Physics 77 Capstone Project

Overview

- Title of the Research: Simulation of a solar system with animation using pygame and analyse using matplotlib
- Group member: Yi-Jhan Huang, Ian Chang

Abstract:

Our research project is based on the current physical detail of solar system.

Technical Approach

- Language of programming: mainly python
- Package requirements: matplotlib, numpy, scipy, pygame, math

References

- https://towardsdatascience.com/simulate-a-tiny-solar-system-with-python-fbbb68d8207b
- https://thepythoncodingbook.com/2021/12/11/simulating-3d-solar-system-python-matplotlib/
- https://towardsdatascience.com/simulate-a-tiny-solar-system-with-python-fbbb68d8207b
- https://github.com/xhinker/orbit/blob/main/solar_orbit_3d_plt.py

Division of Responsibilities

- Ian Chang: Responsible for simulation part of the project
- Yi-Jhan Huang: Responsible for analysis part of the project
- Both: Topic Chose and Structure Design

Project Proposal

We chose this project generally because recently, as you might've known, the James Webb Space telescope sent back marvelous high-resolution images of galaxies, nebulas, black holes, and more. These new images will allow us to have a greater understanding of the history of our universe, recording information about collisions of galaxies and births of stars. Being inspired by this phenomenal event, we've decided to use what we've learnt in the course and try to simulate a solar system, mainly using the libraries matplotlib and numpy.

Additional Project Results

ppt:
https://docs.google.com/presentation/d/1Y3dm2P0M04TKX4FhHT7497pTQgfjgLqFSS9w8bCBXFo/e
usp=sharing

Attributes of Sun and Planets

	Sun	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Radius (km)	696342	2439.7	6051.8	6371.0	3389.5				
Mass (kg)	1.9885e30	3.3011e23	4.8675e24	5.97237e24	6.4171e23				

Importing Necessary Package for following codes

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pygame
import math
```

Class Planet

In [ ]:
```
class Planet:
    AU = 149.6e6 * 1000            # distance from earth to sun
    G = 6.67428e-11                # gravitational constant
    SCALE = 200 / AU               # small value to scale down the solar system to fi
    TIMESTEP = 60 * 60 * 24        # 1 day
    MASS_OF_SUN = 1.98892 * 10 ** 30 # mass of sun
    WIDTH, HEIGHT = 800, 800

    # Constructor
    def __init__(self, name, dis_to_sun, radius, color, mass):
        self.name = name
        self.x = -dis_to_sun * Planet.AU            # x position
        self.y = 0                               # y position
        self.radius = radius                     # radius
        self.color = color                       # color
        self.mass = mass                         # mass
        self.sun = (dis_to_sun == 0)            # if it's the sun
        self.dis_to_sun = dis_to_sun * Planet.AU  # distance from planet to sun
        if (self.sun): Planet.MASS_OF_SUN = mass
        self.orbit = []
        self.x_speed = 0 # x speed
        self.y_speed = 0 if (self.sun) else math.sqrt((Planet.G * Planet.MASS_OF_SUN)
```

```
    # Draw line function for the updated_points
    def move(self):
        # divide by two because pygame (0, 0) is the top left corner
        x = self.x * self.SCALE + self.WIDTH / 2      # x position on screen
        y = self.y * self.SCALE + self.HEIGHT / 2     # y position on screen
        updated_points = []
        for point in self.orbit:
            x, y = point
            x = x * self.SCALE + self.WIDTH / 2
            y = y * self.SCALE + self.HEIGHT / 2
            updated_points.append((x, y))
        return updated_points, (x, y)


    def attraction(self, other):
        distance_x = other.x - self.x # distance between x positions
        distance_y = other.y - self.y # distance between y positions

        distance = math.hypot(distance_x, distance_y) # distance between planets

        force = self.G * self.mass * other.mass / distance ** 2 # force between the pl

        theta = math.atan2(distance_y, distance_x) # angle between the planets x and y

        force_x = math.cos(theta) * force # force in the x direction
        force_y = math.sin(theta) * force # force in the y direction

        return force_x, force_y


    def update_pos(self, planets):
        total_fx = total_fy = 0 # initialize x and y force to zero

        for planet in planets:
            if self == planet: # if the planet is itself
                continue
            fx, fy = self.attraction(planet)
            total_fx += fx
            total_fy += fy

        self.x_speed += total_fx / self.mass * self.TIMESTEP
        self.y_speed += total_fy / self.mass * self.TIMESTEP

        self.x += self.x_speed * self.TIMESTEP
        self.y += self.y_speed * self.TIMESTEP

        self.orbit.append((self.x, self.y))

    def __str__(self):
        return f"""Planet(name={self.name}, radius={self.radius}, color={self.color},
        x={self.x}, y={self.y}, sun={self.sun}, orbit_list={self.orbit[0:5]}, x_speed=
        """
```

Set up constants (Color, window...etc)

In [ ]:

```
WHITE = (255, 255, 255) # color white
YELLOW = (255, 255, 0) # color yellow
```

```python
BLUE = (0, 0, 255) # color blue
RED = (255, 0, 0) # color red
GREY = (100, 100, 100) # color grey
```

Running pygame for visual simulation of solar system

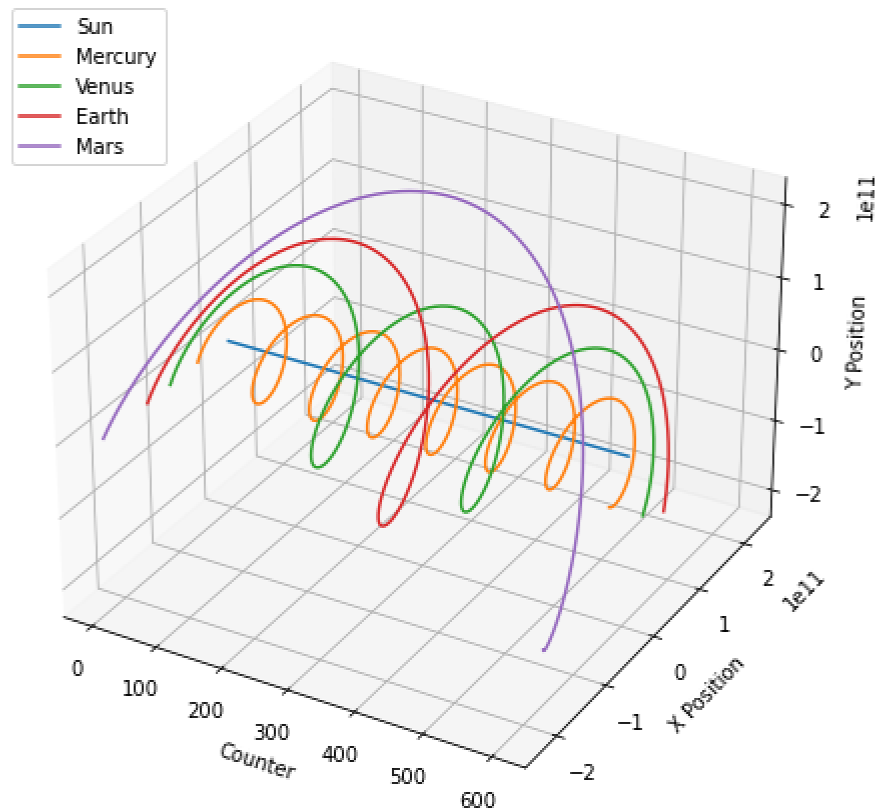
In [ ]:
```python
def simulation(planets, limit=-1, delay=True):
    pygame.init()

    # set pop up screen size
    SCREEN = pygame.display.set_mode((Planet.WIDTH, Planet.HEIGHT))

    FONT = pygame.font.SysFont("garamond", 16)

    # set title of window
    pygame.display.set_caption("Solar System")

    run = True
    clock = pygame.time.Clock()
    count = 0
    while run and count != limit:
        SCREEN.fill((0, 0, 0)) # fill the screen black every frame, or else the old fr

        # Quit loop when receive quit signal
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False

        for planet in planets:
            planet.update_pos(planets) # update the position of the planets using the
            updated_points, (x, y) = planet.move() # draw the planets

            if len(updated_points) >= 2: pygame.draw.lines(SCREEN, planet.color, False

            if not planet.sun:
                planet_name_text = FONT.render(f"{planet.name}", True, WHITE)
                SCREEN.blit(planet_name_text, \
                    (x - planet_name_text.get_width() / 2, y - planet_name_text.get_he

            pygame.draw.circle(SCREEN, planet.color, (x, y), planet.radius) # draw the

        pygame.display.update() # update the display

        if (delay): clock.tick(60) # Set up update delay
        count += 1

    pygame.quit()
```

In [ ]:
```python
planets = [
        Planet('Sun', 0, 30, YELLOW, 1.98892 * 10 ** 30),
        Planet('Mercury', 0.387, 8, GREY, 0.330 * 10 ** 24),
        Planet('Venus', 0.723, 14, WHITE, 4.8685 * 10 ** 24),
        Planet('Earth', 1, 16, BLUE, 5.972 * 10 ** 24),
        Planet('Mars', 1.524, 12, RED, 6.39 * 10 ** 23)
]

simulation(planets)
```

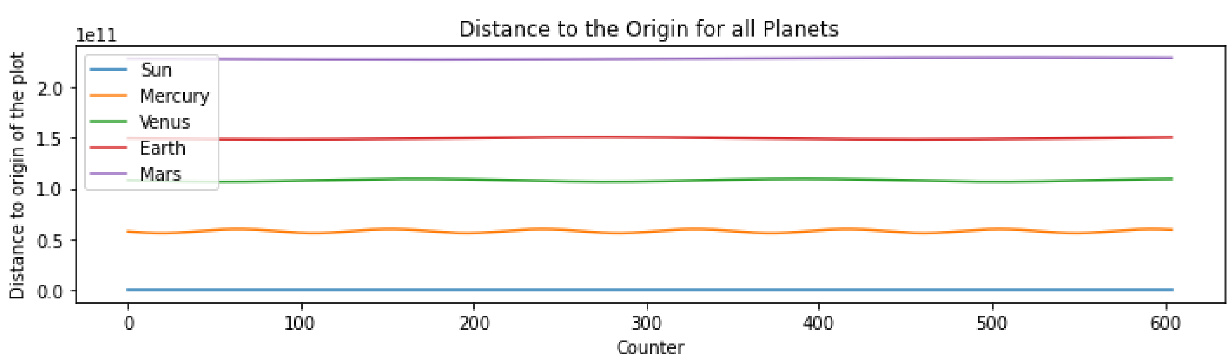
In [ ]:
```python
for planet in planets:
    print(planet)
```

```
Planet(name=Sun, radius=30, color=(255, 255, 0), mass=1.9889200000000002e+30, dis_to_
sun=0.0,
        x=-1062789.015936767, y=10673433.542766603, sun=True, orbit_list=[(-395.46942
426354053, 4.843103645849915e-14), (-1186.6330239537288, 11.685654084580875), (-2373.
254576824716, 46.78863795968859), (-3954.6290135274585, 117.0620652977588), (-5929.57
6161441081, 234.25461154743076)], x_speed=0.1655597193983845, y_speed=0.2568539398425
381)

Planet(name=Mercury, radius=8, color=(100, 100, 100), mass=3.3e+23, dis_to_sun=578952
00000.0,
        x=-37683362738.17094, y=-46013183564.21776, sun=False, orbit_list=[(-57599560
700.35345, 4137169666.4649696), (-57007536003.545845, 8253050912.057345), (-561199343
64.51558, 12326140880.266018), (-54939161637.028015, 16334838612.165207), (-534692550
26.00523, 20257568872.949585)], x_speed=-36327.758054400925, y_speed=29199.3392573231
3)

Planet(name=Venus, radius=14, color=(255, 255, 255), mass=4.8685e+24, dis_to_sun=1081
60800000.0,
        x=34534071395.09854, y=-104052047576.01506, sun=False, orbit_list=[(-10807609
6613.81282, 3026841406.6457696), (-107906656731.28897, 6051309574.138175), (-10765251
3876.02037, 9071027549.041632), (-107313768491.50269, 12083616547.297806), (-10689058
8174.23994, 15086697851.711542)], x_speed=-32602.987782993667, y_speed=-11477.6628375
2641)

Planet(name=Earth, radius=16, color=(0, 0, 255), mass=5.972e+24, dis_to_sun=149600000
000.0,
        x=80438429147.86465, y=-127442536347.96242, sun=False, orbit_list=[(-14955572
0836.4682, 2573705387.3283033), (-149467155971.98764, 5146648801.383676), (-149334311
998.49002, 7718067714.478086), (-149157208658.4194, 10287199373.326721), (-1489358788
61.96133, 12853281026.15999)], x_speed=-24798.156582354455, y_speed=-16107.5563257330
41)

Planet(name=Mars, radius=12, color=(255, 0, 0), mass=6.39e+23, dis_to_sun=22799040000
0.0,
        x=-167555318529.49585, y=-155649333407.48868, sun=False, orbit_list=[(-227971
335251.9316, 2084809296.7789679), (-227933204960.40805, 4169444268.739221), (-2278760
09925.31937, 6253730541.577781), (-227799752542.55234, 8337493726.308527), (-22770443
6804.57364, 10420559433.84463)], x_speed=-16393.06708167503, y_speed=17603.9000494856
45)
```

In [ ]:
```python
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
fig.set_size_inches(6, 6)
ax.set_xlabel("Counter")
ax.set_ylabel("X Position")
ax.set_zlabel("Y Position")

for planet in planets:
    x, y = zip(*planet.orbit)
    ax.plot(np.arange(0, len(x), 1), x, y, label=planet.name)
leg = plt.legend(loc='upper left')
plt.show()
```

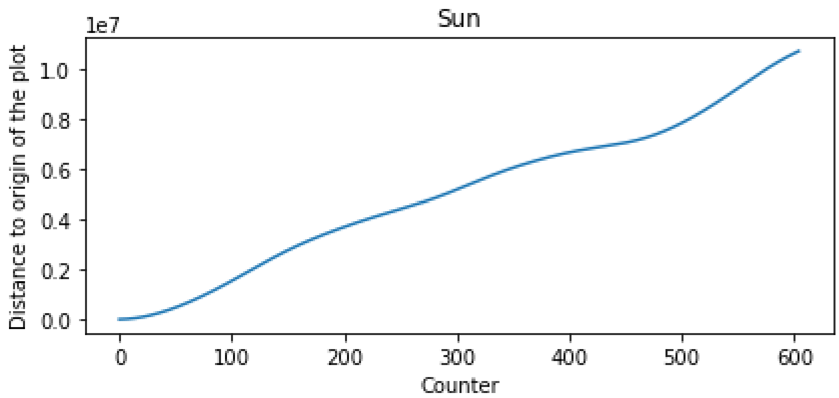


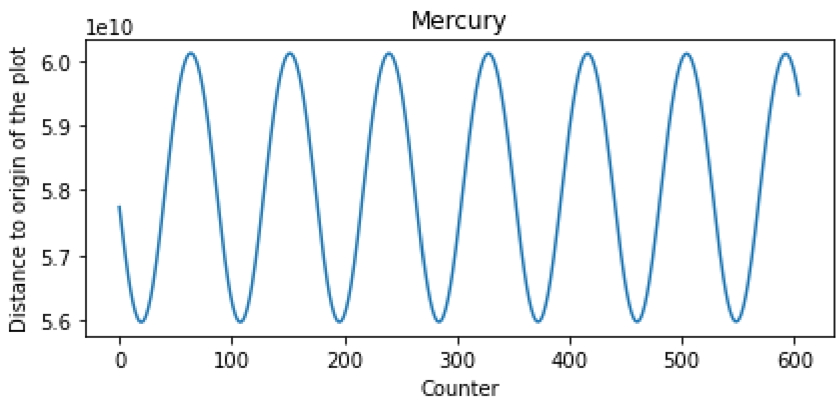
In [ ]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.plot(np.arange(0, len(x), 1), dis, label=planet.name)
plt.title("Distance to the Origin for all Planets")
plt.xlabel("Counter")
plt.ylabel("Distance to origin of the plot")
leg = plt.legend(loc='upper left')
plt.show()
```

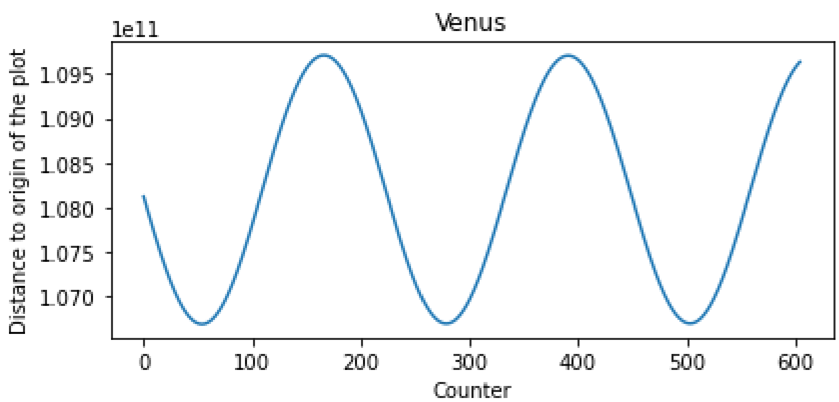


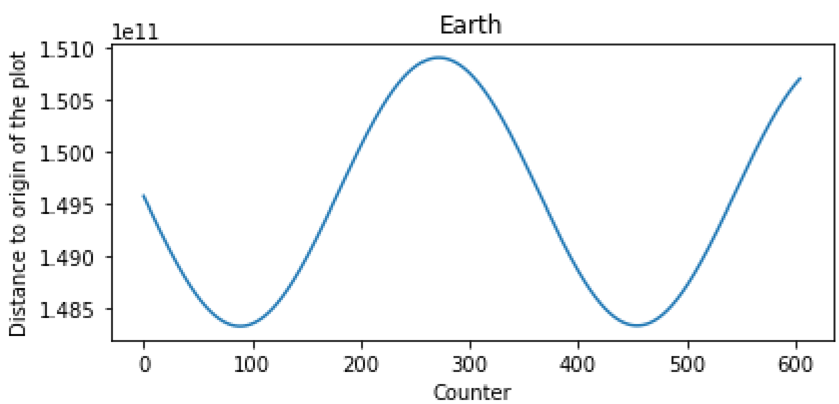
In [ ]:
```
plt.rcParams["figure.figsize"] = [6, 3]
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
```

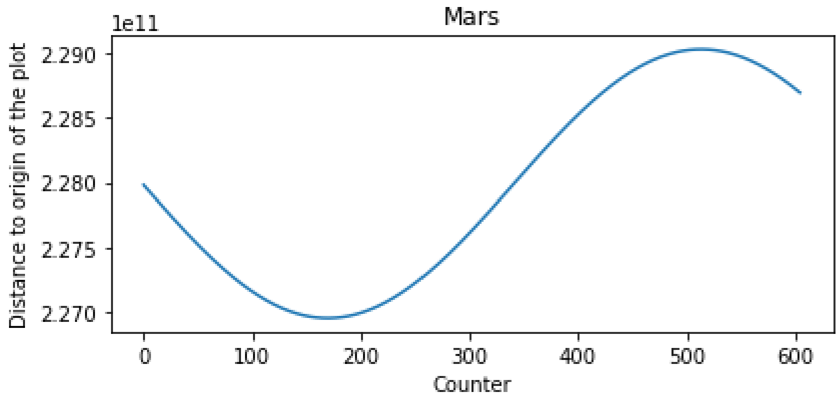
```
    plt.title(planet.name)
    plt.xlabel("Counter")
    plt.ylabel("Distance to origin of the plot")
    plt.plot(np.arange(0, len(x), 1), dis)
    plt.show()
```





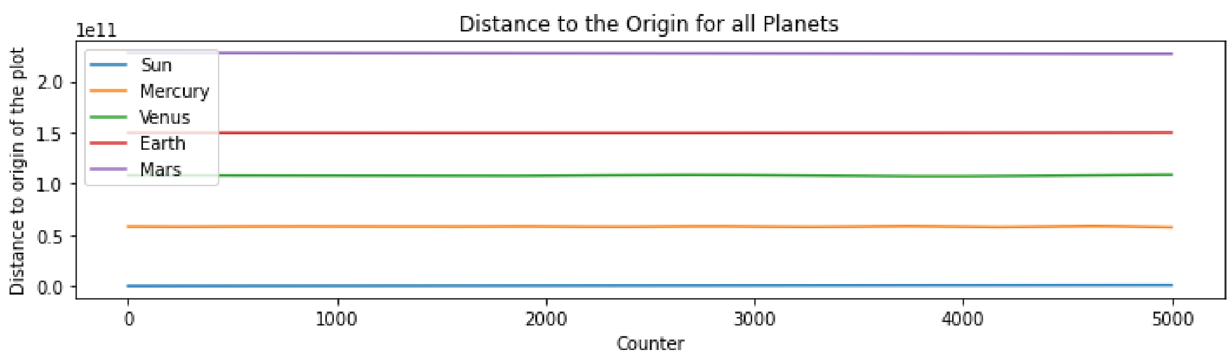






In [ ]:
```
planets = [
        Planet('Sun', 0, 30, YELLOW, 1.98892 * 10 ** 28),
        Planet('Mercury', 0.387, 8, GREY, 0.330 * 10 ** 24),
        Planet('Venus', 0.723, 14, WHITE, 4.8685 * 10 ** 24),
        Planet('Earth', 1, 16, BLUE, 5.972 * 10 ** 24),
        Planet('Mars', 1.524, 12, RED, 6.39 * 10 ** 23)
]

simulation(planets, limit=5000, delay=False)
```

In [ ]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.plot(np.arange(0, len(x), 1), dis, label=planet.name)
plt.title("Distance to the Origin for all Planets")
plt.xlabel("Counter")
plt.ylabel("Distance to origin of the plot")
leg = plt.legend(loc='upper left')
plt.show()
```

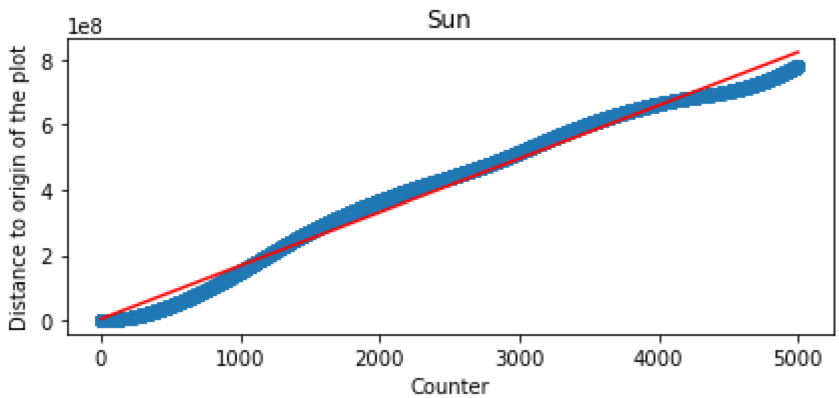


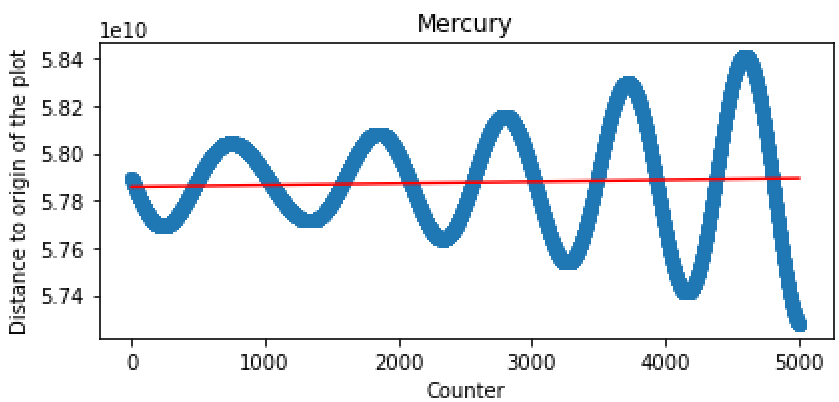
In [ ]:
```
def linear_model(x, slope, intercept):
    '''Model function to use with curve_fit();
       it should take the form of a line'''
    return x * slope + intercept
```

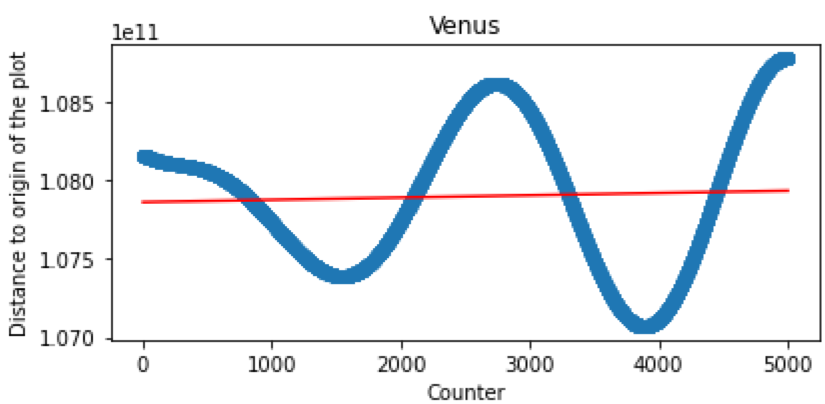
In [ ]:
```
import scipy.optimize as fitter
```

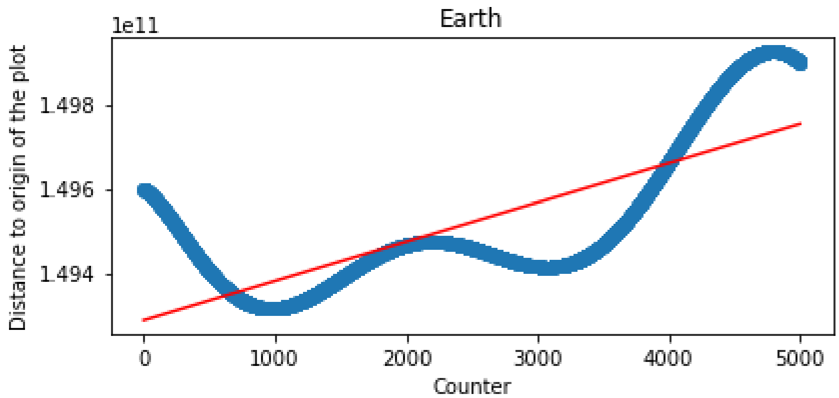
In [ ]:
```
plt.rcParams["figure.figsize"] = [6, 3]
for planet in planets:
```

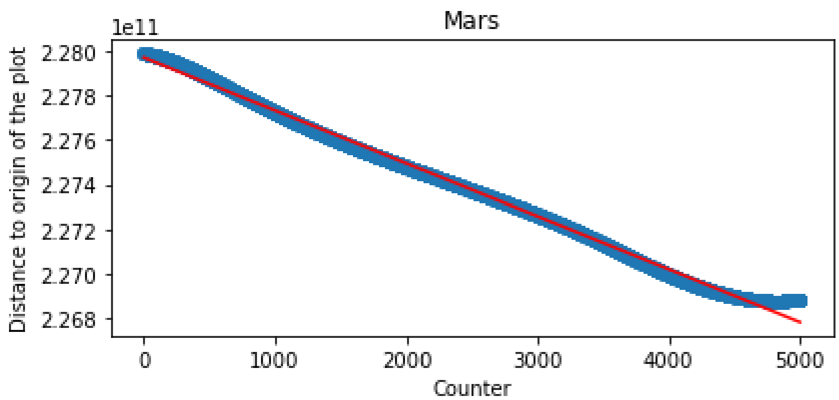
```
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.title(planet.name)
    plt.xlabel("Counter")
    plt.ylabel("Distance to origin of the plot")
    x = np.arange(0, len(x), 1)
    popt, pcov = fitter.curve_fit(linear_model, x, dis)
    plt.plot(x, linear_model(x, *popt), 'r-', label='Line of Best Fit')
    plt.scatter(x, dis)
    plt.plot()
    plt.show()
```





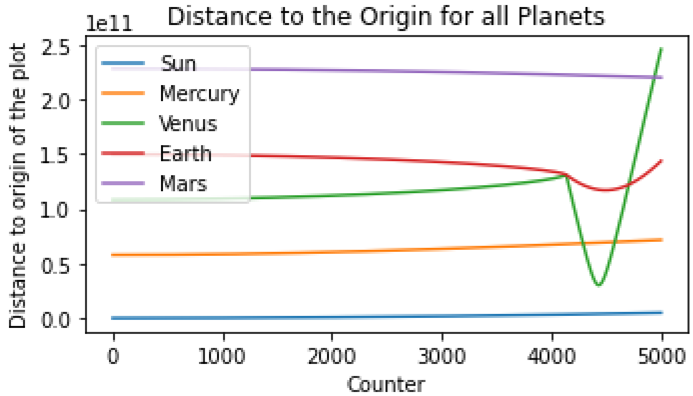




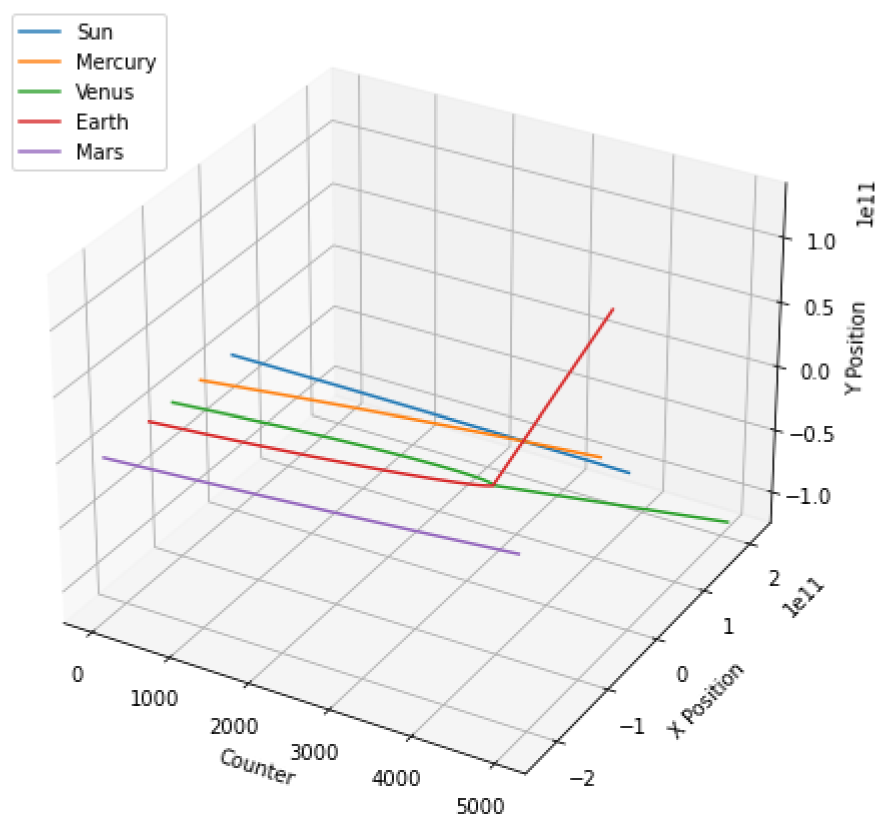


In [ ]:
```
planets = [
        Planet('Sun', 0, 30, YELLOW, 5.972 * 10 ** 24),
        Planet('Mercury', 0.387, 8, GREY, 0.330 * 10 ** 24),
        Planet('Venus', 0.723, 14, WHITE, 4.8685 * 10 ** 24),
        Planet('Earth', 1, 16, BLUE, 5.972 * 10 ** 24),
        Planet('Mars', 1.524, 12, RED, 6.39 * 10 ** 23)
]

simulation(planets, limit=5000, delay=False)
```

In [ ]:
```
plt.rcParams["figure.figsize"] = [5, 3]
plt.rcParams["figure.autolayout"] = True
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.plot(np.arange(0, len(x), 1), dis, label=planet.name)
plt.title("Distance to the Origin for all Planets")
plt.xlabel("Counter")
plt.ylabel("Distance to origin of the plot")
leg = plt.legend(loc='upper left')
plt.show()
```



In [ ]:
```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
fig.set_size_inches(6, 6)
ax.set_xlabel("Counter")
ax.set_ylabel("X Position")
ax.set_zlabel("Y Position")

for planet in planets:
    x, y = zip(*planet.orbit)
    ax.plot(np.arange(0, len(x), 1), x, y, label=planet.name)
leg = plt.legend(loc='upper left')
plt.show()
```

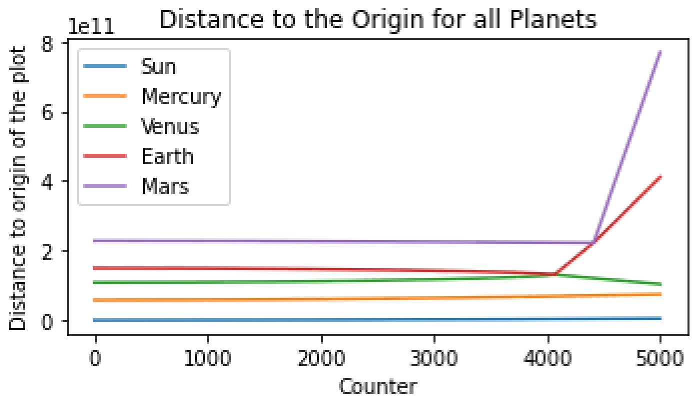


In [ ]:
```
planets = [
        Planet('Sun', 0, 30, YELLOW, 1),
        Planet('Mercury', 0.387, 8, GREY, 0.330 * 10 ** 24),
        Planet('Venus', 0.723, 14, WHITE, 4.8685 * 10 ** 24),
        Planet('Earth', 1, 16, BLUE, 5.972 * 10 ** 24),
        Planet('Mars', 1.524, 12, RED, 6.39 * 10 ** 23)
```

```
]

simulation(planets, limit=5000, delay=False)
```

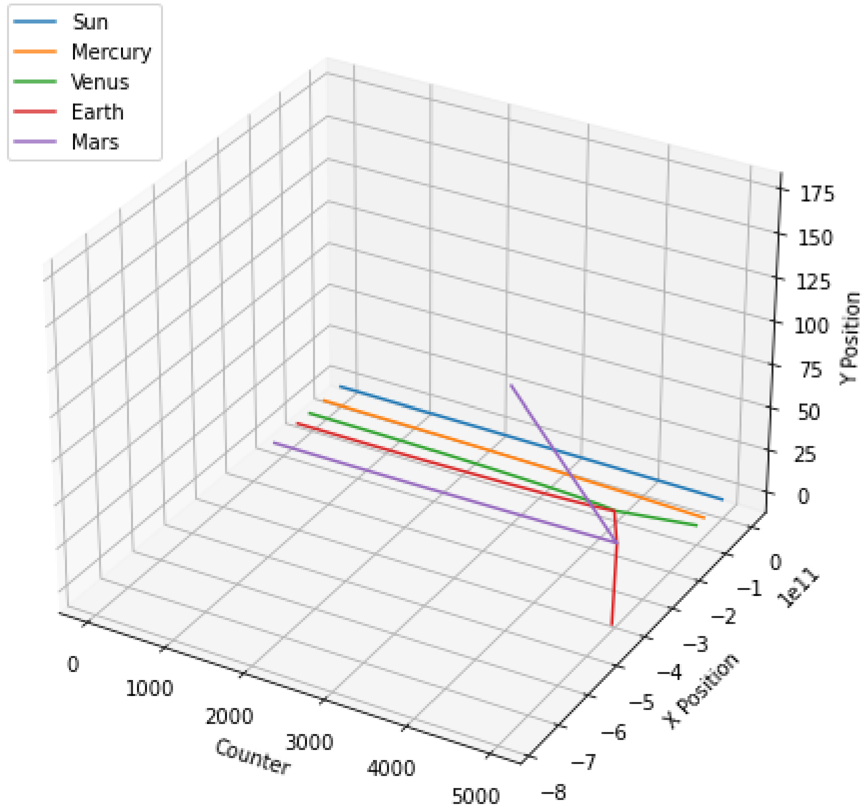
In [ ]:

```
plt.rcParams["figure.figsize"] = [5, 3]
plt.rcParams["figure.autolayout"] = True
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.plot(np.arange(0, len(x), 1), dis, label=planet.name)
plt.title("Distance to the Origin for all Planets")
plt.xlabel("Counter")
plt.ylabel("Distance to origin of the plot")
leg = plt.legend(loc='upper left')
plt.show()
```



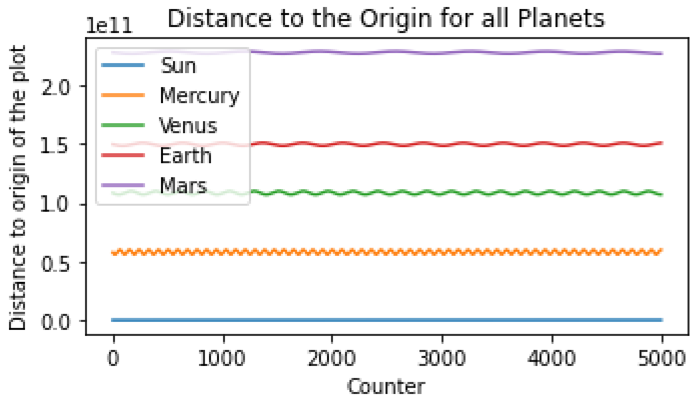
In [ ]:

```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
fig.set_size_inches(6, 6)
ax.set_xlabel("Counter")
ax.set_ylabel("X Position")
ax.set_zlabel("Y Position")

for planet in planets:
    x, y = zip(*planet.orbit)
    ax.plot(np.arange(0, len(x), 1), x, y, label=planet.name)
leg = plt.legend(loc='upper left')
plt.show()
```

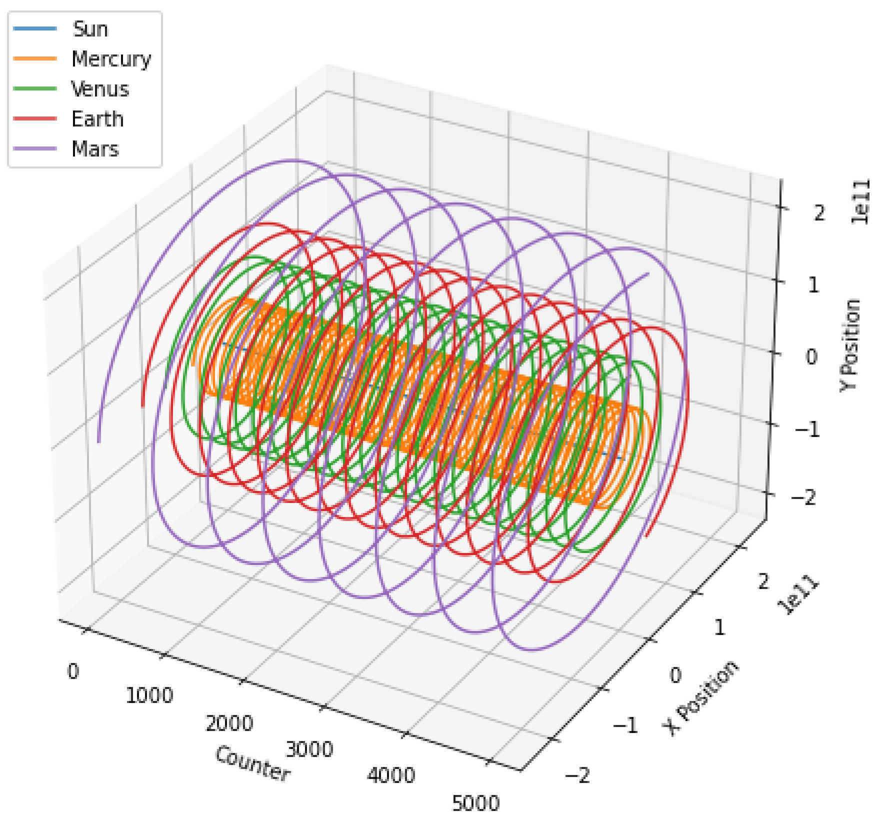


In [ ]:
```
planets = [
        Planet('Sun', 0, 30, YELLOW, 1.98892 * 10 ** 30),
        Planet('Mercury', 0.387, 8, GREY, 0.330 * 10 ** 24),
        Planet('Venus', 0.723, 14, WHITE, 4.8685 * 10 ** 24),
        Planet('Earth', 1, 16, BLUE, 5.972 * 10 ** 24),
        Planet('Mars', 1.524, 12, RED, 6.39 * 10 ** 23)
]

simulation(planets, limit=5000, delay=False)
```

In [ ]:
```
plt.rcParams["figure.figsize"] = [5, 3]
plt.rcParams["figure.autolayout"] = True
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.plot(np.arange(0, len(x), 1), dis, label=planet.name)
plt.title("Distance to the Origin for all Planets")
plt.xlabel("Counter")
plt.ylabel("Distance to origin of the plot")
leg = plt.legend(loc='upper left')
plt.show()
```



In [ ]:
```python
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
fig.set_size_inches(6, 6)
ax.set_xlabel("Counter")
ax.set_ylabel("X Position")
ax.set_zlabel("Y Position")

for planet in planets:
    x, y = zip(*planet.orbit)
    ax.plot(np.arange(0, len(x), 1), x, y, label=planet.name)
leg = plt.legend(loc='upper left')
plt.show()
```

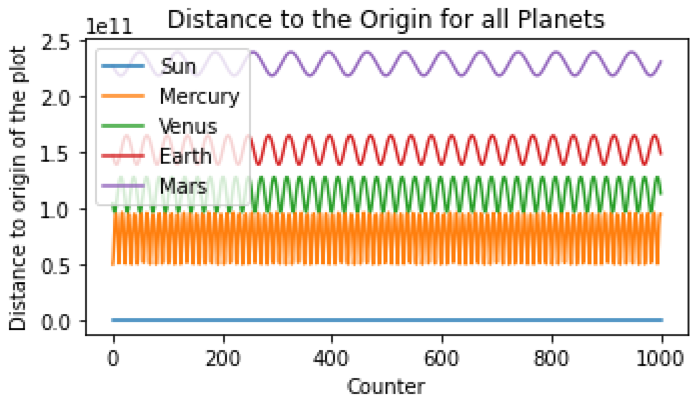


In [ ]:
```python
planets = [
        Planet('Sun', 0, 30, YELLOW, 1.98892 * 10 ** 32),
        Planet('Mercury', 0.387, 8, GREY, 0.330 * 10 ** 24),
        Planet('Venus', 0.723, 14, WHITE, 4.8685 * 10 ** 24),
        Planet('Earth', 1, 16, BLUE, 5.972 * 10 ** 24),
        Planet('Mars', 1.524, 12, RED, 6.39 * 10 ** 23)
```

```
]

simulation(planets, limit=1000, delay=False)
```

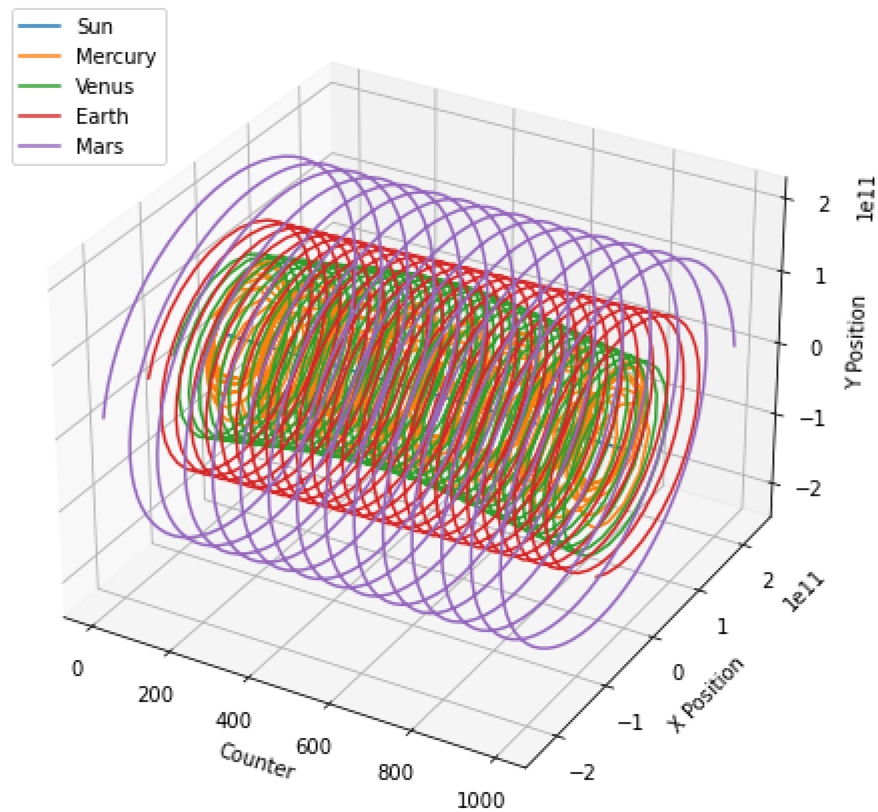
In [ ]:

```
plt.rcParams["figure.figsize"] = [5, 3]
plt.rcParams["figure.autolayout"] = True
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.plot(np.arange(0, len(x), 1), dis, label=planet.name)
plt.title("Distance to the Origin for all Planets")
plt.xlabel("Counter")
plt.ylabel("Distance to origin of the plot")
leg = plt.legend(loc='upper left')
plt.show()
```

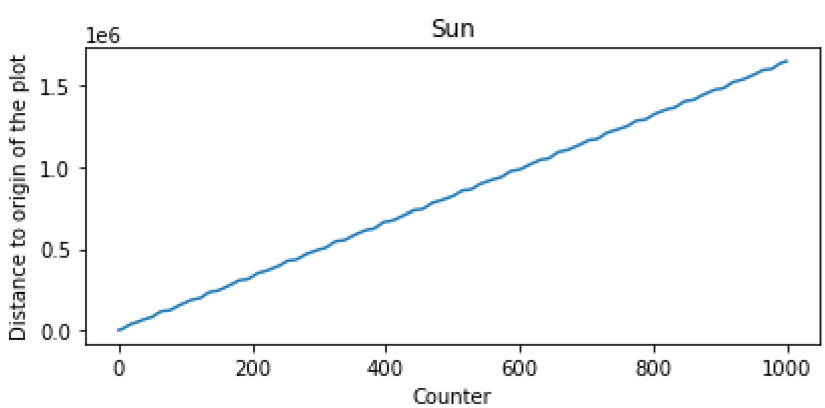


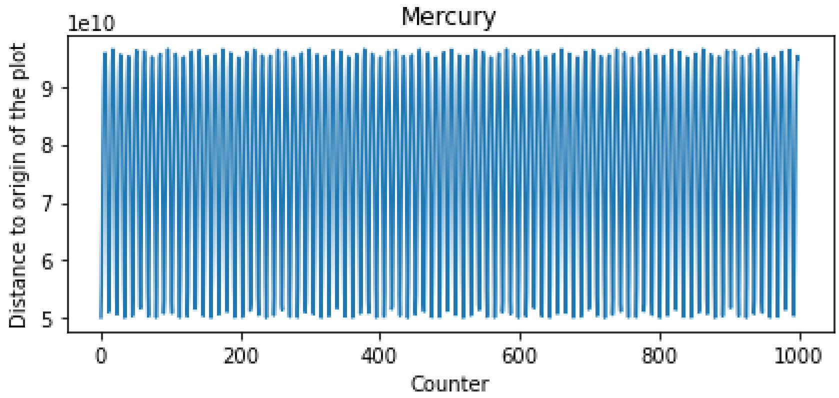
In [ ]:

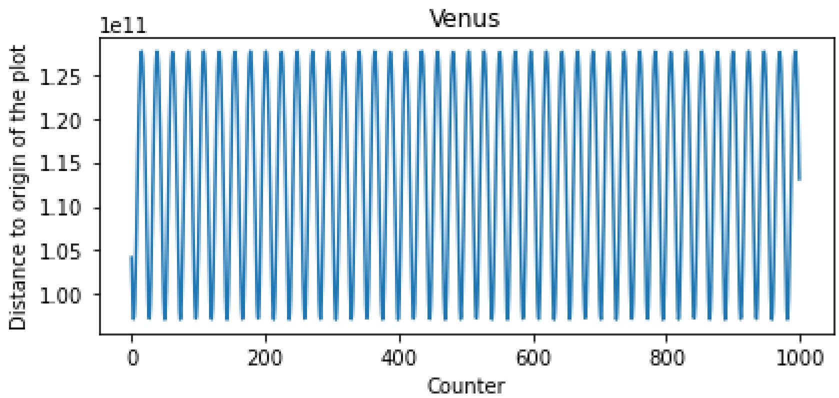
```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
fig.set_size_inches(6, 6)
ax.set_xlabel("Counter")
ax.set_ylabel("X Position")
ax.set_zlabel("Y Position")

for planet in planets:
    x, y = zip(*planet.orbit)
    ax.plot(np.arange(0, len(x), 1), x, y, label=planet.name)
leg = plt.legend(loc='upper left')
plt.show()
```

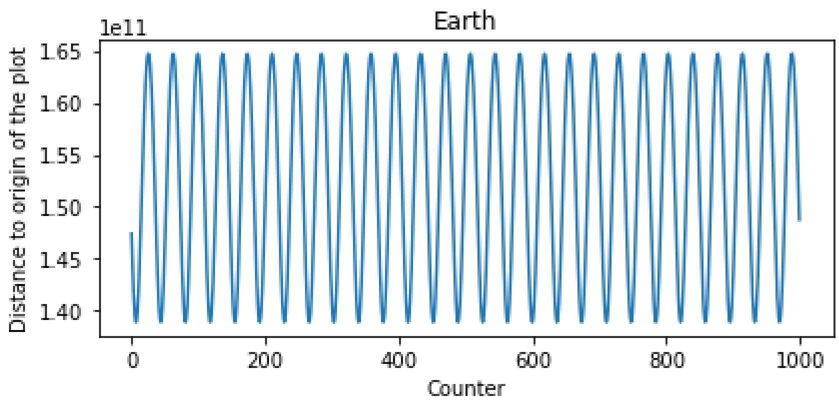


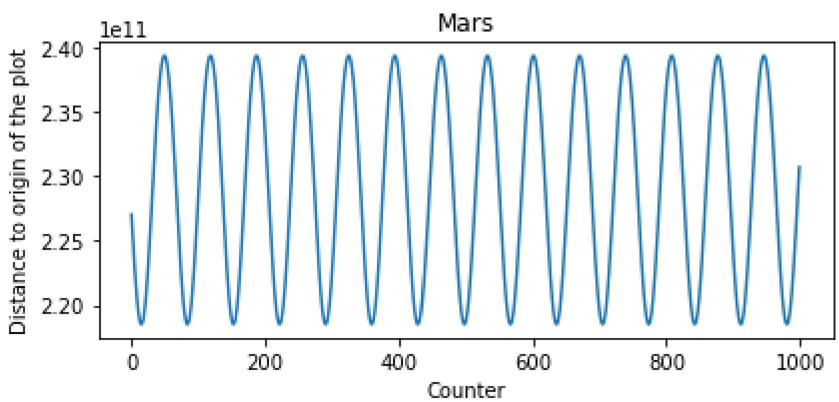
In [ ]:
```
plt.rcParams["figure.figsize"] = [6, 3]
for planet in planets:
    x, y = zip(*planet.orbit)
    dis = []
    for i in range(len(x)):
        dis.append(math.hypot(x[i], y[i]))
    plt.title(planet.name)
    plt.xlabel("Counter")
    plt.ylabel("Distance to origin of the plot")
    plt.plot(np.arange(0, len(x), 1), dis)
    plt.show()
```




Mercury
1e10
Distance to origin of the plot
9
8
7
6
5
0
200
400
600
800
1000
Counter


Venus
1e11
Distance to origin of the plot
1.25
1.20
1.15
1.10
1.05
1.00
0
200
400
600
800
1000
Counter


Earth
1e11
Distance to origin of the plot
1.65
1.60
1.55
1.50
1.45
1.40
0
200
400
600
800
1000
Counter


Mars
1e11
Distance to origin of the plot
2.40
2.35
2.30
2.25
2.20
0
200
400
600
800
1000
Counter