In [1]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
```

QUESTION 1

In [2]:
```
df= pd.read_excel("Measurement.xlsx")
```

In [3]:
```
from datetime import datetime
import matplotlib.pyplot as plt
```

In [4]:
```
numbers = df["Measurement"]
```

In [5]:
```
window_size = 11
```

In [6]:
```
i = 0
```

In [7]:
```
moving_averages = []
```

In [8]:
```
while i < len(numbers) - window_size + 1:
    this_window = numbers[i : i + window_size]

    window_average = sum(this_window) / window_size
    moving_averages.append(window_average)
    i += 1
```

In [9]:
```
print(moving_averages)
```

```
[667.6636363636363, 680.809090909091, 687.9454545454546, 687.1181818181819,
683.0363636363637, 672.3454545454546, 657.7636363636365, 637.3000000000001,
613.3181818181819, 587.690909090909, 561.8818181818181, 539.6636363636363]
```

In [10]:
```
df
```

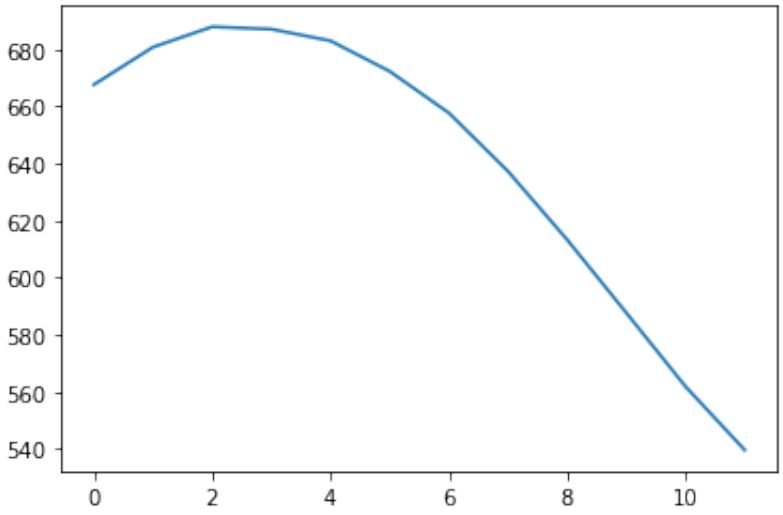
Out[10]:

	Year	Measurement
0	1984	539.9
1	1985	558.1
2	1986	620.1
3	1987	612.5
4	1988	640.6
5	1989	666.9
6	1990	729.6
7	1991	758.2
8	1992	757.7
9	1993	747.1
10	1994	713.6
11	1995	684.5
12	1996	636.6
13	1997	611.0
14	1998	567.6
15	1999	523.0
16	2000	506.5
17	2001	504.5
18	2002	494.4
19	2003	475.8
20	2004	463.2
21	2005	469.2

In [11]:

```
plt.plot(moving_averages)
```

Out[11]: [<matplotlib.lines.Line2D at 0x11b306160>]



In [12]:
```
window_size2 = 5
i = 0
```

In [13]:
```
moving_averages2 = []
num = df["Measurement"]
```

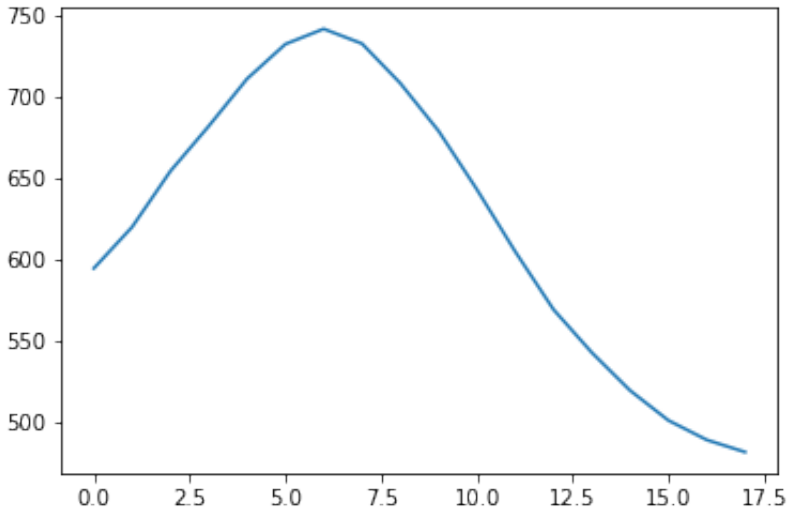
In [14]:
```
while i < len(num) - window_size2 + 1:
    this_window1 = num[i : i + window_size2]

    window_average2 = sum(this_window1) / window_size2
    moving_averages2.append(window_average2)
    i += 1
```

In [15]:
```
print(moving_averages2)
```

```
[594.24, 619.6400000000001, 653.9399999999999, 681.5600000000001, 710.6, 73
1.8999999999999, 741.24, 732.22, 707.9, 678.56, 642.66, 604.54, 568.9399999
999999, 542.52, 519.2, 500.84000000000003, 488.88, 481.41999999999996]
```

In [16]:
```
plt.plot(moving_averages2)
```

Out[16]: `[<matplotlib.lines.Line2D at 0x11b5296d0>]`



c) The effect of changing the span of the simple moving average window was that when it is 5 the moving average reacted faster than when it was 11. You can see the curve being steeper and shorter than the previous one. It looks more "rushed" when N is smaller.

Question 2!

In [17]:
```
df2= pd.read_excel("Yield_Data.xlsx")
```

In [18]:
```
df2.head()
```

Out[18]:

	Hour	Yield, %
0	1	89.0
1	2	90.5
2	3	91.5
3	4	93.2
4	5	93.9

In [19]:
```
array = df2["Yield, %"].to_numpy()
```

In [20]:
```
array
```

Out[20]:
```
array([89. , 90.5, 91.5, 93.2, 93.9, 94.6, 94.7, 93.5, 91.2, 89.3, 85.6,
       80.3, 75.9, 75.3, 78.3, 89.1, 88.3, 89.2, 90.1, 94.3, 97.7, 98.6,
       98.7, 98.9, 99.2, 99.4, 99.6, 99.8, 98.8, 99.9, 98.2, 98.7, 97.5,
       97.9, 98.3, 98.8, 99.1, 99.2, 98.6, 95.3, 94.2, 91.3, 90.6, 91.2,
       88.3, 84.1, 86.5, 88.2, 89.5, 89.5])
```

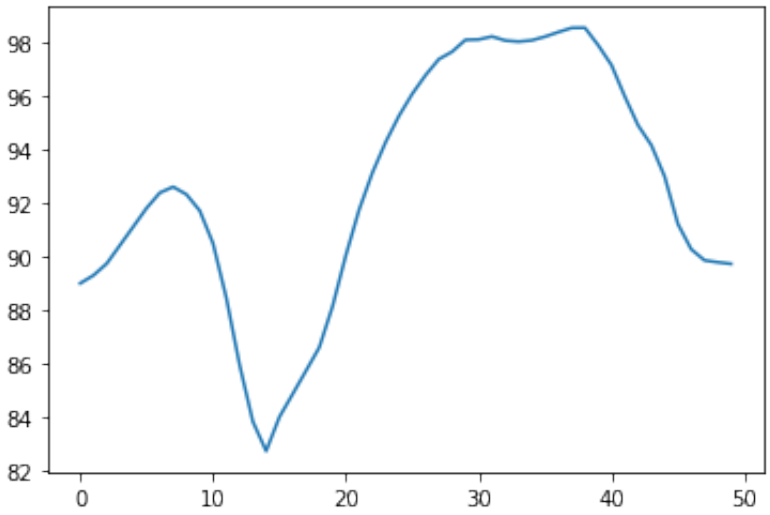
In [25]:
```
alpha= 0.2
```

In [26]:
```
for i in range(len(array)):
    if i ==0 : pass
    else:
        array[i]=(alpha*array[i]) + ((1-alpha)* array[i-1])
```

In [27]:
```
df2["Smooth"]= array.tolist()
```

In [28]:
```
plt.plot(df2["Smooth"])
```

Out[28]:
```
[<matplotlib.lines.Line2D at 0x11b5a0e50>]
```



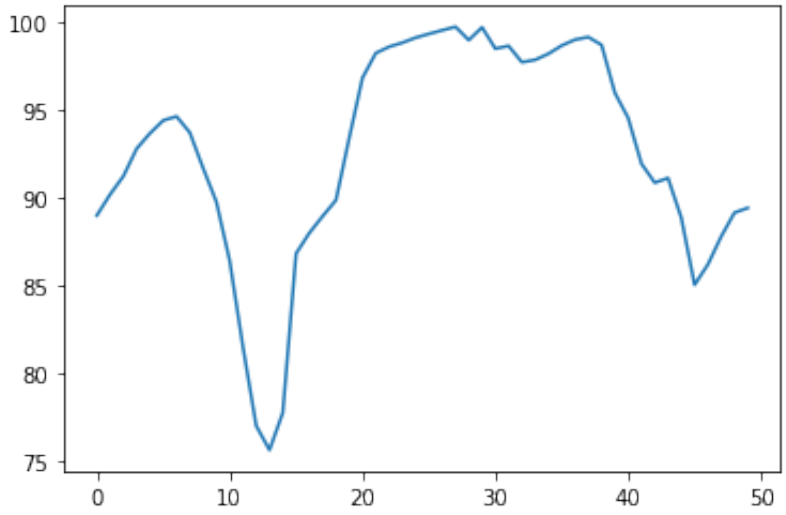
In [21]:
```
new_array = np.array(df2["Yield, %"])
new_alpha= 0.8
```

In [22]:
```
for i in range(len(new_array)):
    if i ==0 : pass
    else:
        new_array[i]=(new_alpha*new_array[i]) + ((1-new_alpha)* new_array[
```

In [23]:
```
df2["Smooth2"]= new_array.tolist()
```

In [24]:
```
plt.plot(df2["Smooth2"])
```

Out[24]: [<matplotlib.lines.Line2D at 0x11b3aa850>]



Just to check :

In [29]:
```
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing
```

In [30]:
```
fit1= SimpleExpSmoothing(array).fit(smoothing_level=0.2)
```

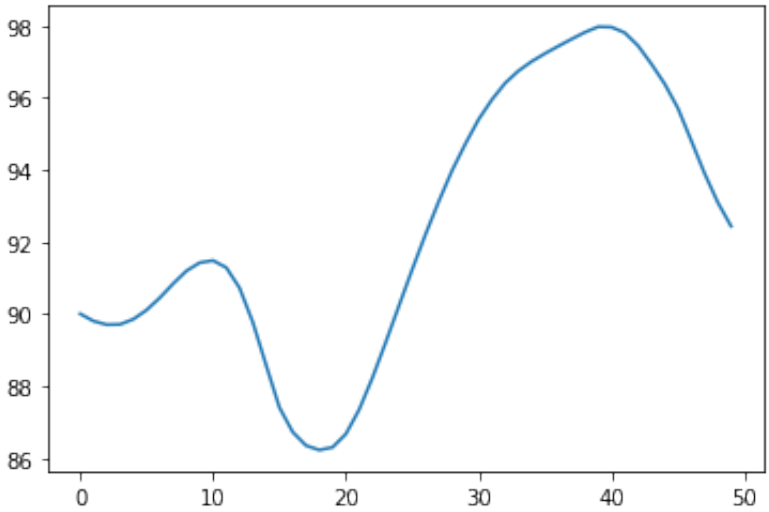
```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/holtwinters/model.py:427: FutureWarning: After 0.13 initialization must be handled at model creation
  warnings.warn(
```

In [31]:
```
print(fit1.fittedvalues)
```

```
[90.00449627 89.80359702 89.70287761 89.71030209 89.85464167 90.10883334
 90.45116267 90.84020694 91.19558699 91.42320674 91.48395512 91.28747587
 90.72223012 89.76758363 88.57790654 87.40539693 86.72277491 86.34898582
 86.22380137 86.29873126 86.66673715 87.34359143 88.22703451 89.2113567
 90.22886863 91.23892153 92.21179852 93.12996785 93.98439751 94.72385659
 95.40415614 95.95138161 96.41155064 96.7495968  97.01196246 97.23139799
 97.43458081 97.63123458 97.81984361 97.97175965 97.96211552 97.80545866
 97.44497993 96.94047434 96.38797179 95.71085128 94.81306011 93.90595136
 93.0971637  92.43565305]
```

In [32]:
```
plt.plot(fit1.fittedvalues)
```

Out[32]: [<matplotlib.lines.Line2D at 0x11c759460>]



In [33]:
```
fit2= SimpleExpSmoothing(array).fit(smoothing_level=0.8)
```

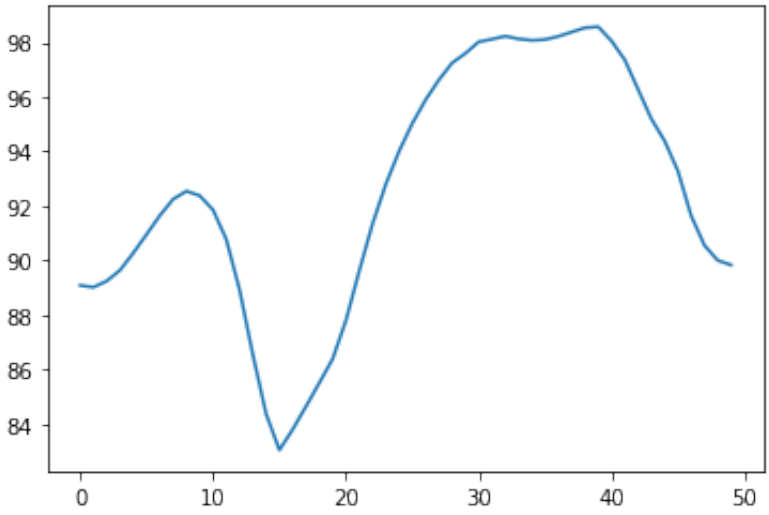
```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/holtwinters/mode
l.py:427: FutureWarning: After 0.13 initialization must be handled at model
creation
  warnings.warn(
```

In [34]:
```
print(fit2.fittedvalues)
```

```
[89.08450656 89.01690131 89.24338026 89.64067605 90.27373521 90.95522704
 91.64742941 92.24659308 92.54300438 92.37554948 91.85666878 90.77258087
 88.92351386 86.54390092 84.3641387  83.04511456 83.80285236 84.64363403
 85.50717766 86.38019621 87.78704779 89.59821639 91.32828875 92.78457412
 93.99604792 95.02251606 95.8871484  96.61954583 97.24560208 97.59447475
 98.01917842 98.11606246 98.22499391 98.12642392 98.07442489 98.10219706
 98.21828908 98.37793755 98.5350113  98.57054129 98.05293948 97.35365288
 96.27318256 95.1925981  94.38088889 93.27807319 91.63313097 90.54863926
 89.9993383  89.83155602]
```

In [35]:
```
plt.plot(fit2.fittedvalues)
```

Out[35]: [<matplotlib.lines.Line2D at 0x11c8883d0>]



Part c

In [59]:
```
mean_squared_error(df2["Yield, %"],fit1.fittedvalues)
```

Out[59]: 9.13376886817349

In [60]:
```
mean_squared_error(df2["Yield, %"],fit2.fittedvalues)
```

Out[60]: 1.4373942964932922

In [61]:
```
mean_squared_error(df2["Yield, %"],df2["Smooth"])
```

Out[61]: 0.0

In [62]:
```
mean_squared_error(df2["Yield, %"],df2["Smooth2"])
```

Out[62]: 12.937887674114036

Smoothing constant of 0.8 produced a lower error.