**Escuela Politécnica Nacional**

**Facultad de Ingeniería de Sistemas**

****

**Métodos Numéricos**

**Integrantes:**

Francis Alian Bravo Alemán

Matthew Israel Cedeño Diaz

Michael Javier Yánez Vela

**Proyecto Telescopio James Webb**

**Ing. Jonathan A. Zea**

Contenido

**No se encontraron entradas de tabla de contenido.**

1. **Introducción**

El telescopio espacial James Webb representa uno de los avances tecnológicos más significativos en la historia de la exploración del universo, marcando un hito en nuestra capacidad para comprender los orígenes del cosmos y los procesos que lo rigen. Este instrumento, diseñado para observar las profundidades del universo en longitudes de onda infrarrojas, combina ingeniería de vanguardia con una arquitectura óptica altamente sofisticada. Su sistema óptico incluye un espejo primario segmentado de 6.5 metros de diámetro y un espejo secundario ajustable, cuya precisión milimétrica es fundamental para garantizar imágenes claras y detalladas de galaxias, exoplanetas y otros fenómenos celestes.

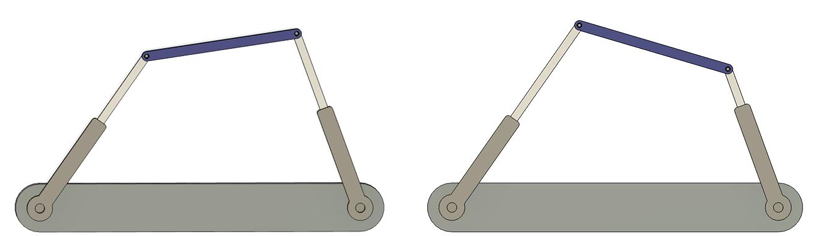


Figura 1: Modelo del Telescopio.

El espejo secundario, pieza clave en el sistema, emplea pistones lineales controlados con extrema exactitud para orientarse y mantener el alineamiento necesario en condiciones extremas del espacio profundo. Inspirados en este fascinante mecanismo, el presente proyecto propone la implementación de un sistema simplificado que modela el comportamiento de un espejo único controlado mediante pistones. Este sistema busca replicar, a pequeña escala, las complejas operaciones de orientación y ajuste, destacando la importancia de la precisión y la lógica computacional en la exploración espacial. A través de simulaciones y cálculos geométricos avanzados, este proyecto no solo rinde homenaje a la tecnología detrás del James Webb, sino que también introduce principios 4 fundamentales aplicables a sistemas ópticos modernos.

1. **Objetivos**

Formular un modelo matemático que represente el comportamiento de un mecanismo de orientación basado en pistones lineales, inspirado en el sistema del espejo secundario del telescopio James Webb. Este modelo permitirá controlar con exactitud la posición e inclinación del espejo respecto a un punto P dado, garantizando que el eje óptico quede perfectamente alineado con el blanco deseado.

Desarrollar una simulación gráfica del mecanismo para contrastar y validar el modelo propuesto en distintas configuraciones. En ella se analizará cómo los cambios en la longitud de los pistones influyen en la orientación del espejo, aportando datos sobre las tolerancias del sistema y sus márgenes de operación.

1. **Metodologia**

El desarrollo de la aplicación se basó en un enfoque modular y iterativo que consta de los siguientes pasos:

1. **Definición de requerimientos**: Identificar las variables clave (L, B, D, d\_max, F, Px, Py) que describen la geometría del espejo y la posición del punto de incidencia.
2. **Diseño modular**: Separar la lógica en cuatro componentes:
   * **Interfaz gráfica** (Tkinter): Entradas de usuario y presentación de resultados.
   * **Cálculo geométrico**: Determinación de longitudes de pistones y orientación del espejo.
   * **Dibujo del sistema** (Matplotlib): Renderizado visual de pistones, espejo y proyección.
   * **Flujo principal** (main.py): Punto de arranque, configuración de fuente e icono.
3. **Iteración y validación**: Probar distintos valores de Px, Py y verificar que el código encuentre soluciones dentro de los límites de los pistones o genere advertencias y recalcule con valores extremos.
4. **Ajustes de presentación**: Incorporar estilos globales (fuente Times New Roman, colores de fondo) y separación de responsabilidades en archivos independientes.

**3.1 Descripción de la solución**

La aplicación recibe parámetros de entrada que definen tanto la geometría del espejo secundario como la posición de un punto P en el espacio. A través de un proceso de barrido:

1. Se recorre un rango de posibles posiciones verticales My del espejo, centrado en Py.
2. Para cada My, se calcula el vector normal N desde P hasta un punto medio en el espejo, y de él se deriva el vector director D (rotación de 90°).
3. Se evalúan las longitudes de pistones x1 y x2 para ver si caen dentro de los límites [D, d\_max]. Si alguna combinación es válida, se detiene el barrido.
4. Si el barrido no arroja soluciones, se asignan valores extremos (d\_max, D) y se reorienta el espejo en función del ángulo al punto P.
5. Finalmente, se calcula el ángulo del espejo respecto a la horizontal, se muestra en pantalla y se dibuja el sistema (pistones, espejo, punto y proyección).

**3.2 Desarrollo matemático**

Dado:

* Punto P = (Px, Py)
* Eje espejo de largo L centrado en un punto variable M = (0, My)
* Vectores básicos:
  + Normal al espejo: N=M−P=(−Px,My−Py)N = M - P = (-Px, My - Py)
  + Vector director del espejo (perpendicular a N):



Para cada extremo del espejo:

* Extremo izquierdo: 
* Extremo derecho: 

Longitudes de pistón:



Condición de validez: D≤xi≤dmax Si no se cumple, usar valores extremos y recalcular con ángulo directo:



Ángulo final del espejo: α=θ+90 (o directamente α=atan2(Dy,Dx)).

**3.3 Diagrama de flujo / Pseudocódigo**

Inicio

Leer L, B, D, d\_max, F, Px, Py

Si Py < 0 o 0 ≥ L ≥ B ≥ d\_max → error y salir

encontrado ← falso

Para My en linspace(Py - L/2, Py + L/2):

N ← (-Px, My - Py)

si N == (0,0) → continuar

D ← rot90(N) / ||N||

E1 ← M - (L/2)\*D; E2 ← M + (L/2)\*D

x1 ← distancia(E1 + (B/2,0)); x2 ← distancia(E2 - (B/2,0))

si D ≤ x1 ≤ d\_max Y D ≤ x2 ≤ d\_max:

encontrado ← verdadero; romper

FinPara

Si ¬encontrado:

x1 ← d\_max; x2 ← D

θ ← atan2(Py, Px)

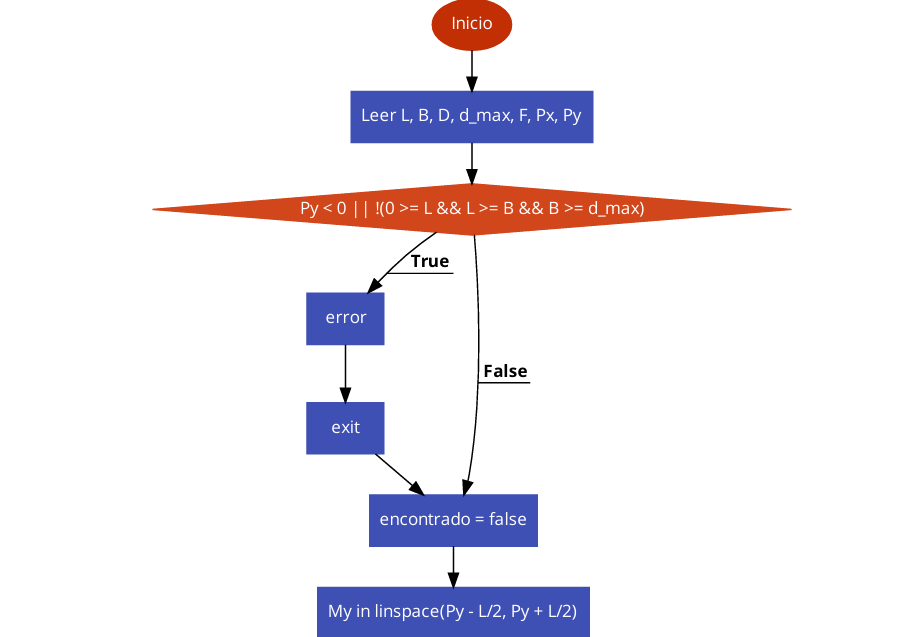
reemplazar E1, E2 con valores extremos y ajustar con θ

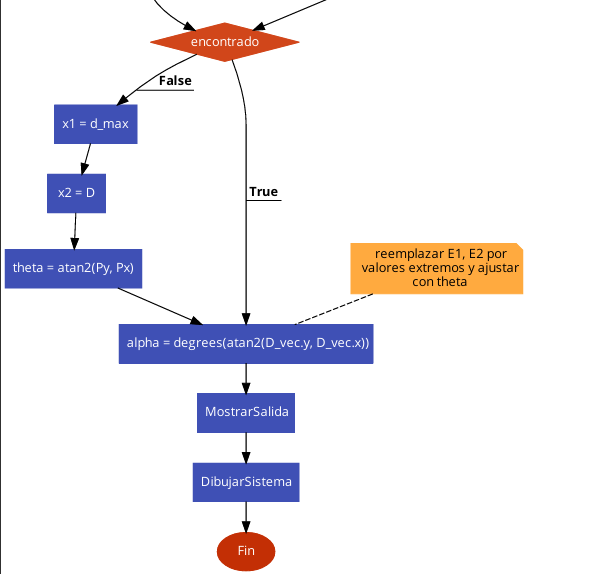
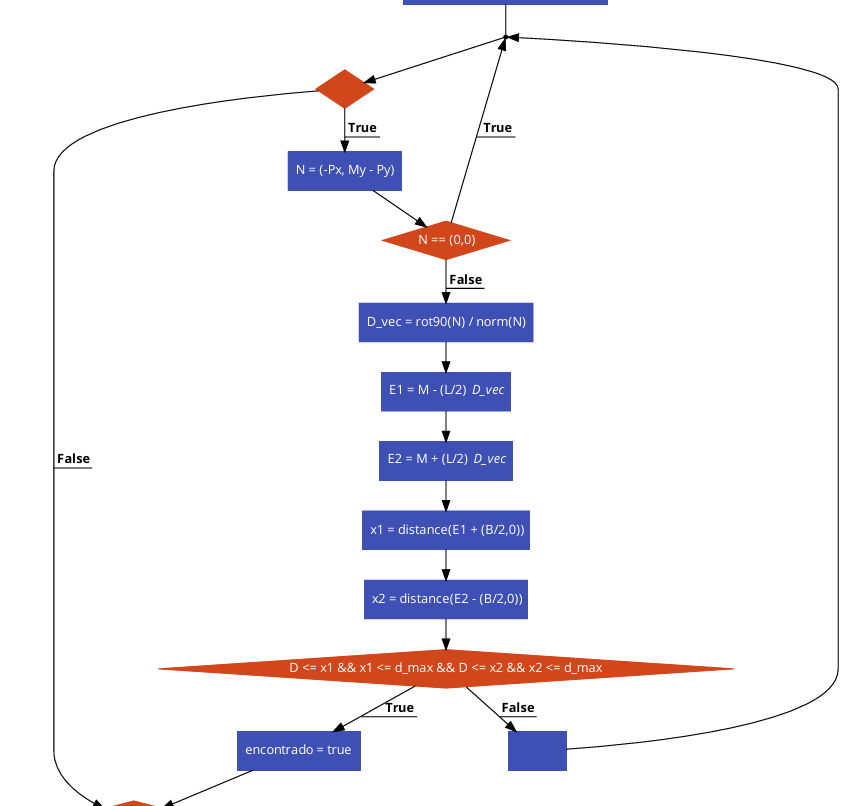
FinSi

α ← grados(atan2(D\_y, D\_x))

Mostrar texto de salida

Dibujar sistema

Fin



**3.4 Detalles importantes de la implementación**

**Modularidad**: Separación en widgets.py, calculo.py, dibujo.py y ventana.py para aislar responsabilidad.

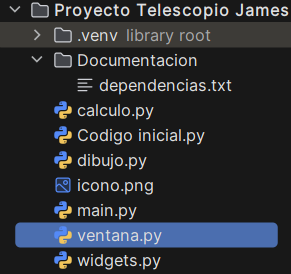
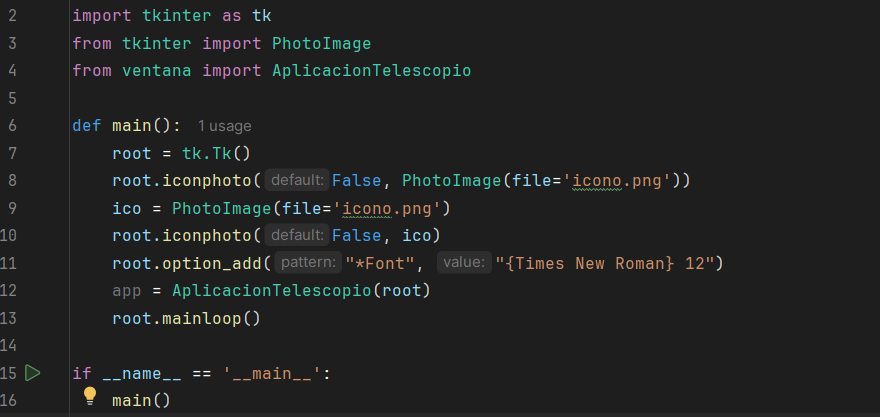


Figura 3. Modularidad

* + 1. **Main**

La función main() inicializa la ventana principal de Tkinter, le asigna icono y fuente global, crea la instancia de AplicacionTelescopio(root) y arranca el bucle de eventos (mainloop).



* + 1. **Calculo**

El módulo **calculo.py** contiene la función calcular\_y\_dibujar(self), que se encarga de:

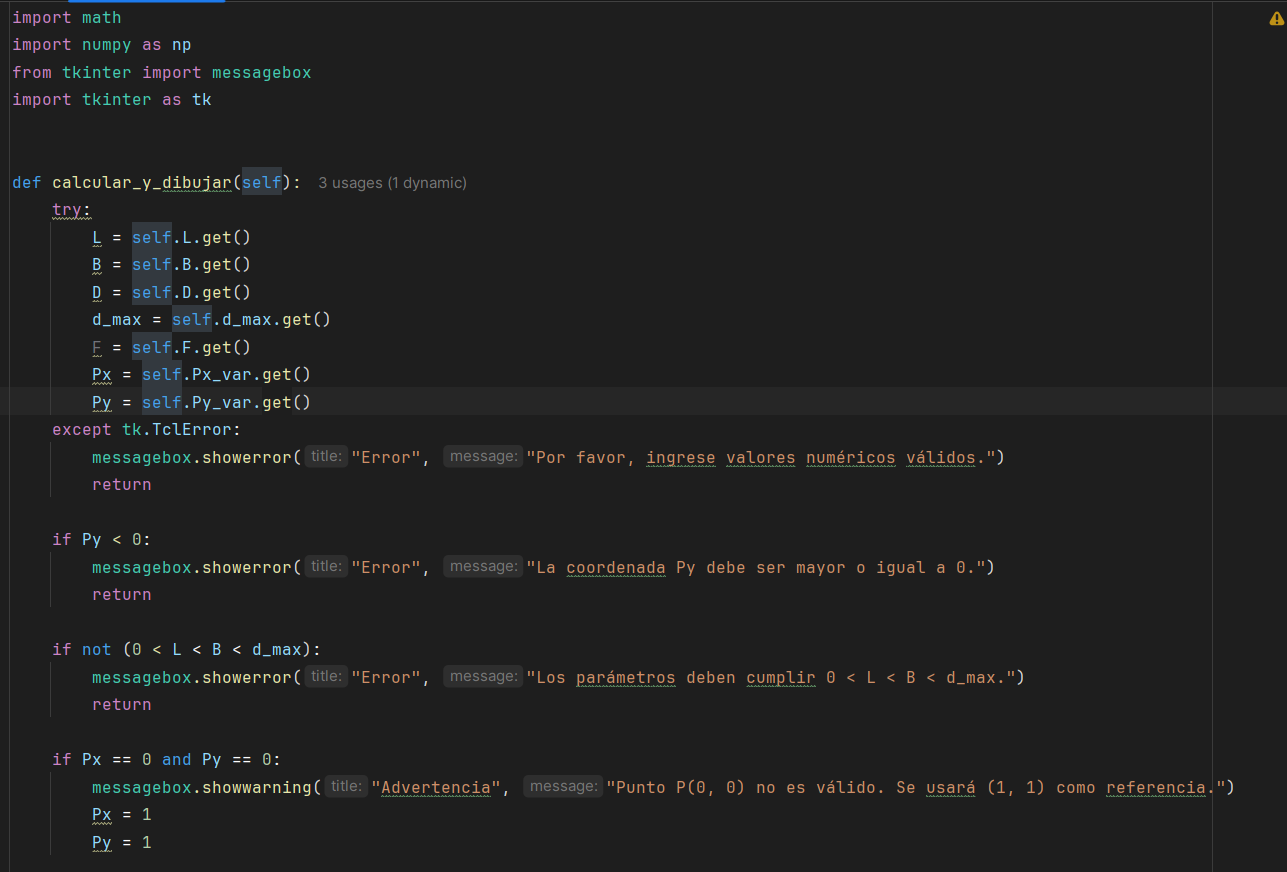
**Leer y validar** los parámetros de entrada (L, B, D, d\_max, F, Px, Py) desde los DoubleVar de la interfaz, mostrando errores si no son numéricos o no cumplen las restricciones geométricas.

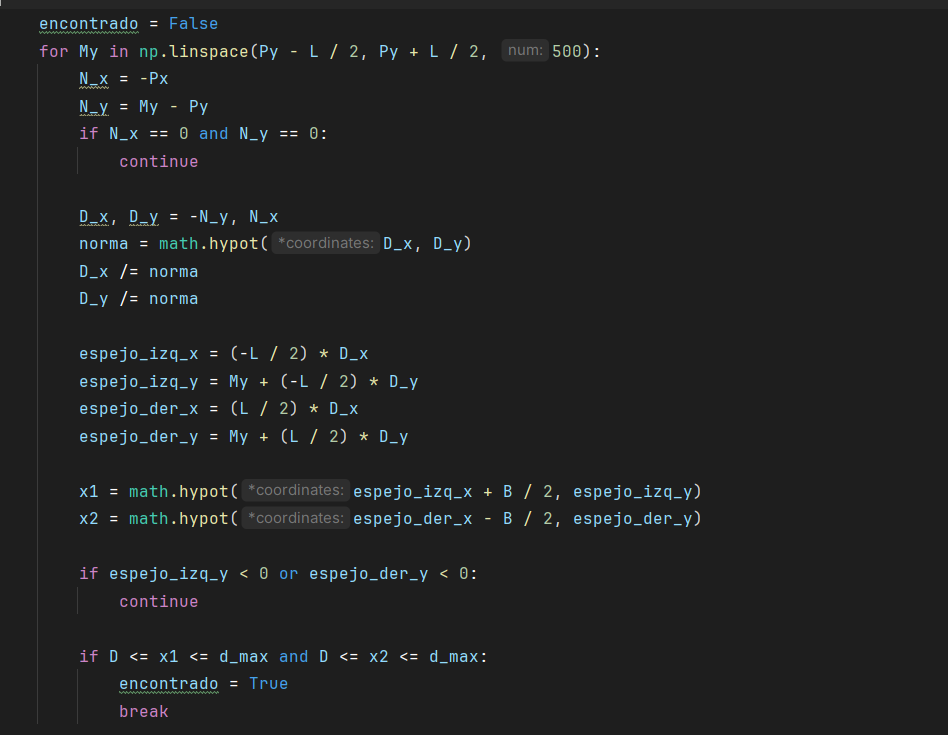
**Determinar** la posición óptima del espejo secundario mediante un barrido de su coordenada vertical My, calculando para cada paso los vectores normales y directores, y evaluando si las longitudes de pistones *x₁* y *x₂* quedan dentro de los límites permitidos.

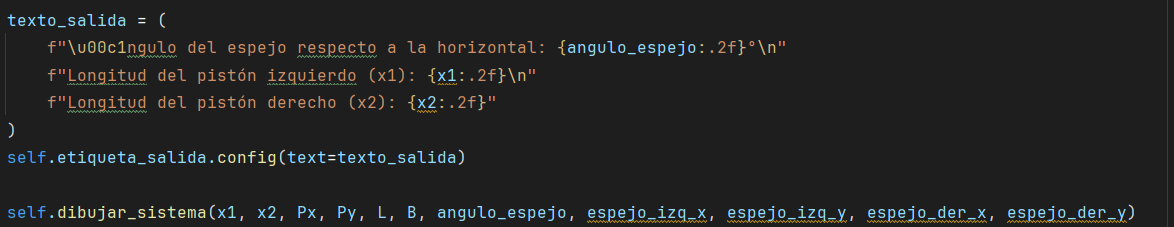
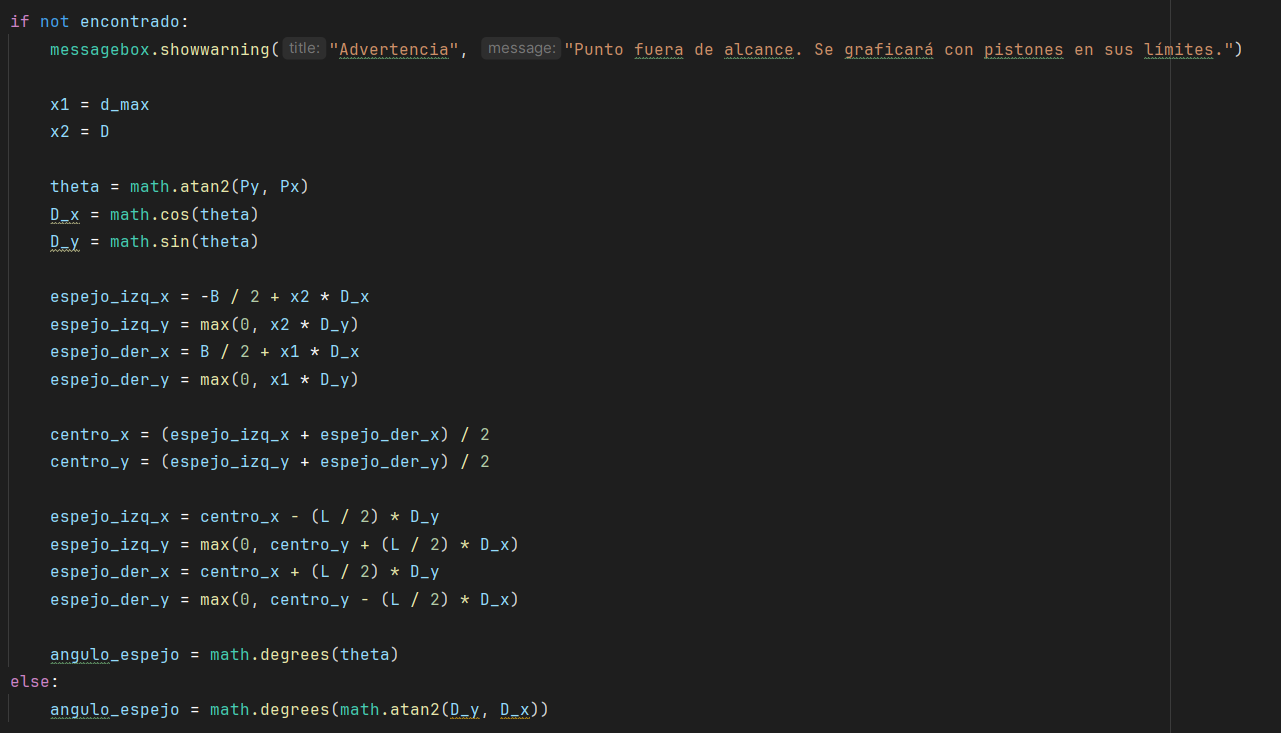
**Manejar** el caso en que no se encuentra solución válida, asignando valores extremos a los pistones y recalculando la orientación respecto al punto P.

**Calcular** el ángulo final del espejo y componer un texto resumen con esos resultados.

**Actualizar** la etiqueta de salida en la GUI con los valores calculados y llamar a dibujar\_sistema(...) para refrescar el gráfico con la nueva configuración.







* + 1. **Ventana**

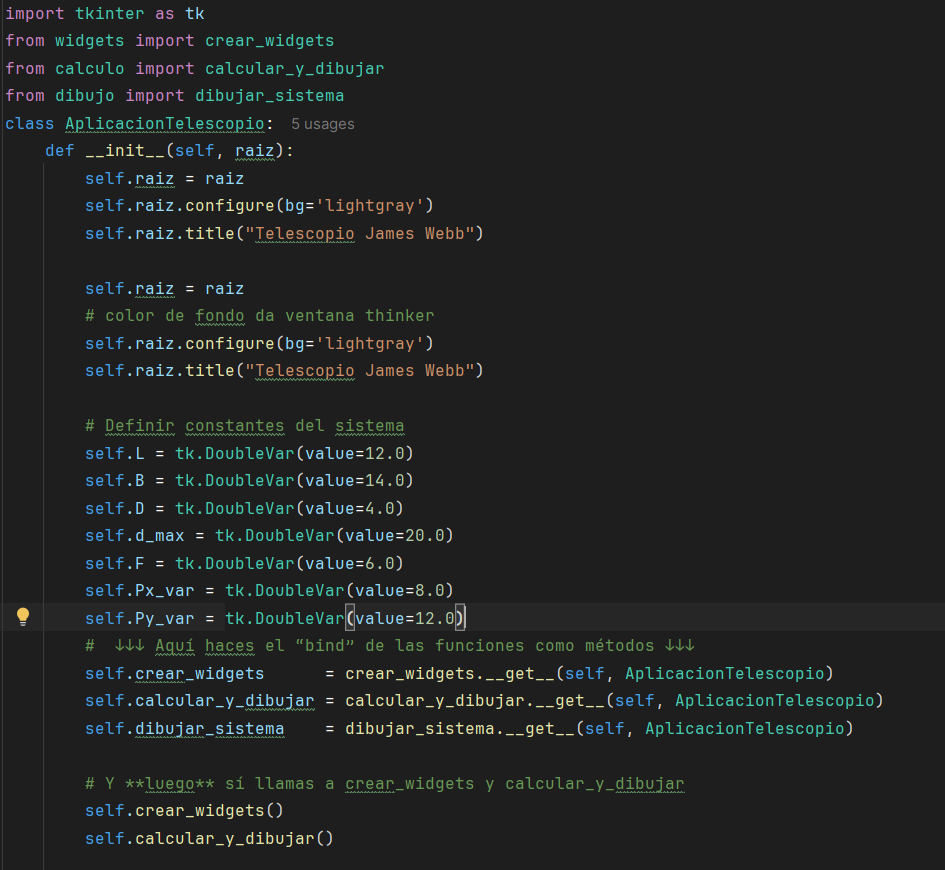
El módulo ventana.py define la clase AplicacionTelescopio, que:

Inicializa la ventana de Tkinter (fondo, título, icono, fuente).

Crea y almacena las variables DoubleVar que representan los parámetros del telescopio.

“Inyecta” (ligándolas con \_\_get\_\_) las funciones externas de creación de widgets, cálculo y dibujo como métodos de la clase.

Llama a crear\_widgets() para montar la interfaz y a calcular\_y\_dibujar() para mostrar el primer resultado y gráfico cuando la aplicación arranca.



* + 1. **Dibujo**

El módulo dibujo.py define la función `dibujar\_sistema(self, x1, x2, Px, Py, L, B, ángulo, espejo\_izq\_x, espejo\_izq\_y, espejo\_der\_x, espejo\_der\_y)`, que se encarga de:

1. Limpiar el eje (`ax.clear()`) y fijar el color de fondo.

2. Dibujar los elementos geométricos:

Líneas de los pistones izquierdo y derecho.

Línea gruesa del espejo.

Punto P como un marcador.

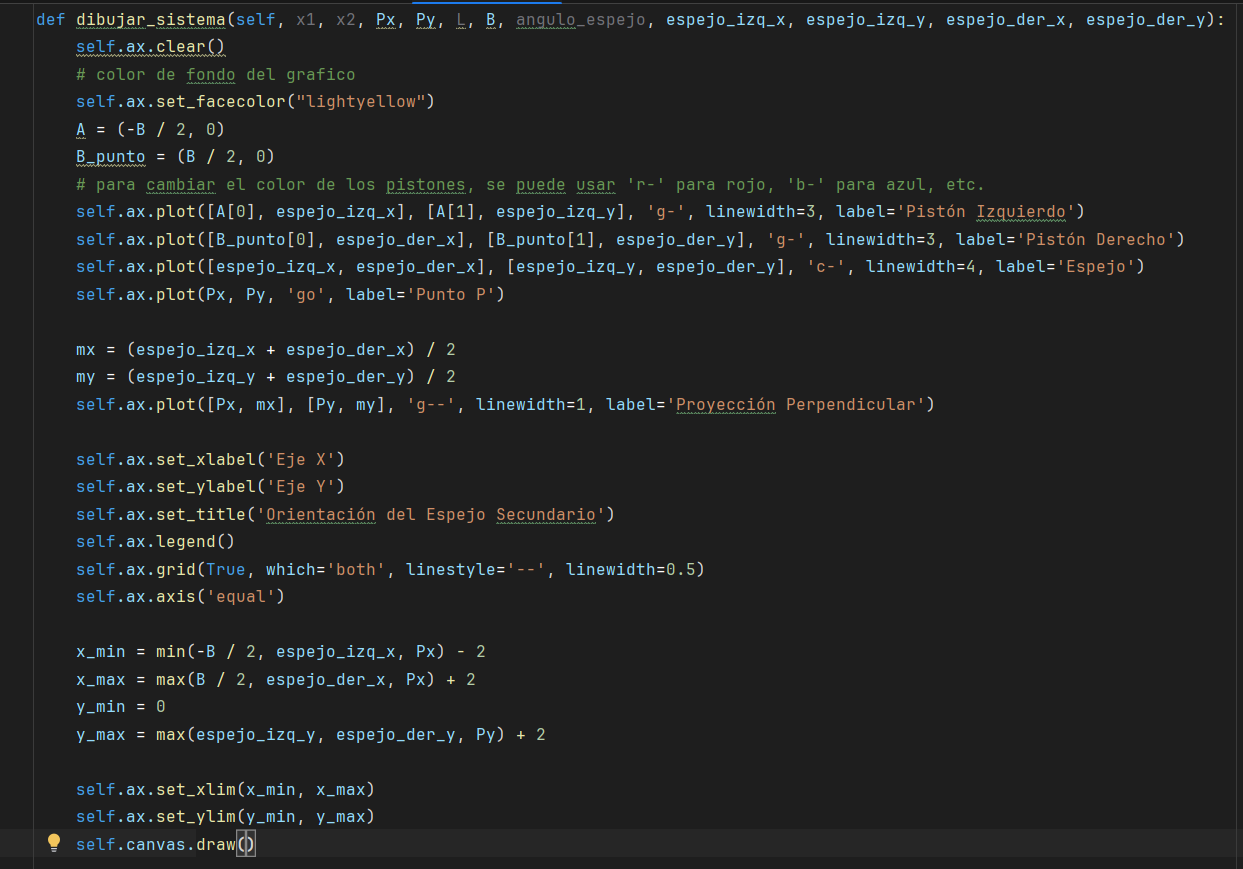
Línea punteada de la proyección perpendicular.

3. Etiquetar ejes (`Eje X`, `Eje Y`), título y leyenda.

4. Añadir la cuadrícula y mantener proporciones iguales (`axis('equal')`).

5. Ajustar los límites de los ejes para encuadrar todos los elementos.

6. Redibujar el canvas de Matplotlib en la ventana con `canvas.draw()`.

****

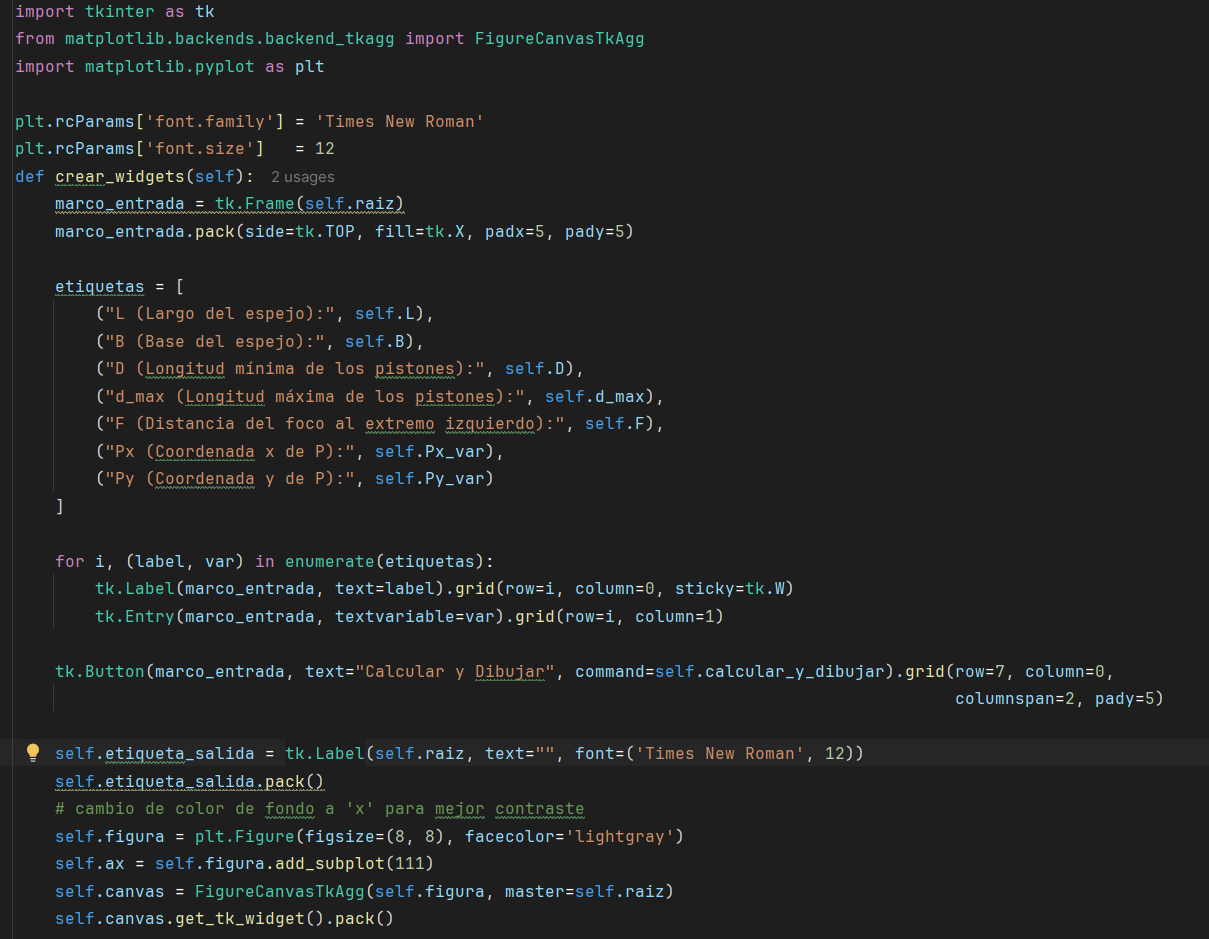
* + 1. **Witgets.py**

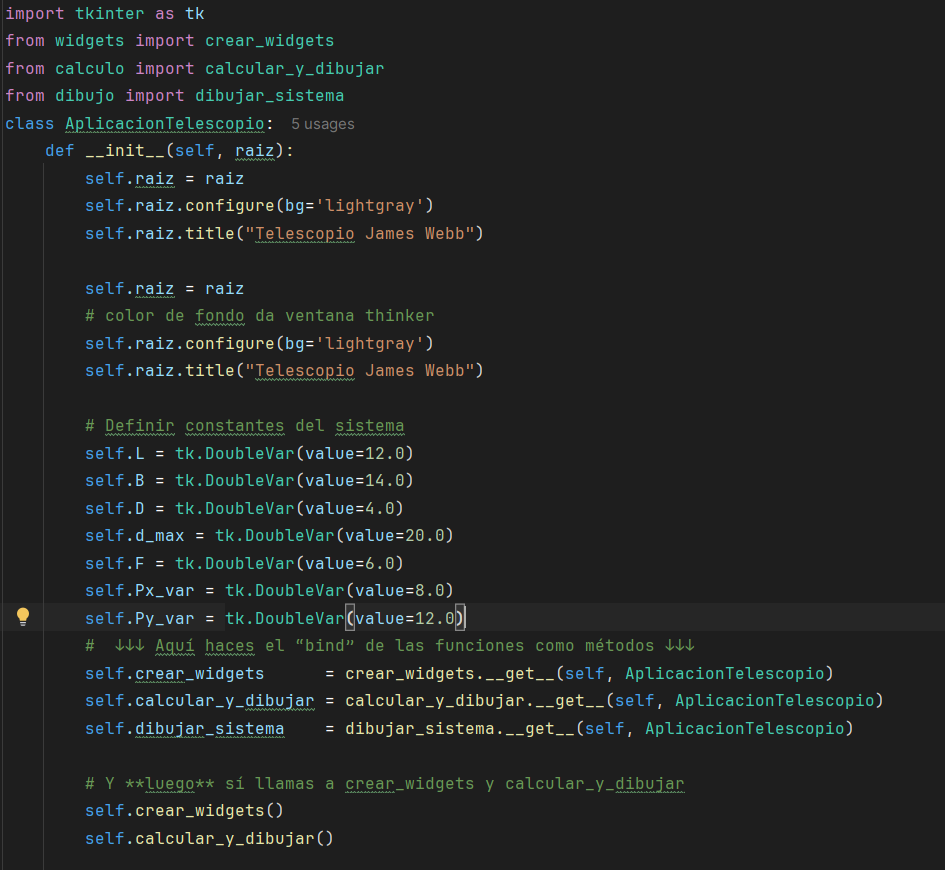
El módulo widgets.py define la función `crear\_widgets(self)`, que arma toda la interfaz gráfica:

1. Crea un frame de entrada y coloca en él etiquetas (`Label`) y campos de texto (`Entry`) para cada parámetro (L, B, D, d\\_max, F, Px, Py).

2. Añade un botón “Calcular y Dibujar” ligado al método de cálculo.

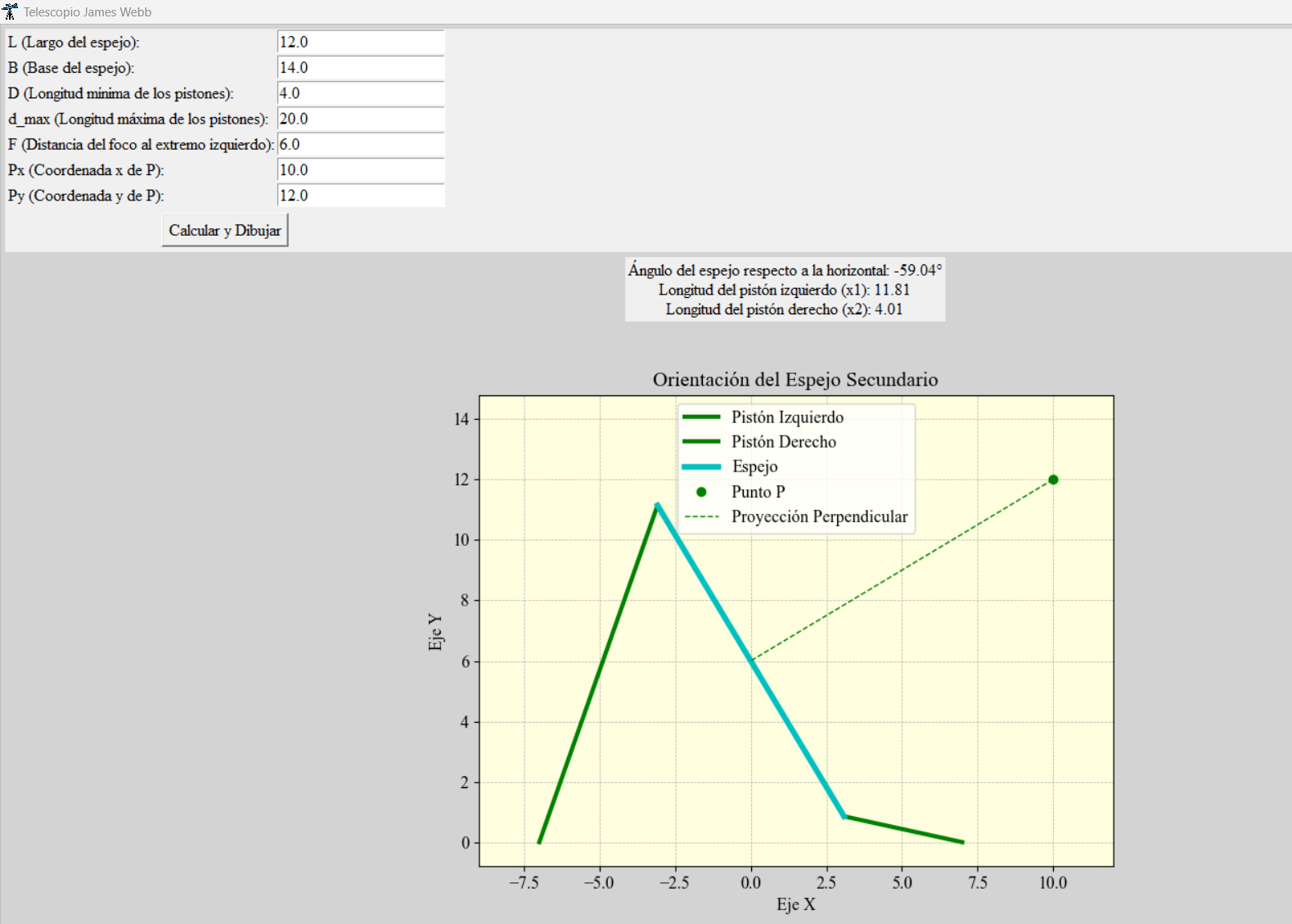
3. Genera la \*\*etiqueta de salida\*\* (`self.etiqueta\_salida`) donde se mostrarán los resultados numéricos.

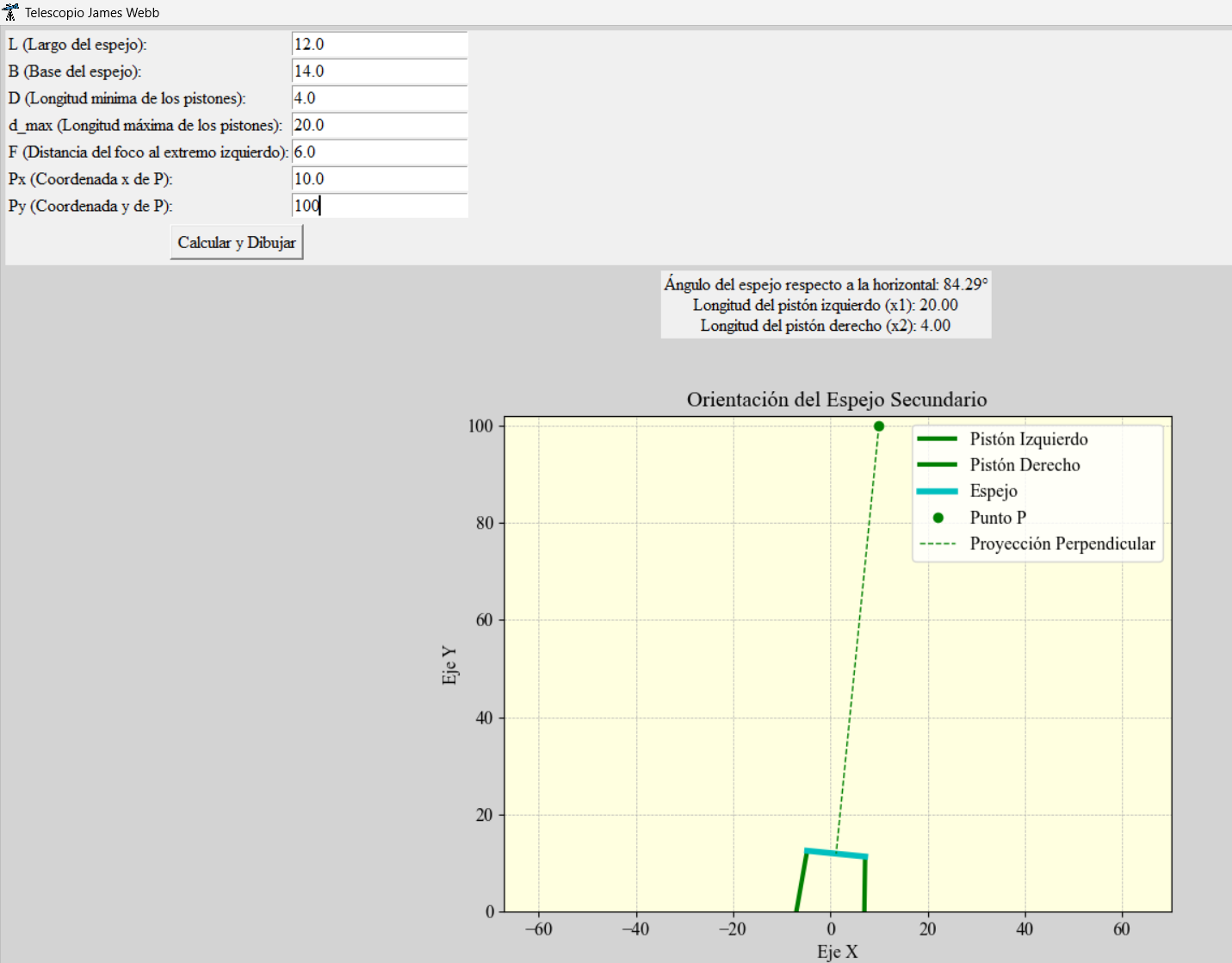
4. Crea la figura de Matplotlib (con fondo y fuente ya configurados), el subplot y el canvas que se incrusta en la ventana para renderizar el gráfico. 

****

1. **Resultados y conclusiones**

**4.1 Resultados**





**4.2 Conclusiones**

Se realizo un modelo matemático o que describe el funcionamiento del sistema de orientación basado en los pistones lineales permitiendo calcular la orientación y posición del espejo en función del punto de interés teniendo una alineación precisa del eje óptico, implementado restricciones geométricas y físicas permitiendo tener un correcto funcionamiento.

La simulación grafica facilita de manera intuitiva la comprensión del sistema del telescopio indicando las posiciones de los pistones y la posición del espejo