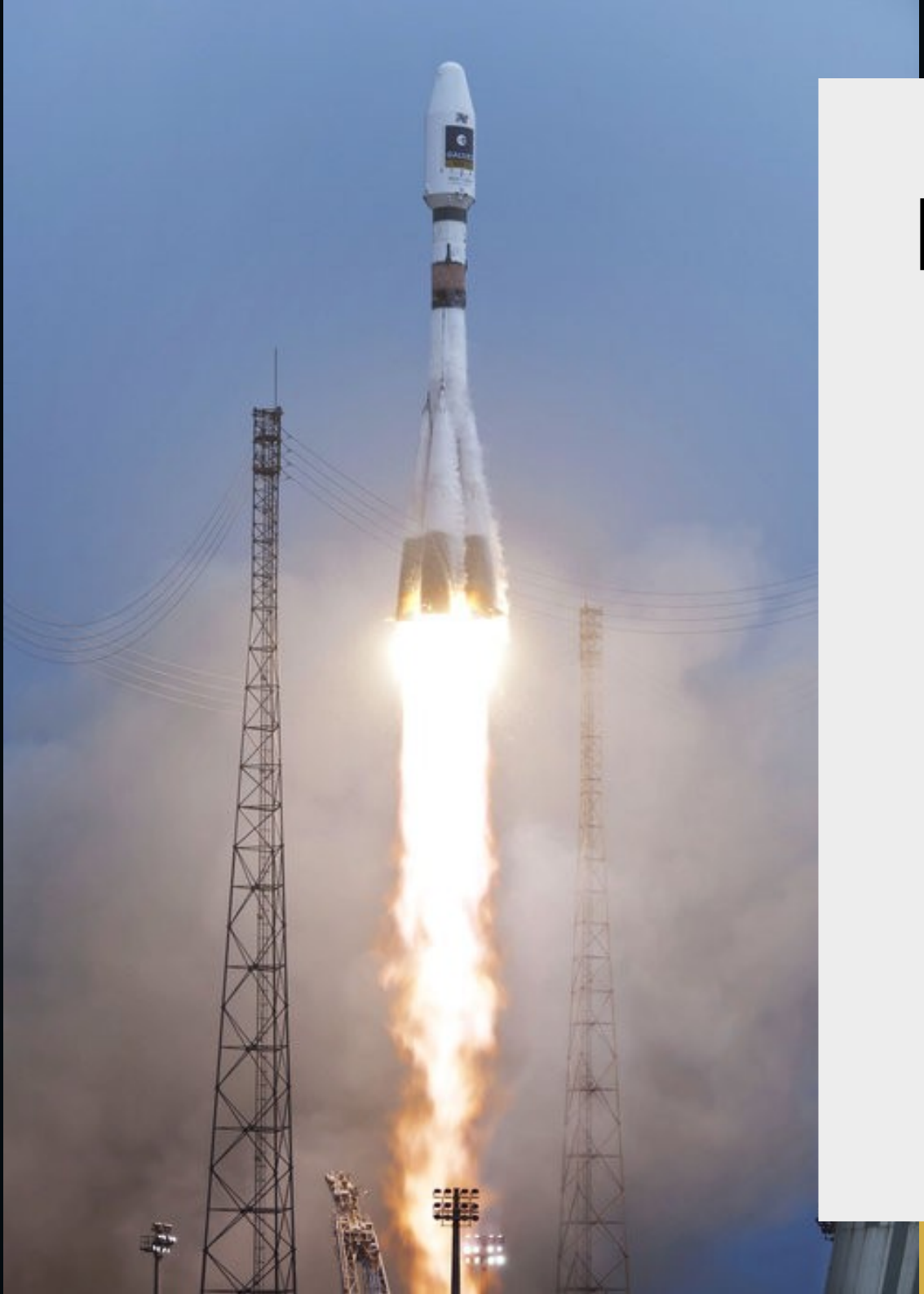


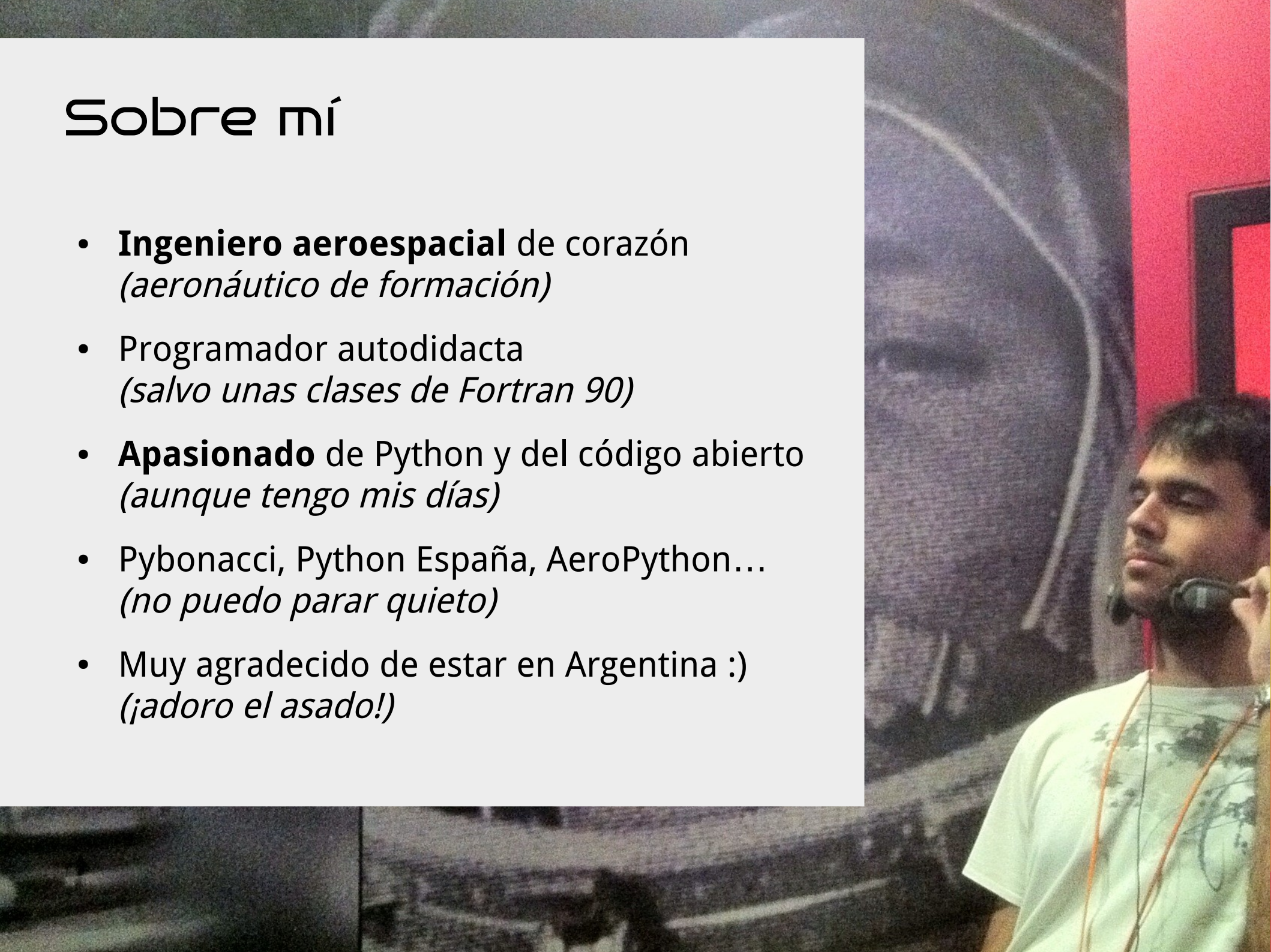
Per Python ad Astra

Python in Astrodynamics and Orbital Mechanics

Juan Luis Cano Rodríguez
<juanlu@pybonacci.org>
Posadas, Misiones, Argentina
2015-05-21

Sobre mí

- **Ingeniero aeroespacial** de corazón
  (aeronáutico de formación)
- Programador autodidacta
  (salvo unas clases de Fortran 90)
- **Apasionado** de Python y del código abierto
  (aunque tengo mis días)
- Pybonacci, Python España, AeroPython…
  (no puedo parar quieto)
- Muy agradecido de estar en Argentina :)
  (¡adoro el asado!)



Warning: This *is* rocket science!

Índice

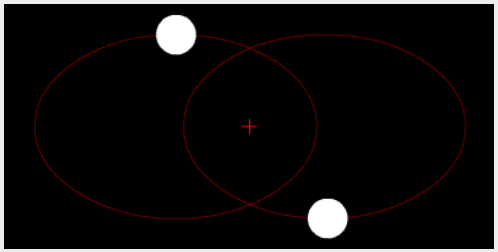
- ¿Qué es la astrodinámica?
- Problemas de Kepler y Lambert
- Mi proyecto personal: poliastro
- «A hombros de gigantes»: astropy, jplephem y numba
- Ventajas de usar solo Python
- Algunos desafíos y futuros desarrollos
- Conclusiones

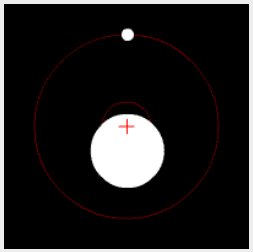
P: ¿Qué es la **astrodinámica**?
R: Jugar al billar… *a escala cósmica ;)*


esa

Problema de los dos cuerpos

- Problema primordial de la mecánica celeste
  - Dos **masas puntuales**
  - Fuerza gravitatoria **exclusivamente**
- ¡Los dos movimientos se desacoplan!





$$\ddot{\mathbf{r}} = -\frac{\mu}{r^2}\hat{\mathbf{r}}$$

Problema de Kepler

- Es el **problema de valor inicial** del problema de los dos cuerpos, también conocido como **propagación**
- *Problema: hallar la posición y velocidad de un satélite en un determinado instante, dados su posición y velocidad en un instante previo*
- Para órbitas elípticas:

$$M = E - e \sin E$$

$$M = n(t - t_0)$$

Problema de Lambert

- Es el **problema de contorno** del problema de los dos cuerpos
- *Problema: hallar la trayectoria entre dos posiciones a recorrer en un intervalo de tiempo dado*
- En transferencias interplanetarias se usa la “patched conic approximation”: se reduce un problema de tres cuerpos a tres problemas de dos cuerpos

poliastro: los inicios

- Proyecto de clase: transferencia Tierra-Venus y análisis de perturbaciones
- Objetivos:
  - Reutilizar software existente
  - Aprender Python
  - ¿Aprobar la asignatura…?
- Resultado: **poliastro**, biblioteca Python para mecánica orbital

POLITECNICO DI MILANO

School of Industrial Engineering
Space Engineering

Orbital Mechanics Project

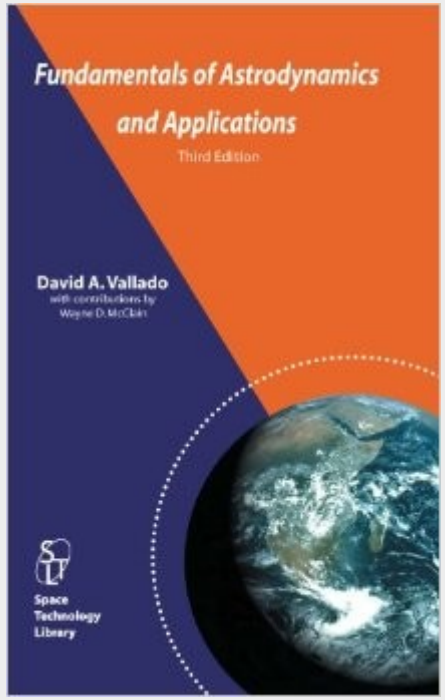
ACADEMIC YEAR 2012-2013

Authors:
Juan Luis CANO - 801222

July 3, 2013

Algoritmos y lenguajes compilados

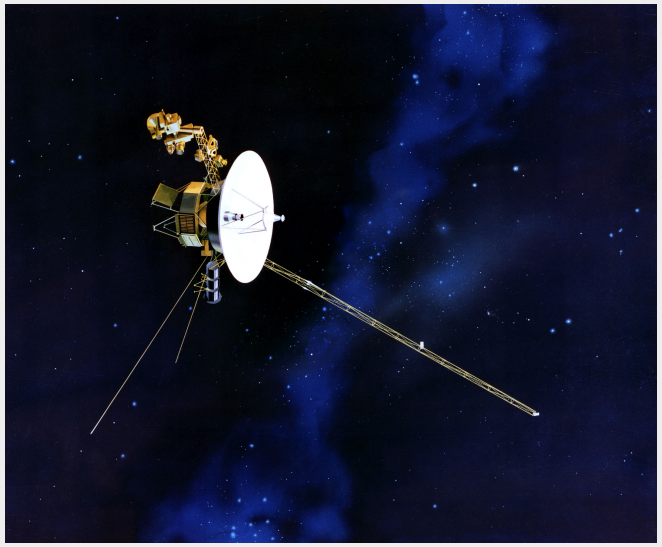
- La mayoría de aplicaciones requieren resolver estos problemas miles de veces
  - Trazas orbitales
  - Ventanas de lanzamiento
  - Optimización de trayectorias
- En Internet: Fortran, C, MATLAB, Java
  - Ventajas: Buen rendimiento sin mucho trabajo de optimización
  - Desventajas: Los avances en calidad del código de los últimos años brillan por su ausencia
- ¡Esto tenía que cambiar!



astropy: Astronomía en Python

- Base común de cualquier desarrollo futuro de astronomía en Python
  - **Unidades físicas** (`astropy.units`): declaración de tipos para ingenieros 😀
  - **Tiempos y fechas** (`astropy.time`): vectores de tiempos, conversión a fechas julianas (JD)
  - Cosas que usaré: **conversión entre sistemas de referencia** (`astropy.coordinates`)
- Otras cosas interesantes: cálculos cosmológicos (`astropy.cosmology`), datos FITS (`astropy.io.fits`)

jplephem: efemérides planetarias



- La NASA nos brinda datos con los que calcular las posiciones de los planetas con gran precisión (**efemérides**) en forma de archivos binarios (SPK kernels)
- jplephem, escrita por Brandon Rhodes♥, permite leer archivos SPK

♥Otras bibliotecas: **python-sgp4**, **python-skyfield**

numba: JIT para Python

- Compilador “just-in-time” para código Python *numérico*
- Optimizado para trabajar con arrays de NumPy
- Prueba: problemas de Kepler y Lambert en Python acelerados con numba
- Son algoritmos simples resueltos iterativamente (método de Newton)
- **Resultado: ~90 % del rendimiento de Fortran**

¡Adiós Fortran!

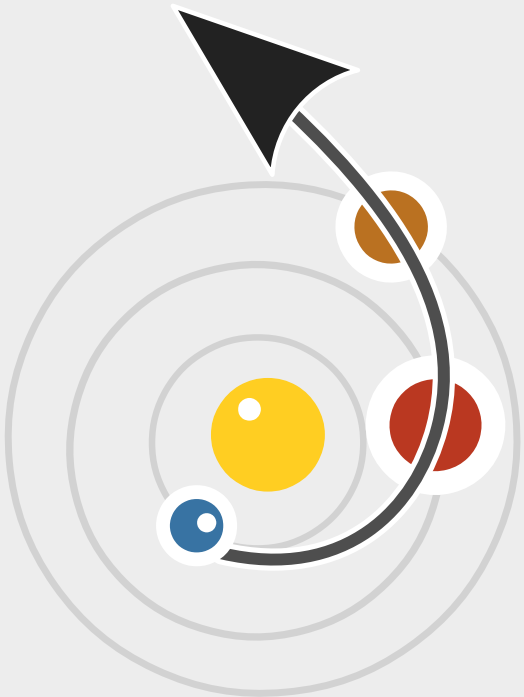
Ventajas de usar (solo) Python

- **Máxima simplicidad de instalación**[1]
- **Accesibilidad**: Python es un lenguaje legible y popular[2], ¡más contribuidores!
- **Propósito general**: Herramientas y documentación de máximo nivel
- **Introspección**: ¡métricas sobre la calidad del código!
- **No soy programador**: Python es el último lenguaje que quiero aprender[3]

[1] ¡Funciona con pip! numba es opcional
[2] Tercero en GitHub (`http://githut.info`)
[3] ¿O el penúltimo…?

¡Dentro demo!

https://github.com/poliastro/poliastro



Desafíos y futuros desarrollos

- Mantener compromiso legibilidad/complejidad
- Mejores algoritmos
- Representación de órbitas en 3D y trazas
- Diversas representaciones orbitales
- NEO (Near-Earth Objects): propagación SGP4, lectura de TLE

Mil gracias a todos