c195161lculo-de-viajes-espaciales

October 9, 2023

T.A.R.S. - AstroCalc: Módulo de Información Planetaria y Cálculo de Viajes Espaciales Autor: J. Gabriel Gudiño Lara
Equipo de Desarrollo: Endurance
Fecha de Creación:07-10-2023
DescripciónModulo filtrador y procesador de datos para T.A.R.S.

1 TARS - AstroCalc: Módulo de Información Planetaria y Cálculo de Viajes Espaciales

Descripción:

Este módulo está dedicado a la recolección, procesamiento y presentación de información relacionada con las características físicas de los planetas, así como a la estimación de tiempos y costos de viajes espaciales utilizando diferentes cohetes.

Fuentes de Datos: - API SCOUT de la NASA - API SOLAR SYSTEM OPEN DATA - Bases de datos de Kaggle solar system

Funcionalidades Principales:

1. **Recolección y Unificación de Datos:**
   - Se obtuvo y unificó la información de diversas fuentes en un solo archivo CSV.
2. **Procesamiento con Pandas:**
   - Se realizó una limpieza y análisis inicial de los datos usando la librería pandas.
   - Se generó un glosario para entender mejor cada columna o característica de los planetas.
3. **Serializacion a JSON:**
   - Se creó un dataframe final que el equipo de herramientas de integración adaptó para convertir la información en formato JSON, compatible con TARS, que está desarrollado en JavaScript.
4. **Cálculo de Posiciones Planetarias:**
   - Se desarrolló un algoritmo utilizando las librerías `astropy` y `skyfield` para determinar la posición de los planetas en coordenadas (x,y,z) con respecto al Sol, ya sea en tiempo real o en una fecha específica.
   - Se convirtieron las coordenadas a coordenadas eclípticas para facilitar el cálculo de distancias entre planetas.
   - Se determinaron los rangos en los que la librería puede calcular las coordenadas de los planetas.
5. **Diccionario de Cohetes:**

- Se creó un diccionario con las características, "amenidades de viaje", y costo hipotético de cada cohete para que TARS pueda informar al usuario y ayudarle a tomar decisiones sobre su viaje.

6. **Estimación de Tiempos de Viaje:**
   - Usando las distancias entre planetas y la velocidad de cohetes específicos, se estimó el tiempo de llegada que TARS utiliza para proporcionar respuestas precisas al usuario.
   - Utilizando una funcion que calcula la fecha donde la distancia entre planetas se generó un dataframe con las fechas optimas de viaje donde la distancia es minima por la tanto el tiempo y costo tambien.

Integración: - Todas las herramientas y funcionalidades desarrolladas en este módulo fueron integradas por el equipo de desarrollo de herramientas de integración.

1.1 Informacion planetaria:

1.2 Dataset Glossary

Planet - Name of the Planet.

Color - Color of the Planet.

Mass (10^24kg) - This is the mass of the planet in septillion (1 followed by 24 zeros) kilograms or sextillion (1 followed by 21 zeros) tons. Strictly speaking, tons are measures of weight, not mass, but are used here to represent the mass of one ton of material under Earth gravity.

Diameter (km) - The diameter of the planet at the equator, the distance through the center of the planet from one point on the equator to the opposite side, in kilometers or miles.

Density (kg/m^3) - The average density (mass divided by volume) of the whole planet (not including the atmosphere for the terrestrial planets) in kilograms per cubic meter or pounds per cubic foot. Strictly speaking, pounds are measures of weight, not mass, but are used here to represent the mass of one pound of material under Earth's gravity.

Gravity (m/s^2) - The gravitational acceleration on the surface at the equator in meters per second squared or feet per second squared, including the effects of rotation. For the gas giant planets, the gravity is given at the 1 bar pressure level in the atmosphere. The gravity on Earth is designated as 1 "G", so the Earth ratio fact sheets give the gravity of the other planets in G's.

Escape Velocity (km/s) - Initial velocity, in kilometers per second or miles per second, needed at the surface (at the 1 bar pressure level for the gas giants) to escape the body's gravitational pull, ignoring atmospheric drag.

Rotation Period (hours) - This is the time it takes for the planet to complete one rotation relative to the fixed background stars (not relative to the Sun) in hours. Negative numbers indicate retrograde (backward relative to the Earth) rotation.

Length of Day (hours) - The average time in hours for the Sun to move from the noon position in the sky at a point on the equator back to the same position.

Distance from Sun (10^6 km) - This is the average distance from the planet to the Sun in millions of kilometers or millions of miles, also known as the semi-major axis. All planets have orbits that are elliptical, not perfectly circular, so there is a point in the orbit at which the planet is closest to the Sun, the perihelion, and a point furthest from the Sun, the aphelion. The average distance from the Sun is midway between these two values. The average distance from the Earth to the Sun is defined as 1 Astronomical Unit (AU), so the ratio table gives this distance in AU.

Perihelion, Aphelion (10^6 km) - The closest and furthest points in a planet's orbit about the Sun, see "Distance from Sun" above.

Orbital Period (days) - This is the time in Earth days for a planet to orbit the Sun from one vernal equinox to the next. Also known as the tropical orbit period, this is equal to a year on Earth.

Orbital Velocity (km/s) - The average velocity or speed of the planet as it orbits the Sun, in kilometers per second or miles per second.

Orbital Inclination (degrees) - The angle in degrees at which a planets orbit around the Sun is tilted relative to the ecliptic plane. The ecliptic plane is defined as the plane containing the Earths orbit, so the Earth's inclination is 0.

Orbital Eccentricity - This is a measure of how far a planets orbit about the Sun (or the Moons orbit about the Earth) is from being circular. The larger the eccentricity, the more elongated the orbit, an eccentricity of 0 means the orbit is a perfect circle. There are no units for eccentricity.

Obliquity to Orbit (degrees) - The angle in degrees the axis of a planet (the imaginary line running through the center of the planet from the north to south poles) is tilted relative to a line perpendicular to the planet's orbit around the Sun, north pole defined by the right-hand rule. Venus rotates in a retrograde direction, opposite the other planets, so the tilt is almost 180 degrees, it is considered to be spinning with its "top", or north pole pointing "downward" (southward). Uranus rotates almost on its side relative to the orbit, Pluto is pointing slightly "down". The ratios with Earth refer to the axis without reference to north or south.

Mean Temperature (C) - This is the average temperature over the whole planets surface (or for the gas giants at the one bar level) in degrees C (Celsius or Centigrade) or degrees F (Fahrenheit). For Mercury and the Moon, for example, this is an average over the sunlit (very hot) and dark (very cold) hemispheres and so is not representative of any given region on the planet, and most of the surface is quite different from this average value. As with the Earth, there will tend to be variations in temperature from the equator to the poles, from the day to night sides, and seasonal changes on most planets.

Surface Pressure (bars) -

Number of Moons - This gives the number of IAU officially confirmed moons orbiting the planet. New moons are still being discovered.

Ring System? - This tells whether a planet has a set of rings around it, Saturn being the most obvious example.

Global Magnetic Field? - This tells whether the planet has a measurable large-scale magnetic field. Mars and the Moon have localized regional magnetic fields but no global field.

Surface Temperature (C) - Surface temperature of the planet in degrees Celsius.

Atmospheric Composition - Composition of the planet's atmosphere.

Atmospheric Pressure (bars) - Atmospheric pressure throughout the planet's atmosphere in bars.

Surface Features - Notable features on the planet's surface.

Composition - Composition of the planet's materials.

[1]:
```python
# Cargar todas las librerías
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import re
from astropy.coordinates import solar_system_ephemeris, get_body
from astropy.coordinates.sky_coordinate import SkyCoord
import astropy.units as u
from astropy.time import Time
from astropy import coordinates as coord
from skyfield.api import load
from skyfield.errors import EphemerisRangeError
from datetime import datetime
import math
```

[2]:
```python
# Mostrar base de datos
planets_df = pd.read_csv('planets_updated.csv')
display(planets_df)
```

```
    Planet                                          Color  Mass (10^24kg)  \
0  Mercury                                           Grey           0.330
1    Venus                                 Brown and Grey           4.870
2    Earth                   Blue, Brown Green and White           5.970
3     Mars                             Red, Brown and Tan           0.642
4  Jupiter  Brown, Orange and Tan, with White cloud stripes        1898.000
5   Saturn                     Golden, Brown, and Blue-Grey         568.000
6   Uranus                                     Blue-Green          86.800
7  Neptune                                           Blue         102.000

   Diameter (km)  Density (kg/m^3)  Surface Gravity(m/s^2)  \
0           4879              5429                     3.7
1          12104              5243                     8.9
2          12756              5514                     9.8
3           6792              3934                     3.7
4         142984              1326                    23.1
5         120536               687                     9.0
6          51118              1270                     8.7
7          49528              1638                    11.0

   Escape Velocity (km/s)  Rotation Period (hours)  Length of Day (hours)  \
0                     4.3                   1407.6                 4222.6
1                    10.4                  -5832.5                 2802.0
```

2	11.2	23.9	24.0
3	5.0	24.6	24.7
4	59.5	9.9	9.9
5	35.5	10.7	10.7
6	21.3	-17.2	17.2
7	23.5	16.1	16.1

	Distance from Sun (10^6 km)	…	Mean Temperature (C)	\
0	57.9	…	167	
1	108.2	…	464	
2	149.6	…	15	
3	228.0	…	-65	
4	778.5	…	-110	
5	1432.0	…	-140	
6	2867.0	…	-195	
7	4515.0	…	-200	

	Surface Pressure (bars)	Number of Moons	Ring System?	\
0	0	0	No	
1	92	0	No	
2	1	1	No	
3	0.01	2	No	
4	Unknown	79	Yes	
5	Unknown	82	Yes	
6	Unknown	27	Yes	
7	Unknown	14	Yes	

	Global Magnetic Field?	Surface Temperature (C)	Atmospheric Composition	\
0	Yes	-173 to 427	Mostly None	
1	No	462	Carbon Dioxide (96.5%)	
2	Yes	-89 to 58	Nitrogen (78.1%), Oxygen	
3	No	-153 to 20	Carbon Dioxide (95.3%)	
4	Yes	-108 to -150	Hydrogen, Helium	
5	Yes	-178 to -228	Hydrogen, Helium	
6	Yes	-197	Hydrogen, Helium	
7	Yes	-201	Hydrogen, Helium	

	Atmospheric Pressure (bars)	Surface Features	\
0	Trace	Craters, Scarps	
1	92	Volcanoes, Venusian Plains	
2	1	Mountains, Oceans, Forests	
3	0.006	Valles Marineris, Olympus Mons	
4	Unknown	Great Red Spot, Jupiter's Rings	
5	Unknown	Rings, Cassini Division, Saturn's Hexagon	
6	Unknown	Rings, Miranda's Cliff, Oberon's Craters	
7	Unknown	Great Dark Spot, Triton's Geysers	

Composition

```
0        Rock and Metal
1        Rock and Metal
2  Rock, Water, and Air
3          Rock and Ice
4        Gas and Liquid
5        Gas and Liquid
6           Gas and Ice
7           Gas and Ice

[8 rows x 27 columns]
```

[3]:
```
planets_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 27 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Planet                         8 non-null      object
 1   Color                          8 non-null      object
 2   Mass (10^24kg)                 8 non-null      float64
 3   Diameter (km)                  8 non-null      int64
 4   Density (kg/m^3)               8 non-null      int64
 5   Surface Gravity(m/s^2)         8 non-null      float64
 6   Escape Velocity (km/s)         8 non-null      float64
 7   Rotation Period (hours)        8 non-null      float64
 8   Length of Day (hours)          8 non-null      float64
 9   Distance from Sun (10^6 km)    8 non-null      float64
 10  Perihelion (10^6 km)           8 non-null      float64
 11  Aphelion (10^6 km)             8 non-null      float64
 12  Orbital Period (days)          8 non-null      object
 13  Orbital Velocity (km/s)        8 non-null      float64
 14  Orbital Inclination (degrees)  8 non-null      float64
 15  Orbital Eccentricity           8 non-null      float64
 16  Obliquity to Orbit (degrees)   8 non-null      float64
 17  Mean Temperature (C)           8 non-null      int64
 18  Surface Pressure (bars)        8 non-null      object
 19  Number of Moons                8 non-null      int64
 20  Ring System?                   8 non-null      object
 21  Global Magnetic Field?         8 non-null      object
 22  Surface Temperature (C)        8 non-null      object
 23  Atmospheric Composition        8 non-null      object
 24  Atmospheric Pressure (bars)    8 non-null      object
 25  Surface Features               8 non-null      object
 26  Composition                    8 non-null      object
dtypes: float64(12), int64(4), object(11)
memory usage: 1.8+ KB
```

[4]:
```
planets_df.describe()
```

[4]:
```
       Mass (10^24kg)  Diameter (km)  Density (kg/m^3)  \
count        8.000000       8.000000          8.000000
mean       333.326500   50087.125000       3130.125000
std        660.538057   53916.366175       2104.022368
min          0.330000    4879.000000        687.000000
25%          3.813000   10776.000000       1312.000000
50%         46.385000   31142.000000       2786.000000
75%        218.500000   68472.500000       5289.500000
max       1898.000000  142984.000000       5514.000000

       Surface Gravity(m/s^2)  Escape Velocity (km/s)  \
count                8.000000                 8.00000
mean                 9.737500                21.33750
std                  6.040089                18.67473
min                  3.700000                 4.30000
25%                  7.450000                 9.05000
50%                  8.950000                16.25000
75%                 10.100000                26.50000
max                 23.100000                59.50000

       Rotation Period (hours)  Length of Day (hours)  \
count                 8.000000                8.00000
mean               -544.612500              890.90000
std                2191.819718             1661.92169
min               -5832.500000                9.90000
25%                   3.125000               14.75000
50%                  13.400000               20.60000
75%                  24.075000              719.02500
max                1407.600000             4222.60000

       Distance from Sun (10^6 km)  Perihelion (10^6 km)  Aphelion (10^6 km)  \
count                     8.000000              8.000000            8.000000
mean                   1267.025000           1226.162500         1307.912500
std                    1626.047146           1599.115396         1653.781713
min                      57.900000             46.000000           69.800000
25%                     139.250000            137.200000          141.300000
50%                     503.250000            473.650000          532.850000
75%                    1790.750000           1701.375000         1880.225000
max                    4515.000000           4471.100000         4558.900000

       Orbital Velocity (km/s)  Orbital Inclination (degrees)  \
count                 8.000000                       8.000000
mean                 21.412500                       2.325000
std                  15.183397                       2.150581
min                   5.400000                       0.000000
```

```
25%                    8.975000                        1.175000
50%                   18.600000                        1.800000
75%                   31.100000                        2.725000
max                   47.400000                        7.000000

       Orbital Eccentricity  Obliquity to Orbit (degrees)  \
count              8.000000                      8.000000
mean               0.060250                     47.741750
std                0.065447                     60.364836
min                0.007000                      0.034000
25%                0.015250                     18.325000
50%                0.048000                     25.950000
75%                0.062500                     45.675000
max                0.206000                    177.400000

       Mean Temperature (C)  Number of Moons
count              8.000000         8.000000
mean              -8.000000        25.625000
std              225.783714        35.136215
min             -200.000000         0.000000
25%             -153.750000         0.750000
50%              -87.500000         8.000000
75%               53.000000        40.000000
max              464.000000        82.000000
```

[5]:

```
# Nombres de columnas originales
original_columns = planets_df.columns
print(original_columns)
```

```
Index(['Planet', 'Color', 'Mass (10^24kg)', 'Diameter (km)',
       'Density (kg/m^3)', 'Surface Gravity(m/s^2)', 'Escape Velocity (km/s)',
       'Rotation Period (hours)', 'Length of Day (hours)',
       'Distance from Sun (10^6 km)', 'Perihelion (10^6 km)',
       'Aphelion (10^6 km)', 'Orbital Period (days)',
       'Orbital Velocity (km/s)', 'Orbital Inclination (degrees)',
       'Orbital Eccentricity', 'Obliquity to Orbit (degrees)',
       'Mean Temperature (C)', 'Surface Pressure (bars)', 'Number of Moons',
       'Ring System?', 'Global Magnetic Field?', 'Surface Temperature (C)',
       'Atmospheric Composition', 'Atmospheric Pressure (bars)',
       'Surface Features', 'Composition'],
      dtype='object')
```

[6]:

```
# Función para convertir a snake_case
def to_snake_case(column_name):
    s = column_name.lower()  # Convertir a minúsculas
    s = re.sub(r'[^a-z0-9]+', '_', s)  # Reemplazar caracteres no alfanuméricos
 ↪con guiones bajos
```

```
    s = s.strip('_')  # Eliminar guiones bajos extra al principio o al final
    return s
```

[7]:
```
# Función para quitar las unidades (y cualquier información entre paréntesis)
def remove_units(column_name):
    return re.sub(r'\(.*\)', '', column_name).strip()

# Quitar las unidades de los nombres de las columnas
columns_without_units = [remove_units(col) for col in original_columns]

# Convertir cada nombre de columna (sin unidades) a snake_case
snake_case_columns_without_units = [to_snake_case(col) for col in␣
 ↪columns_without_units]
snake_case_columns_without_units
```

[7]:
```
['planet',
 'color',
 'mass',
 'diameter',
 'density',
 'surface_gravity',
 'escape_velocity',
 'rotation_period',
 'length_of_day',
 'distance_from_sun',
 'perihelion',
 'aphelion',
 'orbital_period',
 'orbital_velocity',
 'orbital_inclination',
 'orbital_eccentricity',
 'obliquity_to_orbit',
 'mean_temperature',
 'surface_pressure',
 'number_of_moons',
 'ring_system',
 'global_magnetic_field',
 'surface_temperature',
 'atmospheric_composition',
 'atmospheric_pressure',
 'surface_features',
 'composition']
```

[8]:
```
#remplazo las columnas en el dataframe
planets_df.columns = snake_case_columns_without_units
```

[9]:
```python
# renombra las columnas
planets_df = planets_df.rename(columns = {
        'surface_gravity':'gravity',
        'surface_pressure':'pressure',
        'surface_temperature':'temperature',
        'Day': 'day',
        }
)
```

[10]:
```python
#Elimino la coliumna pressure porque es ambiagua con respecto a␣
↪'atmospheric_pressure'
planets_df.drop('pressure', axis=1, inplace=True)
```

[11]:
```python
#Nueva base de datos
planets_df.to_csv('planets_system.csv', index=False)
```

[12]:
```python
print(planets_df["planet"].tolist())
```

```
['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune']
```

[13]:
```python
planets_df.sort_values(by='escape_velocity', ascending=True)
```

[13]:
```
    planet                                        color      mass  \
0  Mercury                                         Grey     0.330
3     Mars                           Red, Brown and Tan     0.642
1    Venus                               Brown and Grey     4.870
2    Earth                     Blue, Brown Green and White     5.970
6   Uranus                                   Blue-Green    86.800
7  Neptune                                         Blue   102.000
5   Saturn                   Golden, Brown, and Blue-Grey   568.000
4  Jupiter  Brown, Orange and Tan, with White cloud stripes  1898.000

   diameter  density  gravity  escape_velocity  rotation_period  \
0      4879     5429      3.7              4.3           1407.6
3      6792     3934      3.7              5.0             24.6
1     12104     5243      8.9             10.4          -5832.5
2     12756     5514      9.8             11.2             23.9
6     51118     1270      8.7             21.3            -17.2
7     49528     1638     11.0             23.5             16.1
5    120536      687      9.0             35.5             10.7
4    142984     1326     23.1             59.5              9.9

   length_of_day  distance_from_sun  …  obliquity_to_orbit  \
0         4222.6               57.9  …               0.034
3           24.7              228.0  …              25.200
1         2802.0              108.2  …             177.400
2           24.0              149.6  …              23.400
```

```
6          17.2           2867.0  …          97.800
7          16.1           4515.0  …          28.300
5          10.7           1432.0  …          26.700
4           9.9            778.5  …           3.100

   mean_temperature number_of_moons  ring_system  global_magnetic_field  \
0               167               0           No                    Yes
3               -65               2           No                     No
1               464               0           No                     No
2                15               1           No                    Yes
6              -195              27          Yes                    Yes
7              -200              14          Yes                    Yes
5              -140              82          Yes                    Yes
4              -110              79          Yes                    Yes

    temperature   atmospheric_composition  atmospheric_pressure  \
0   -173 to 427               Mostly None                 Trace
3    -153 to 20    Carbon Dioxide (95.3%)                 0.006
1           462    Carbon Dioxide (96.5%)                    92
2     -89 to 58  Nitrogen (78.1%), Oxygen                     1
6          -197          Hydrogen, Helium               Unknown
7          -201          Hydrogen, Helium               Unknown
5  -178 to -228          Hydrogen, Helium               Unknown
4  -108 to -150          Hydrogen, Helium               Unknown

                             surface_features          composition
0                             Craters, Scarps        Rock and Metal
3               Valles Marineris, Olympus Mons          Rock and Ice
1                   Volcanoes, Venusian Plains        Rock and Metal
2                   Mountains, Oceans, Forests  Rock, Water, and Air
6   Rings, Miranda's Cliff, Oberon's Craters          Gas and Ice
7            Great Dark Spot, Triton's Geysers          Gas and Ice
5  Rings, Cassini Division, Saturn's Hexagon        Gas and Liquid
4             Great Red Spot, Jupiter's Rings        Gas and Liquid

[8 rows x 26 columns]
```

1.3 Descripcion de informacion de cada cohete y convesion a JSON

1.4 "Amenidades de viaje"

[14]:
```
# Crear el diccionario con la información solicitada y luego convertirlo a␣
↪formato JSON

rockets_info = {
    "SLS": {
```

```
        "Description": "Con 98 metros de altura, el SLS es el cohete más␣
↪potente jamás construido por la NASA, diseñado para generar una fuerza de␣
↪empuje de 8,8 millones de libras. Está compuesto por dos etapas, donde la␣
↪primera utiliza cuatro motores RS-25 y la segunda un motor J-2X, ambos␣
↪funcionando con hidrógeno y oxígeno líquidos.",
        "Cost": 2000,
        "Speed": "7.78 km/s",
        "Amenities": "Un salón de observación exclusivo, asientos ergonómicos␣
↪de lujo con función de masaje, servicio de comidas gourmet, sistema de␣
↪entretenimiento a bordo y habitaciones privadas para descansar."
    },
    "Saturn V": {
        "Description": "El Saturn V, con 110.6 metros de altura, es el cohete␣
↪más grande y poderoso jamás construido, utilizado principalmente para␣
↪misiones lunares. Genera una fuerza de empuje de 7,5 millones de libras y␣
↪está compuesto por tres etapas con motores F-1 y J-2.",
        "Cost": 1500,
        "Speed": "11 km/s",
        "Amenities": "Un bar y lounge espacial, terraza con vistas al espacio,␣
↪servicio de guía turístico espacial, habitaciones con camas de gravedad cero␣
↪y un gimnasio espacial."
    },
    "Falcon Heavy": {
        "Description": "Desarrollado por SpaceX, el Falcon Heavy mide 70 metros␣
↪y es el cohete más potente en servicio actualmente. Genera una fuerza de␣
↪empuje de 5,1 millones de libras y se compone de tres núcleos Falcon 9␣
↪equipados con motores Merlin.",
        "Cost": 150,
        "Speed": "7.5 km/s",
        "Amenities": "Cápsula con ventanas de 360 grados, asientos reclinables␣
↪de alta tecnología, barra de oxígeno y aromaterapia, sesiones de meditación␣
↪en el espacio y internet de alta velocidad."
    },
    "Delta IV Heavy": {
        "Description": "El Delta IV Heavy mide 72 metros y es el segundo cohete␣
↪más grande y potente en servicio. Puede generar una fuerza de empuje de 2,1␣
↪millones de libras y se compone de tres núcleos Delta IV con motores RS-68A.
↪",
        "Cost": 350,
        "Speed": "9.2 km/s",
        "Amenities": "Biblioteca espacial, observatorio con telescopios, clases␣
↪de astronomía, habitaciones suites con vistas al espacio y servicio de spa y␣
↪bienestar."
    }
}
```

```python
import json
rockets_json = json.dumps(rockets_info, indent=4)
rockets_json
```

[14]:

```
'{\n    "SLS": {\n        "Description": "Con 98 metros de altura, el SLS es el cohete m\\u00e1s potente jam\\u00e1s construido por la NASA, dise\\u00f1ado para generar una fuerza de empuje de 8,8 millones de libras. Est\\u00e1 compuesto por dos etapas, donde la primera utiliza cuatro motores RS-25 y la segunda un motor J-2X, ambos funcionando con hidr\\u00f3geno y ox\\u00edgeno l\\u00edquidos.",\n        "Cost": 2000,\n        "Speed": "7.78 km/s",\n        "Amenities": "Un sal\\u00f3n de observaci\\u00f3n exclusivo, asientos ergon\\u00f3micos de lujo con funci\\u00f3n de masaje, servicio de comidas gourmet, sistema de entretenimiento a bordo y habitaciones privadas para descansar."\n    },\n    "Saturn V": {\n        "Description": "El Saturn V, con 110.6 metros de altura, es el cohete m\\u00e1s grande y poderoso jam\\u00e1s construido, utilizado principalmente para misiones lunares. Genera una fuerza de empuje de 7,5 millones de libras y est\\u00e1 compuesto por tres etapas con motores F-1 y J-2.",\n        "Cost": 1500,\n        "Speed": "11 km/s",\n        "Amenities": "Un bar y lounge espacial, terraza con vistas al espacio, servicio de gu\\u00eda tur\\u00edstico espacial, habitaciones con camas de gravedad cero y un gimnasio espacial."\n    },\n    "Falcon Heavy": {\n        "Description": "Desarrollado por SpaceX, el Falcon Heavy mide 70 metros y es el cohete m\\u00e1s potente en servicio actualmente. Genera una fuerza de empuje de 5,1 millones de libras y se compone de tres n\\u00facleos Falcon 9 equipados con motores Merlin.",\n        "Cost": 150,\n        "Speed": "7.5 km/s",\n        "Amenities": "C\\u00e1psula con ventanas de 360 grados, asientos reclinables de alta tecnolog\\u00eda, barra de ox\\u00edgeno y aromaterapia, sesiones de meditaci\\u00f3n en el espacio y internet de alta velocidad."\n    },\n    "Delta IV Heavy": {\n        "Description": "El Delta IV Heavy mide 72 metros y es el segundo cohete m\\u00e1s grande y potente en servicio. Puede generar una fuerza de empuje de 2,1 millones de libras y se compone de tres n\\u00facleos Delta IV con motores RS-68A.",\n        "Cost": 350,\n        "Speed": "9.2 km/s",\n        "Amenities": "Biblioteca espacial, observatorio con telescopios, clases de astronom\\u00eda, habitaciones suites con vistas al espacio y servicio de spa y bienestar."\n    }\n}'
```


2 Calcular la posicion en cordenadas de cada planeta en tiempo real o en una fecha determinada

2.1 Librerias

[15]:
```python
from astropy import units as u
from astropy import coordinates as coord
from skyfield.api import load
from skyfield.errors import EphemerisRangeError
from datetime import datetime
```

```python
import math
```

2.2 Normalizacion de nombres

[16]:
```python
# Mapeo de nombres a identificadores de Skyfield
planet_mapping = {
    'Mercury': 'mercury',
    'Venus': 'venus',
    'Earth': 'earth',
    'Mars': 'mars',
    'Jupiter': 'jupiter barycenter',
    'Saturn': 'saturn barycenter',
    'Uranus': 'uranus barycenter',
    'Neptune': 'neptune barycenter'
}

# Velocidades de los cohetes en km/s y mapeo de nombres cortos
rocket_speeds = {
    "SLS": 28000 / 3600,
    "Saturn V": 11,
    "Falcon Heavy": 7.5,
    "Delta IV Heavy": 9.2
}
```

2.3 2. Funciones de Utilidad

[17]:
```python
def get_planet_positions(date_str=None):
    # Cargar efemérides
    eph = load('de421.bsp')

    # Cargar tiempos
    ts = load.timescale()
    if date_str:
        date_obj = datetime.strptime(date_str, '%Y-%m-%d')
        t = ts.utc(date_obj.year, date_obj.month, date_obj.day)
    else:
        t = ts.now()

    planet_data = {}

    for planet_name, planet_id in planet_mapping.items():
        body = eph[planet_id]
        astrometric = body.at(t)
        lat, lon, distance = astrometric.ecliptic_latlon()
        planet_data[planet_name] = {
            'lat': lat.degrees,
            'lon': lon.degrees,
```

```
            'distance': distance.au
        }

    # Convertir las coordenadas eclípticas a cartesianas
    for planet, data in planet_data.items():
        r = data['distance']
        lon = math.radians(data['lon'])
        lat = math.radians(data['lat'])
        x = r * math.cos(lat) * math.cos(lon)
        y = r * math.cos(lat) * math.sin(lon)
        z = r * math.sin(lat)
        planet_data[planet]['x'] = x
        planet_data[planet]['y'] = y
        planet_data[planet]['z'] = z

    return planet_data

def calculate_distance(planet_data, planet1, planet2):
    x1, y1, z1 = planet_data[planet1]['x'], planet_data[planet1]['y'],␣
 ↪planet_data[planet1]['z']
    x2, y2, z2 = planet_data[planet2]['x'], planet_data[planet2]['y'],␣
 ↪planet_data[planet2]['z']
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2 + (z2 - z1)**2)

def calculate_travel_time(distance, speed):
    return distance * 1.496e8 / speed  # Convertir UA a km y dividir por la␣
 ↪velocidad para obtener el tiempo en segundos
```

2.4 3. Interaccion con el usuario

[18]:
```
##### Solicitar entrada del usuario
user_input = input("Ingrese una fecha (DD/MM/YYYY) o presione Enter para usar␣
 ↪la fecha actual: ")
#Prueba que la fehca este dentro del rango cubierto por el archivo de␣
 ↪efemérides.
try:
    if user_input:
        day, month, year = map(int, user_input.split('/'))
        planet_data = get_planet_positions(f"{year}-{month}-{day}")
    else:
        planet_data = get_planet_positions()

except EphemerisRangeError:
    print("La fecha ingresada está fuera del rango cubierto por el archivo de␣
 ↪efemérides. Por favor, elige una fecha entre 1899-07-29 y 2053-10-09.")
    planet_data = None
```

```python
# Asegúrate de que solo continuamos si planet_data es válido
if planet_data:
    planet1 = input("Ingrese el primer planeta (Mercury, Venus, Earth, Mars,␣
↪Jupiter, Saturn, Uranus, Neptune): ")
    planet2 = input("Ingrese el segundo planeta (Mercury, Venus, Earth, Mars,␣
↪Jupiter, Saturn, Uranus, Neptune): ")

    # Permitir al usuario elegir un cohete por su nombre corto
    print("Elija un cohete:")
    for rocket in rocket_speeds:
        print(rocket)

    selected_rocket = input()
    if selected_rocket not in rocket_speeds:
        print("Cohete no reconocido.")
        exit()

    distance = calculate_distance(planet_data, planet1, planet2)
    time_seconds = calculate_travel_time(distance,␣
↪rocket_speeds[selected_rocket])

    hours = time_seconds // 3600
    minutes = (time_seconds % 3600) // 60
    ,
    seconds = time_seconds % 60

    print(f"El tiempo de viaje entre {planet1} y {planet2} usando el␣
↪{selected_rocket} es aproximadamente {int(hours)} horas, {int(minutes)}␣
↪minutos y {seconds:.2f} segundos.")
```

```
Ingrese una fecha (DD/MM/YYYY) o presione Enter para usar la fecha actual:
25/10/2025
Ingrese el primer planeta (Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus,
Neptune):  Mercury
Ingrese el segundo planeta (Mercury, Venus, Earth, Mars, Jupiter, Saturn,
Uranus, Neptune):  Uranus

Elija un cohete:
SLS
Saturn V
Falcon Heavy
Delta IV Heavy

 SLS

El tiempo de viaje entre Mercury y Uranus usando el SLS es aproximadamente
105366 horas, 16 minutos y 54.84 segundos.
```

[19]:
```
def gen():

    lst1: list = [key.lower() for key in planet_mapping.keys()]
    lst2: list = lst1.copy()

    print(lst1)

    lst2 = lst2[1:]

    rtn = list()

    for p1 in lst1:

        for p2 in lst2:

            rtn.append(calc_min_dist(p1,p2))

        lst1 = lst1[1:]
        lst2 = lst2[1:]


    return rtn
```

[20]:
```
from astropy.coordinates import solar_system_ephemeris, get_body
from astropy.coordinates.sky_coordinate import SkyCoord
import astropy.units as u
from astropy.time import Time

planet_mapping = {
    'Mercury': 'mercury',
    'Venus': 'venus',
    'Earth': 'earth',
    'Mars': 'mars',
    'Jupiter': 'jupiter barycenter',
    'Saturn': 'saturn barycenter',
    'Uranus': 'uranus barycenter',
    'Neptune': 'neptune barycenter'
}

# Definir la fecha y hora para la que deseas obtener las coordenadas␣
 ↪heliocéntricas
fecha = Time('2023-10-08')

# Nombre del planeta que deseas obtener (por ejemplo, Marte)
nombre_planeta = 'mars'
```

```
def calc_min_dist(planet1 : str, planet2 : str, date = None, offset = 3,div =␣
 ↪3):

    def get_sq_dist(sky_coords : SkyCoord):

        a = sky_coords.cartesian

        return a.x.value**2+a.y.value**2+a.z.value**2



    # initial date
    if (date):
        date = Time(date)

    else:
        # Obtener la fecha y hora actuales
        date = Time.now()


        date = Time(date)

    # list to save
    date_lst = list()
    dist_list = list()
    planet_list = list()

    for iter in range(1,365//div):

        # Obtener las coordenadas heliocéntricas del planeta en la fecha␣
 ↪especificada
        with solar_system_ephemeris.set('builtin'):
            p1 = get_body(planet1, date)
            p2 = get_body(planet2,date)

        # Obtener las coordenadas heliocéntricas
        c1 : SkyCoord = p1.heliocentrictrueecliptic
        c2 : SkyCoord = p2.heliocentrictrueecliptic

        # get square distance
        d1 = get_sq_dist(c1)
        d2 = get_sq_dist(c2)

        # add date to list
        date_lst.append(date)
        # add dist to list
```

```
        dist_list.append(abs(d1-d2))
        # add planets
        planet_list.append((planet1,planet2))

        # sum date
        date += offset*u.day


    index = dist_list.index(min(dist_list))

    return (planet_list[index],dist_list[index],date_lst[index].
 ↪strftime('%Y-%m-%d'))

def gen():

    lst1: list = [key.lower() for key in planet_mapping.keys()]
    lst2: list = lst1.copy()

    print(lst1)

    lst2 = lst2[1:]

    rtn = list()

    for p1 in lst1:

        for p2 in lst2:

            rtn.append(calc_min_dist(p1,p2))

        lst1 = lst1[1:]
        lst2 = lst2[1:]


    return rtn

if __name__ == "__main__":

    print(gen())

    pass
```

```
['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune']
[(('mercury', 'venus'), 0.2994674689613539, '2024-07-25'), (('mercury',
'earth'), 0.7533882584118824, '2024-02-02'), (('mercury', 'mars'),
1.6919344867561035, '2024-05-02'), (('mercury', 'jupiter'), 24.52708147763117,
'2023-11-01'), (('mercury', 'saturn'), 93.1325391074201, '2024-09-29'),
```

```
(('mercury', 'uranus'), 382.7389091962094, '2024-09-29'), (('mercury',
'neptune'), 893.696755910214, '2024-09-29'), (('venus', 'earth'),
0.4454819855216243, '2024-01-24'), (('venus', 'mars'), 1.3835305758024956,
'2024-05-02'), (('venus', 'jupiter'), 24.182732876220836, '2023-10-08'),
(('venus', 'saturn'), 92.75924516541696, '2024-09-29'), (('venus', 'uranus'),
382.3656152650828, '2024-09-29'), (('venus', 'neptune'), 893.32346282933,
'2024-09-29'), (('earth', 'mars'), 0.8872617968374739, '2024-05-17'), (('earth',
'jupiter'), 23.706523506733422, '2023-10-08'), (('earth', 'saturn'),
92.28394935544947, '2024-09-29'), (('earth', 'uranus'), 381.89031941487235,
'2024-09-29'), (('earth', 'neptune'), 892.8481676212766, '2024-09-29'),
(('mars', 'jupiter'), 22.18804095809393, '2023-10-08'), (('mars', 'saturn'),
91.03329301367255, '2024-09-29'), (('mars', 'uranus'), 380.63966284806935,
'2024-09-29'), (('mars', 'neptune'), 891.5975114495218, '2024-09-29'),
(('jupiter', 'saturn'), 67.753482193449, '2024-09-29'), (('jupiter', 'uranus'),
357.3598520373278, '2024-09-29'), (('jupiter', 'neptune'), 868.3177009728183,
'2024-09-29'), (('saturn', 'uranus'), 289.6063700120447, '2024-09-29'),
(('saturn', 'neptune'), 799.1039667226822, '2023-10-08'), (('uranus',
'neptune'), 509.243604250352, '2023-10-08')]
```

[21]:
```python
# Datos proporcionados
data_dis_min = [(('mercury', 'venus'), 0.29939066853237073, '2024-07-28'),
         (('mercury', 'earth'), 0.7533166101091904, '2024-02-02'),
         (('mercury', 'mars'), 1.691861958254788, '2024-05-02'),
         (('mercury', 'jupiter'), 24.527008132027493, '2023-11-01'),
         (('mercury', 'saturn'), 93.1350485949965, '2024-09-29'),
         (('mercury', 'uranus'), 382.7415942742422, '2024-09-29'),
         (('mercury', 'neptune'), 893.6982233097619, '2024-09-29'),
         (('venus', 'earth'), 0.4454817707495228, '2024-01-24'),
         (('venus', 'mars'), 1.3835417155671903, '2024-05-02'),
         (('venus', 'jupiter'), 24.18232140781379, '2023-10-08'),
         (('venus', 'saturn'), 92.76070777844512, '2024-09-29'),
         (('venus', 'uranus'), 382.36725345788494, '2024-09-29'),
         (('venus', 'neptune'), 893.323883014894, '2024-09-29'),
         (('earth', 'mars'), 0.8872714044057355, '2024-05-17'),
         (('earth', 'jupiter'), 23.706021098477986, '2023-10-08'),
         (('earth', 'saturn'), 92.28523588189123, '2024-09-29'),
         (('earth', 'uranus'), 381.8917815245582, '2024-09-29'),
         (('earth', 'neptune'), 892.8484114498792, '2024-09-29'),
         (('mars', 'jupiter'), 22.186799843804252, '2023-10-08'),
         (('mars', 'saturn'), 91.03568615367539, '2024-09-29'),
         (('mars', 'uranus'), 380.642231685531, '2024-09-29'),
         (('mars', 'neptune'), 891.5988618090599, '2024-09-29'),
         (('jupiter', 'saturn'), 67.75566292258303, '2024-09-29'),
         (('jupiter', 'uranus'), 357.3622084228153, '2024-09-29'),
         (('jupiter', 'neptune'), 868.3188388133096, '2024-09-29'),
         (('saturn', 'uranus'), 289.6065456190637, '2024-09-29'),
         (('saturn', 'neptune'), 799.1029959778213, '2023-10-08'),
```

```python
      (('uranus', 'neptune'), 509.24244805616433, '2023-10-08')]

# Convertir a DataFrame
df_dist_min = pd.DataFrame(data_dis_min, columns=['planetas', 'distancia',␣
 ↪'fecha_aproximada'])
df_dist_min['planeta_origen'] = df_dist_min['planetas'].apply(lambda x: x[0])
df_dist_min['planeta_destino'] = df_dist_min['planetas'].apply(lambda x: x[1])
df_dist_min.drop(columns='planetas', inplace=True)

# Reordenar columnas
df_dist_min = df_dist_min[['planeta_origen', 'planeta_destino', 'distancia',␣
 ↪'fecha_aproximada']]
# Raiz de distancia
df_dist_min['distancia'] = df_dist_min['distancia'].apply(lambda x: x**0.5)

df_dist_min
```

[21]:

	planeta_origen	planeta_destino	distancia	fecha_aproximada
0	mercury	venus	0.547166	2024-07-28
1	mercury	earth	0.867938	2024-02-02
2	mercury	mars	1.300716	2024-05-02
3	mercury	jupiter	4.952475	2023-11-01
4	mercury	saturn	9.650650	2024-09-29
5	mercury	uranus	19.563783	2024-09-29
6	mercury	neptune	29.894786	2024-09-29
7	venus	earth	0.667444	2024-01-24
8	venus	mars	1.176241	2024-05-02
9	venus	jupiter	4.917552	2023-10-08
10	venus	saturn	9.631236	2024-09-29
11	venus	uranus	19.554213	2024-09-29
12	venus	neptune	29.888524	2024-09-29
13	earth	mars	0.941951	2024-05-17
14	earth	jupiter	4.868883	2023-10-08
15	earth	saturn	9.606520	2024-09-29
16	earth	uranus	19.542052	2024-09-29
17	earth	neptune	29.880569	2024-09-29
18	mars	jupiter	4.710287	2023-10-08
19	mars	saturn	9.541262	2024-09-29
20	mars	uranus	19.510055	2024-09-29
21	mars	neptune	29.859653	2024-09-29
22	jupiter	saturn	8.231383	2024-09-29
23	jupiter	uranus	18.904026	2024-09-29
24	jupiter	neptune	29.467250	2024-09-29
25	saturn	uranus	17.017830	2024-09-29
26	saturn	neptune	28.268410	2023-10-08
27	uranus	neptune	22.566401	2023-10-08

[22]:
```python
#Nueva base de datos de distancia minima entre 2 planetas
df_dist_min.to_csv('df_dist_min.csv', index=False)
```

[ ]:
```python

```

[ ]:
```python

```