1. The nine measurements that follow are furnace temperatures recorded on successive batches in a semiconductor manufacturing process (units are °F):
   953, 955, 948, 951, 957, 949, 954, 950, 959

a. Find the sample median of these data.

Ans.

```
>>> Temperatures = [953, 955, 948, 951, 957, 949, 954, 950, 959]
>>> np.median(Temperatures)
np.float64(953.0)
```

b. How much could the largest temperature measurement increase without changing the sample median?

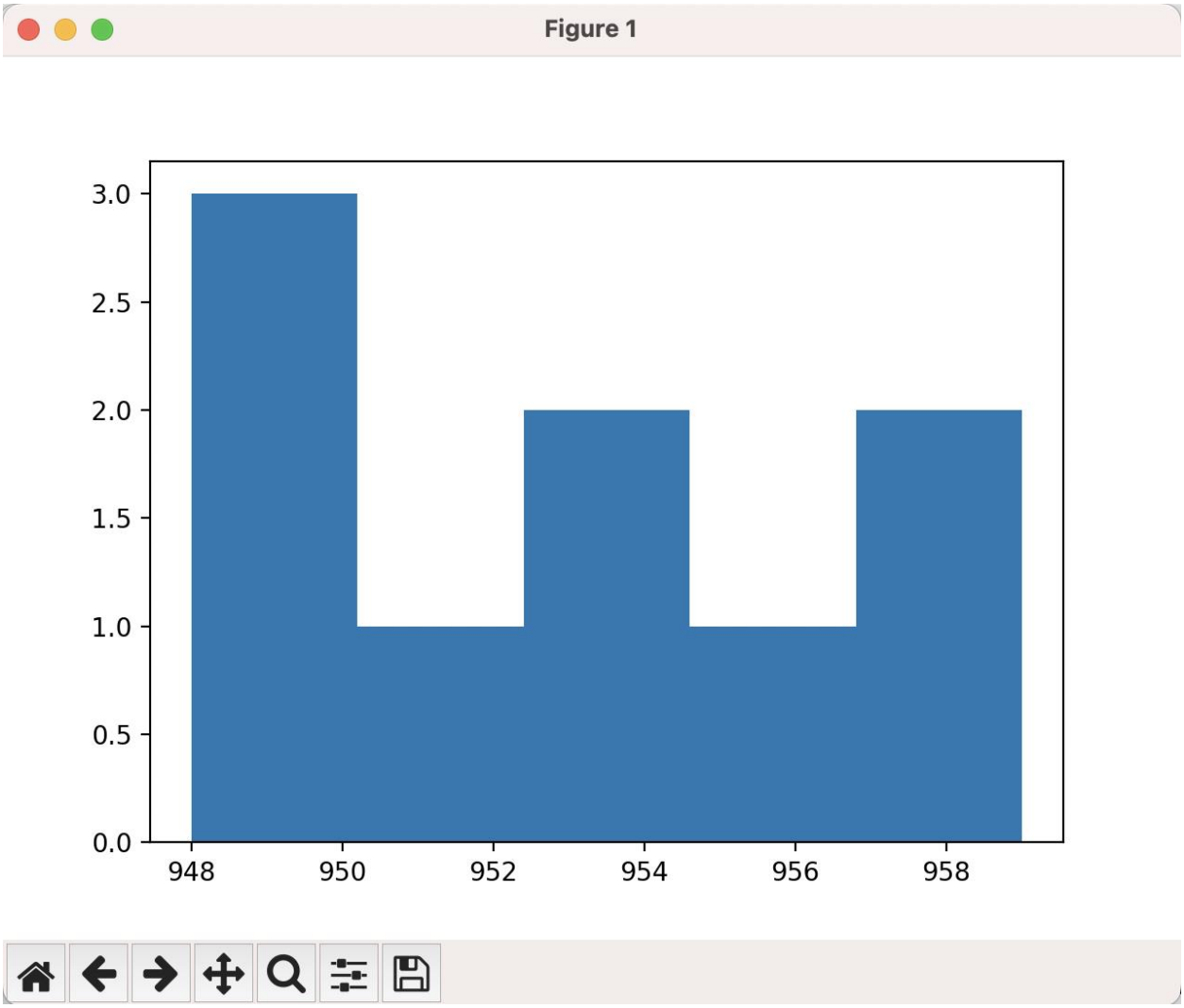
Ans. Median is the middle value after we sort(ascending) the temperatures. If we increase the largest value it doesn't matter, median will be remain the same. Largest temperature can be increased indefinitely.

c. Calculate the sample average and standard deviation.

```
>>> avg = sum(Temperatures)/len(Temperatures)
>>> avg
952.8888888888889
```

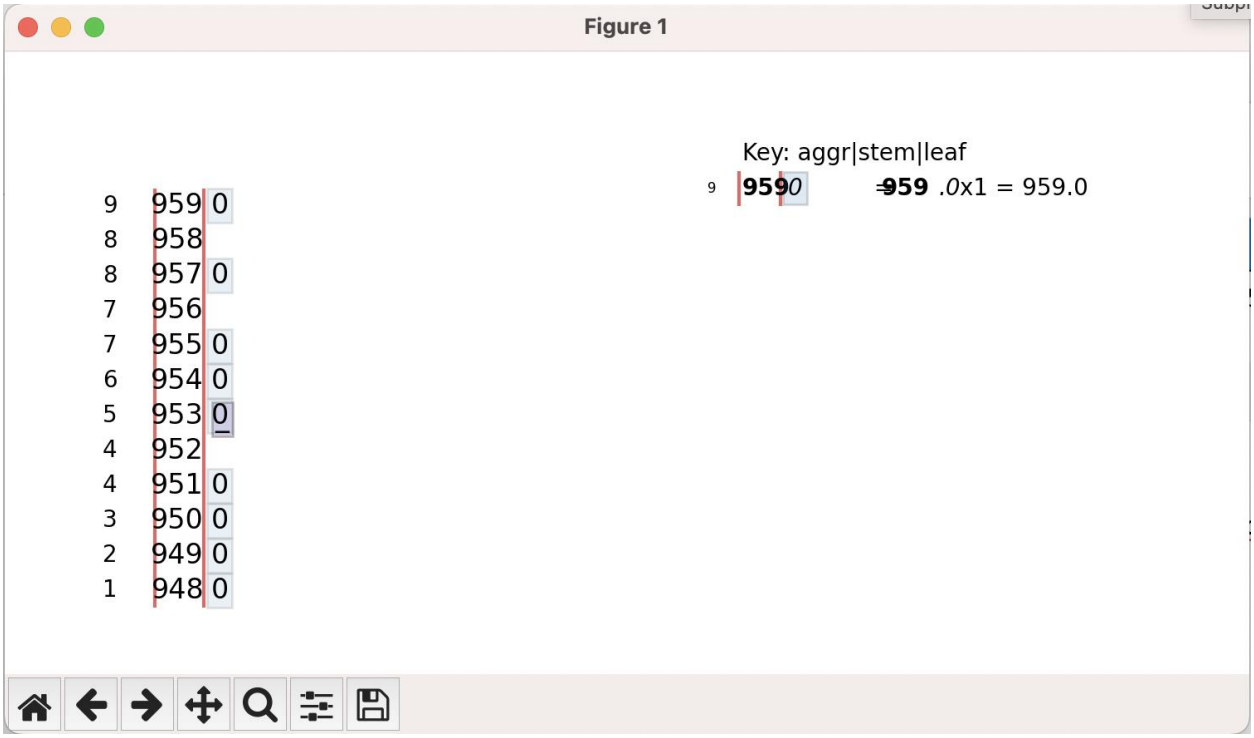
d. Construct a histogram.

```
>>> plt.hist(Temperatures, bins='auto')
... plt.show()
```



e. Construct a stem-and-leaf plot.

```
>>> stemgraphic.stem_graphic(Temperatures, scale=1, leaf_order=True)
... plt.show()
```

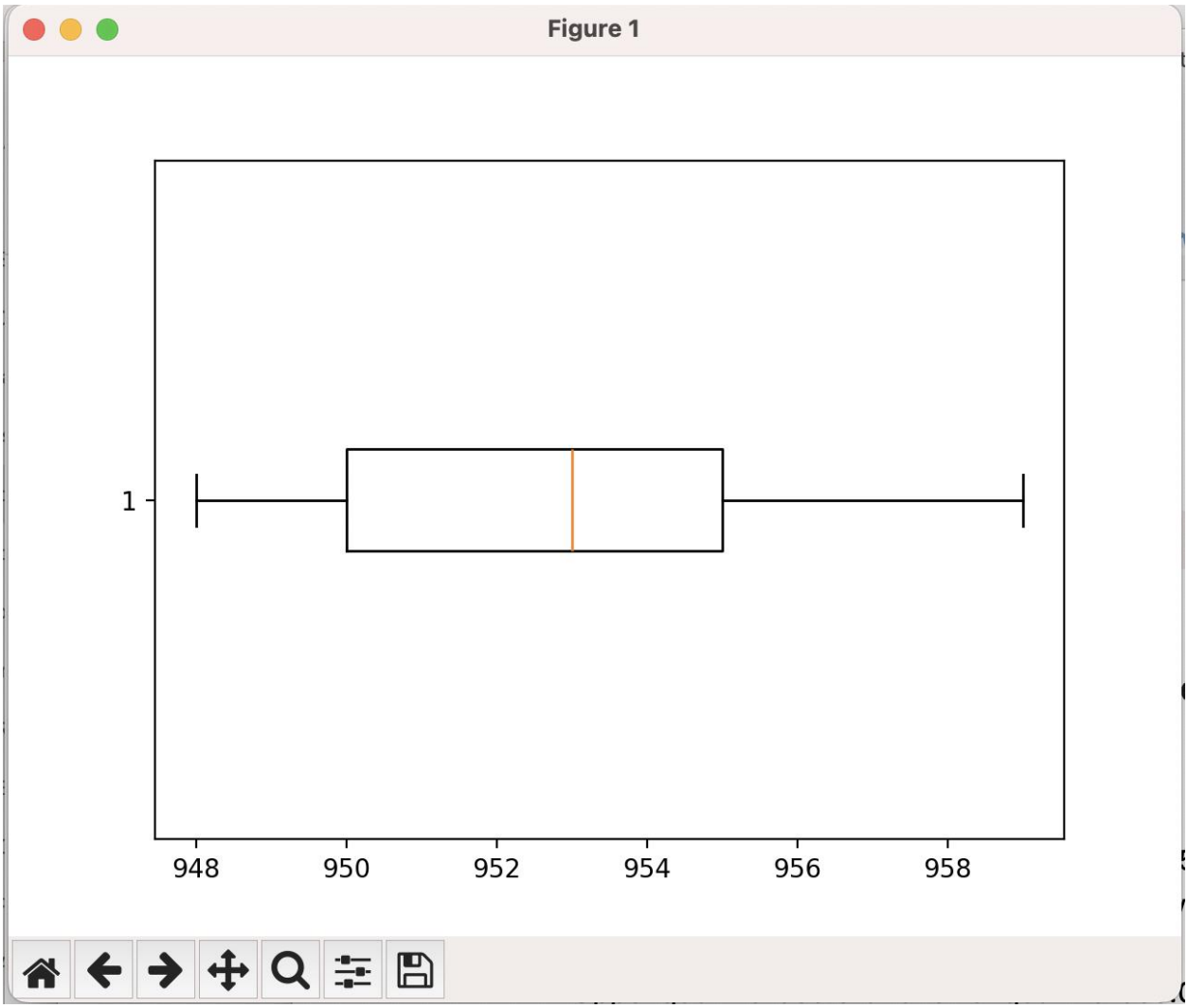


f. Find the sample median and the lower and upper quartiles.

```
>>> np.median(Temperatures)
np.float64(953.0)
>>>
>>> quarts = np.quantile(Temperatures, [.25, .50, .75])
... print("Upper quartile " , quarts[2] , " and lower quartile " , quarts[0])
...
Upper quartile  955.0  and lower quartile  950.0
```

g. Construct a boxplot.

```
>>> plt.boxplot(Temperatures,vert=False)
... plt.show()
```



h.

Based on some/all of the above, provide a brief written description of this sample.

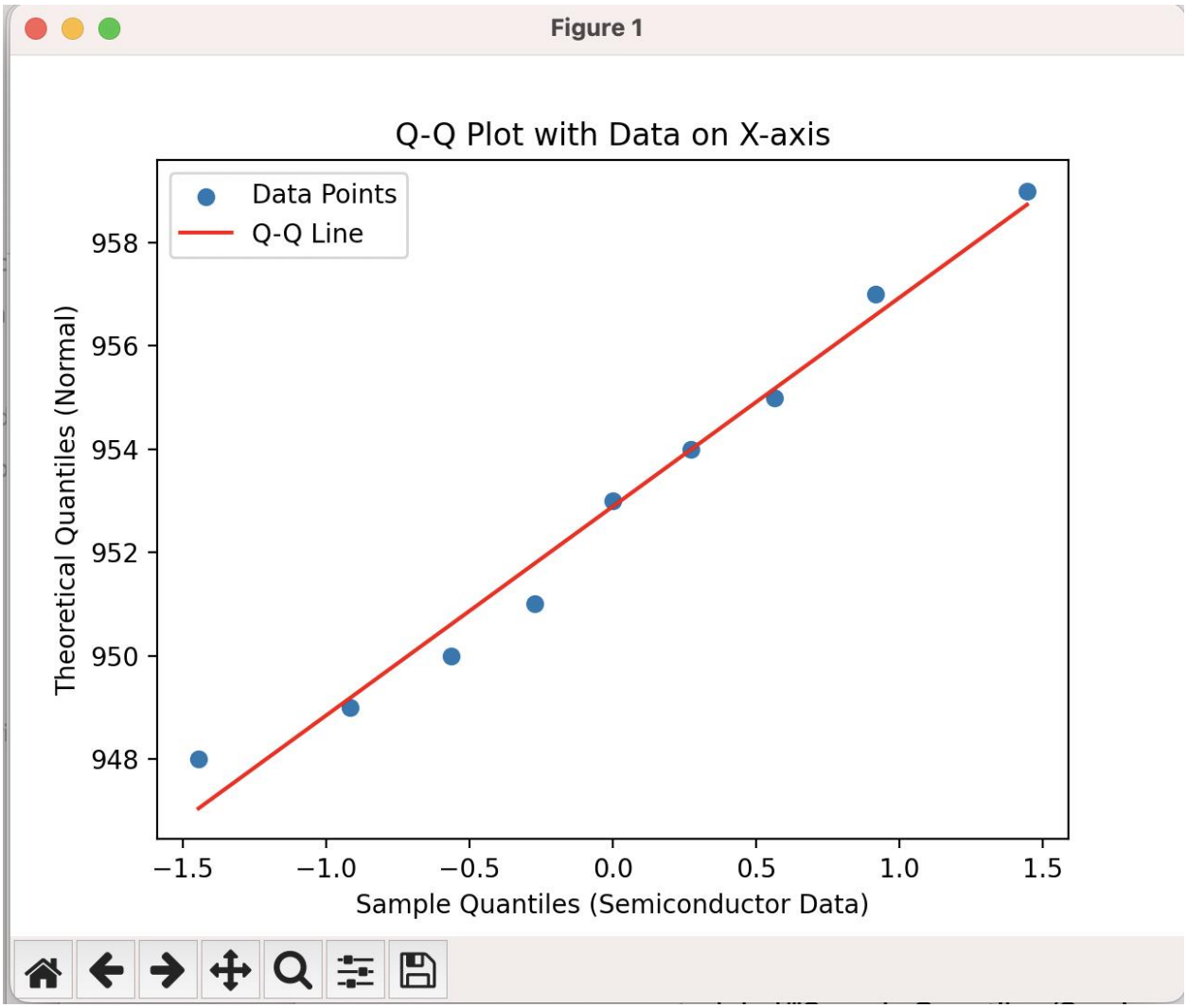
Ans. From this boxplot we can derive below info

- Value starts from 948 to 958
- A wider box indicates higher variability and narrow box indicates lower variability.
- The median is somewhere at 953 and it close to upper quantile and far from lower quantile.
- There is no outliers otherwise it would have been marked with dot.

i. Construct a normal probability plot. Does the assumption that failure time for this component is well modeled by a normal distribution seem reasonable?

Ans.

```
fig, ax = plt.subplots()
... res = probplot(Temperatures, dist="norm")
... ax.scatter(res[0][0], res[0][1], label="Data Points")  # Scatter plot for octane data
... ax.plot(res[0][0], res[1][1] + res[1][0] * res[0][0], color="red", label="Q-Q Line")  #
Reference line
...
... # Set labels and title
... ax.set_title("Q-Q Plot with Data on X-axis")
... ax.set_xlabel("Sample Quantiles (Semiconductor Data)")
... ax.set_ylabel("Theoretical Quantiles (Normal)")
...
... # Show legend and plot
... ax.legend()
... plt.show()
```



Failure time for this component is not modeled by a normal distribution. In case of failure, there might be significant deviation. The data may not follow a normal distribution.

2. Patients arriving at an outpatient clinic are routinely screened for high blood pressure. Assume that this condition occurs in 15% of the population.

a. What is the probability that the third patient of the day is the first to have high blood pressure?

Ans. This follows geometric distribution. The formula for geometric distribution

P(X=k) = (1-p)^(k-1)*p
P=0.15(probability of high blood pressure)
K= 3(random variable for 3rd patient to have blood pressure)
1-p = 0.85(number of failure 1-.15)

```
>>> p = 0.15
... k=3
... probability = geom.pmf(k, p)
... print(probability)
...
0.10837499999999999
```

So probability that the third patient of the day is the first to have high blood pressure is 10.8374%.

b. What is the average number of patients that must be seen to find the first patient with high blood pressure?

Ans. The average number of patients will be 1/P.

```
>>> average=1/p
... print(average)
...
6.666666666666667
```

6.667 is the average number of patients.

c. If the clinic typically sees 50 patients each day, what is the probability of finding exactly 10 patients with high blood pressure?

Ans. This follows a binomial distribution.

```
>>> n = 50  # Number of trials
... p = 0.15 # Probability of success
... k=10 # number of success
... prob=binom.pmf(k,n,p)
... print(prob)
...
0.0889890131316163
>>>
```

So probability of finding exactly 10 patients with high blood pressure is 8.8989%.

3. An electronic component for a medical X-ray unit is produced in lots of size $N = 25$. An acceptance testing procedure is used by the purchaser to protect against lots that contain too many nonconforming components. The procedure consists of selecting five components at random from the lot (without replacement) and testing them. If none of the components is nonconforming, the lot is accepted.

a. If the lot contains two nonconforming components, what is the probability of lot acceptance?

Ans. It follows a hypergeometric distribution.

```
>>> N = 25  # Total components in the lot
... K = 2   # Nonconforming components
... n = 5   # Sample size
... k = 0   # No nonconforming components in the sample
...
... # Compute hypergeometric probability
... probability = hypergeom.pmf(k, N, K, n)
... print(probability)
...
0.6333333333333333
```

Probability of lot acceptance is 63.33%

b. Calculate the desired probability in (a) using the binomial approximation. Is this approximation satisfactory? Why or why not?

```
>>> n = 5   # Sample size
... p = 2/25  # Probability of selecting a nonconforming component
... k = 0   # No nonconforming components
...
... # Compute binomial probability
... binomial_approx = binom.pmf(k, n, p)
... print(binomial_approx)
0.6590815231999997
```

For binomial distribution(with replacement) acceptance will be 65.908%

if the binomial approximation is close to the exact hypergeometric probability, it is satisfactory. Otherwise, the hypergeometric approach is preferred.

So in my understanding binomial approximation is not satisfactory.

b. Suppose the lot size was $N = 150$. Would the binomial approximation be satisfactory in this case?

```
N = 150   # Total components in the lot

... K = 2     # Nonconforming components

... n = 5     # Sample size

... k = 0     # No of nonconforming components

...

... # hypergeometric probability

... hypergeom_prob = hypergeom.pmf(k, N, K, n)

...

... #  binomial probability approximation

... p = K / N  # Probability of selecting a nonconforming component

... binomial_prob = binom.pmf(k, n, p)

...

... print("hypergeom_prob =", hypergeom_prob)

... print("binomial_prob = ", binomial_prob)

...

hypergeom_prob = 0.934228187919463

binomial_prob =  0.935087565010699
```

Yes. For N=150, binomial approximation is satisfactory.

c. Suppose that the purchaser will reject the lot (of $N = 25$) with the decision rule of finding one or more nonconforming components in a sample of size n, and wants the lot to be rejected with probability at least 0.95 if the lot contains five or more nonconforming components. How large should the sample size be?

```
N = 25  # Total lot size
... K = 5   # Nonconforming components
... target_prob = 0.95  # Desired rejection probability
... k=0
...
... # Find the smallest n such that P(reject) >= 0.95
... for n in range(1, N + 1):
...     P_accept = hypergeom.pmf(k, N, K, n)  # Probability of accepting the lot
...     P_reject = 1 - P_accept  # Probability of rejecting the lot
...
...     if P_reject >= target_prob:
...         print("Minimum sample size required: ", n)
...         break
...
Minimum sample size required:  11
```

4. Surface-finish defects in a small electric appliance occur at random with a mean rate of 0.1 defects per unit. Find the probability that a randomly selected unit will contain at least one surface-finish defect.

Ans. It follows poisson distribution

```
>>> lambda_ = 0.1
...
...
... probability = 1 - poisson.pmf(0, lambda_)
... print(probability)
...
0.09516258196404048
```

The probability is 9.516%

5. A lightbulb has a normally distributed light output with mean 5000 end foot-candles and standard deviation of 50 end foot-candles. Find a lower specification limit such that only 0.5% of the bulbs will not exceed this limit.

```
It follows normal distribution
mu = 5000
... sigma = 50
... probability = 0.005  # Lower 0.5% tail
...
...
... lower_limit = norm.ppf(probability, loc=mu, scale=sigma)
...
... print(lower_limit)
...
4871.208534822555
```

It means only .5% of the bulbs have the output less than 4871.208 ft candle

6. The time to failure of a product is well described by the following probability distribution:

   $f(x) = 0.1e^{-0.1x}, x \geq 0$

   If time is measured in hours, what is the probability that a unit fails before 100 hours? Also, what is the probability that a unit fails after 5 hours?

   Ans. It follows exponential distribution.

```
>>> lambda_ = 0.1
...
... # Define the exponential distribution scale parameter (1/lambda)
... scale = 1 / lambda_
...
... # Probability of failure before 100 hours
... P_fail_100 = expon.cdf(100, scale=scale)
...
... # Probability of failure after 5 hours
... P_survive_5 = 1 - expon.cdf(5, scale=scale)
...
... print("probability that unit fails before 100 hours is " , P_fail_100*100,"%")
... print("probability that unit fails after 5 hours is " , P_survive_5*100,"%")
```

```
...
probability that unit fails before 100 hours is  99.99546000702375 %
probability that unit fails after 5 hours is  60.653065971263345 %
```

7. The output of a manufacturing process is normally distributed with mean 100 and standard deviation 2. Suppose that the lower specification limit is 97 and the upper specification limit is at 102. What proportion of the process output is within the specifications?

```
>>> P_LSL = norm.cdf(LSL, loc=mu, scale=sigma)
... print(P_LSL)
... P_USL = norm.cdf(USL, loc=mu, scale=sigma)
... print(P_USL)
... # Proportion within specifications
... P_within_specs = P_USL - P_LSL
... print(P_within_specs*100, "% is withinn specification")
...
0.06680720126885807
0.8413447460685429
77.45375447996848 % is withinn specification
```

Now suppose that units that are above the upper specification must be scrapped at a cost of $5 per unit, while units that are below the lower specification limit can be reworked at a cost of $1 per unit. If the process mean can be adjusted relatively easily, what action would you recommend to minimize the costs that are being incurred due to scrap and rework?

Ans. Let's find out what is the probability of go beyond upper limit and lower limit

```
mu_current = 100
... sigma = 2
... LSL = 97
... USL = 102
...
... # Compute probabilities for current mean
... P_LSL_current = norm.cdf(LSL, loc=mu_current, scale=sigma)
... print("Probability of lower limit=", P_LSL_current*100)
```

```
... P_USL_current = 1 - norm.cdf(USL, loc=mu_current, scale=sigma)
... print("Probability of upper limit=", P_USL_current*100)
...
Probability of lower limit= 6.680720126885807
Probability of upper limit= 15.865525393145708
```

So cost for one unit including scrap and rework

```
>>> cost_scrap = 5  # Cost per scrapped unit
... cost_rework = 1  # Cost per reworked unit
...
... cost_current = (P_LSL_current * cost_rework) + (P_USL_current * cost_scrap)
... print("current cost for one unit::", cost_current)
...
current cost for one unit:: 0.8600834709261435
```

Since scrapping cost is more, If we lower the mean slightly below 100, more units will fall within the lower specification limit, reducing expensive scrap.
If we make it 98

```
>>> mu_new = 98
... P_LSL_new = norm.cdf(LSL, loc=mu_new, scale=sigma)
... P_USL_new = 1 - norm.cdf(USL, loc=mu_new, scale=sigma)
...
... cost_new = (P_LSL_new * cost_rework) + (P_USL_new * cost_scrap)
... print("new cost for one unit::", cost_new)
...
new cost for one unit:: 0.4222881984668829
```

It's a significant reduce from .86 to .42228

8. Laura loves going to Vegas and playing Roulette. Each time she plays, she bets on the first column, which has a 12/38 = 31.58% chance of winning. On her next trip, she will play Roulette with this bet 50 times. Using the appropriate approximation, approximate the probability that she wins at least 12 times.

```
>>> n = 50  # Number of bets
... p = 12 / 38  # Probability of winning
... k = 11
...
... # Compute probability using binomial distribution directly
... probability_binom = 1 - binom.cdf(k, n, p)  # Calculating P(X > 11) which is P(X >=
12)
```

```
... print("Probability of winning at least 12 times ", probability_binom*100)
...
Probability of winning at least 12 times  90.68316477330218
```