UNIVERSIDAD NACIONAL DE SAN AGUSTIN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación
Aprobación: 2022/03/01 Código: GUIA-PRLE-001

INFORMACIÓN BÁSICA					
ASIGNATURA:	FÍSICA COMPUTACIONAL				
TÍTULO DE LA PRÁCTICA:	Implementación de fórmulas de Kepler y Ley de Gravitación.				
NÚMERO DE PRÁCTICA:	04	**AÑO LECTIVO:**	2022	**NRO. SEMESTRE:**	2022A
FECHA DE PRESENTACIÓN	6/06/2022	**HORA DE PRESENTACIÓN**	7:00 pm		
INTEGRANTE (s): • Alván Ventura, Edsel Yael				**NOTA:**	
DOCENTE(s): **Danny Giancarlo Apaza Veliz**					

Laboratorio 04.

Escrito por
Alván Ventura, Edsel Yael
ealvan@unsa.edu.pe

Profesor
Apaza Veliz, Danny Giancarlo
dapazav@unsa.edu.pe

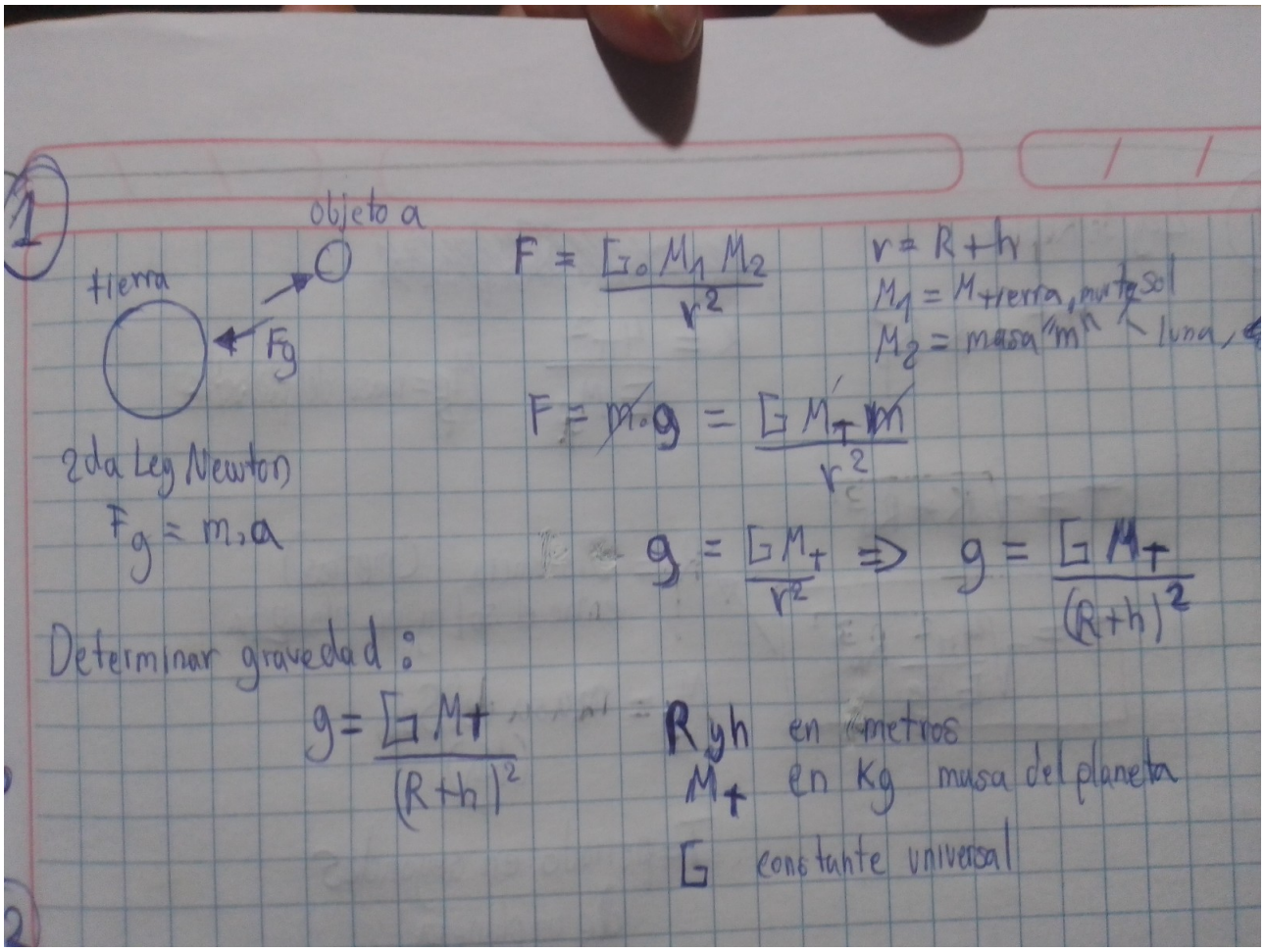
June 6, 2022

1 Problema 1

A partir de la segunda ley de movimiento de Newton y la ley de gravitación universal realice un código que permita determinar el valor de la gravedad para cualquier planeta del sistema planetario solar.

1.1 Análisis

Para obtener la gravedad en cualquier planeta del sistema solar, se debe igualar la segunda ley de newton y la ley de fuerza gravitacional entre dos cuerpos, pues en este caso la fuerza de ambas formulas son las mismas. A continuación se muestra la igualdad y el despeje:



1.2 Programación

Para la programación se obto por recolectar datos de los planetas. Para este caso en especifico se recolecto la masa y el radio de cada planeta. Además se puso la constante universal de gravitación.

Todo los datos anteriores(masa,radio) se pusieron en un archivo llamado data.py, este archivo contiene un diccionario en python con todos los datos(este archivo se usara continuamente para los siguiente ejercicios).

Archivo data.py

```
1  CONSTANTE_GRAVITACIONAL=6.672*10**-11
2  AU = 1.49597871*10**11#Astronomical Unit
3  #AU es una medida con respecto de la distancia del sol a la tierra.
4  DATA = {
5      "tierra":{
6          "masa":5.972*10**24,#kg
7          "radio":6375*10**3,#metros
8          "sunDistancia":1.49597871*10**11,#1.496*10**11,#metros,
9          "periodo": 365.26,#en dias usado unicamente para el ejer3
10         #en el ejercio 4, usamos la formula
11         #para encontrar el periodo en segundos
12     },
13     "marte":{
14         "masa":6.39*10**23,
15         "radio":3389.5*10**3,
16         "sunDistancia":2.2794*10**11,
17         "periodo": 686.98,
18     },
19     "mercurio":{
20         "masa":0.33010*10**24,
21         "radio":2439.7*10**3,
22         "sunDistancia":0.38*AU,
23         "periodo": 87.97,
24     },
25     "venus":{
26         "masa":4.867*10**24,
27         "radio":6051.8*10**3,
28         "sunDistancia":0.72*AU,
29         "periodo": 224.70,
30     },
31     "jupiter":{#24.79
32         "masa":1.898*10**27,
33         "radio":69911*10**3,
34         "sunDistancia":5.2*AU,
35         "periodo": 4332.71,
36     },
37     "saturno":{#10.44
38         "masa":5.683*10**26,
39         "radio":58232*10**3,
40         "sunDistancia":9.5*AU,
41         "periodo": 10759.50,
42     },
43     "urano":{
44         "masa":8.681*10**25,
45         "radio":25362*10**3,
46         "sunDistancia":2870972200*10**3,#19.8*AU,
47         "periodo": 30685.00,
48     },
49     "neptuno":{
50         "masa":1.024*10**26,
51         "radio":24622*10**3,
```

```
        "sunDistancia":30*AU,
        "periodo": 60190.00,
    },
}
```

Finalmente tambien se agrego la condición:

```
if h > radio:
    print("No puede ser mayor al radio del planeta")
    exit(1)#salir del programa
```

A continuacion, la siguiente implementación usa el data.py para obtener los calculos con los datos de masa y radio de los planetas.

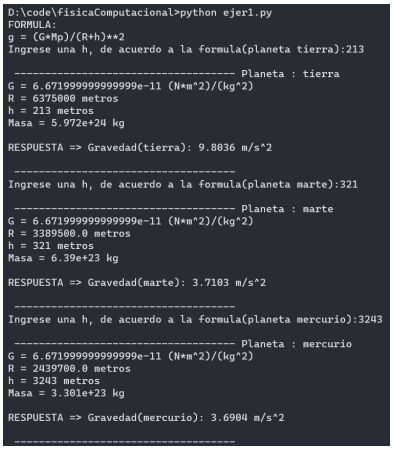
Archivo ejer1.py

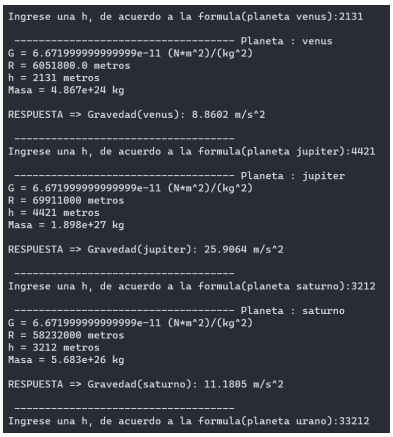
```
from data import DATA,CONSTANTE_GRAVITACIONAL
import random
#EJERCICIO 1
# A partir de la segunda ley de movimiento de Newton y
# la ley de gravitacion universal realice un codigo que
# permita determinar el valor de la gravedad para
# cualquier planeta del sistema planetario solar.

#Formula = (G*masaPlaneta)/(R+h)^2

#unidades:
    #masas en Kg
    #G = constante universal (N*m^2)/(kg^2)
    #R y h = Distancias en metros

def getGravity(planet,h):
    planetData =DATA[planet]
    mass = planetData["masa"]
    radius = planetData["radio"]
    if not(mass and radius):
        print("No existe este planeta en nuestra base de datos...")
        exit(1)
    if h > radius:
        print("No puede ser mayor al radio del planeta")
        exit(1)
    G = CONSTANTE_GRAVITACIONAL
    gravity = (G*mass)/((radius+h)**2)
    return gravity
def printData(planeta,h,gravity):
    G = CONSTANTE_GRAVITACIONAL
    planet_data = DATA[planeta]
    masa = planet_data["masa"]
    R = planet_data["radio"]
    str1 = f"""Planeta : {planeta}
G = {G} (N*m^2)/(kg^2)
R = {R} metros
h = {h} metros
Masa = {masa} kg

RESPUESTA => Gravedad({planeta}): {round(gravity,4)} m/s^2
"""
    print("\n","-"*36,str1,"\n","-"*36)

def main():
    print("FORMULA: \ng = (G*Mp)/(R+h)**2")
    for k in DATA.keys():
```

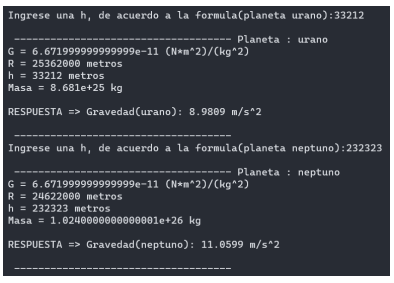
```
47          h = input(f"Ingrese una h, de acuerdo a la formula(planeta {k}):")
48          if(h == "exit"):
49              print("Usted ha salido con exito del programa...")
50              exit(0)
51          h = int(h)
52          while(h < 0):
53              print("La distancia no puede ser negativa...")
54              h = input(f"Ingrese una h, de acuerdo a la formula(planeta {k}):")
55              if(h == "exit"):
56                  print("Usted ha salido con exito del programa...")
57                  exit(0)
58              h = int(h)
59          gravity = getGravity(k,h)
60          printData(k,h,gravity)#imprimiendo datos
61      for k in DATA.keys():
62          print("\n","Planeta = ",k,"Gravedad=",getGravity(k,0),"m/s^2")
63
64  if __name__ == "__main__":
65      main()
```

1.3 Resultados

A continuación, se muestran los calculos hechos por el algoritmo.







En la siguiente imagen, se muestra una comparación con datos reales:

Space object	g, gravitational field strength (N/kg)
The Sun (star)	293.0
Mercury	3.7
Venus	8.8
Earth	9.8
Moon (satellite)	1.7
Mars	3.7
Ceres (dwarf planet)	0.27
Jupiter	24.7
Saturn	10.5
Uranus	9.0
Neptune	11.7
Pluto (dwarf planet)	0.49

2 Problema 2

Del problema anterior realice un código para poder determinar la densidad de cualquier planeta del sistema planetario solar.

2.1 Análisis

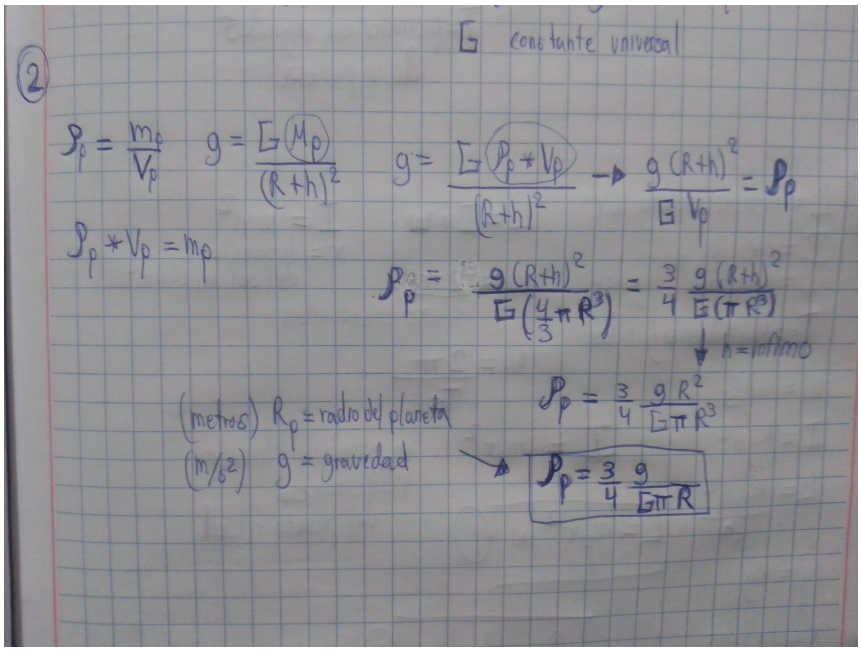
En el siguiente analisis, se despeja la variable densidad. Sabiendo que: La densidad del planeta es:

$$\rho_p = \frac{M_p}{V_p} \tag{1}$$

La gravedad del planeta es:

$$g_p = \frac{GM_p}{R^2} \tag{2}$$

Y su despeje seria:



Por lo tanto la formula seria:

$$\rho_p = (\frac{3}{4})(\frac{g_p}{G\pi R}) \tag{3}$$

2.2 Programación

En el siguiente codigo, se muestra como se implemento el codigo en el archivo ejer2.py.

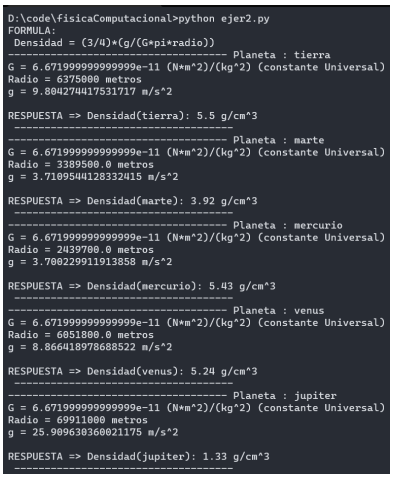
Archivo ejer2.py

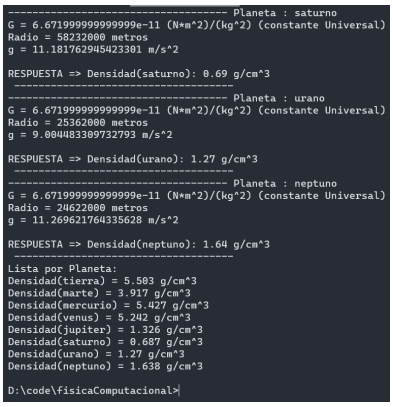
```
1  import math
2
3  from data import DATA,CONSTANTE_GRAVITACIONAL
4  from ejer1 import getGravity
5
6  # EJERCICIO 2:
7  #  Del problema anterior realice un codigo
8  #  para poder determinar la densidad de
9  #  cualquier planeta del sistema planetario solar.
10
11 # densidad = (3/4)*(g/(G*pi*R))
12
```

```
#Unidades:
    # g = gravedad (m/s^2)
    # G = constante universal (N*m^2)/(kg^2)
    # R = radio del planeta en (metros)
#DENSIDAD en kg/m^3 -> g/cm^3
gravity = 0
def getDensity(planet):
    planetdata = DATA[planet]
    radio = planetdata.get("radio")
    if(not(planetdata and radio)):
        print("El planeta {planet} o su radio de {radio} metros no existe...")
        exit(1)
    global gravity
    gravity = getGravity(planet,0)
    pi = math.pi
    G = CONSTANTE_GRAVITACIONAL
    densidad = (3/4)*((gravity)/(G*pi*radio))
    #cambiando de  kg/m^3 -> g/cm^3  con (10^-3)
    densidad = densidad*(10**-3)
    densidad = round(densidad,4) #redondeo de 4 decimales
    return densidad

def printData(planeta,densidad):
    G = CONSTANTE_GRAVITACIONAL
    planet_data = DATA[planeta]
    R = planet_data["radio"]
    str1 = f"""Planeta : {planeta}
G = {G} (N*m^2)/(kg^2) (constante Universal)
Radio = {R} metros
g = {gravity} m/s^2

RESPUESTA => Densidad({planeta}): {round(densidad,2)} g/cm^3
"""
    print("-"*36,str1,"-"*36)

def main():
    print("FORMULA:\n Densidad = (3/4)*(g/(G*pi*radio))")
    for k in DATA.keys():
        density = getDensity(k)
        printData(k,density)
    print("Lista por Planeta:")
    for k in DATA.keys():
        density = getDensity(k)
        print(f"Densidad({k}) = {round(density,3)} g/cm^3")

if __name__ == "__main__":
    main()
```

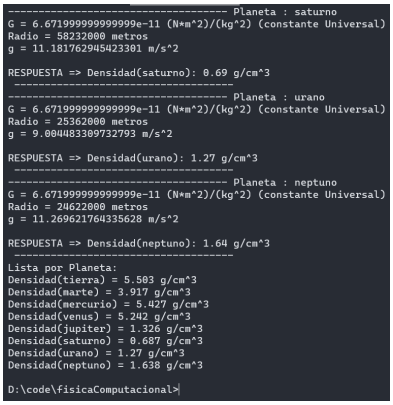
2.3 Resultados

A continuacion se muestra los resultados obtenidos por el algoritmo:





En la siguiente imagen se hace una comparacion con datos reales:



Planet	Average Density (gm/cm^3)
Mercury	5.4
Venus	5.2
Earth	5.5
Mars	3.9
Jupiter	1.3
Saturn	0.7
Uranus	1.3
Neptune	1.6

Como se puede ver los datos obtenidos son muy aproximados a los datos reales de la densidades de cada planeta.

3 Problema 3

Implementar un código computacional para la solución de la segunda ley de Kepler.

3.1 Análisis

La segunda ley de kepler nos dice que un planeta en tiempos iguales barren un area igual.

Esto significa, un planeta cuando esta mas cerca del Sol recorre mas distancia angular en el mismo tiempo, mientras que el mismo planeta lejos del sol barre la misma area en el mismo tiempo, debido a que la distancia entre el sol y el planeta es lo suficientemente grande para barrer la misma área.

Las siguientes ecuaciónes que se presenta se llama momento de Inercia y velocidad angular:

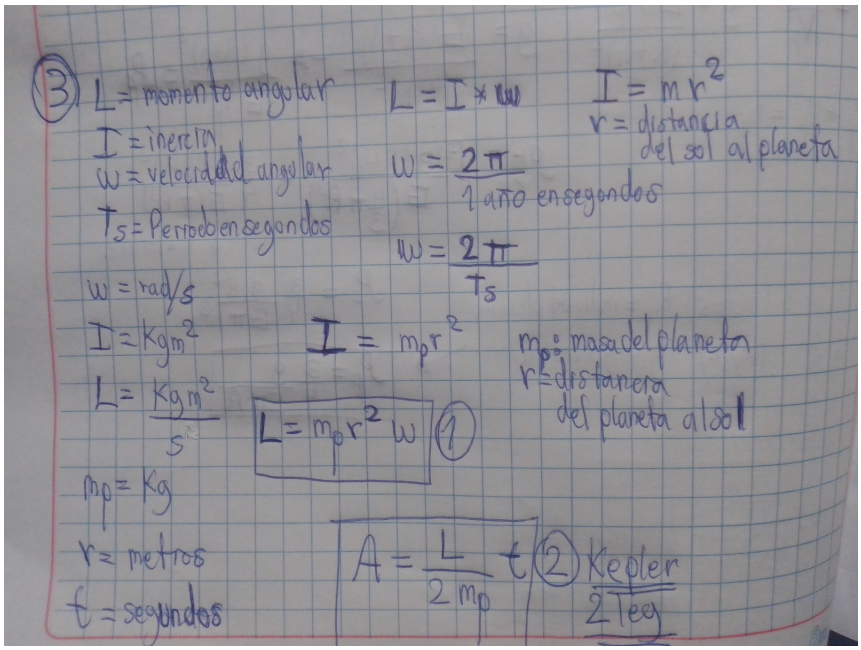
$$I = M_p * R^2 \tag{4}$$

$$w = \frac{2\pi}{1 a\tilde{n}oensegundos} \tag{5}$$

Y el momento angular del planeta con respecto al Sol es:

$$L = Iw \tag{6}$$

A continuacion, se ve como obtener el Area barrida por un planeta con respecto al tiempo:



Y la formula de la 2da ley de Kepler es:

$$A = (\frac{L_p}{2} M_p) * t \tag{7}$$

Donde M_p es la masa del planeta, t es el tiempo y L_p el momento angular del planeta con respecto al Sol.

3.2 Programación

En este caso se uso el archivo data.py, de este archivo usamos el periodo de un planeta(debemos convertirlo a segundos), este nos servira para obtener "w" con mucha más exactitud.

A continuación se muestra la implementación:

Archivo ejer3.py

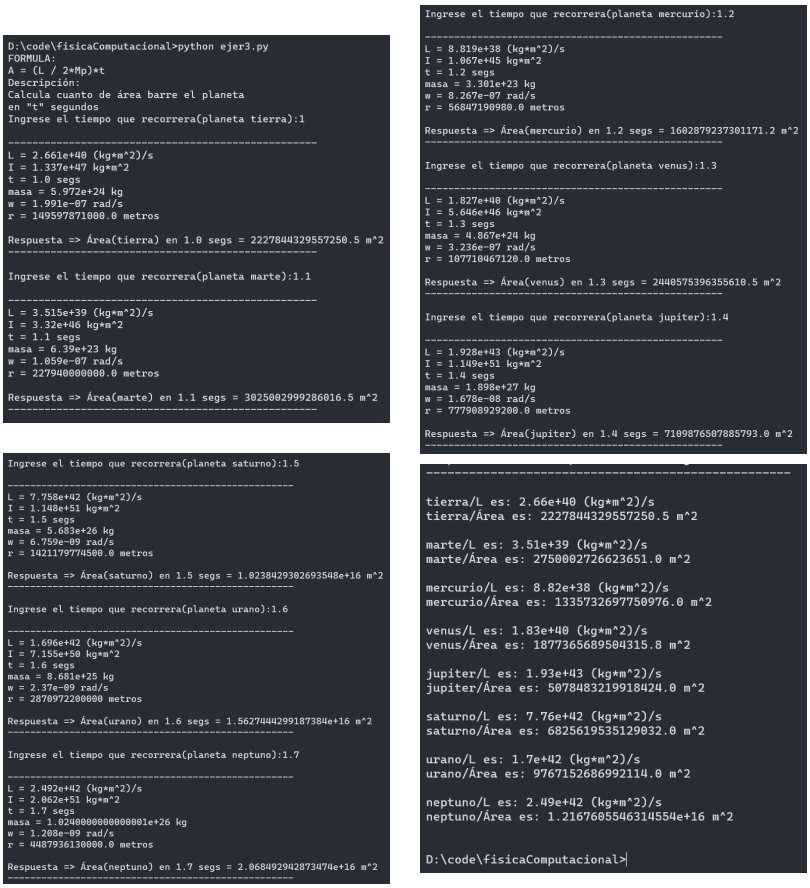
```
1  import math
2  #Ley de kepler #2
3  #areas iguales en tiempos iguales
4  from data import CONSTANTE_GRAVITACIONAL, DATA
5
6  # Implementar un codigo computacional
7  # para la solucion de la segunda
8  # ley de Kepler.
9
10 #FORMULAS
11     # L = I * w
12     # I = Mp*(r^2)
13     # w = (2*pi) / (1 anio en segs)
14     # A = (L / 2*Mp)*t
15
16 #Significados y Unidades
17     # t = tiempo en (segundos)
```

```
18      # L = momento angular((Kg*m^2)/s)
19      # Mp= Masa del Planeta (Kg)
20      # I = inercia (Kg*m^2)
21      # r = distancia del sol al planeta desde sus centros (metros)
22  w = 0 #velocida angular
23  L = 0 # momento angular
24  I = 0 #inercia
25  def getAngularMomentum(planet):
26      global w,I,L
27      # L = I*w
28      # I = m*r^2
29      # w = 2*pi / 1 anio en segundos
30      planet_data = DATA[planet]
31      I = planet_data["masa"]*(planet_data["sunDistancia"])**2
32      pi = math.pi
33      # period = period*60*60*24
34      w = (2*pi)/(planet_data["periodo"]*24*60*60)
35      L = I*w
36      return L
37
38  def getAreaKepler2law(planet,tiempo):
39      # A = (L / 2*Mp)*t
40      if(tiempo < 0):
41          print("El tiempo debe ser positivo")
42          exit(1)
43      L = getAngularMomentum(planet)
44      masaPlaneta = DATA[planet]["masa"]
45      area = (L/(2*masaPlaneta))*tiempo
46      return area
47
48  def printData(planet,time,area):
49      planet_data = DATA[planet]
50      masa = planet_data["masa"]
51      distanciaSol = planet_data["sunDistancia"]
52      str1 = f"""
53  ---------------------------------------------------------
54  L = {float(f"{L:.3e}")} (kg*m^2)/s
55  I = {float(f"{I:.3e}")} kg*m^2
56  t = {time} segs
57  masa = {masa} kg
58  w = {float(f"{w:.3e}")} rad/s
59  r = {distanciaSol} metros
60
61  Respuesta => Area({planet}) en {time} segs = {round(area,4)} m^2
62  ---------------------------------------------------------
63  """
64      print(str1)
65
66  def main():
67      print("FORMULA: \nA = (L / 2*Mp)*t")
68      print("Descripcion:\nCalcula cuanto de area barre el planeta\nen \"t\" segundos")
69      for k in DATA.keys():
70          h = input(f"Ingrese el tiempo que recorrera(planeta {k}):")
71          if(h == "exit"):
72              print("Usted ha salido con exito del programa...")
73              exit(0)
74          h = float(h)
75          while(h < 0):
76              print("La distancia no puede ser negativa...")
77              h = input(f"Ingrese el tiempo que recorrera(planeta {k}):")
78              if(h == "exit"):
79                  print("Usted ha salido con exito del programa...")
```

```
                    exit(0)
                h = float(h)
            area = getAreaKepler2law(k,h)
            printData(k,h,area)
        for planet in DATA.keys():
            momentum = getAngularMomentum(planet)
            area = getAreaKepler2law(planet,1)
            print(f"{planet}/L es: {float('{:0.2e}'.format(momentum))} (kg*m^2)/s")
            print(f"{planet}/Area es: {round(area,3)} m^2\n")

if __name__ == "__main__":
    main()
```

3.3 Resultados

Los resultados del momento angular y los de el Area recorrida se muestran en las siguientes imágenes:



A continuación se muestra una comparación entre el momento angular de por planeta real y el que calculamos:

```
tierra/L es: 2.66e+40 (kg*m^2)/s
tierra/Área es: 2227844329557250.5 m^2

marte/L es: 3.51e+39 (kg*m^2)/s
marte/Área es: 2750002726623651.0 m^2

mercurio/L es: 8.82e+38 (kg*m^2)/s
mercurio/Área es: 1335732697750976.0 m^2

venus/L es: 1.83e+40 (kg*m^2)/s
venus/Área es: 1877365689504315.8 m^2

jupiter/L es: 1.93e+43 (kg*m^2)/s
jupiter/Área es: 5078483219918424.0 m^2

saturno/L es: 7.76e+42 (kg*m^2)/s
saturno/Área es: 6825619535129032.0 m^2

urano/L es: 1.7e+42 (kg*m^2)/s
urano/Área es: 9767152686992114.0 m^2

neptuno/L es: 2.49e+42 (kg*m^2)/s
neptuno/Área es: 1.2167605546314554e+16 m^2


D:\code\fisicaComputacional>
```

body	orbit radius (km)	orbit period (days)	mass (kg)	L (kg m²/s)
Mercury	58.e6	87.97	3.30e23	9.1e38
Venus	108.e6	224.70	4.87e24	1.8e40
Earth	150.e6	365.26	5.97e24	2.7e40
Mars	228.e6	686.98	6.42e23	3.5e39
Jupiter	778.e6	4332.71	1.90e27	1.9e43
Saturn	1429.e6	10759.50	5.68e26	7.8e42
Uranus	2871.e6	30685.00	8.68e25	1.7e42
Neptune	4504.e6	60190.00	1.02e26	2.5e42
Pluto	5914.e6	90800	1.27e22	3.6e38
				3.1e43

Como se vio, las aproximaciones entre lo calculado y lo real, son muy aproximadas.

4 Problema 4

Implementar un código computacional para determinar la solución de la tercera ley de Kepler para cualquier planeta que describa una órbita elíptica.

5 Análisis

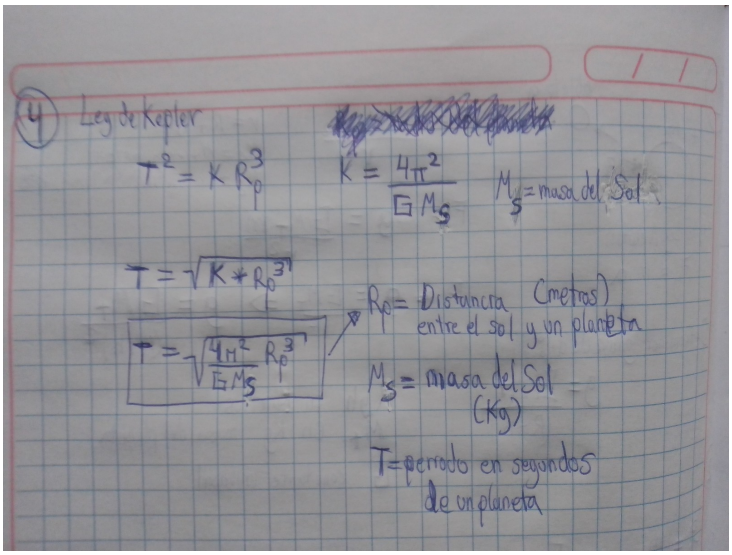
La 3ra ley de Kepler nos indica como obtener el periodo de un cuerpo alrededor de otro. En este problema calcularemos los periodos de los planetas alrededor del Sol y comprobaremos nuestros valores con los reales, para ver la proximidad de nuestros calculos. La 3ra Ley de Kepler es:

$$T^2 = KR^3 \tag{8}$$

Tomando en cuenta que K es igual a:

$$K = \frac{4 * pi^2}{GM_s} \tag{9}$$

A continuación integrando estas dos ecuaciones, se muestra la resolución en la siguiente imagen:



6 Programación

Para la implementación de las formulas anteriores, se decidio tener una función llamada *getPeriodPlanet()*, este se encargara de obtener los datos del archivo data.py y devolver el periodo para cada planeta.

A continuación se muestra el codigo de implementación:

Archivo ejer4.py

```
1  from data import CONSTANTE_GRAVITACIONAL,DATA
2  import math
3
4  #EJERCICIO 3:
5  #tercera ley de kepler
6  #Formula para obtener el periodo de un
7  #Planeta alrededor del sol
8  #ejemplo la tierra da una vuelta en 365 segundos
9
10 # Formula:
11 #      K = (4*pi)/(G*Ms)
12     # Ms = masa del sol (kg)
13     # G = constante universal (N*m^2)/(kg^2)
14 # Formula:
15 #      T^2 = K * Rp^3
16     # Rp = distancia entre sol y planeta desde sus centros (metros)
17     # K = constante K en (s^2/m^3)
18     # T = periodo (segundos)
19
20 K = 0 # constante K
21 R = 0 # Rp distancia al sol
22 toSun = 0#distancia al sol Ms
23 sol = {
24         "masa": 1.989*10**30,
25         "radio": 696340*10**3
26 }
27 def getPeriodPlanet(planet):
28
29     planet_data = DATA[planet]
30     planet_radio = planet_data["radio"]
31     toSun = planet_data["sunDistancia"]
32     G = CONSTANTE_GRAVITACIONAL
33     sqrt = math.sqrt
34     pi = math.pi
35     K = (4*pi**2)/(G*sol["masa"])
36     #distancia del sol al planeta
37     R = sol["radio"]+toSun+planet_radio
38     periodo = sqrt(K*R**3)
39     return periodo
40
41 def printData(planet,period):
42     #K T Ms Rp
43     magnitug = "dias"
44     if(period/(3600*24) >= 400):
45         period = period/(3.1536*10**7)
46         magnitug = "anios"
47     else:
48         pedPiod = period/(3600*24)
49         magnitug = "dias"
50     str1 = f"""
51 Planeta : {planet}
52 K = {K} s^2*m^-3
53 R = {R} metros
```

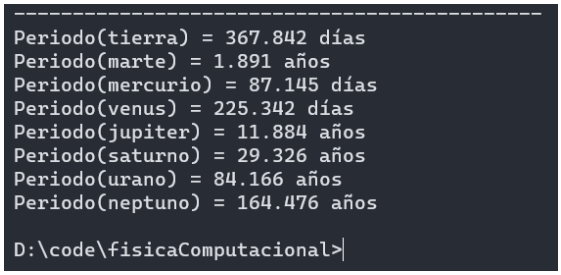
```
54 Masa del Sol = {sol["masa"]} kg
55 G = {CONSTANTE_GRAVITACIONAL} (N*m^2)/(kg^2)
56
57 Respuesta => Periodo({planet}) = {round(period,4)} {magnitug}
58 ----------------------------------------------"""
59     print(str1)
60
61
62 def main():
63     for planet in DATA.keys():
64         period = getPeriodPlanet(planet)
65         printData(planet,period)
66     print("Lista de Periodos por planeta.")
67     for planet in DATA.keys():
68         period = getPeriodPlanet(planet)
69         magnitug = "dias"
70         if(period/(3600*24) >= 400):
71             period = period/(3.1536*10**7)
72             magnitug = "anios"
73         else:
74             period = period/(3600*24)
75             magnitug = "dias"
76         print(f"Periodo({planet}) = {round(period,3)} {magnitug}")
77
78 if __name__ == "__main__":
79     main()
```

7 Resultados

Los resultados se muestran a continuación:

```
D:\code\fisicaComputacional>python ejer4.py
-----------------------------------------------
Planeta : tierra
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(tierra) = 367.8423 días
-----------------------------------------------
-----------------------------------------------
Planeta : marte
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(marte) = 1.8908 años
-----------------------------------------------
-----------------------------------------------
Planeta : mercurio
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(mercurio) = 87.1451 días
-----------------------------------------------
-----------------------------------------------
Planeta : venus
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(venus) = 225.3419 días
-----------------------------------------------
-----------------------------------------------
Planeta : jupiter
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(jupiter) = 11.884 años
-----------------------------------------------
-----------------------------------------------
Planeta : saturno
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(saturno) = 29.3256 años
-----------------------------------------------
-----------------------------------------------
Planeta : urano
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(urano) = 84.1657 años
-----------------------------------------------
-----------------------------------------------
Planeta : neptuno
K = 0 s^2*m^-3
R = 0 metros
Masa del Sol = 1.9890000000000002e+30 kg
G = 6.671999999999999e-11 (N*m^2)/(kg^2)

Respuesta => Periodo(neptuno) = 164.476 años
-----------------------------------------------
-----------------------------------------------
Periodo(tierra) = 367.842 días
Periodo(marte) = 1.891 años
Periodo(mercurio) = 87.145 días
Periodo(venus) = 225.342 días
Periodo(jupiter) = 11.884 años
Periodo(saturno) = 29.326 años
Periodo(urano) = 84.166 años
Periodo(neptuno) = 164.476 años

D:\code\fisicaComputacional>
```

A continuación, se muestra una comparación entre los calculos y datos reales de los periodos.



Planet	Rotation Period	Revolution Period
Mercury	58.6 days	87.97 days
Venus	243 days	224.7 days
Earth	0.99 days	365.26 days
Mars	1.03 days	1.88 years
Jupiter	0.41 days	11.86 years
Saturn	0.45 days	29.46 years
Uranus	0.72 days	84.01 years
Neptune	0.67 days	164.79 years
Pluto	6.39 days	248.59 years

Como se ve en la imagen anterior, los calculos se asemejan mucho a los valores reales.