

IE 306 - Fall 2023 Assignment

1) Consider an inlet valve mechanism that is used to fill a water reservoir. The valve works such that the water filling rate is a function of the gap between the actual water height (h) and the target water height (h^*). The valve is produced to provide an inflow that will close the 10% of the gap between h^* and h per hour. Additionally, there is a small hole in the reservoir through which some water leaks out. The leakage rate is proportional to the water height, and this hole alone leads to a 5% fall in the water height per hour. Initial height of water is 10 cm, and the target water height is 60 cm.

a. Identify the state variables involved in this problem and give the model equation(s) that capture the dynamics of the state variable(s).

Firstly, I made some calculations to understand the question. After that, I started to solve it. I tried to solve it analytically, then faced with a differential equation. To see the behavior, I used an web app to solve it, and spot the graph.

Handwritten notes and calculations on a notebook page, showing the derivation of a differential equation for water height $h(t)$ and a table of values for $h(t)$ over time.

Left Page:

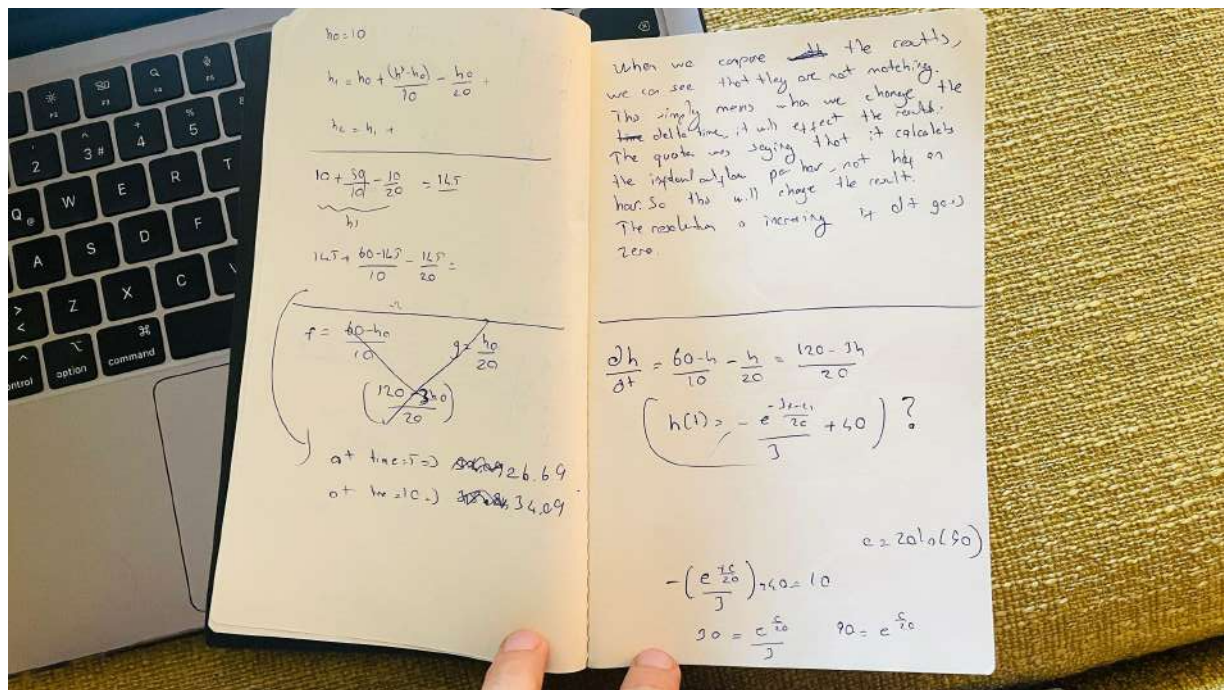
- Initial condition: $h(0) = 10$
- Target height: $h^* = 60$
- Gap: $h^* - h$
- Inflow rate: $f(h) = \frac{h^* - h}{10}$
- Leakage rate: $g(h) = \frac{h}{20}$
- Differential equation: $\frac{dh}{dt} = f(h) - g(h)$
- Discrete approximation: $h(t+1) = h(t) + \frac{h^* - h(t)}{10} - \frac{h(t)}{20}$
- Continuous approximation: $\frac{dh}{dt} = \frac{60 - h}{10} - \frac{h}{20}$
- Initial condition: $h(0) = 10$
- Final condition: $h(1) = 10$

Right Page:

- Diagram of a water reservoir with height h .
- Equations: $f(h) = \frac{h^* - h}{10}$, $g(h) = \frac{h}{20}$
- Table of values for $h(t)$ over time:

Time	States	$h(t)$	$f(t)$	$g(t)$	$h(t+dt)$
0	10	10	5	0.5	10
0.5	10	10	5	0.5	10
1	10	10	5	0.5	10
1.5	10	10	5	0.5	10
2	10	10	5	0.5	10
2.5	10	10	5	0.5	10
3	10	10	5	0.5	10
3.5	10	10	5	0.5	10
4	10	10	5	0.5	10
4.5	10	10	5	0.5	10
5	10	10	5	0.5	10

on computer



b) Next give the simulation equations and simulate the model for 10 hours, using a step size $dt=0.5$, tabulating the water height (h). (You may do it by calculator or using a spreadsheet software, or by computer programming).

The code:

```
times = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10]
timesi = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
haim = 60
h = 10
print(" t h* h f g h'")
for t in times:
    h = 10 if t==0 else h
    f = 0.05 * (haim - h)
    g = h * 0.025
    nexth = h + f - g

    gstr = "%.2f"%g
    fstr = "%.2f"%f
    hstr = "%.2f"%(h)
    nexts = "%.2f"%(nexth)

    if t != 10:
        print(" "+str(t) if int(t)==t else t, ' ', haim, ' ', hstr, fstr, gstr, " ", nexts)
    else:
        print(" "+str(t) if int(t)==t else t, ' ', haim, ' ', hstr, fstr, gstr, " ", nexts)
    h = nexth
```

```

main.py > [t]
1 times = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10]
2 timesi = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3 haim = 60
4 h = 10
5 print(" t h* h f g h'")
6 for t in times:
7     h = 10 if t==0 else h
8     f = 0.05 * (haim - h)
9     g = h * 0.025
10    nexth = h + f - g
11
12    gstr = "%.2f"%g
13    fstr = "%.2f"%f
14    hstr = "%.2f"%(h)
15    nexts = "%.2f"%(nexth)
16    if t != 10:
17        print(" "+str(t) if int(t)==t else t, ' ', haim, ' ', hstr, fstr, gstr, " ", nexts)
18    else:
19        print(" "+str(t) if int(t)==t else t, ' ', haim, ' ', hstr, fstr, gstr, " ", nexts)
20    h = nexth

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```

kadirgokhansezer ~/Desktop $% py main.py
t h* h f g h'
0 60 10.00 2.50 0.25 12.25
0.5 60 12.25 2.39 0.31 14.33
1 60 14.33 2.28 0.36 16.26
1.5 60 16.26 2.19 0.41 18.04
2 60 18.04 2.10 0.45 19.68
2.5 60 19.68 2.02 0.49 21.21
3 60 21.21 1.94 0.53 22.62
3.5 60 22.62 1.87 0.57 23.92
4 60 23.92 1.80 0.60 25.13
4.5 60 25.13 1.74 0.63 26.24
5 60 26.24 1.69 0.66 27.27
5.5 60 27.27 1.64 0.68 28.23
6 60 28.23 1.59 0.71 29.11
6.5 60 29.11 1.54 0.73 29.93
7 60 29.93 1.50 0.75 30.68
7.5 60 30.68 1.47 0.77 31.38
8 60 31.38 1.43 0.78 32.03
8.5 60 32.03 1.40 0.80 32.63
9 60 32.63 1.37 0.82 33.18
9.5 60 33.18 1.34 0.83 33.69
10 60 33.69 1.32 0.84 34.16
kadirgokhansezer ~/Desktop $%

```

c. Next solve the model equations analytically in order to find the exact water height value at $t=5$ and $t=10$. Compare the exact values with the simulated values you find in (b)

In the section a, i shared the way i tried to solve it. Then just calculated it with Python.

```

main.py > [?] t
1 times = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10]
2 timesi = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3 haim = 60
4 h = 10
5 print(" t h* h f g h'")
6 for t in timesi:
7     h = 10 if t==0 else h
8     f = 0.1 * (haim - h)
9     g = h * 0.05
10    nexth = h + f - g
11
12    gstr = "%.2f"%g
13    fstr = "%.2f"%f
14    hstr = "%.2f"%(h)
15    nexts = "%.2f"%(nexth)
16    if t != 10:
17        print(" "+str(t) if int(t)==t else t, ' ', haim, ' ', hstr, fstr, gstr, " ", nexts)
18    else:
19        print(" "+str(t) if int(t)==t else t, ' ', haim, ' ', hstr, fstr, gstr, " ", nexts)
20    h = nexth

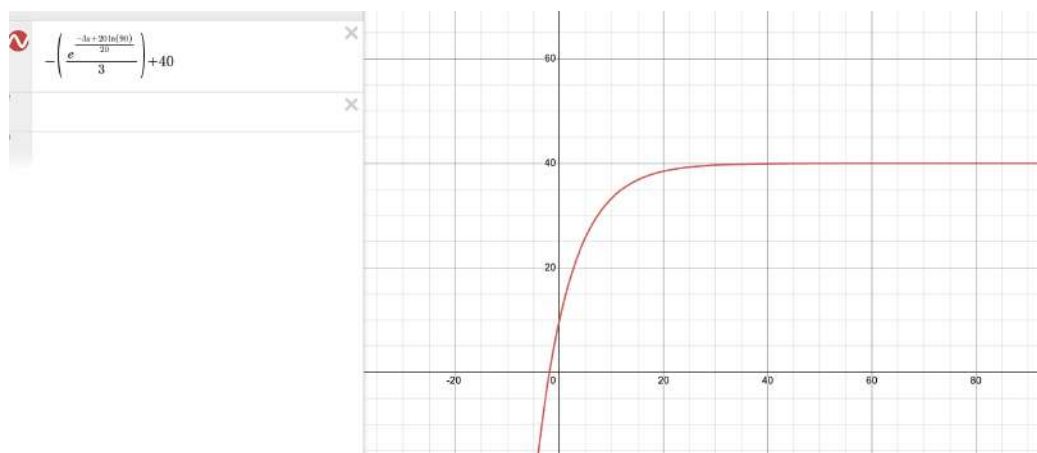
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS COMMENTS

```

kadirgokhansezer ~/Desktop $% py main.py
t h* h f g h'
0 60 10.00 5.00 0.50 14.50
1 60 14.50 4.55 0.73 18.32
2 60 18.32 4.17 0.92 21.58
3 60 21.58 3.84 1.08 24.34
4 60 24.34 3.57 1.22 26.69
5 60 26.69 3.33 1.33 28.69
6 60 28.69 3.13 1.43 30.38
7 60 30.38 2.96 1.52 31.83
8 60 31.83 2.82 1.59 33.05
9 60 33.05 2.69 1.65 34.09
10 60 34.09 2.59 1.70 34.98
kadirgokhansezer ~/Desktop $%

```



ing: $-\frac{e^{\frac{-3x-c_1}{20}}}{3} + 40$ assuming $c_1 = 1$

2)

a) Consider the pdf of standard Normal random variable z , which is known, but impossible to integrate analytically. Using the simulation method discussed in class, approximately compute the integral of this function in the interval $(0.56, 2.4)$. Use the

random generator of Excel, R or any other programming language and do the simulation for 1000 random number pairs. At the end, how well do you think is the approximation?

```
import random
import math
results = []

def pdf(x):
    stdev = 1
    mean = 0
    return (1.0 / (stdev * math.sqrt(2*math.pi))) * math.exp(-0.5*((x - mean) / stdev) ** 2)

maxxx = pdf(0.56)

def getpair():
    global maxxx, results
    maxx = 2.4
    minn = 0.56
    diff = maxx - minn
    x = random.random()*diff + minn
    yfrompdf = pdf(x)
    y = random.random()*(maxxx)
    if y <= yfrompdf:
        results.append(1)
    else:
        results.append(0)

for i in range(10000000):
    getpair()

fullarea =maxxx * (2.4 - 0.56)
area = 0
for i in results:
    if i == 1:
        area += 1

print(area/len(results) * fullarea)
```



```

q2.py > ...
1  import random
2  import math
3  results = []
4
5  def pdf(x):
6      stdev = 1
7      mean = 0
8      return (1.0 / (stdev * math.sqrt(2*math.pi))) * math.exp(-0.5*((x - mean) / stdev) ** 2)
9
10 maxxx = pdf(0.56)
11
12 def getpair():
13     global maxxx, results
14     maxx = 2.4
15     minn = 0.56
16     diff = maxx - minn
17     x = random.random()*diff + minn
18     yfrompdf = pdf(x)
19     y = random.random()*(maxxx)
20
21     if y <= yfrompdf:
22         results.append(1)
23     else:
24         results.append(0)
25
26 for i in range(1000000):
27     getpair()
28
29 fullarea =maxxx * (2.4 - 0.56)
30 area = 0
31 for i in results:
32     if i == 1:
33         area += 1
34
35 print(area/len(results) * fullarea)
36
37

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS COMMENTS

```

● kadirgokhansezer ~/Desktop $% py q2.py
0.27995111889174207
● kadirgokhansezer ~/Desktop $% py q2.py
0.27935497085321626
● kadirgokhansezer ~/Desktop $% py q2.py
0.27892197911997113
● kadirgokhansezer ~/Desktop $% py q2.py
0.27961225579615895
● kadirgokhansezer ~/Desktop $% py q2.py
0.2795799382972283
○ kadirgokhansezer ~/Desktop $% 

```

The z value I calculated from the script is pretty enough trustable. The more I iterate, the more it goes the real z value. This method is really good idea that it would be so useful when there is no chance to solve equations.

b) The famous central limit theorem in statistics states that if you take the sum of N identical independent random variables, the sum would be Normally distributed, if N is large enough. If we apply this to (say) 10 independent U(0,1) random variables, then the sum (Y) should be Normal(5, 0.83), since the mean of U(0,1) is 0.5 and its variance is 1/12. Write a computer program to generate 750 such Y values. To see if the theorem works, estimate the mean and variance of Y, and plot its histogram (using a suitable number of classes). In both parts (a) and (b), you should turn in not only your results, but also your formulas used in the cells (if you do it by spreadsheet), or you code.

```

import numpy as np
import matplotlib.pyplot as plt

class CLT:
def __init__(self, noSamples, noVariables):
self.noSamples = noSamples
self.noVariables = noVariables
self.sums = np.sum(np.random.rand(noSamples, noVariables), axis=1)
self.sample_mean = np.mean(self.sums)
self.sample_variance = np.var(self.sums)

```

```

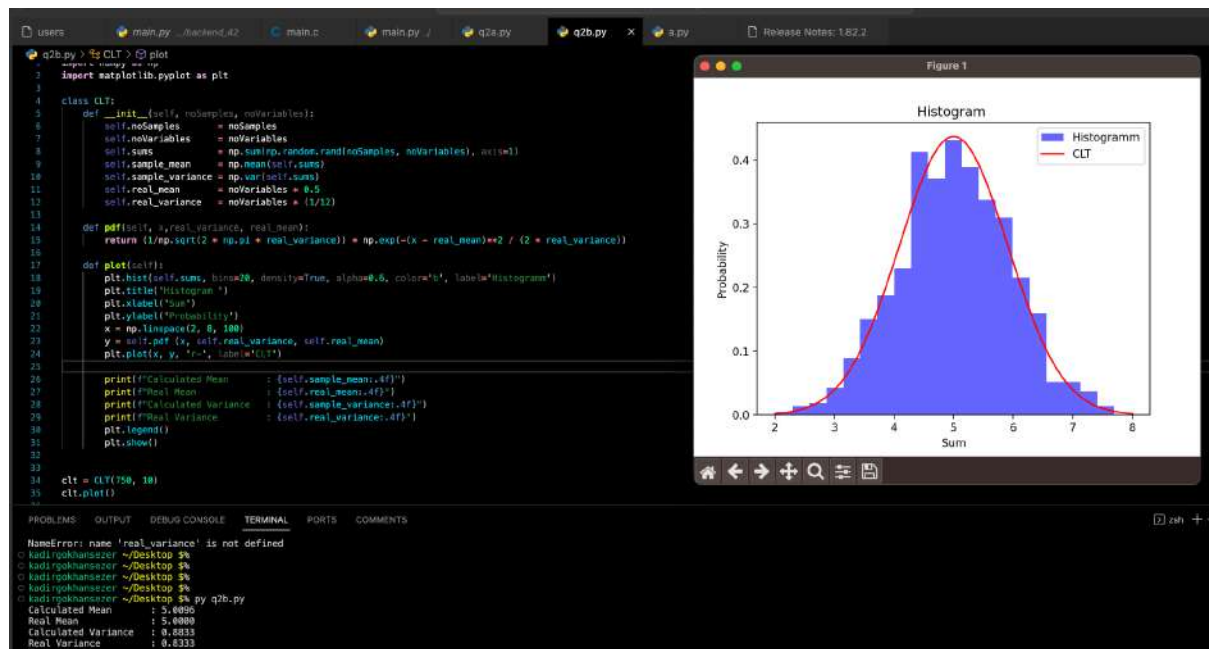
self.real_mean = noVariables * 0.5
self.real_variance = noVariables * (1/12)

def pdf(self, x, real_variance, real_mean):
    return (1/np.sqrt(2 * np.pi * real_variance)) * np.exp(-(x - real_mean)**2 / (2 * real_variance))
def plot(self):
    plt.hist(self.sums, bins=20, density=True, alpha=0.6, color='b', label='Histogramm')
    plt.title('Histogram ')
    plt.xlabel('Sum')
    plt.ylabel('Probability')
    x = np.linspace(2, 8, 100)
    y = self.pdf(x, self.real_variance, self.real_mean)
    plt.plot(x, y, 'r-', label='CLT')

print(f"Calculated Mean : {self.sample_mean:.4f}")
print(f"Real Mean : {self.real_mean:.4f}")
print(f"Calculated Variance : {self.sample_variance:.4f}")
print(f"Real Variance : {self.real_variance:.4f}")
plt.legend()
plt.show()

clt = CLT(750, 10)
clt.plot()

```

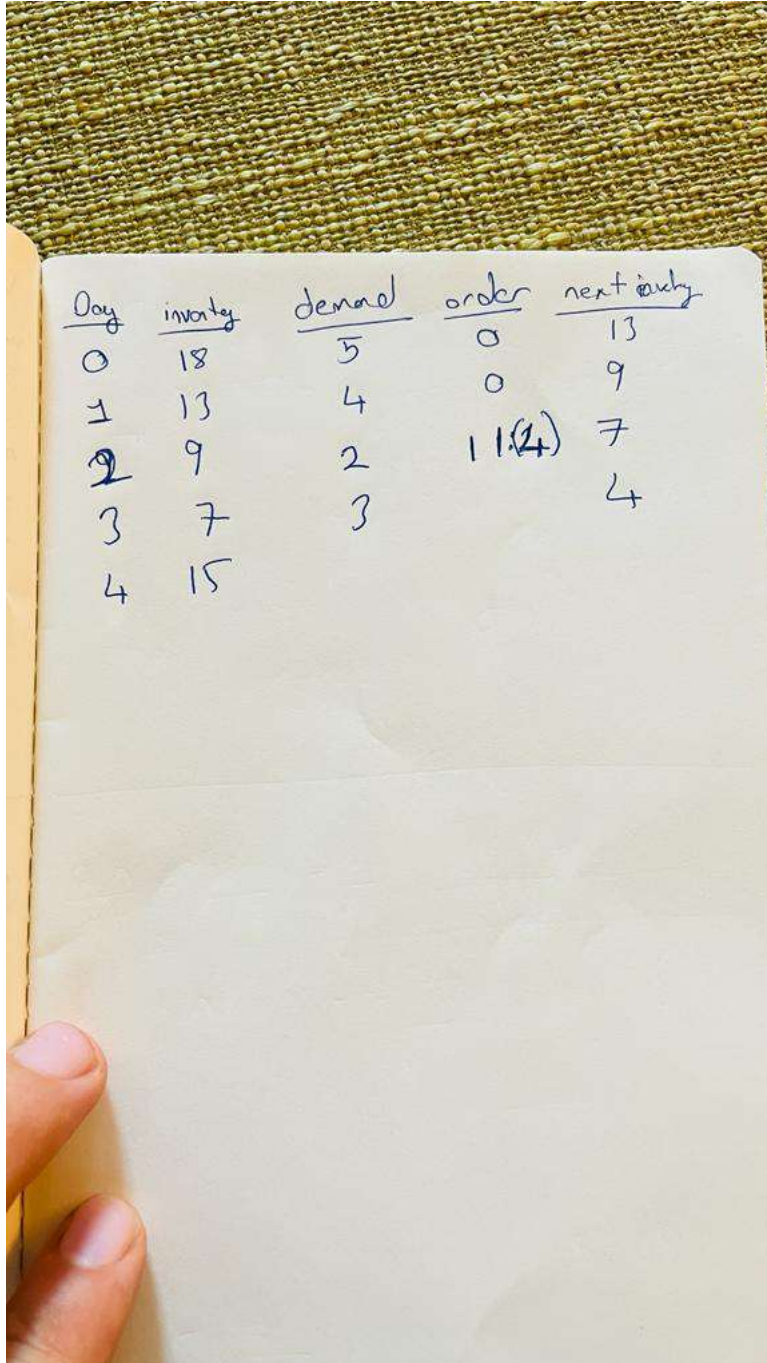


3. Consider the following inventory system; a. Whenever the inventory falls to or below 10 units, an order is placed. Only one order can be outstanding at a time (i.e. no new orders until the placed order arrives). b. The size of each order is equal to target inventor level (20) minus the current inventory level (I) c. If a demand occurs during a period when the inventory level is zero, the sale is lost d. Daily demand is discrete between 2 and 7, where probabilities are given as; Demand 2 3 4 5 6 7 Probability 0,10 0,18 0,20 0,22 0,20 0,10 e. Lead time of orders is discrete uniform distributed between zero and 5 days. For simplicity, assume that orders are placed at the close of the business day and receive after the lead time has occurred. Thus, if lead time is one day, the order is available for distribution on the morning of the second day of business following the placement of the order. f. The unsatisfied portion of the daily demand is considered as lost sales Estimate by simulation, the average number of

lost sales per day, and the average inventory level for his inventory system. The simulation will start with 18 unit in inventory and let the simulation run for 10 days.

Answer: Firstly, to understand the question and the simulation I am supposed to do, I made a hand simulation so that I could see the pattern and what to do next.

PS: in the code, lead_time shows when an order comes. In other words, it calculates the random related lead time and add it with the day it is running on.



A handwritten table on a piece of paper, showing the results of a simulation over 5 days. The table has five columns: Day, inventory, demand, order, and next inventory. The data is as follows:

Day	inventory	demand	order	next inventory
0	18	5	0	13
1	13	4	0	9
2	9	2	1 (2)	7
3	7	3		4
4	15			

```
import random
days = [0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 ]
```



```

invlevel= []

def get_demand():
    rand = random.random()
    if rand > .90: #10% chance
        return 7
    elif rand > .70: #20% chance
        return 6
    elif rand > .48: #22% chance
        return 5
    elif rand > .28: #20% chance
        return 4
    elif rand > .10: #18% chance
        return 3
    else: #10% chance
        return 2
    inventory = 18
    total_lost_sales = 0
    total_inventory = 0
    order_size = 0
    lead_time = 0
    lost = 0
    print("day", "inventory", "demand", "order", "lead_time", "lost",
          "next_inventory", "total_lost")
    for day in days:
        if day == lead_time and day != 0:
            inventory += order_size
            lead_time = 0
            inv_of_day = inventory
            invlevel.append(inv_of_day)
            if inventory <= 10 and lead_time == 0:
                order_size = 20 - inventory
                lead_time = random.randint(0, 5) + day
                demand = get_demand()

            if inventory >= demand:
                inventory -= demand
                next_inventory = inventory
            else:
                total_lost_sales += demand - inventory
                lost = demand - inventory
                inventory = 0
                next_inventory = inventory

```

```

print(day,\
      '*'(3 if day>9 else 4), \
      '*'(1 if inv_of_day>9 else 2), inv_of_day,\
      '*'(3 if demand>9 else 4), demand,\
      '*'(2 if order_size>9 else 3), order_size,\
      '*'(4 if lead_time>9 or lead_time<0 else 5), lead_time,\
      '*'(4 if lost>9 else 5), lost,\
      '*'(4 if next_inventory>9 else 5), next_inventory,\
      '*'(9 if total_lost_sales>9 else 10), total_lost_sales)
lost = 0

avg_total_lost_sales_per_day = total_lost_sales / 10
avg_inventory_level = total_inventory / 10

print(f"Average Lost Sales per Day: {avg_total_lost_sales_per_day:.2f}")

print(f"Average Inventory Level: {sum(invlevel)/10:.2f}")

```

```

kadirgokhansezer ~/Desktop $% py a.py
day inventory demand order lead_time lost next_inventory total_lost
0      18      4      0      0      0      14      0
1      14      6      0      0      0      8      0
2       8      7     12      3      0      1      0
3      13      4     12      0      0      9      0
4       9      5     11      7      0      4      0
5       4      5     11      7      1      0      1
6       0      2     11      7      2      0      3
7      11      2     11      0      0      9      3
8       9      7     11     11      0      2      3
9       2      2     11     11      0      0      3
Average Lost Sales per Day: 0.30
Average Inventory Level: 8.80

```

```

kadirgokhansezer ~/Desktop $% py a.py
day inventory demand order lead_time lost next_inventory total_lost
0      18      2      0      0      0      16      0
1      16      5      0      0      0      11      0
2      11      6      0      0      0      5      0
3       5      6     15      5      1      0      1
4       0      7     15      5      7      0      8
5      15      6     15      0      0      9      8
6       9      4     11      9      0      5      8
7       5      4     11      9      0      1      8
8       1      4     11      9      3      0     11
9      11      5     11      0      0      6     11
Average Lost Sales per Day: 1.10
Average Inventory Level: 9.10

```

```
● kadirgokhansezer ~/Desktop $% py a.py
day inventory demand order lead_time lost next_inventory total_lost
0      18        6      0      0      0      12          0
1      12        4      0      0      0       8          0
2       8        3     12      5      0       5          0
3       5        3     12      5      0       2          0
4       2        5     12      5      3       0          3
5      12        3     12      0      0       9          3
6       9        3     11      9      0       6          3
7       6        4     11      9      0       2          3
8       2        5     11      9      3       0          6
9      11        7     11      0      0       4          6
Average Lost Sales per Day: 0.60
Average Inventory Level: 8.50
```