

## Kasidhad Physics Final Project

### Given Instructions:

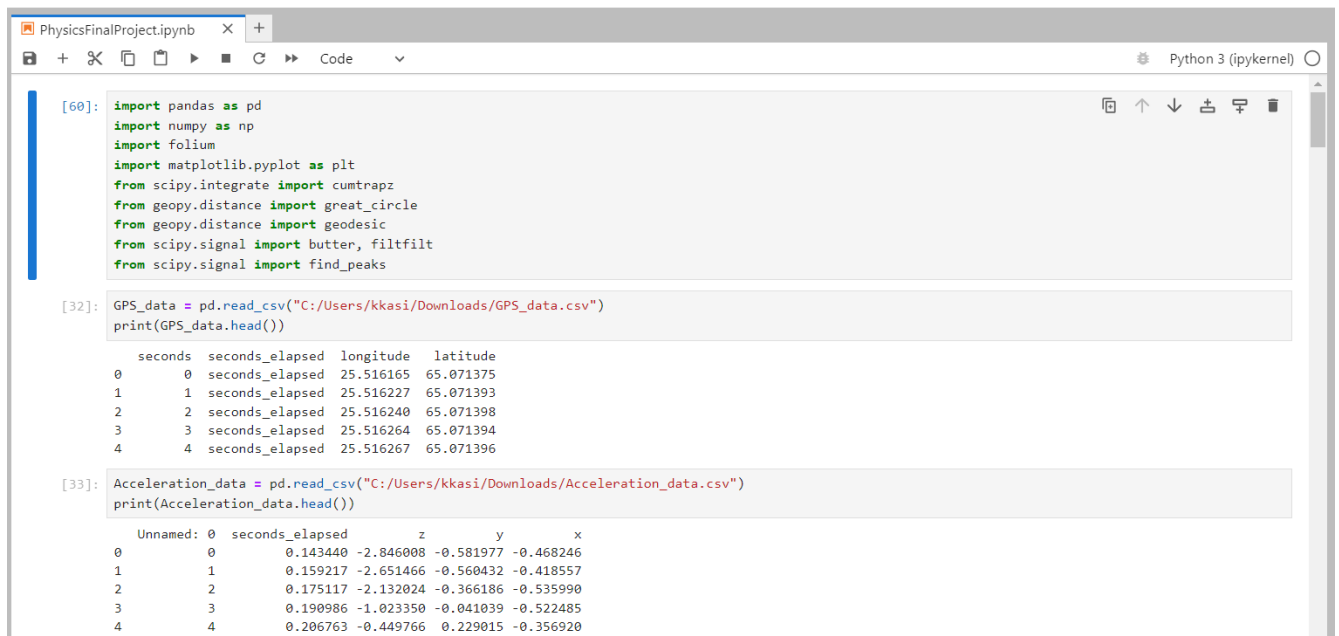
The task is to write a Jupyter Notebook in Python where the following things are calculated and visualized either as a diagram or as numbers.

### Show the following things.

1. A map of movement (Folium)
2. Distance graph (increasing from zero)
3. Velocity graph (calculated from coordinates)
4. Acceleration graph (observed [three components] and calculated from speed [one component]).  
Note that these will be very different from each other!
5. Average speed as a number
6. Traveled distance as a number
7. The number of steps

Below will be screenshots of the codes and the outputs to complete the Physics Final Project.

In this first image, I imported all the necessary extensions to complete the project. I printed out the first columns of the data to show that I had added a title ('seconds') to the GPS\_data which would help me in the later tasks.



```
[60]: import pandas as pd
import numpy as np
import folium
import matplotlib.pyplot as plt
from scipy.integrate import cumtrapz
from geopy.distance import great_circle
from geopy.distance import geodesic
from scipy.signal import butter, filtfilt
from scipy.signal import find_peaks

[32]: GPS_data = pd.read_csv("C:/Users/kkasi/Downloads/GPS_data.csv")
print(GPS_data.head())
```

	seconds	seconds_elapsed	longitude	latitude
0	0	seconds_elapsed	25.516165	65.071375
1	1	seconds_elapsed	25.516227	65.071393
2	2	seconds_elapsed	25.516240	65.071398
3	3	seconds_elapsed	25.516264	65.071394
4	4	seconds_elapsed	25.516267	65.071396

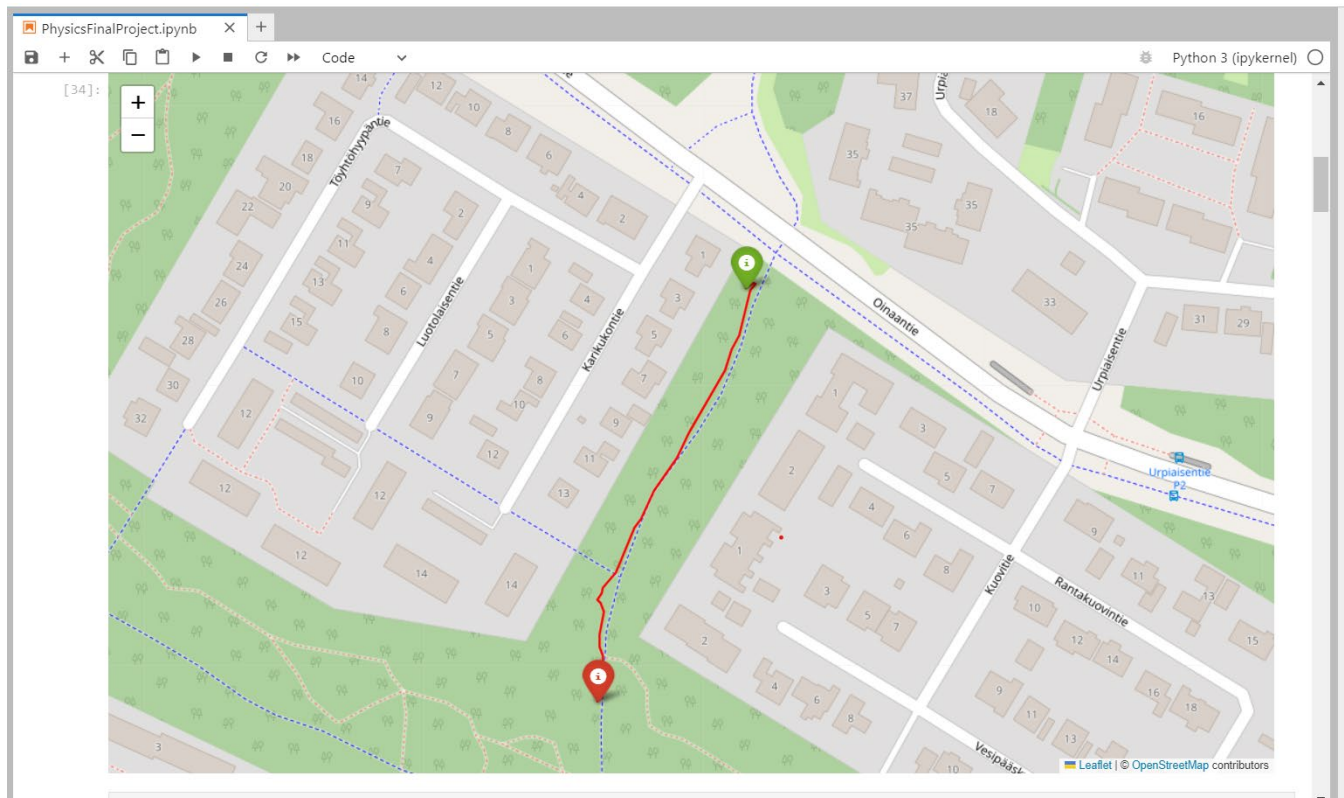
```
[33]: Acceleration_data = pd.read_csv("C:/Users/kkasi/Downloads/Acceleration_data.csv")
print(Acceleration_data.head())
```

Unnamed: 0	seconds_elapsed	z	y	x	
0	0	0.143440	-2.846008	-0.581977	-0.468246
1	1	0.159217	-2.651466	-0.560432	-0.418557
2	2	0.175117	-2.132024	-0.366186	-0.535990
3	3	0.190986	-1.023350	-0.041039	-0.522485
4	4	0.206763	-0.449766	0.229015	-0.356920

1. In these 2 images, I had written the code to display the map of movement. I made the line red to make it clearer and added markers to help improve the ease of graph readings.

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel)

[34]: def create_movement_map(GPS_data):
#I assume that the data sorted going down the group means location from start to finish
start_location = GPS_data.iloc[0][['latitude', 'longitude']]
movement_map = folium.Map(location=start_location, zoom_start=16, tiles="OpenStreetMap")
folium.PolyLine(GPS_data[['latitude', 'longitude']].values, color="red", weight=2, opacity=1).add_to(movement_map)
#Making it look nice (I searched this on YouTube)
folium.Marker(GPS_data.iloc[0][['latitude', 'longitude']].values, popup='Start', icon=folium.Icon(color='green')).add_to(movement_map)
folium.Marker(GPS_data.iloc[-1][['latitude', 'longitude']].values, popup='End', icon=folium.Icon(color='red')).add_to(movement_map)
return movement_map
movement_map = create_movement_map(GPS_data)
movement_map
```



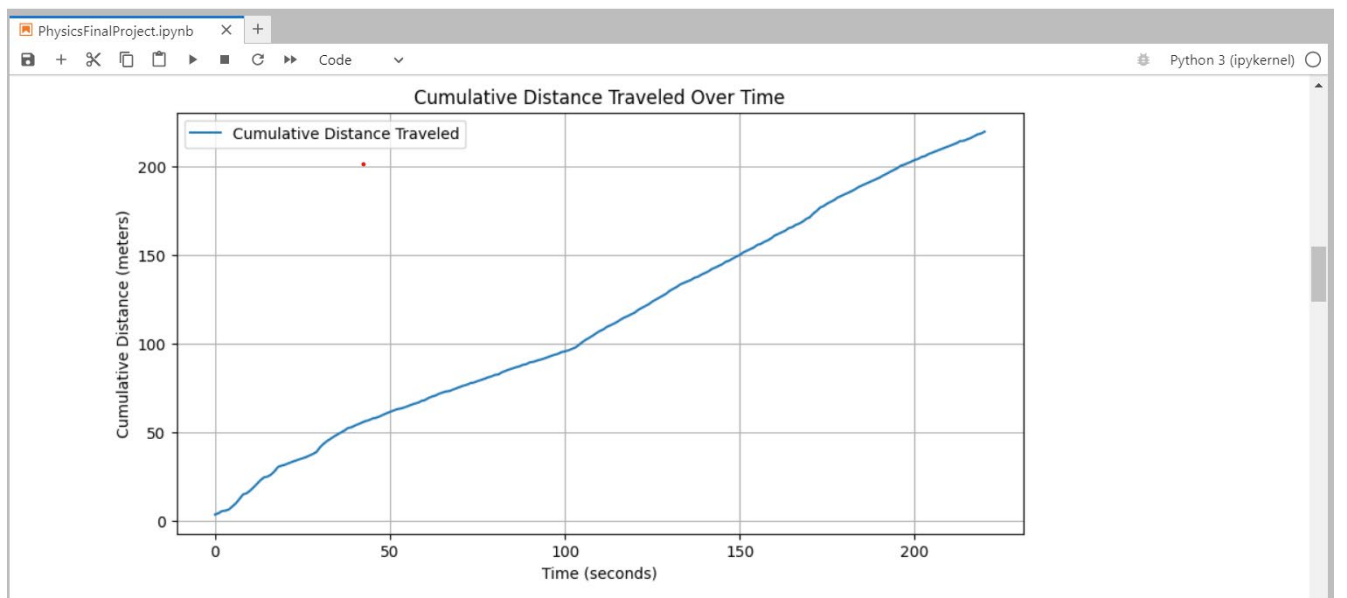
2. In the next 3 images, I wrote the code to calculate and display the cumulative distance travelled over time as a graph. I assumed that each time interval was taken every 1 second, then calculated the approximate distance travelled (in meters) every time interval.

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel)

[35]: def calculate_distances(latitudes, longitudes):
        distances = []
        for i in range(1, len(latitudes)):
            point1 = (latitudes[i-1], longitudes[i-1])
            point2 = (latitudes[i], longitudes[i])
            distance = geodesic(point1, point2).meters
            distances.append(distance)
        return np.array(distances)

[36]: latitudes = GPS_data['latitude'].values
        longitudes = GPS_data['longitude'].values
        distances = calculate_distances(latitudes, longitudes)
        cumulative_distances = np.cumsum(distances)

[37]: plt.figure(figsize=(10, 5))
        plt.plot(cumulative_distances, label='Cumulative Distance Traveled')
        plt.xlabel('Time (seconds)')
        plt.ylabel('Cumulative Distance (meters)')
        plt.title('Cumulative Distance Traveled Over Time')
        plt.legend()
        plt.grid(True)
        plt.show()
```



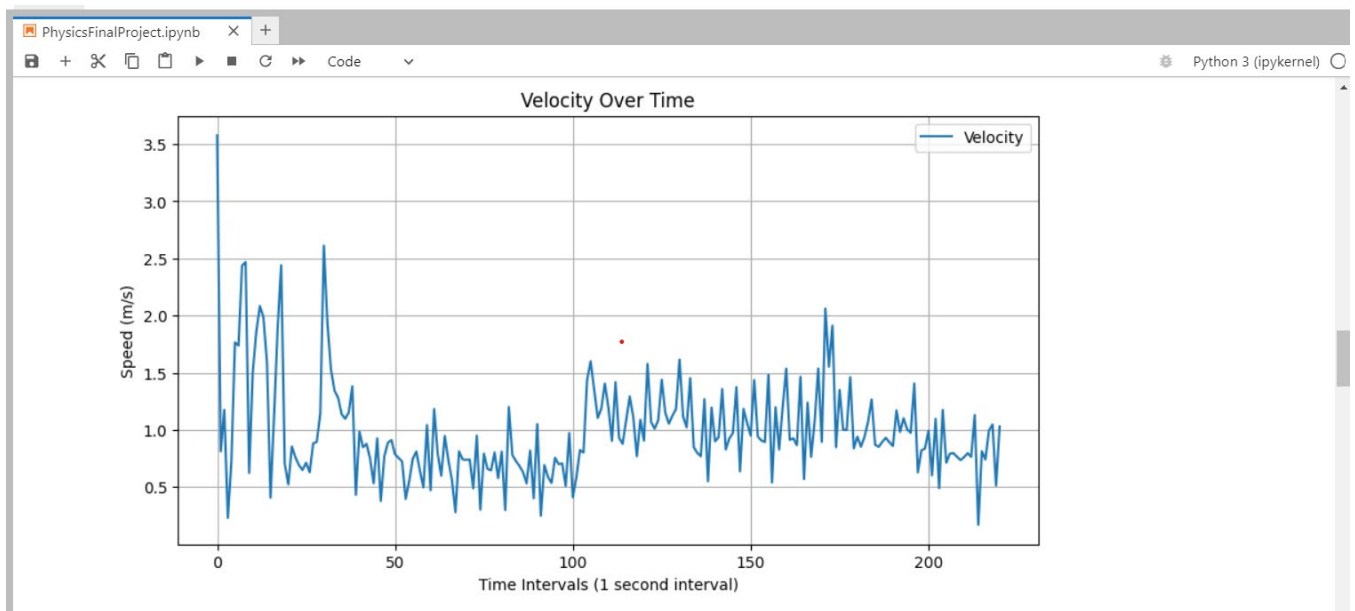
3. To write out the code to display a velocity graph, I used the Haversine formula to calculate the distances and the distances between the points.

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel) C

[39]: #calculating the distances using the Haversine formula
def calculate_distances(latitudes, longitudes):
    distances = []
    for i in range(1, len(latitudes)):
        point1 = (latitudes[i-1], longitudes[i-1])
        point2 = (latitudes[i], longitudes[i])
        distance = geodesic(point1, point2).meters
        distances.append(distance)
    return np.array(distances)

#calculating distances between successive points
latitudes = GPS_data['latitude'].values
longitudes = GPS_data['longitude'].values
distances = calculate_distances(latitudes, longitudes)
time_interval = 1 #I assume each time interval was 1
speeds = distances / time_interval

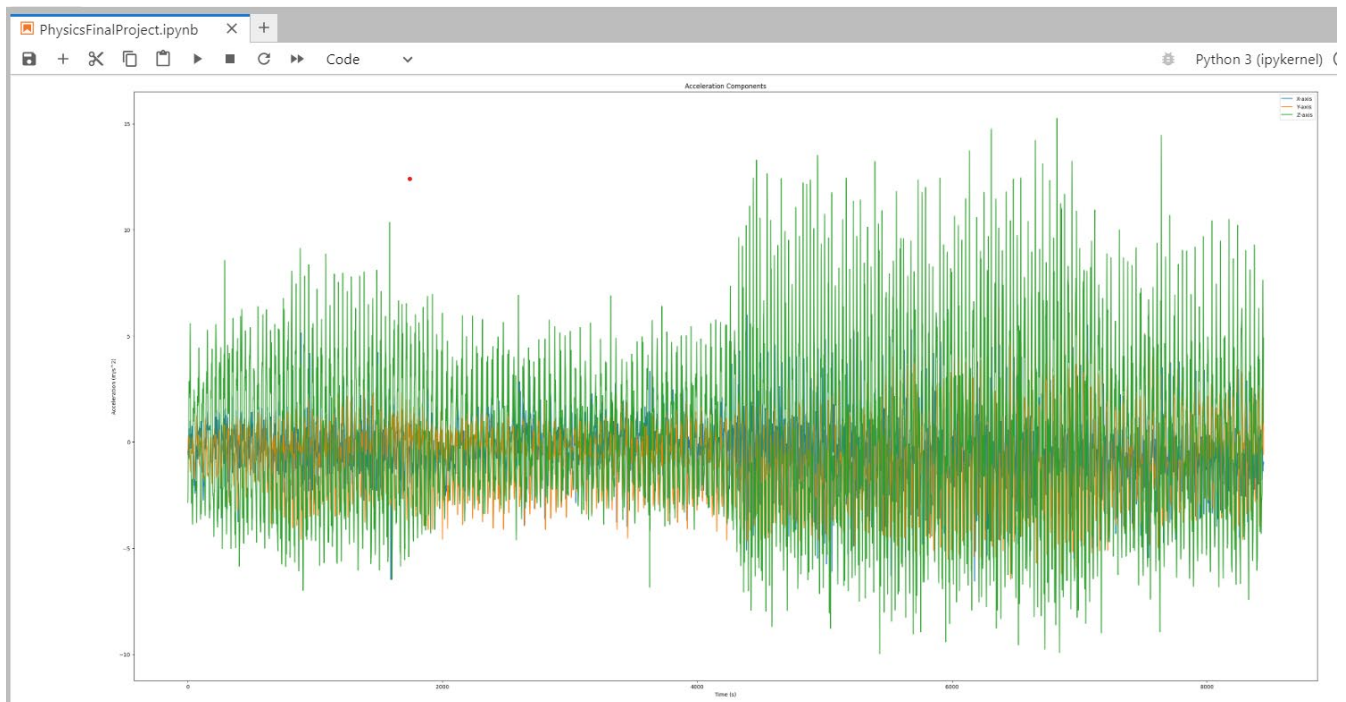
[40]: plt.figure(figsize=(10, 5))
plt.plot(speeds, label='Velocity')
plt.xlabel('Time Intervals (1 second interval)')
plt.ylabel('Speed (m/s)')
plt.title('Velocity Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



4. I printed out the acceleration of all 3 components in 3 different manners. Firstly, I printed them all out into 1 large graph, however, this first method would be difficult to read, even after adjusting the figure size several times. Hence, I decided to also plot the x, y, and z components as 3 separate graphs. Furthermore, I also filtered the data and plotted them into 3 separate graphs to analyze between the unfiltered versions. The x, y, and z components' colors remain constant within all 3 versions of the graph where  $x$ =blue,  $y$ =orange, and  $z$ =green.

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel)

[41]: plt.figure(figsize=(40,20))
      plt.plot(Acceleration_data['x'], label='X-axis')
      plt.plot(Acceleration_data['y'], label='Y-axis')
      plt.plot(Acceleration_data['z'], label='Z-axis')
      plt.legend()
      plt.title('Acceleration Components')
      plt.xlabel('Time (s)')
      plt.ylabel('Acceleration (m/s^2)')
      plt.show()
```



```
PhysicsFinalProject.ipynb Python 3 (ipykernel)

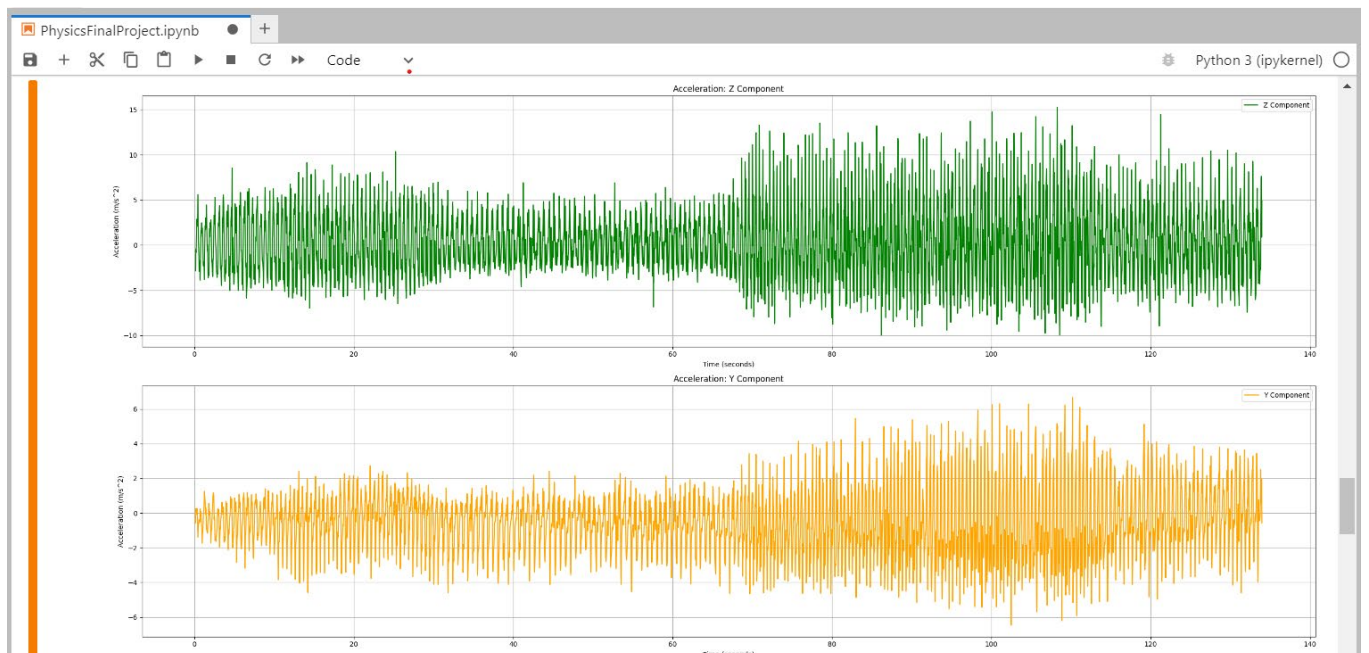
•[271]: #plotting the observed acceleration components into 3 separate graphs
plt.figure(figsize=(25, 18))

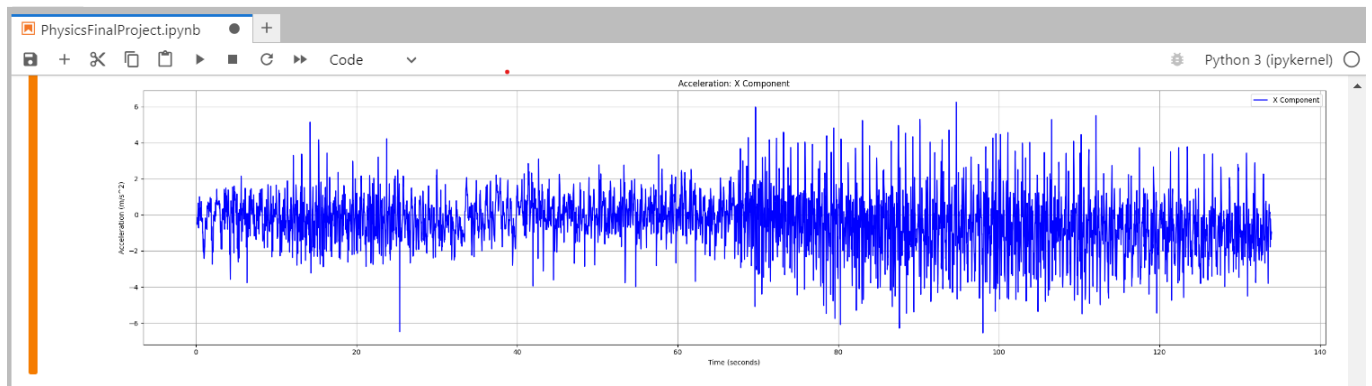
plt.subplot(3, 1, 3)
plt.plot(Acceleration_data['seconds_elapsed'], Acceleration_data['x'], label='X Component', color='blue')
plt.xlabel('Time (seconds)')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Acceleration: X Component')
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(Acceleration_data['seconds_elapsed'], Acceleration_data['y'], label='Y Component', color='orange')
plt.xlabel('Time (seconds)')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Acceleration: Y Component')
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 1)
plt.plot(Acceleration_data['seconds_elapsed'], Acceleration_data['z'], label='Z Component', color='green')
plt.xlabel('Time (seconds)')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Acceleration: Z Component')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```





PhysicsFinalProject.ipynb Python 3 (ipykernel)

```
[274]: #Filtering the data
def low_pass_filter(data, cutoff_freq, fs, order=5):
    nyquist_freq = 0.5 * fs
    normal_cutoff = cutoff_freq / nyquist_freq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    filtered_data = filtfilt(b, a, data)
    return filtered_data

fs = 50
cutoff_freq = 2 #I googled a good cutoff frequency and it suggested around 2-3
filtered_x = low_pass_filter(Acceleration_data['x'], cutoff_freq, fs)
filtered_y = low_pass_filter(Acceleration_data['y'], cutoff_freq, fs)
filtered_z = low_pass_filter(Acceleration_data['z'], cutoff_freq, fs)
```

PhysicsFinalProject.ipynb Python 3 (ipykernel)

```
[11]: #plotting filtered data into 3 seperate graphs
plt.figure(figsize=(25, 18))

plt.subplot(3, 1, 3)
plt.plot(Acceleration_data['seconds_elapsed'], filtered_x, label='Filtered X Component', color='blue')
plt.xlabel('Time (seconds)')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Filtered Acceleration: X Component')
plt.legend()
plt.grid(True)

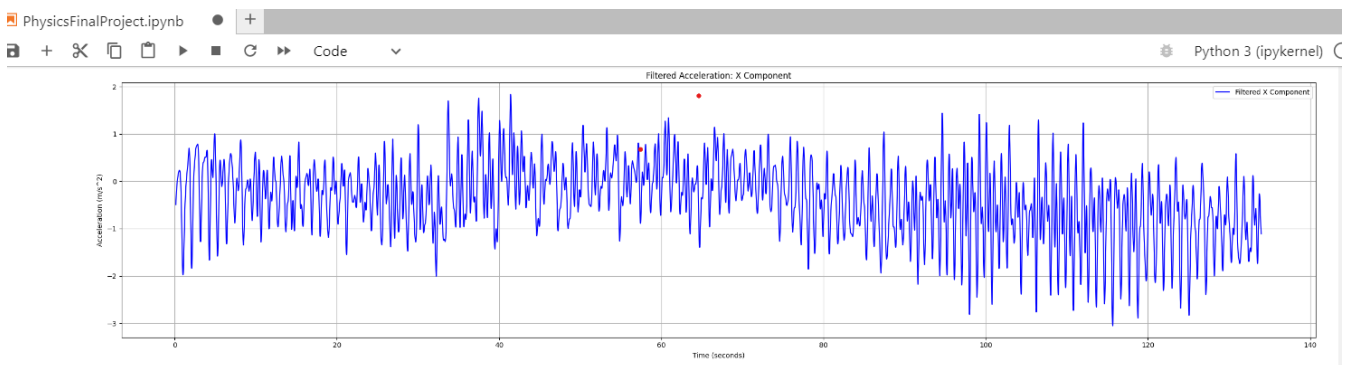
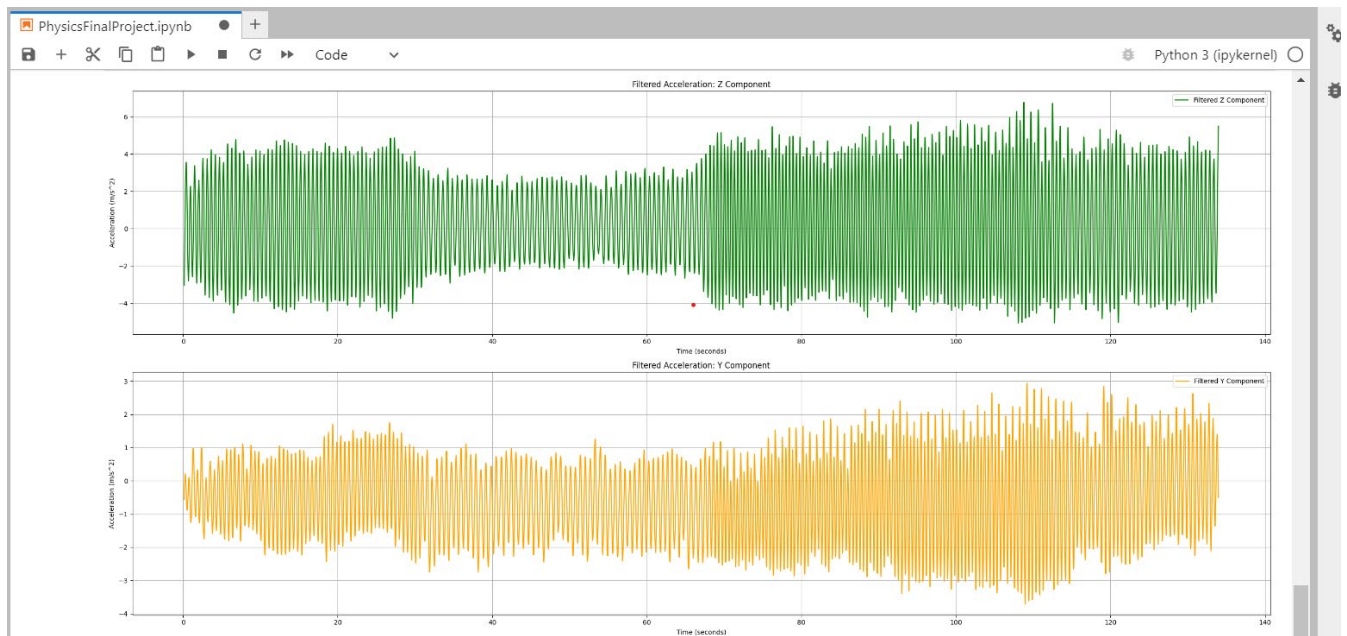
plt.subplot(3, 1, 2)
plt.plot(Acceleration_data['seconds_elapsed'], filtered_y, label='Filtered Y Component', color='orange')
plt.xlabel('Time (seconds)')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Filtered Acceleration: Y Component')
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 1)
plt.plot(Acceleration_data['seconds_elapsed'], filtered_z, label='Filtered Z Component', color='green')
plt.xlabel('Time (seconds)')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Filtered Acceleration: Z Component')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

Filtered Acceleration: Z Component







## 5. Calculating the average speed (in meters per second)

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel)

[63]: def calculate_speeds(latitudes, longitudes):
        speeds = []
        for i in range(1, len(latitudes)):
            distance = geodesic((latitudes[i-1], longitudes[i-1]), (latitudes[i], longitudes[i])).meters
            speed = distance / 1
            speeds.append(speed)
        return np.array(speeds)
    speeds = calculate_speeds(GPS_data['latitude'], GPS_data['longitude'])
    average_speed_m_per_s = np.mean(speeds)
    print(f"The average speed is approximately: {average_speed_m_per_s:.2f} m/s")

The average speed is approximately: 0.99 m/s
```

## 6. To calculate the approximate distance travelled, I used the data from the GPS\_data file to measure the distances between the start and end longitude and latitude.

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel)

[64]: def calculate_total_distance(latitudes, longitudes):
        total_distance = 0
        for i in range(1, len(latitudes)):
            start = (latitudes[i-1], longitudes[i-1])
            end = (latitudes[i], longitudes[i])
            total_distance += geodesic(start, end).meters
        return total_distance

    latitudes = GPS_data['latitude']
    longitudes = GPS_data['longitude']

    total_distance_meters = calculate_total_distance(latitudes, longitudes)

    print(f"The total distance travelled is Approximately: {total_distance_meters:.2f} meters")

The total distance travelled is Approximately: 219.34 meters
```

## 7. Similar to the week 4 assignment, I used the filtered data set to calculate the steps. To ensure that the steps were not exponentially overcalculated, I had to set the initial steps to be 0 and set each wave in the data to be 0.5 as each step represents the entire wave of up and down.

```
PhysicsFinalProject.ipynb X +
Python 3 (ipykernel)

[71]: def low_pass_filter(data, cutoff_freq, fs, order=5):
        nyquist_freq = 0.5 * fs
        normal_cutoff = cutoff_freq / nyquist_freq
        b, a = butter(order, normal_cutoff, btype='low', analog=False)
        filtered_data = filtfilt(b, a, data)
        return filtered_data

    fs = 50
    cutoff_freq = 2
    filtered_x = low_pass_filter(Acceleration_data['x'], cutoff_freq, fs)
    filtered_y = low_pass_filter(Acceleration_data['y'], cutoff_freq, fs)
    filtered_z = low_pass_filter(Acceleration_data['z'], cutoff_freq, fs)

    steps = 0
    for i in range(1, len(filtered_z)):
        if filtered_z[i-1] > 0 and filtered_z[i] < 0:
            steps += 0.5

    print(f"The number of steps is approximately: {steps}")

The number of steps is approximately: 128.0
```

**Answer the following questions:**

- 1. Does the acceleration calculated from the speed match the observed acceleration? What differences do you notice? What could be their cause?**

The average speed that was calculated averaged at approximately 0.99m/s, when compared to the graph of the distance travelled over time as a gradient, it would be logical to assume that the calculated average speed matched what was plotted as the gradient seems to be increasing by approximately 0.99 units. Nevertheless, it is not perfect and even if each acceleration point was calculated, there are bound to be differences between the calculated and observed data. This is due to several reasons.

Firstly, the filter cutoff has a high possibility of varying. This can cause the graphs to have a different gradient or peaks, which could cause the different results in acceleration. Next, as the average speed was calculated, all the errors of every data point were taken into account which may have also affected the results of the calculated speed.

- 2. Is the number of steps logical?**

The number of steps can be considered to be a sensible and logical number. This is because when the distance (approximately 220m) and the average speed (approximately 1m/s) is taken into consideration, it would be logical that 128 steps were taken over 220m in a time frame of (assuming that each time interval given was 1 second) of over 200 seconds. As the speed was calculated to be an average, it could be considered that during the period of the walk, there were more steps per second during one taken data point compared to another.

In conclusion, I would consider that the number of steps were undercalculated, however, it would still be logical to consider the number of steps to be 128.