Usada: vf = vi + a * t")

        print("")

        print("Ingrese la velocidad inicial en m/seg (vi): ")
        vi = float(input())

        print("")

        print("Ingrese la aceleración en m/seg^2 (a): ")
        a = float(input())

        print("")
```

```
    print("Ingrese el tiempo en segundos (t): ")
    t = float(input())

    vf = vi + a * t

    print("")

    print(f"La velocidad final es: {vf:.2f} m/seg")

elif opcion == 2:

    print("Ecuacion usada: t1,2 = (-vi +- sqrt(vi**2 + 2 * a * (x - xi))) / a")

    print("")

    print("Ingrese la velocidad inicial en m/seg (vi): ")
    vi = float(input())

    print("")

    print("Ingrese la aceleracion en m/seg^2 (a): ")
    a = float(input())

    print("")

    print("Ingrese la posicion inicial en metros (xi): ")
    xi = float(input())

    print("")

    print("Ingrese la posicion final en metros (x): ")
    x = float(input())

    discriminante = vi**2 + 2 * a * (x - xi)

    if discriminante < 0:
        print("No es posible calcular el tiempo con los parámetros dados.")

    else:
        t1 = (-vi + np.sqrt(discriminante)) / a
        t2 = (-vi - np.sqrt(discriminante)) / a

        t = max(t1, t2)

        print(f"El tiempo es: {t:.2f} seg")

elif opcion == 3:

    print("Ecuacion usada: a = 2 * (x - xi - vi * t) / t**2")

    print("")

    print("Ingrese la velocidad inicial en m/s (vi): ")
    vi = float(input())
```

```
            print("")

            print("Ingrese el tiempo en segundos (t): ")
            t = float(input())

            print("")

            print("Ingrese la posición inicial en metros (xi): ")
            xi = float(input())

            print("")

            print("Ingrese la posicion final en metros (x): ")
            x = float(input())

            a = 2 * (x - xi - vi * t) / t**2

            print("")

            print(f"La aceleracion es: {a:.2f} m/seg^2")

        elif opcion == 4:

            print("Ecuacion usada: xi = vf * t - 0.5 * a * t**2")

            print("")

            print("Ingrese la velocidad final en m/seg (vf): ")
            vf = float(input())

            print("")

            print("Ingrese la aceleracion en m/seg^2 (a): ")
            a = float(input())

            print("")

            print("Ingrese el tiempo en segundos (t): ")
            t = float(input())

            xi = vf * t - 0.5 * a * t**2

            print("")

            print(f"La posicion inicial es: {xi:.2f} m")

        else:
            print("Opción inválida.")

    def resolver_incongnitas_tiro_oblicuo():

        print("¿Que incognita quiere calcular o encontrar?")
        print("")
        print("1. Altura máxima")
        print("2. Tiempo de vuelo")
        print("3. Alcance máximo")
```

```
print("Ingrese la opción deseada: ")
opcion = int(input())

print("")

if opcion == 1:

    print("Ecuacion usada:  h_max = ((vi * sen(θ))**2) / (2 * g)")

    print("")

    print("Ingrese la velocidad inicial en m/seg (vi): ")
    vi = float(input())

    print("")

    print("Ingrese el angulo de lanzamiento en grados: ")
    tita = float(input())

    g = 9.81

    tita_rad = np.radians(tita)

    vy = vi * np.sin(tita_rad)

    h_max = (vy**2) / (2 * g)

    print("")

    print(f"La altura máxima es: {h_max:.2f} m")

elif opcion == 2:

    print("Ecuacion usada: t_vuelo = (2 * vi * sen(θ)) / g")

    print("")

    print("Ingrese la velocidad inicial en m/seg (vi): ")
    vi = float(input())

    print("")

    print("Ingrese el ángulo de lanzamiento en grados: ")
    tita = float(input())

    g = 9.81

    tita_rad = np.radians(tita)

    vy = vi * np.sin(tita_rad)

    t_vuelo = (2 * vy) / g

    print("")
```

```
        print(f"El tiempo de vuelo es: {t_vuelo:.2f} seg")

    elif opcion == 3:

        print("Ecuacion usada: alcance_max = vi * cos(θ) * (2 * vi * sen(θ)) / g")

        print("")

        print("Ingrese la velocidad inicial en m/seg (vi): ")
        vi = float(input())

        print("")

        print("Ingrese el angulo de lanzamiento en grados: ")
        tita = float(input())

        g = 9.81

        tita_rad = np.radians(tita)

        vx = vi * np.cos(tita_rad)
        vy = vi * np.sin(tita_rad)

        t_vuelo = (2 * vy) / g

        alcance_max = vx * t_vuelo

        print("")

        print(f"El alcance máximo es: {alcance_max:.2f} m")

    else:
        print("Opción no válida.")

def resolver_incongnitas_caida_libre():

    print("¿Que incognita quiere calcular o encontrar?")
    print("")
    print("1. Altura (y)")
    print("2. Tiempo (t)")

    print("")

    print("Ingrese la opción deseada: ")
    opcion = int(input())

    print("")

    if opcion == 1:

        print("Ecuacion Usada: y = 0.5 * g * t**2")

        print("")

        print("Ingrese el tiempo en segundos (t): ")
        t = float(input())
```

```
        g = 9.81
        y = 0.5 * g * t**2

        print("")

        print(f"La altura es: {y:.2f} m")

    elif opcion == 2:

        print("Ecuacion usada:  t = sqrt(2 * y / g)")

        print("Ingrese la altura en metros (y): ")
        y = float(input())

        g = 9.81
        t = np.sqrt(2 * y / g)

        print("")

        print(f"El tiempo es: {t:.2f} seg")

    else:
        print("Opcion no válida.")

# MENU ----------------------------------------------------------------------

def menu():

    while True:

        print("")
        print("-----------------------------------------------------")
        print("Seleccione una de las opciones:")
        print("-----------------------------------------------------")
        print("1. Graficar Movimiento Rectilíneo Uniforme (MRU)")
        print("2. Graficar Movimiento Rectilíneo Uniformemente Variado (MRUV)")
        print("3. Graficar Tiro Oblicuo")
        print("4. Graficar Caída Libre")
        print("5. Resolver incógnitas de MRU")
        print("6. Resolver incógnitas de MRUV")
        print("7. Resolver incógnitas de Tiro Oblicuo")
        print("8. Resolver incógnitas de Caída Libre")
        print("9. Salir")
        print("-----------------------------------------------------")

        print("")

# Opciones ------------------------------------------------------------------

        print("Ingrese una opcion: ")
        opcion = int(input())

        print("")

        if opcion == 1:
```

```
    print("Ecuacion usada: x = xi + v * t")

    print("")

    print("Ingrese la velocidad en m/seg: ")
    v = float(input())

    print("")

    print("Ingrese el tiempo total en seg: ")
    t_total = float(input())

    print("")

    print("Ingrese la posicion inicial en m: ")
    xi = float(input())

    d = MRU(v, t_total, xi)

    print("")

    print(f"Distancia recorrida: {d:.2f} m")

    # Graficar
    graficar_MRU(v, t_total, xi)

elif opcion == 2:

    print("Ecuacion usada: x = xi + vi * tiempo + 0.5 * a * tiempo^2")

    print("")

    print("Ingrese la velocidad inicial en m/seg: ")
    vi = float(input())

    print("")

    print("Ingrese la aceleración en m/seg^2: ")
    a = float(input())

    print("")

    print("Ingrese el tiempo total en seg: ")
    t_total = float(input())

    print("")

    print("Ingrese la posición inicial m: ")
    xi = float(input())

    d = MRUV(vi, a, t_total, xi)

    print("")

    print(f"Distancia recorrida: {d:.2f} m")
```

```
            graficar_MRUV(vi, a, t_total, xi)

        elif opcion == 3:

            print("ecuacion usada, para X: x = vi * cos(θ) * t, y para Y: y = vi * sen(θ) * t + 0.5 * g *
t**2")

            print("")

            print("Ingrese la velocidad inicial en m/seg: ")
            vi = float(input())

            print("")

            print("Ingrese el angulo de lanzamiento en grados: ")
            tita = float(input())

            print("")

            print("Ingrese el tiempo total en seg: ")
            t_total = float(input())

            d, h = calcular_altura_maxima_y_alcance(vi, tita)

            print("")

            print(f"Distancia horizontal recorrida: {d:.2f} m")

            print("")

            print(f"Altura maxima alcanzada: {h:.2f} m")

            graficar_tiro_oblicuo(vi, tita, t_total)



        elif opcion == 4:

            print("Ecuacion usada: y = 0.5 * g * t**2")

            print("")

            print("Ingrese el tiempo total en seg: ")
            t_total = float(input())

            h = caida_libre(t_total)

            print("")

            print(f"Altura alcanzada: {h:.2f} m")

            graficar_caida_libre(t_total)

        elif opcion == 5:
```

```
            resolver_incongnitas_MRU()

        elif opcion == 6:

            resolver_incongnitas_MRUV()

        elif opcion == 7:

            resolver_incongnitas_tiro_oblicuo()

        elif opcion == 8:

            resolver_incongnitas_caida_libre()

        elif opcion == 9:

            print("Fin")

            break

        else:
            print("Opción inválida.")


# Llamo a la funcion MENU
menu()
```