Learning and Memory Assignment-2

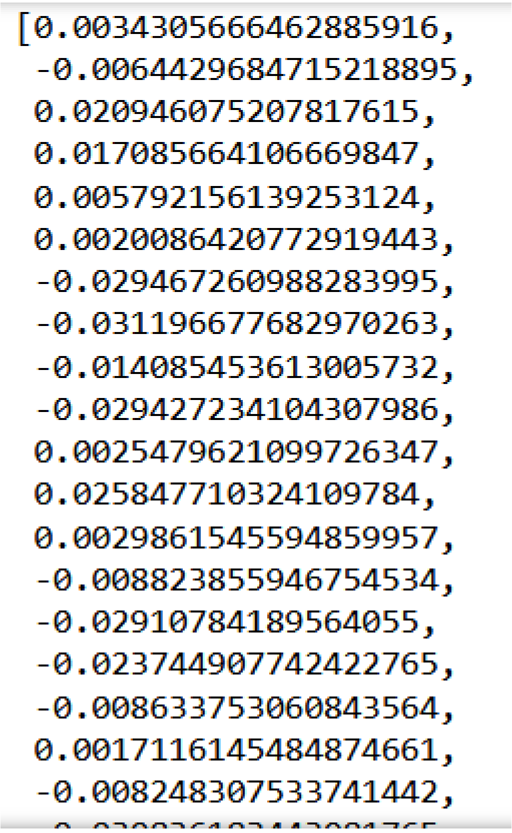
a.

Average data per day 1 is 0.129938212695416 sec
Average data per day 2 is 0.1518654402902913 sec

[redacted]

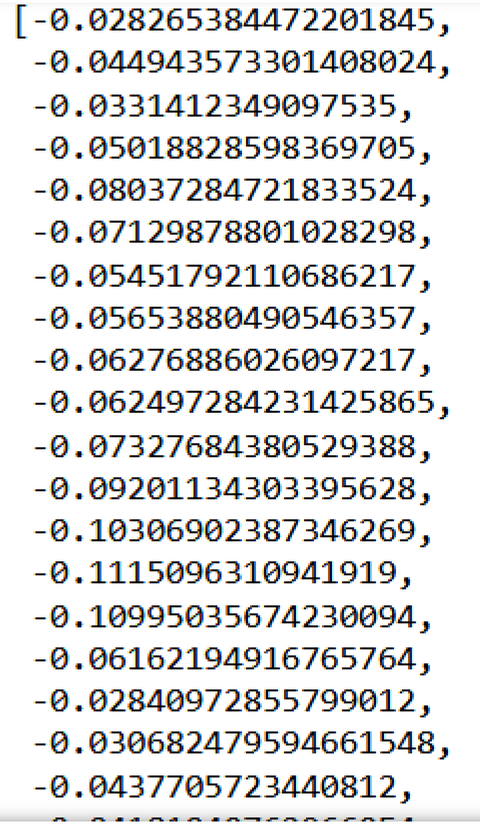
b.

For Day1



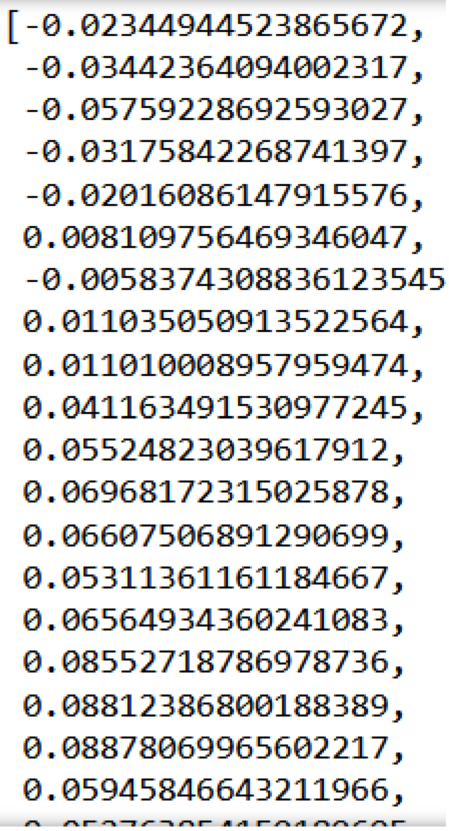
```
[0.0034305666462885916,
 -0.0064429684715218895,
 0.020946075207817615,
 0.017085664106669847,
 0.005792156139253124,
 0.0020086420772919443,
 -0.029467260988283995,
 -0.031196677682970263,
 -0.014085453613005732,
 -0.029427234104307986,
 0.0025479621099726347,
 0.025847710324109784,
 0.0029861545594859957,
 -0.008823855946754534,
 -0.02910784189564055,
 -0.023744907742422765,
 -0.008633753060843564,
 0.0017116145484874661,
 -0.008248307533741442,
```

For Day 2



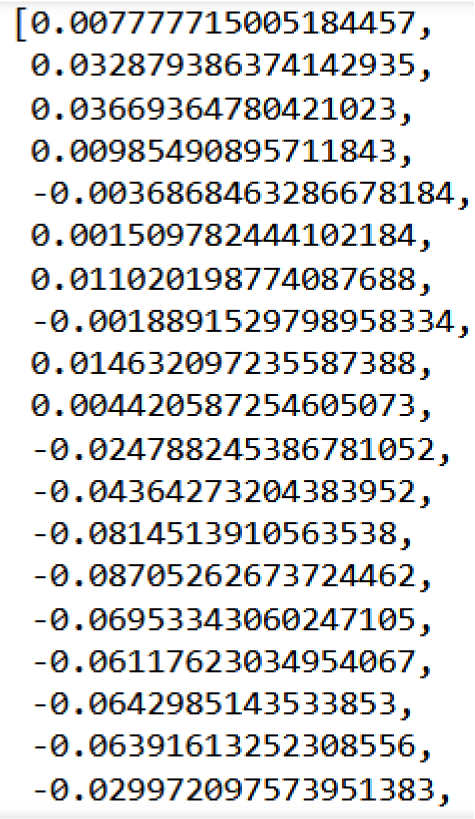
```
[-0.028265384472201845,
 -0.044943573301408024,
 -0.0331412349097535,
 -0.05018828598369705,
 -0.08037284721833524,
 -0.07129878801028298,
 -0.05451792110686217,
 -0.05653880490546357,
 -0.06276886026097217,
 -0.062497284231425865,
 -0.07327684380529388,
 -0.09201134303395628,
 -0.10306902387346269,
 -0.1115096310941919,
 -0.10995035674230094,
 -0.06162194916765764,
 -0.02840972855799012,
 -0.030682479594661548,
 -0.0437705723440812,
```

Day 3



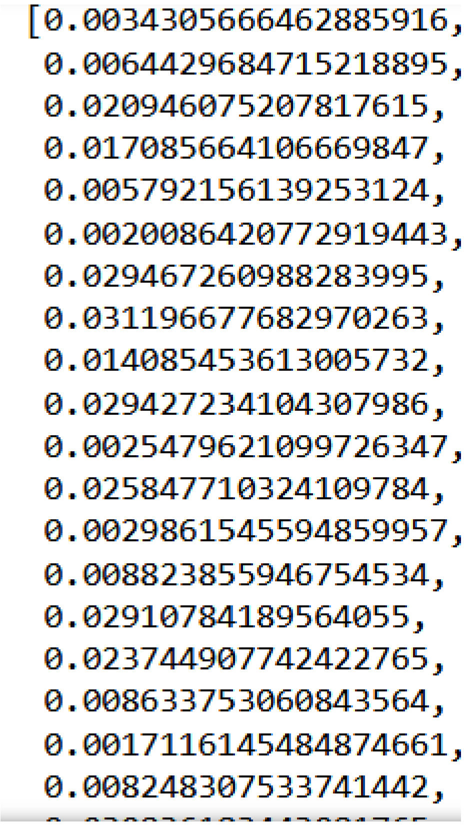
```
[-0.02344944523865672,
 -0.03442364094002317,
 -0.05759228692593027,
 -0.03175842268741397,
 -0.02016086147915576,
 0.008109756469346047,
 -0.0058374308836123545
 0.011035050913522564,
 0.011010008957959474,
 0.04116349153097245,
 0.0552482303961791 2,
 0.0696817231502587 8,
 0.06607506891290699,
 0.0531136116118466 7,
 0.06564934360241083,
 0.08552718786978736,
 0.08812386800188389,
 0.08878069965602217,
 0.05945846643211966,
```

Day 4



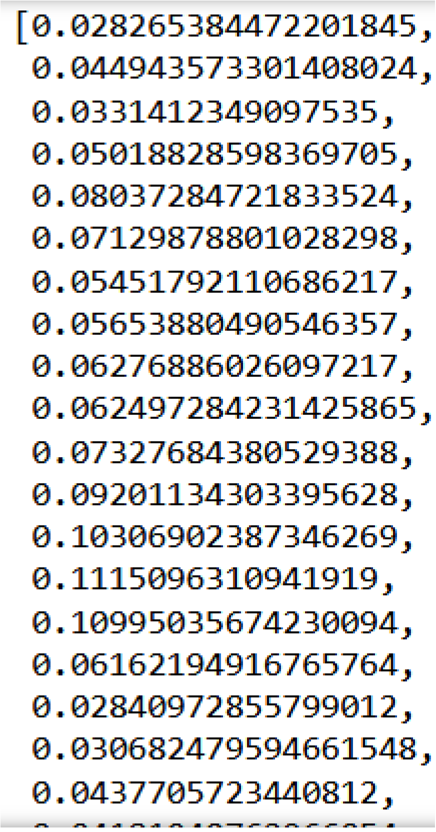
```
[0.007777715005184457,
 0.032879386374142935,
 0.03669364780421023,
 0.00985490895711843,
 -0.0036868463286678184,
 0.001509782444102184,
 0.011020198774087688,
 -0.0018891529798958334,
 0.014632097235587388,
 0.004420587254605073,
 -0.024788245386781052,
 -0.04364273204383952,
 -0.0814513910563538,
 -0.08705262673724462,
 -0.06953343060247105,
 -0.06117623034954067,
 -0.0642985143533853,
 -0.06391613252308556,
 -0.029972097573951383,
```

c. For day 1



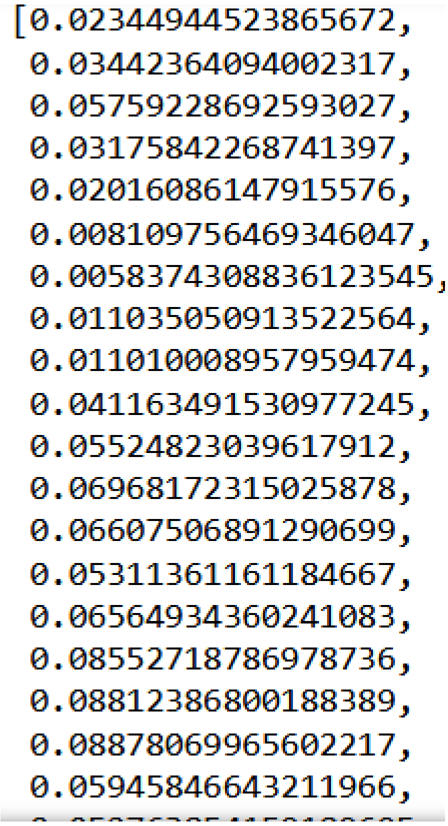
```
[0.003430566646288591б,
 0.0064429684715218895,
```

Day 2



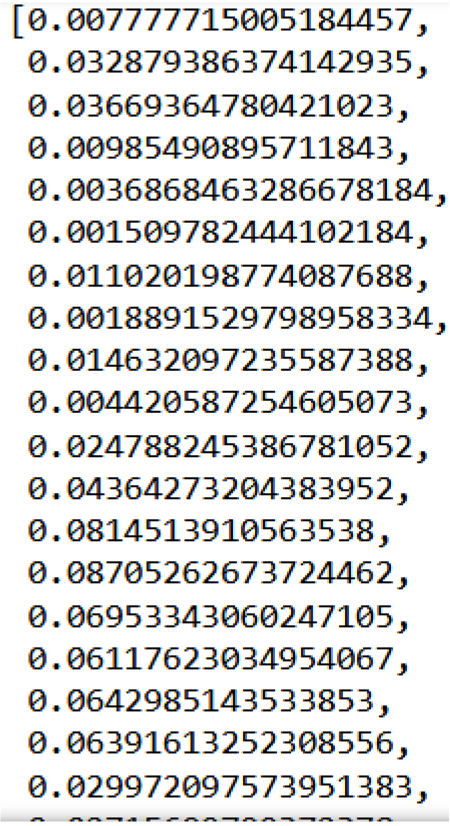
```
[0.028265384472201845,
 0.044943573301408024,
 0.0331412349097535,
 0.05018828598369705,
```

Day 3



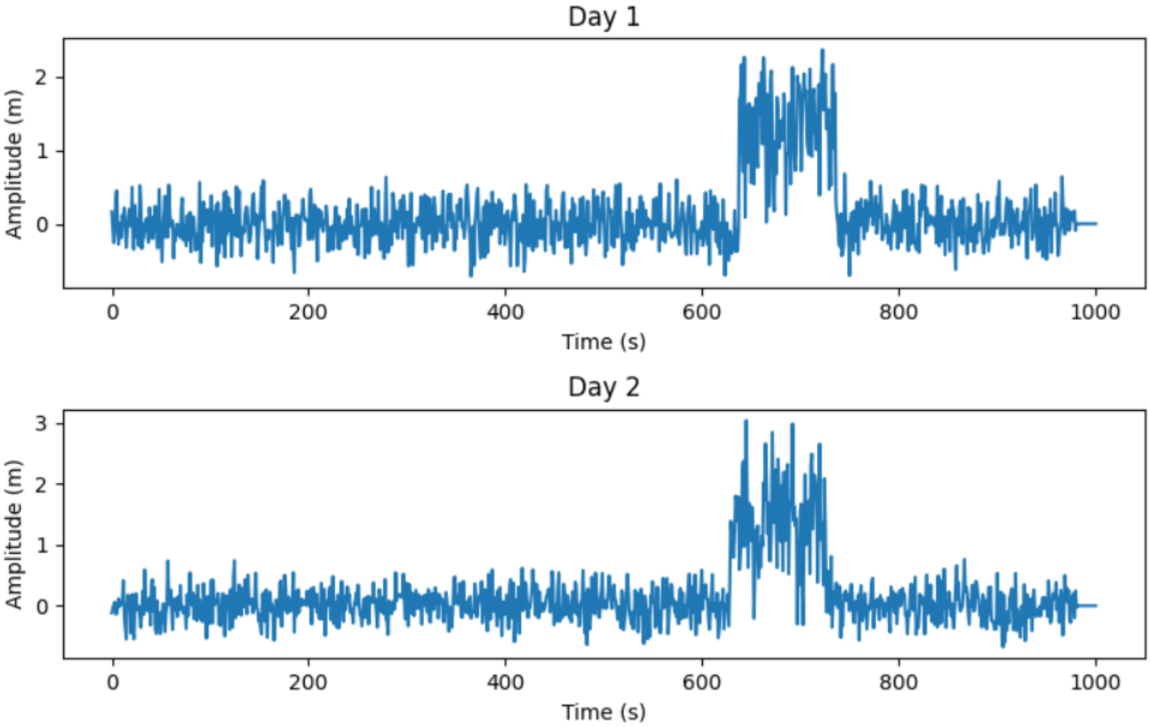
```
[0.02344944523865672,
 0.03442364094002317,
 0.05759228692593027,
 0.03175842268741397,
 0.02016086147915576,
 0.008109756469346047,
 0.0058374308836123545,
 0.011035050913522564,
 0.011010008957959474,
 0.041163491530977245,
 0.05524823039617912,
 0.06968172315025878,
 0.06607506891290699,
 0.05311361161184667,
 0.06564934360241083,
 0.08552718786978736,
 0.08812386800188389,
 0.08878069965602217,
 0.05945846643211966,
```

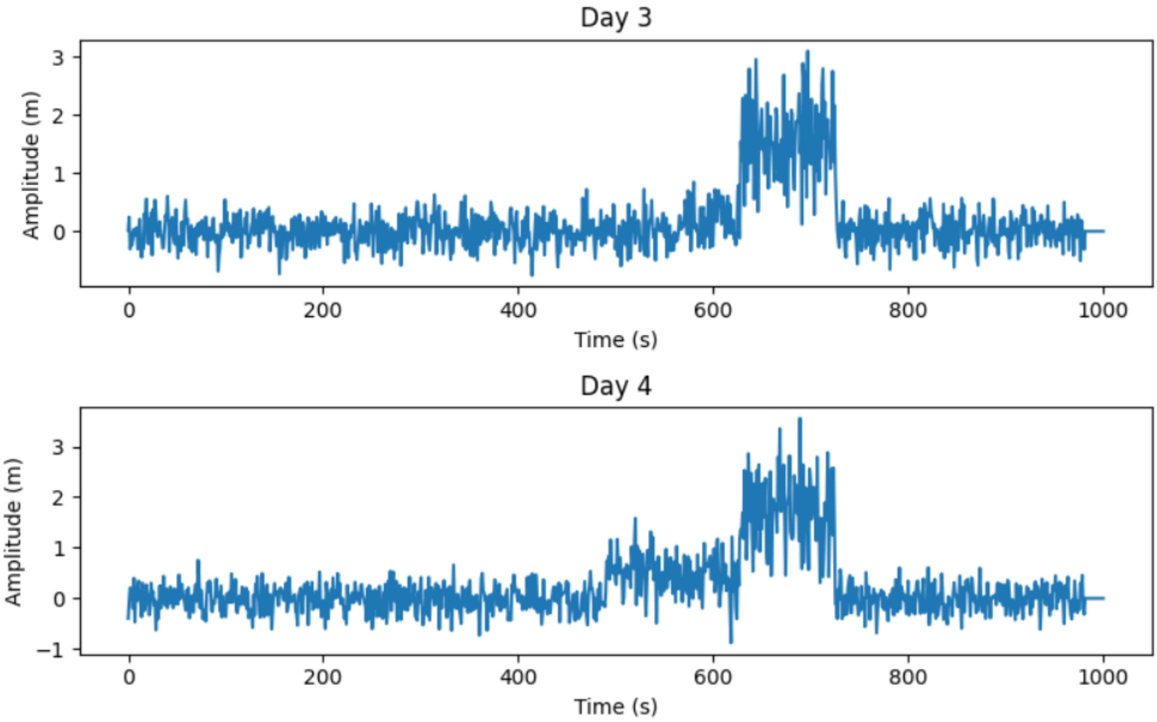
Day 4



```
[0.007777715005184457,
 0.032879386374142935,
 0.03669364780421023,
 0.00985490895711843,
 0.0036868463286678184,
 0.001509782444102184,
 0.011020198774087688,
 0.0018891529798958334,
 0.014632097235587388,
 0.004420587254605073,
 0.024788245386781052,
 0.04364273204383952,
 0.0814513910563538,
 0.08705262673724462,
 0.06953343060247105,
 0.06117623034954067,
 0.0642985143533853,
 0.06391613252308556,
 0.029972097573951383,
```

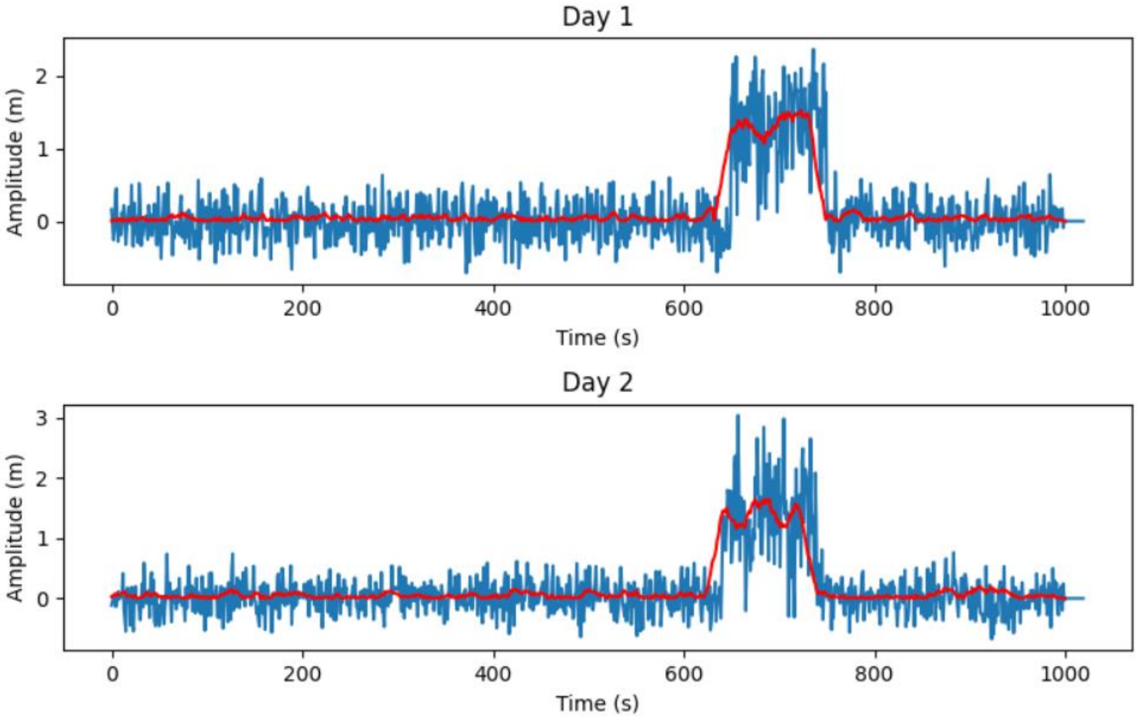
d.



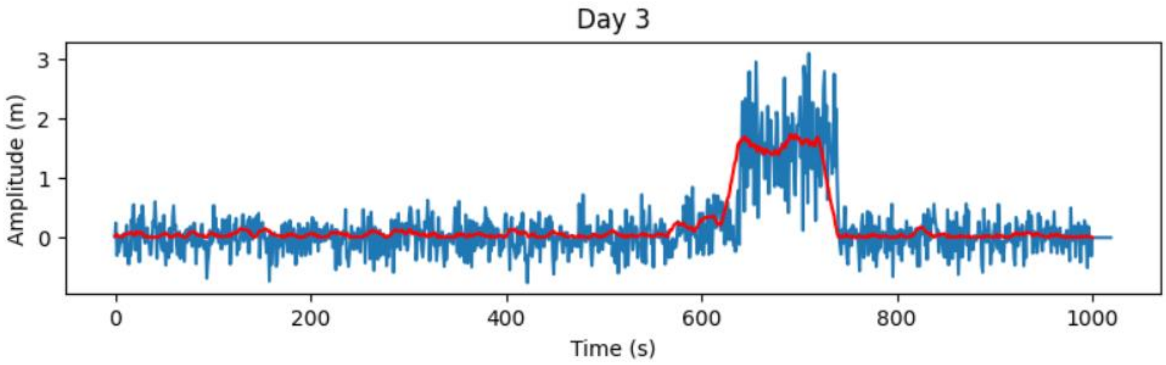

Day 3

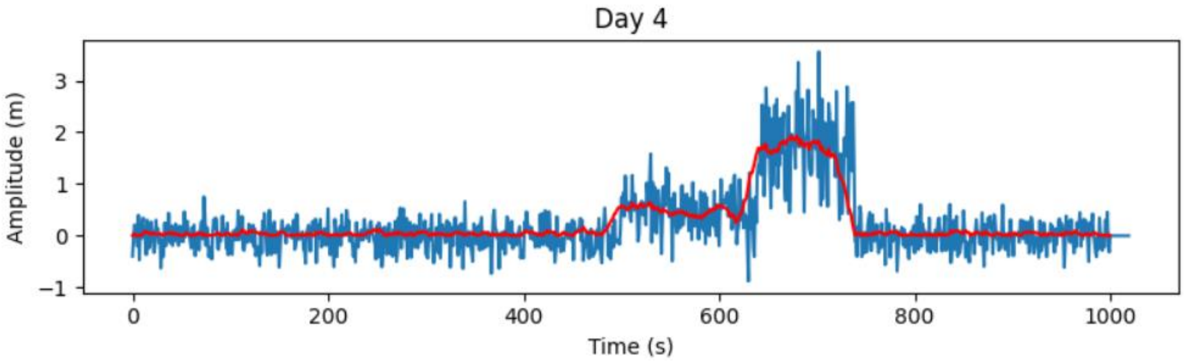
Day 4

e.


Day 1

Day 2





Code:

The learning mechanism involved is Conditioned Stimulus which makes a response with an UnConditioned Stimulus.

The necessary another thing will be involved will be:

1. Fast Recovery
2. Elimination
3. Make that these stimuli don't correspond to other predicted stimuli.

Peaks:

Day 1:

736

Day 2

657

Day 3

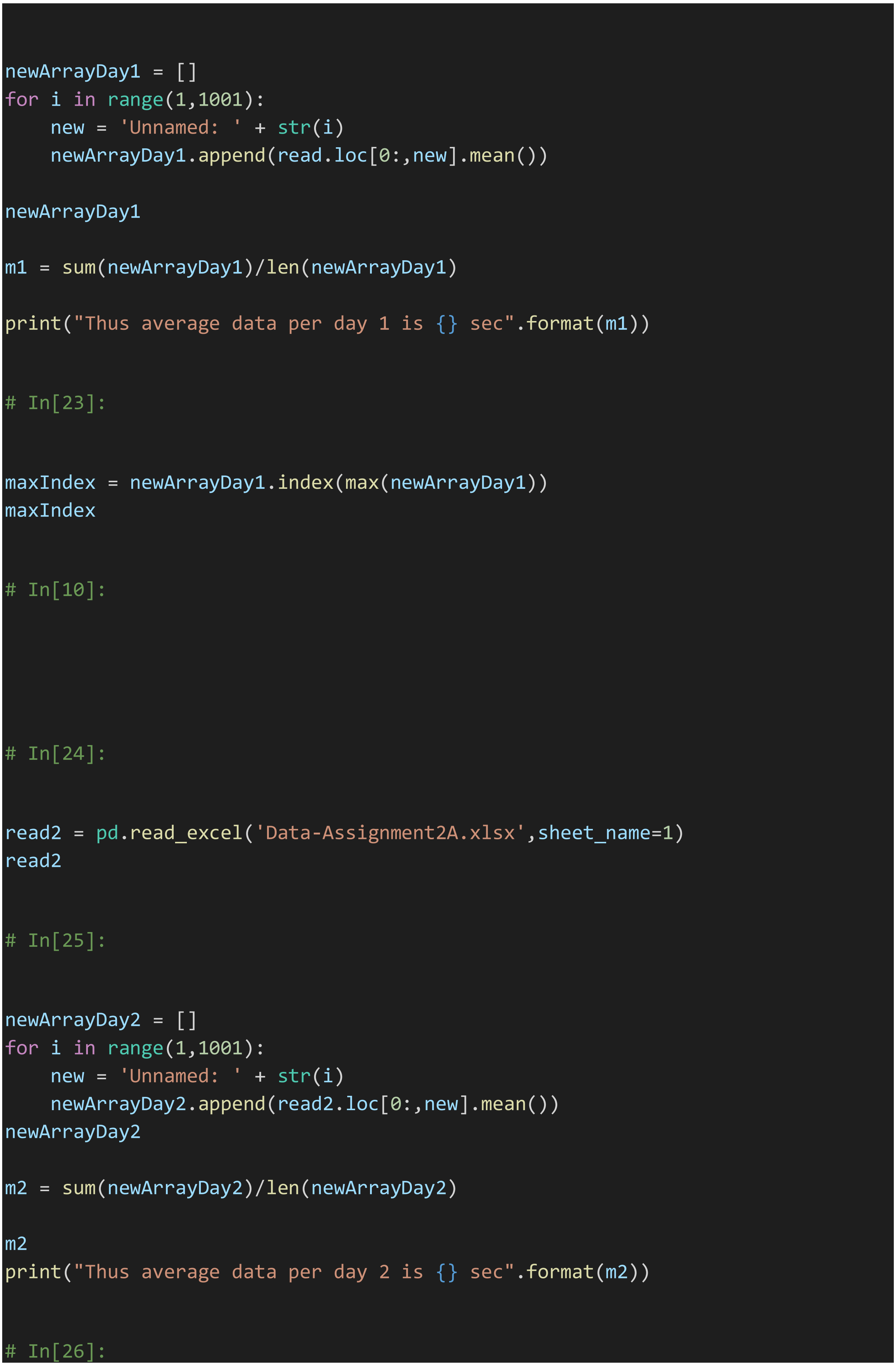
710

Day 4

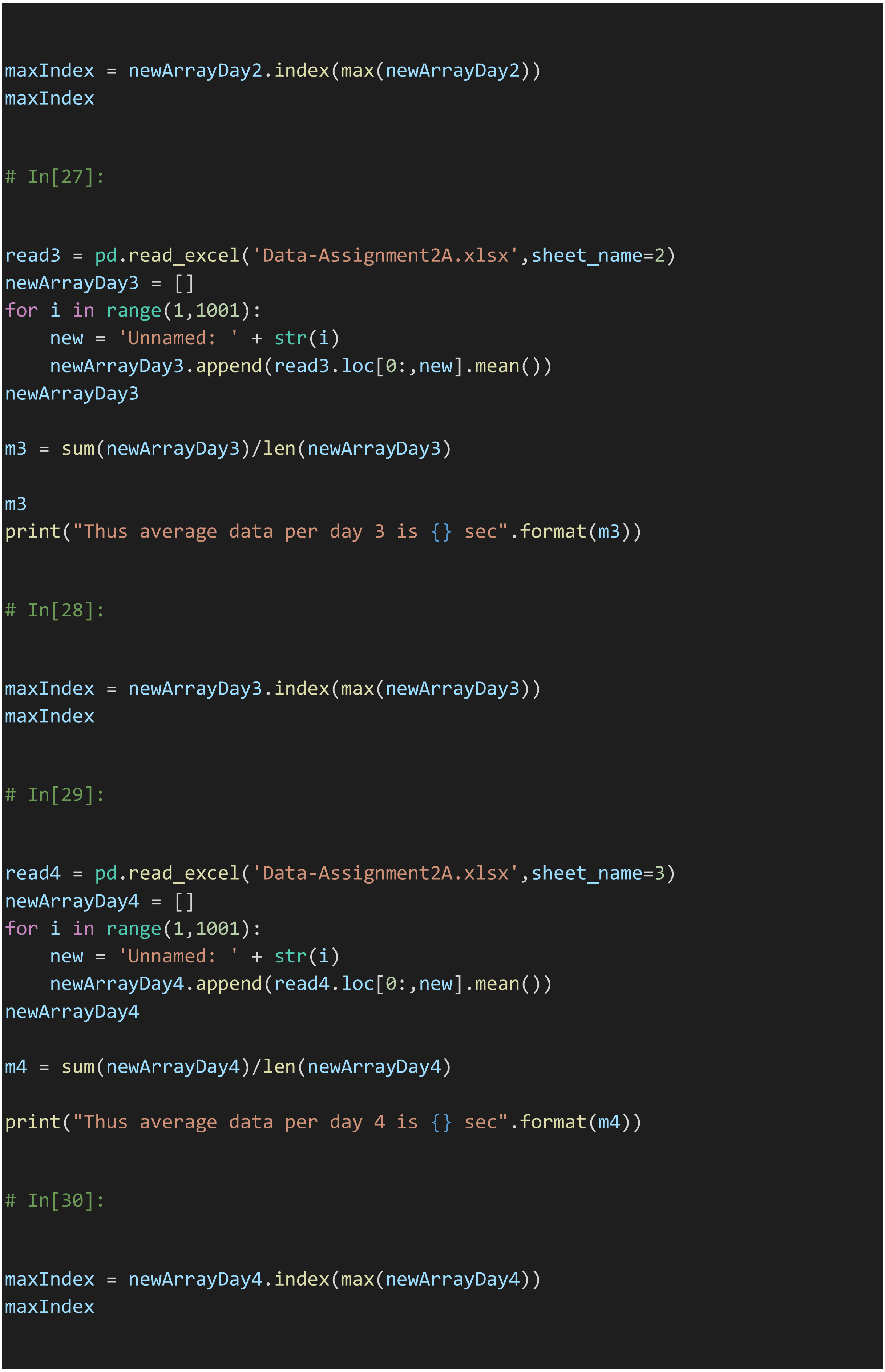
702

Code:

```
#!/usr/bin/env python
# coding: utf-8

# In[14]:


import pandas as pd
import numpy as np
from scipy.signal import find_peaks


# In[15]:


read = pd.read_excel('Data-Assignment2A.xlsx')


# In[16]:


read


# In[17]:


col1 = read.loc[0:,"Unnamed: 1"]
col1


# In[18]:


s = 'Unnamed: '
w = str(1)
s + w
new= s+ w
read.loc[0:,new]


# In[19]:


newArrayDay1.clear()


# In[20]:
```



```
newArrayDay1 = []
for i in range(1,1001):
    new = 'Unnamed: ' + str(i)
    newArrayDay1.append(read.loc[0:,new].mean())

newArrayDay1

m1 = sum(newArrayDay1)/len(newArrayDay1)

print("Thus average data per day 1 is {} sec".format(m1))


# In[23]:


maxIndex = newArrayDay1.index(max(newArrayDay1))
maxIndex


# In[10]:





# In[24]:


read2 = pd.read_excel('Data-Assignment2A.xlsx',sheet_name=1)
read2


# In[25]:


newArrayDay2 = []
for i in range(1,1001):
    new = 'Unnamed: ' + str(i)
    newArrayDay2.append(read2.loc[0:,new].mean())
newArrayDay2

m2 = sum(newArrayDay2)/len(newArrayDay2)

m2
print("Thus average data per day 2 is {} sec".format(m2))


# In[26]:
```



```
maxIndex = newArrayDay2.index(max(newArrayDay2))
maxIndex


# In[27]:


read3 = pd.read_excel('Data-Assignment2A.xlsx',sheet_name=2)
newArrayDay3 = []
for i in range(1,1001):
    new = 'Unnamed: ' + str(i)
    newArrayDay3.append(read3.loc[0:,new].mean())
newArrayDay3

m3 = sum(newArrayDay3)/len(newArrayDay3)

m3
print("Thus average data per day 3 is {} sec".format(m3))


# In[28]:


maxIndex = newArrayDay3.index(max(newArrayDay3))
maxIndex


# In[29]:


read4 = pd.read_excel('Data-Assignment2A.xlsx',sheet_name=3)
newArrayDay4 = []
for i in range(1,1001):
    new = 'Unnamed: ' + str(i)
    newArrayDay4.append(read4.loc[0:,new].mean())
newArrayDay4

m4 = sum(newArrayDay4)/len(newArrayDay4)

print("Thus average data per day 4 is {} sec".format(m4))


# In[30]:


maxIndex = newArrayDay4.index(max(newArrayDay4))
maxIndex
```

```python
# In[13]:



# In[14]:


# Example usage
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
window_size = 3

moving_average_data = moving_average(newArrayDay1, 20)
print(len(moving_average_data))


# In[15]:


len(newArrayDay1)


# Moving average filter&#39; across the averaged signal with a window width of 20 ms to get a
# filtered signal. Ensure that the raw and filtered signal are of the same length.

# In[16]:


for i in range(20):
    newArrayDay1.append(0)


# In[17]:


AverageFilterDay1 = []
last = 20
for i in range(0,1001):
    m1 = sum(newArrayDay1[i:last])/len(newArrayDay1[i:last])
    AverageFilterDay1.append(m1)
    last+=1
AverageFilterDay1


# In[ ]:
```

```
# In[19]:


for i in range(20):
    newArrayDay2.append(0)
AverageFilterDay2 = []
last = 20
for i in range(1001):
    m1 = sum(newArrayDay2[i:last])/len(newArrayDay2[i:last])
    AverageFilterDay2.append(m1)
    last+=1
AverageFilterDay2


# In[20]:


for i in range(20):
    newArrayDay3.append(0)
AverageFilterDay3 = []
last = 20
for i in range(1001):
    m1 = sum(newArrayDay3[i:last])/len(newArrayDay3[i:last])
    AverageFilterDay3.append(m1)
    last+=1
AverageFilterDay3


# In[21]:


for i in range(20):
    newArrayDay4.append(0)
AverageFilterDay4 = []
last = 20
for i in range(1001):
    m1 = sum(newArrayDay4[i:last])/len(newArrayDay4[i:last])
    AverageFilterDay4.append(m1)
    last+=1
AverageFilterDay4


# In[21]:


AverageFilterDay1


# Do a full wave rectification of the above moving average filtered signal.
# In[22]:


rectifiedDay1.clear()


# In[22]:


rectifiedDay1=[]
for i in range(len(AverageFilterDay1)):
    rectifiedDay1.append(np.abs(AverageFilterDay1[i]))
rectifiedDay1


# In[24]:


rectifiedDay2.clear()


# In[23]:


rectifiedDay2=[]
for i in range(len(AverageFilterDay2)):
    rectifiedDay2.append(np.abs(AverageFilterDay2[i]))
rectifiedDay2


# In[24]:


rectifiedDay3=[]
for i in range(len(AverageFilterDay3)):
    rectifiedDay3.append(np.abs(AverageFilterDay3[i]))
rectifiedDay3


# In[27]:


rectifiedDay4.clear()


# In[25]:


rectifiedDay4=[]
for i in range(len(AverageFilterDay4)):
    rectifiedDay4.append(np.abs(AverageFilterDay4[i]))
```

```
rectifiedDay4


# In[29]:


amplitude average ka hai



# In[26]:


import matplotlib.pyplot as plt


# In[32]:


time1 = np.linspace(0,1000,1020)

fig, axs = plt.subplots(nrows=4, ncols=1, figsize=(8, 10))

axs[0].plot(time1,newArrayDay1)
time2 = np.linspace(0,1000,1020)
axs[1].plot(time2,newArrayDay2)
time3 = np.linspace(0,1000,1020)
axs[2].plot(time3,newArrayDay3)
axs[3].plot(time3,newArrayDay4)

axs[0].set_xlabel("Time (s)")
axs[1].set_xlabel("Time (s)")
axs[2].set_xlabel("Time (s)")
axs[3].set_xlabel("Time (s)")

axs[0].set_ylabel("Amplitude (m)")
axs[1].set_ylabel("Amplitude (m)")
axs[2].set_ylabel("Amplitude (m)")
axs[3].set_ylabel("Amplitude (m)")

axs[0].set_title("Day 1")
axs[1].set_title("Day 2")
axs[2].set_title("Day 3")
axs[3].set_title("Day 4")

fig.tight_layout()
# Display the plot
plt.show()
```

```
# Plot the amplitude vs time of the filtered and rectified signal (as red
curve) - one signal for each
# day on top of the raw signal in the same subplots.

# In[ ]:


newArrayDay1.append(0)


# In[31]:


#filered Signal
rectifiedDay1

time1 = np.linspace(0,1000,1001)

fig, axs = plt.subplots(nrows=4, ncols=1, figsize=(8, 10))

axs[0].plot(newArrayDay1)
axs[0].plot(rectifiedDay1,color='red')

axs[1].plot(newArrayDay2)
axs[1].plot(rectifiedDay2,color='red')

axs[2].plot(newArrayDay3)
axs[2].plot(rectifiedDay3,color='red')

axs[3].plot(newArrayDay4)
axs[3].plot(rectifiedDay4,color='red')

axs[0].set_xlabel("Time (s)")
axs[1].set_xlabel("Time (s)")
axs[2].set_xlabel("Time (s)")
axs[3].set_xlabel("Time (s)")

axs[0].set_ylabel("Amplitude (m)")
axs[1].set_ylabel("Amplitude (m)")
axs[2].set_ylabel("Amplitude (m)")
axs[3].set_ylabel("Amplitude (m)")

axs[0].set_title("Day 1")
axs[1].set_title("Day 2")
axs[2].set_title("Day 3")
axs[3].set_title("Day 4")
```

```
fig.tight_layout()
# Display the plot
plt.show()


# In[ ]:




# In[ ]:




# In[ ]:



```

2.

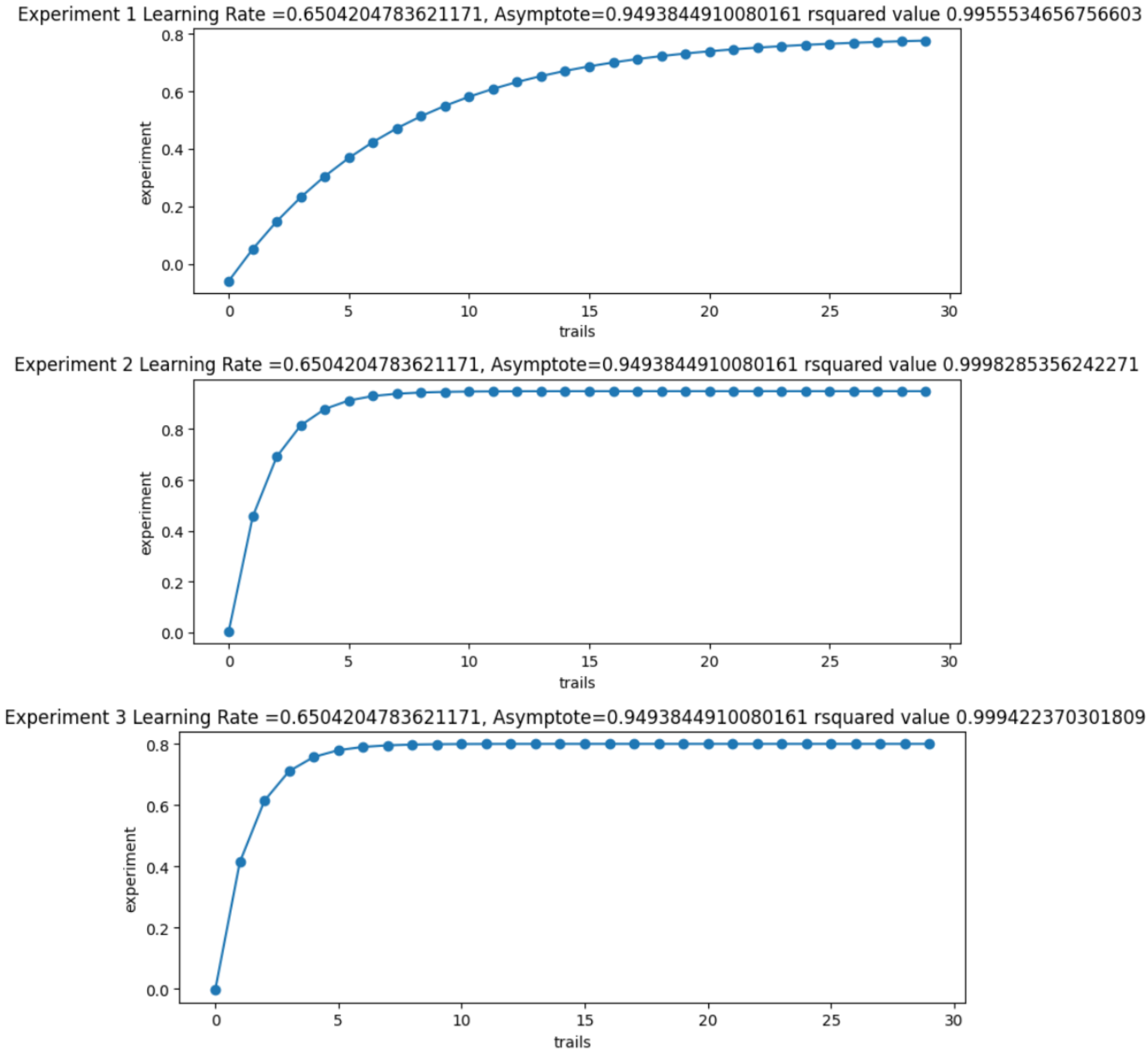
i. Exp1 :

Asymptote learning for experiment 1 is 0.7913065284884426
Learning rate for experiment 1 is 0.1399442931489643
R sqared of experiment 1 is 0.9955534656756603

Exp2:
Asymptote learning for experiment 2 is 0.9493844910080161
Asymptote for experiment 2 is 0.6504204783621171
R sqared of experiment 2 is 0.9998285356242271

Exp3:
Asymptote learning for experiment 3 is 0.8008236290427434
Learning rate for experiment 3 is 0.7334091931092159
R sqared of experiment 3 is 0.999422370301809

d



Experiment 1 Learning Rate =0.6504204783621171, Asymptote=0.9493844910080161 rsquared value 0.9955534656756603

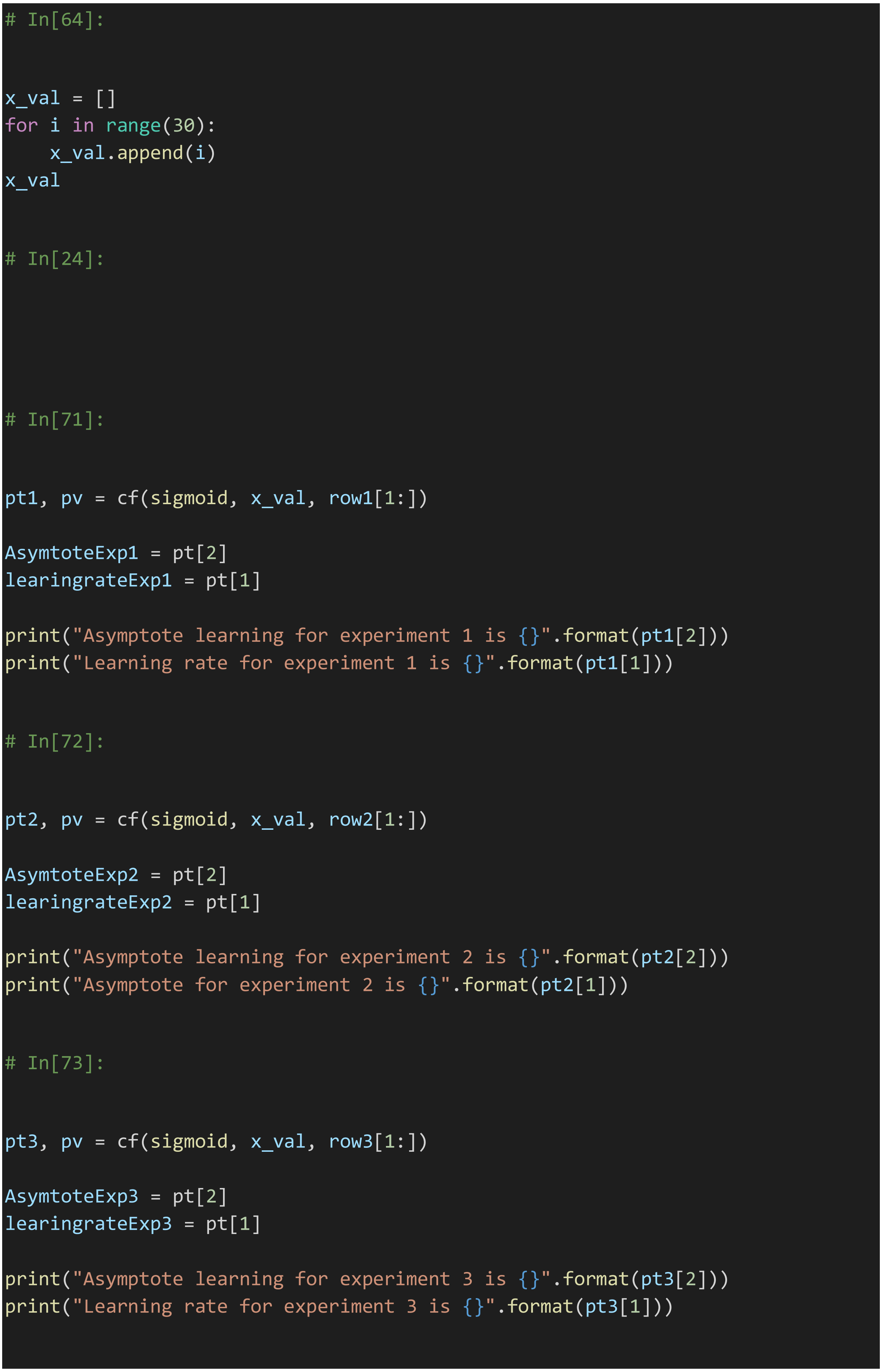
Experiment 2 Learning Rate =0.6504204783621171, Asymptote=0.9493844910080161 rsquared value 0.9998285356242271

Experiment 3 Learning Rate =0.6504204783621171, Asymptote=0.9493844910080161 rsquared value 0.999422370301809

Code:

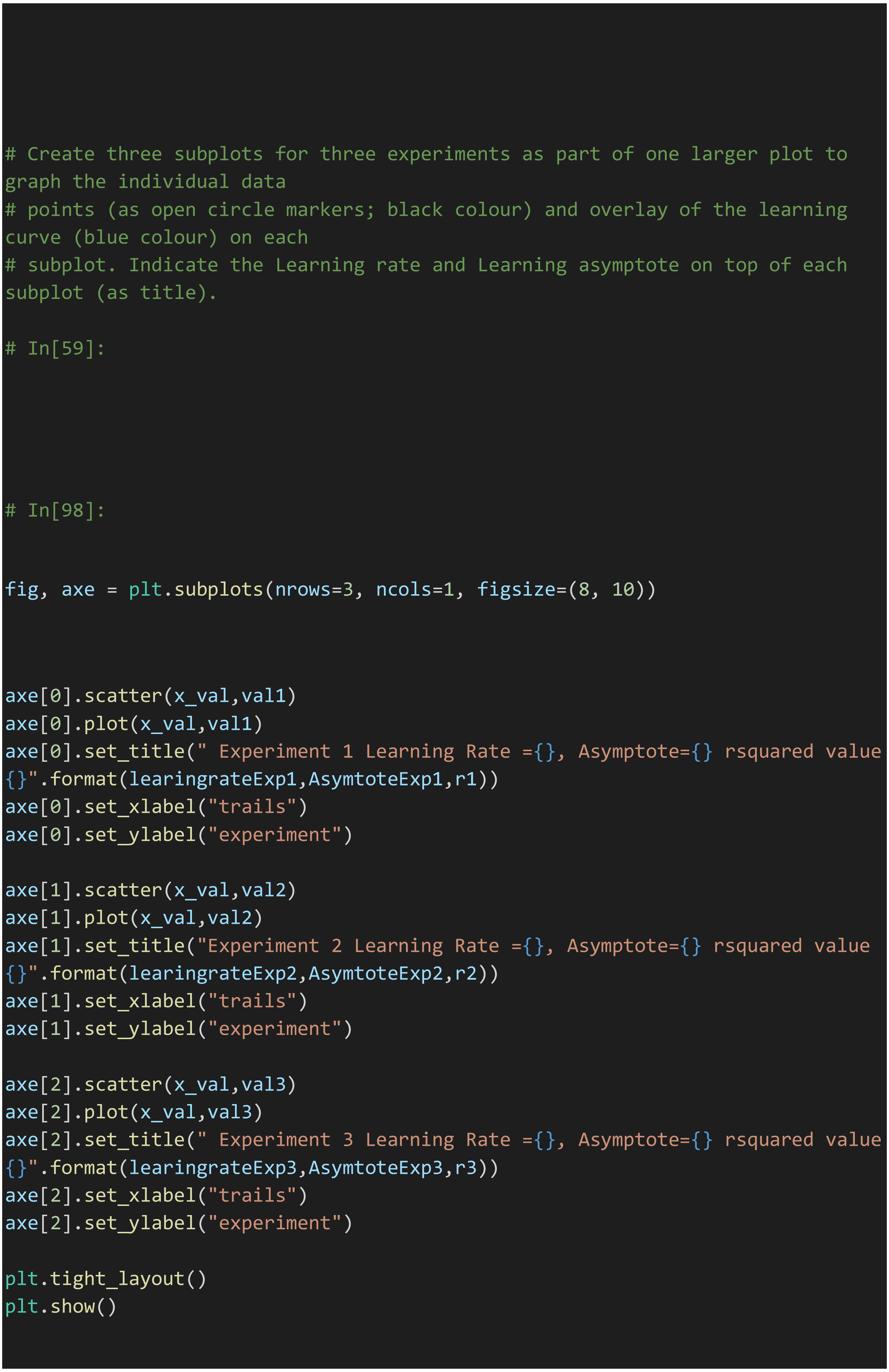
```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
from lmfit import Model as md
from scipy.optimize import curve_fit as cf
import matplotlib.pyplot as plt
```

```python
from sklearn.model_selection import learning_curve
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split


# In[2]:


read = pd.read_excel('Data-Assignment2B.xlsx',header=None)


# In[3]:


read


# In[14]:





# In[4]:


row1 = read.iloc[0]
row2 = read.iloc[1]
row3 = read.iloc[2]


# In[5]:


row1


# In[11]:


def sigmoid(xVAli, aVAli, bVaLi, cVali):
    return aVAli * np.exp(-bVaLi * xVAli) + cVali


# In[63]:


x_val.clear()

```



```
# In[64]:


x_val = []
for i in range(30):
    x_val.append(i)
x_val


# In[24]:






# In[71]:


pt1, pv = cf(sigmoid, x_val, row1[1:])

AsymtoteExp1 = pt[2]
learingrateExp1 = pt[1]

print("Asymptote learning for experiment 1 is {}".format(pt1[2]))
print("Learning rate for experiment 1 is {}".format(pt1[1]))


# In[72]:


pt2, pv = cf(sigmoid, x_val, row2[1:])

AsymtoteExp2 = pt[2]
learingrateExp2 = pt[1]

print("Asymptote learning for experiment 2 is {}".format(pt2[2]))
print("Asymptote for experiment 2 is {}".format(pt2[1]))


# In[73]:


pt3, pv = cf(sigmoid, x_val, row3[1:])

AsymtoteExp3 = pt[2]
learingrateExp3 = pt[1]

print("Asymptote learning for experiment 3 is {}".format(pt3[2]))
print("Learning rate for experiment 3 is {}".format(pt3[1]))
```

```
# In[51]:


val1 = [sigmoid(x_val,*pt1) for x_val in x_val]
val2 = [sigmoid(x_val,*pt2) for x_val in x_val]
val3 = [sigmoid(x_val,*pt3) for x_val in x_val]


# In[57]:


from sklearn.metrics import r2_score
sqq = sigmoid(1000,*pt1)
r1 = r2_score(row1[1:],val1)
r1
print("R sqared of experiment 1 is {}".format(r1))


# In[58]:


sqq = sigmoid(1000,*pt2)
r2 = r2_score(row2[1:],val2)
r2
print("R sqared of experiment 2 is {}".format(r2))


# In[60]:


sqq = sigmoid(1000,*pt3)
r3 = r2_score(row3[1:],val3)
r3
print("R sqared of experiment 3 is {}".format(r3))


# In[53]:





# In[54]:





# In[55]:
```



```
# Create three subplots for three experiments as part of one larger plot to graph the individual data
# points (as open circle markers; black colour) and overlay of the learning curve (blue colour) on each
# subplot. Indicate the Learning rate and Learning asymptote on top of each subplot (as title).

# In[59]:




# In[98]:


fig, axe = plt.subplots(nrows=3, ncols=1, figsize=(8, 10))



axe[0].scatter(x_val,val1)
axe[0].plot(x_val,val1)
axe[0].set_title(" Experiment 1 Learning Rate ={}, Asymptote={} rsquared value {}".format(learingrateExp1,AsymtoteExp1,r1))
axe[0].set_xlabel("trails")
axe[0].set_ylabel("experiment")

axe[1].scatter(x_val,val2)
axe[1].plot(x_val,val2)
axe[1].set_title("Experiment 2 Learning Rate ={}, Asymptote={} rsquared value {}".format(learingrateExp2,AsymtoteExp2,r2))
axe[1].set_xlabel("trails")
axe[1].set_ylabel("experiment")

axe[2].scatter(x_val,val3)
axe[2].plot(x_val,val3)
axe[2].set_title(" Experiment 3 Learning Rate ={}, Asymptote={} rsquared value {}".format(learingrateExp3,AsymtoteExp3,r3))
axe[2].set_xlabel("trails")
axe[2].set_ylabel("experiment")

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

```
# In[ ]:




# In[22]:




# In[ ]:




# In[ ]:
```

For one metric goodness of fit, I can r squared value. Higher the value better is the results

b. That will be strength in which an Unconditioned stimuli provides a good response with the conditioned stimuli. Here at start it generates a strong response then after it becomes constant as in Pavlov's dog experiment. It knows the sound already so it don't generated strong response.