In [ ]:
```
from src.model import Beach
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import statistics
```

In [ ]:
```
n_LWRobots=10
n_palm_trees = 6
n_heavy_plastic = 10
n_cleaning_tanks = 3
width=50
height=50
added_functionality=False
rubbish_collection_robots=False
```

In [ ]:
```
#standard beach cleaning simulation with no added features

#standard simulation low robots
sslr = Beach(n_LWRobots=2, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 1, width=50, height=50, added_functionality=False, ru

#standard simulation default robots
ssdr = Beach(n_LWRobots=5, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 2, width=50, height=50, added_functionality=False, ru

#standard simulation high robots
sshr = Beach(n_LWRobots=10, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 3, width=50, height=50, added_functionality=False, ru

while sslr.running:
    sslr.step()

while ssdr.running:
    ssdr.step()

while sshr.running:
    sshr.step()
```

In [ ]:
```
#added functionality simulation low robots
afslr = Beach(n_LWRobots=2, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 1, width=50, height=50, added_functionality=True, rub

#added function default robots
afsdr = Beach(n_LWRobots=5, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 2, width=50, height=50, added_functionality=True, rub

#added function high robots
afshr = Beach(n_LWRobots=10, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 3, width=50, height=50, added_functionality=True, rub

while afslr.running:
    afslr.step()

while afsdr.running:
    afsdr.step()

while afshr.running:
    afshr.step()
```

In [ ]:
```
#rubbish collection robot simulation low robots
rcrslr = Beach(n_LWRobots=2, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 1, width=50, height=50, added_functionality=False, ru

#rubbish collection robot simulation default robots
rcrsdr = Beach(n_LWRobots=5, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 2, width=50, height=50, added_functionality=False, ru

#rubbish collection robot simulation high robots
rcrshr = Beach(n_LWRobots=10, n_palm_trees = 6, n_heavy_plastic = 15, \
    n_cleaning_tanks = 3, width=50, height=50, added_functionality=False, ru

while rcrslr.running:
    rcrslr.step()

while rcrsdr.running:
    rcrsdr.step()

while rcrshr.running:
    rcrshr.step()
```

In [ ]:
```
#standard simulation low robots
SSLR_beach_df = sslr.datacollector.get_model_vars_dataframe()
SSLR_agents_df = sslr.datacollector.get_agent_vars_dataframe()

#standard simulation default robots
SSDR_beach_df = ssdr.datacollector.get_model_vars_dataframe()
SSDR_agents_df = ssdr.datacollector.get_agent_vars_dataframe()

#standrd simulation high robots
SSHR_beach_df = sshr.datacollector.get_model_vars_dataframe()
SSHR_agents_df = sshr.datacollector.get_agent_vars_dataframe()

#added functionality simulation low robots
AFSLR_beach_df = afslr.datacollector.get_model_vars_dataframe()
AFSLR_agents_df = afslr.datacollector.get_agent_vars_dataframe()

#added functionality simulation default robots
AFSDR_beach_df = afsdr.datacollector.get_model_vars_dataframe()
AFSDR_agents_df = afsdr.datacollector.get_agent_vars_dataframe()

#added functionality simulation high robots
AFSHR_beach_df = afshr.datacollector.get_model_vars_dataframe()
AFSHR_agents_df = afshr.datacollector.get_agent_vars_dataframe()

#rubbish collection robot simulation low robots
RCRSLR_beach_df = rcrslr.datacollector.get_model_vars_dataframe()
RCRSLR_agents_df = rcrslr.datacollector.get_agent_vars_dataframe()

#rubbish collection robot default robots
RCRSDR_beach_df = rcrsdr.datacollector.get_model_vars_dataframe()
RCRSDR_agents_df = rcrsdr.datacollector.get_agent_vars_dataframe()

#rubbish collection robot high robots
RCRSHR_beach_df = rcrshr.datacollector.get_model_vars_dataframe()
RCRSHR_agents_df = rcrshr.datacollector.get_agent_vars_dataframe()
```

Analysis One

The following figures (1-3) show the collection of plastic over time

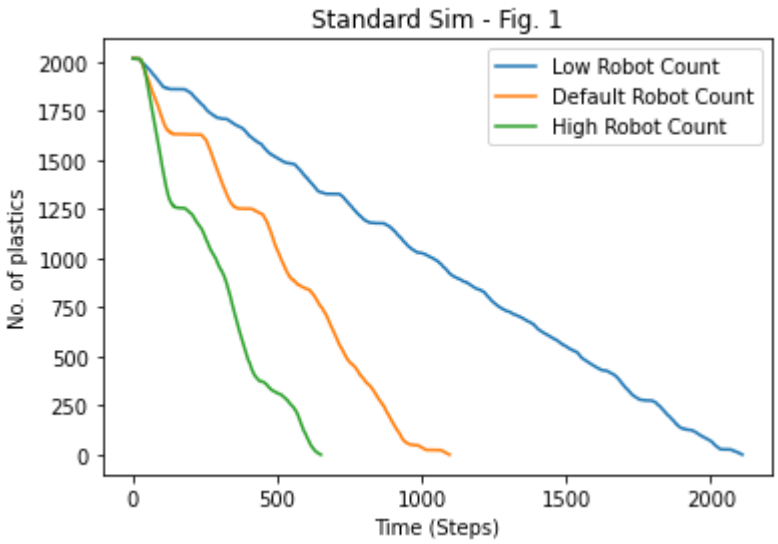
Figure 1 is for the standard simulation

Figure 2 is for the simulation with added functionality

Figure 3 is for the simulation with rubbish collection robots

All figures show metrics from simulations run with the lowest possible robot count, the defualt robot count, and the highest possible robot count.

In [ ]:
```
plt.plot(SSLR_beach_df.overall_plastic, label= "Low Robot Count")
plt.plot(SSDR_beach_df.overall_plastic, label= "Default Robot Count")
plt.plot(SSHR_beach_df.overall_plastic, label= "High Robot Count")
plt.xlabel("Time (Steps)")
plt.ylabel("No. of plastics")
plt.legend()
plt.title("Standard Sim - Fig. 1")
plt.show()
```



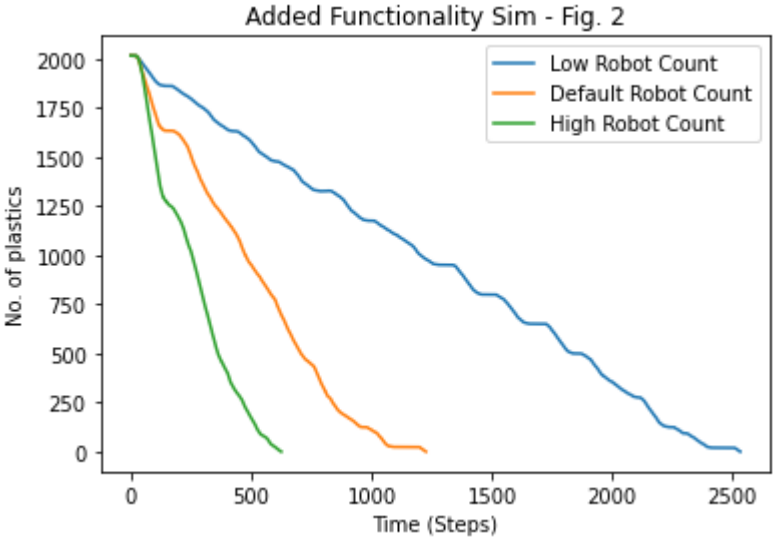
Standard Sim

The results of the first graph is unsurprising and predictable. The more robots the simulation has the faster it cleans.

The shape of data can also be explained. The step like gradient indicates a stop in collection by some or all robots for a brief period. This will be caused by robots needing to empty their hoppers and therefore moving towards the bin unable to collect.

The gradient generally flattens out towards the end, this is caused by some robots finishing collection before others and so the rate of collection slows down.

In [ ]:
```
plt.plot(AFSLR_beach_df.overall_plastic, label = "Low Robot Count")
plt.plot(AFSDR_beach_df.overall_plastic, label = "Default Robot Count")
plt.plot(AFSHR_beach_df.overall_plastic, label = "High Robot Count")
plt.xlabel("Time (Steps)")
plt.ylabel("No. of plastics")
plt.legend()
plt.title("Added Functionality Sim - Fig. 2")
plt.show()
```

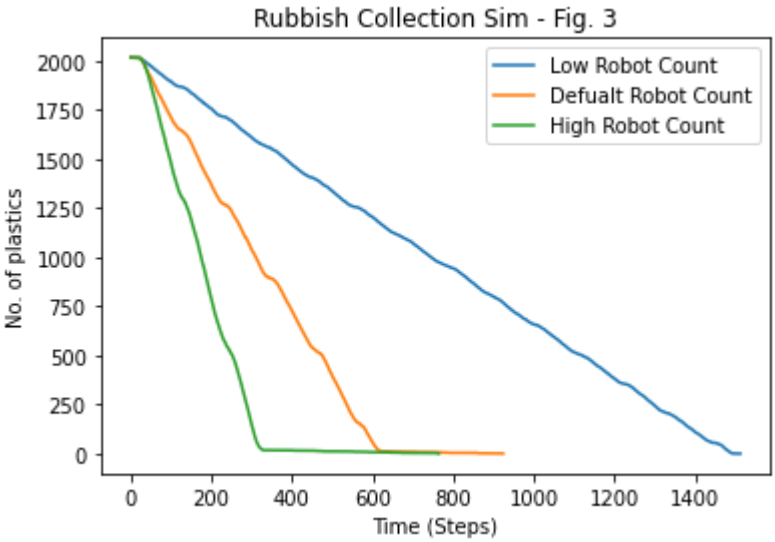


Added Functionality Sim

Figure 2 shows a very similar gradient shape to figure 1. Which is unsurprising as they both have the same robot movement behaviour.

Interestingly it seems that the added functionality simulation finished a little slower than the standard one.

In [ ]:
```
plt.plot(RCRSLR_beach_df.overall_plastic, label = "Low Robot Count")
plt.plot(RCRSDR_beach_df.overall_plastic, label = "Defualt Robot Count")
plt.plot(RCRSHR_beach_df.overall_plastic, label = "High Robot Count")
plt.xlabel("Time (Steps)")
plt.ylabel("No. of plastics")
plt.legend()
plt.title("Rubbish Collection Sim - Fig. 3")
plt.show()
```



Rubbish Collection Robot Sim

The above figure (3) is quite different from the first 2.

Firstly, this simulation completes around a thousand steps quicker than the other two simulations when running a high robot count. As well as, this the default robot count finishes

a couple hundred steps quicker when compare to the other two. This is a drastic improvement.

Secondly, the steps in the gradient of the graph above are greatly reduced. This can be put down to the robots not needing to move to the bin and therefore being able to continuously collect. However, the steps are still present because as the light robots fill their speed decreases.

In the low robot count and defualt robot count we can see the gradient flattens at the end. This is because of the speed improvements that the collection robots give the light robots. Therefore, they can finish collecting long before the cleaning tanks have discovered the remaining heavy plastics.

Analysis Two

Rate of collision between robots compaired to the number of robots on the beach

For this comparison I will be comparing the standard simulation low and high robot counts. The low having just 3 and the high having 13. With the rubbish collection robot simulation low and high robot counts. The low having 6 and the high having 16.

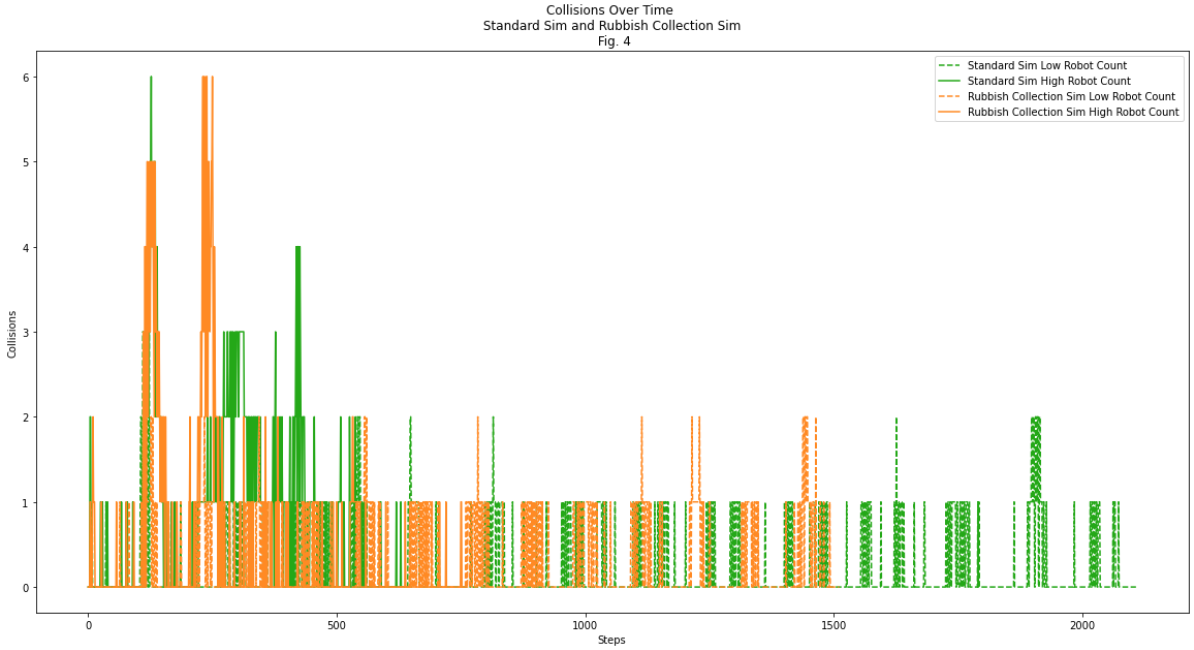
I have choosen not to compare the added functionality simulation as it has the same amount of robots and the same robot movement behaivour as the standard simulation.

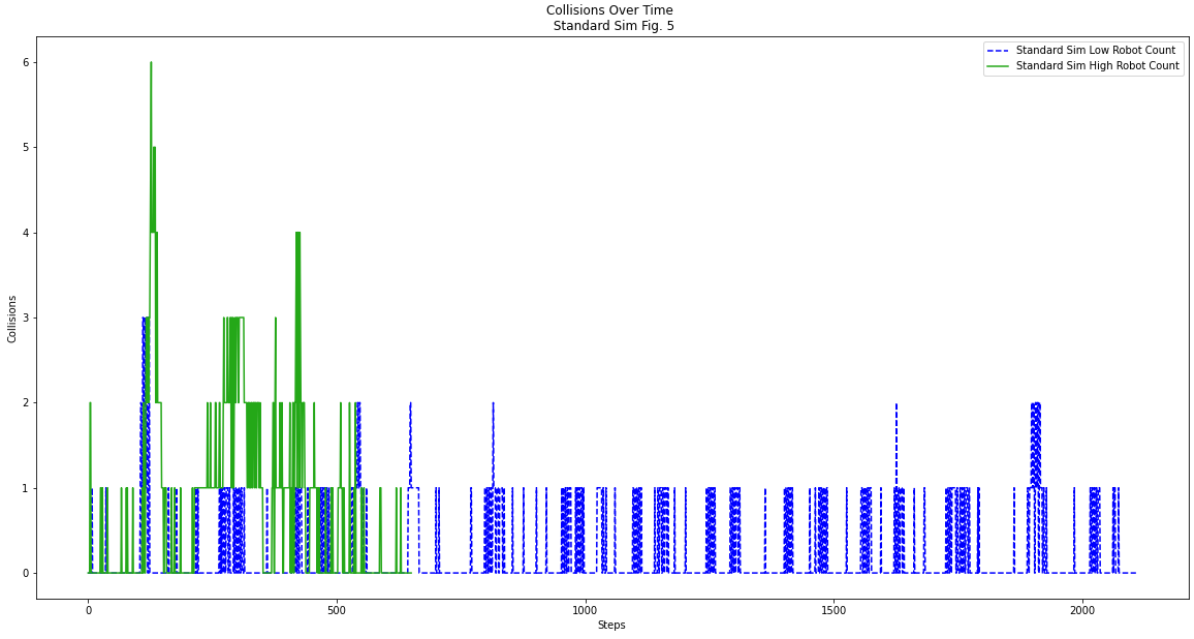
An important note to mention is that this metric doesn't just measure robot on robot collisions, it measures any time a robot collides with anything on the beach.

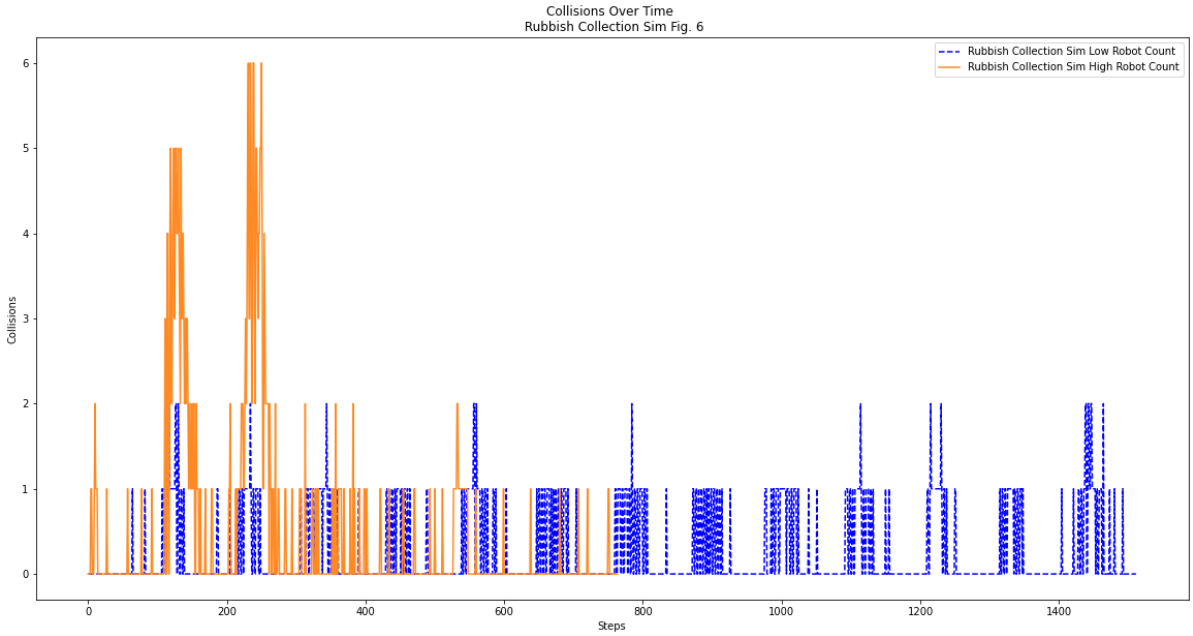
In [ ]:
```
#get a dataframe of states per step

SSLR_states_per_step = SSLR_agents_df.groupby('Step')['state'].value_counts
SSHR_states_per_step = SSHR_agents_df.groupby('Step')['state'].value_counts

RCRSLR_states_per_step = RCRSLR_agents_df.groupby('Step')['state'].value_cou
RCRSHR_states_per_step = RCRSHR_agents_df.groupby('Step')['state'].value_cou
```

In [ ]:
```
plt.figure(figsize=(20,10))
plt.plot(np.arange(len(SSLR_states_per_step)), SSLR_states_per_step["AVOIDAN
plt.plot(np.arange(len(SSHR_states_per_step)), SSHR_states_per_step["AVOIDAN
plt.plot(np.arange(len(RCRSLR_states_per_step)), RCRSLR_states_per_step["AV
plt.plot(np.arange(len(RCRSHR_states_per_step)), RCRSHR_states_per_step["AV
plt.xlabel("Steps")
plt.ylabel("Collisions")
plt.title("Collisions Over Time \n Standard Sim and Rubbish Collection Sim \
plt.legend()
plt.show()

plt.figure(figsize=(20,10))
plt.plot(np.arange(len(SSLR_states_per_step)), SSLR_states_per_step["AVOIDAN
plt.plot(np.arange(len(SSHR_states_per_step)), SSHR_states_per_step["AVOIDAN
plt.xlabel("Steps")
plt.ylabel("Collisions")
plt.title("Collisions Over Time \n Standard Sim Fig. 5")
plt.legend()
plt.show()
```

```
plt.figure(figsize=(20,10))
plt.plot(np.arange(len(RCRSLR_states_per_step)), RCRSLR_states_per_step["AVG
plt.plot(np.arange(len(RCRSHR_states_per_step)), RCRSHR_states_per_step["AVG
plt.xlabel("Steps")
plt.ylabel("Collisions")
plt.title("Collisions Over Time \n Rubbish Collection Sim Fig. 6")
plt.legend()
plt.show()
```







The above graphs (4,5,6) display the number of collisions per step between the four different simulations labelled in the legend.

The peaks in the graph for the high robot count simulations are higher and more frequent. This is to be expected as there are more robots on the beach and therefore more agents to collide with.

The rubbish collection sim with high robot count has 2 large sections of collision at the start and then nothing that matches it after. I believe this could be reflecting when light tanks all empty their hopper simultaneously. When this happens the collection robots stay on the beach for long period, therefor causing more collisions. Although you would expect to see at least one other peak before the high robot sim finishes. We don't see this peak because the light robots finish searching a few steps after they empty for the third time. Meaning, there are fewer moving objects for robots to collide with.

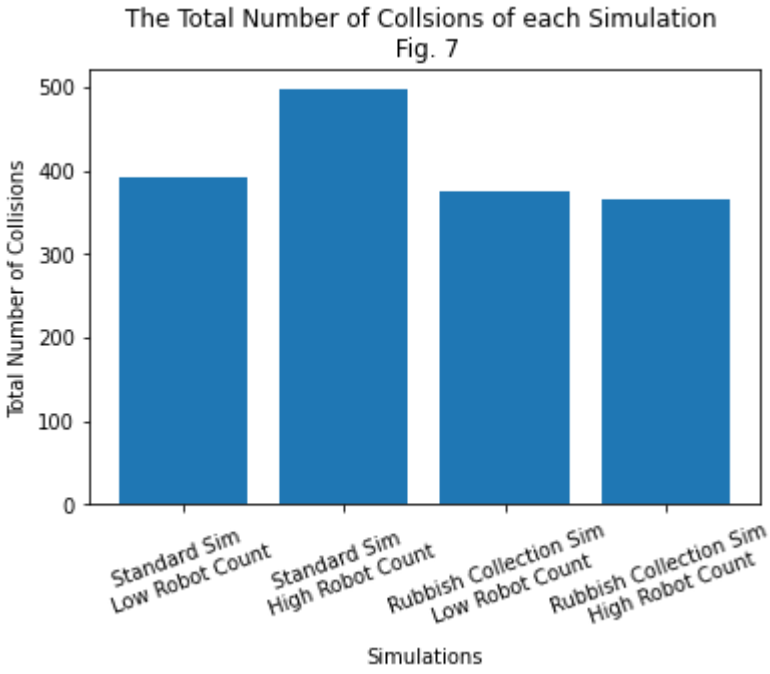
The Standard sim shows a behaviour that is similar. However, there are 3 peaks rather than 2. The high robot count peaks once at the start, but the subsequent peaks are lower and more drawn out. I believe peak one is caused by the initial simultaneous movement of light robots to the bin. I believe the more drawn-out 2nd and 3rd peaks are caused because each robot must travel a different distance and then wait it's turn to empty. Because of this, robots return to cleaning at different times. Therefor the 2nd and 3rd round of emptying's are not simultaneous and are more drawn out.

The low robot standard sim has one peak at its start which I believe to be the two light tanks emptying at the same time. After the first empty, the cleaning tanks no longer fill at a similar rate, because of this they no longer collide with one another at the bin. I believe this why we see a peak at the start and then not again.

There are no notable peaks with the low robot rubbish collection sim even though there are more robots than the standard simulation. I believe this is because light robots keep to their own sections and never interact with each other as they aren't required to use the bin.

In [ ]:
```
y = [sum(SSLR_states_per_step["AVOIDANCE"].tolist()), sum(SSHR_states_per_st
    sum(RCRSHR_states_per_step["AVOIDANCE"].tolist())]
```

```
x = ["Standard Sim \n Low Robot Count", "Standard Sim \n High Robot Count",
    "Rubbish Collection Sim \n Low Robot Count", "Rubbish Collection Sim \n High Robot Count"]

plt.bar(x, y)
plt.xlabel("Simulations")
plt.xticks(rotation=20)
plt.ylabel("Total Number of Collisions")
plt.title("The Total Number of Collsions of each Simulation \n Fig. 7")
plt.show()
```



The results of the above bar graph are interesting. It would seem that adding rubbish collection robots decreases the number of collisions, quite dramatically in the case of high robot count vs high robot count. This is interesting as the rubbish collection simulation has a higher number of total robots. However, what I didn't expect to see is that with the two simulations that are running rubbish collection robots the higher robot count simulation has less collisions than the low robot count. Although it isn't by much. Nevertheless, I would have expected to see the simulation with high robots have higher collisions. As the differences are quite small, we could put this down to the stochastic nature of the collisions in the Beach cleaning simulation. It would seem that the most collisions occur when robots empty their hopper, and so without this the collisions are capped.

The highest number of collisions is suffered by the standard simulation with a high robot count, which is what I would expect to see.

Analysis 3

In my final analysis I am going to look into whether auctioning jobs to cleaning tanks increases or descreases their effecientcy.

I will compare results from the standard simulation with the default number of robots aswell as the added functionailty simulation with the default amount of robots.

In [ ]:
```
#figure out the average speed for the added functionality simulation
```

```
AFSDR_list_of_speeds = []

#figure when they complete collecting
stop = 0
for x in AFSDR_beach_df["heavy_plastic"].tolist():
    stop += 1
    if x == 0:
        break

for step in range(1, stop):
    AFSDR_list_of_speeds.append(abs(AFSDR_beach_df["heavy_plastic"].iloc[ste

mean = 0
for number in AFSDR_list_of_speeds:
    mean += number

AFSDR_speed = mean/len(AFSDR_list_of_speeds)
print(AFSDR_list_of_speeds)
print(AFSDR_speed)
```

```
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1]
0.02631578947368421
```

In [ ]:

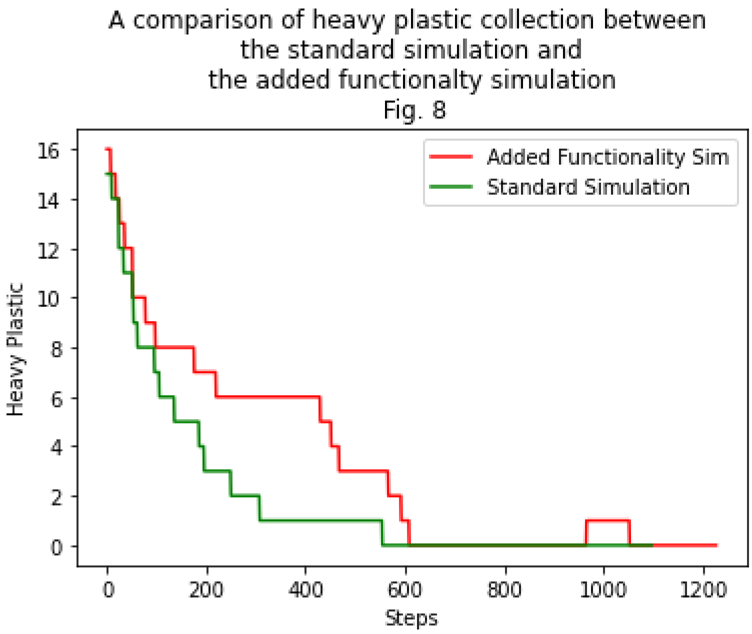
```
#figure out the average speed for the standard simulation

SSDR_list_of_speeds = []

#figure when they complete collecting
stop = 0
for x in SSDR_beach_df["heavy_plastic"].tolist():
    stop += 1
    if x == 0:
        break

for step in range(1, stop):
    SSDR_list_of_speeds.append(abs(SSDR_beach_df["heavy_plastic"].iloc[step]

mean = 0
for number in SSDR_list_of_speeds:
    mean += number
```

```
SSDR_speed = mean/len(SSDR_list_of_speeds)
print(SSDR_list_of_speeds)
print(SSDR_speed)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1]
0.02707581227436823
```

In [ ]:

```
plt.plot(AFSDR_beach_df["heavy_plastic"], color = "Red", label = "Added Func
plt.plot(SSDR_beach_df["heavy_plastic"], color = "Green", label = "Standard
plt.legend()
plt.title("A comparison of heavy plastic collection between \n the standard
plt.xlabel("Steps")
plt.ylabel("Heavy Plastic")
plt.show()

plt.figure(figsize=(6,7))
plt.bar(["Added Functionality Sim", "Standard Simulation"], [AFSDR_speed, SS
plt.title("A comparison of heavy plastic collection speeds \n Fig. 9")
plt.show()
```



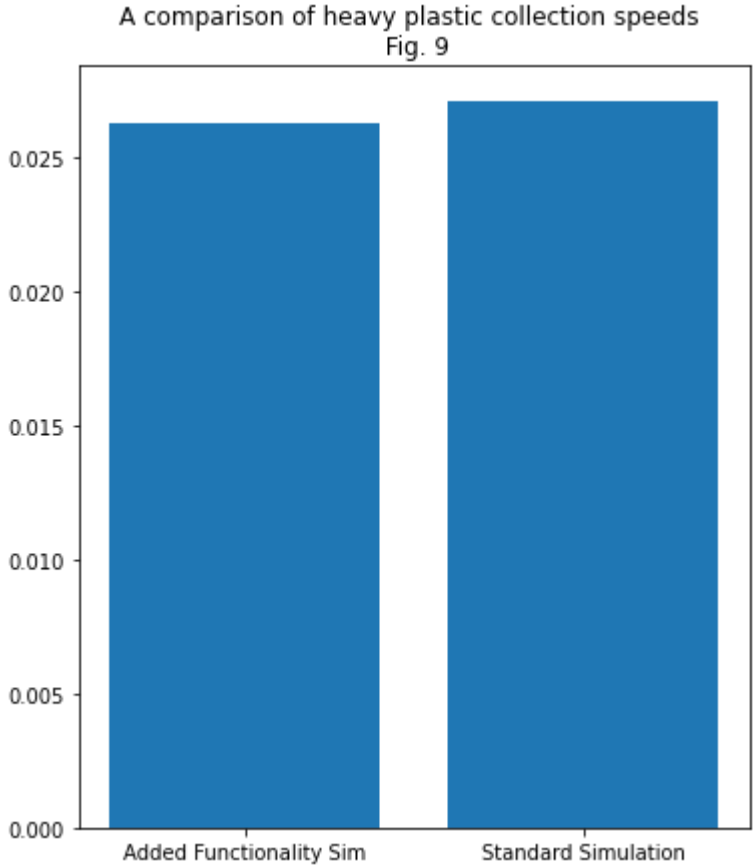


Figure 8 displays the number of heavy plastic pieces collected over time.

Figure 9 displays the average speed of collection. Measured in plastic pieces collected per step. I measured every step until the cleaning tanks finish collecting, which happens at different times for each simulation, but around step 600. Ignoring the plastic piece that spawned in around step 1000 for the added functionality sim.

These figures show that the auctioning process slows down the speed at which cleaning tanks collect. I believe this is because of the cleaning tank prioritising jobs coming from the auction rather than jobs that are close to them.

An improvement that could be made is to have the clean tanks assess their soundings and pick jobs close to them before entering the auction.