1. The weights of steers in a herd are distributed normally. The variance is 40,000 and the mean steer weight is 1300 lbs. Find the probability that the randomly selected steer is greater than 979 lbs. (round your answer to 4 decimal places.)

We can use the property of the normal distribution to find the probability that the weight of a randomly selected steer is greater than 979 lbs. The weights are normally distributed with a mean of 1300 lbs and varience of 40,000. so, we can use Z-score formula to standardize the value of 979 and then find the probability. Z = (X - μ)/σ

X = 979 , μ = 1300, σ = √40000 = 200

```
from scipy.stats import norm
import math
X = 979
μ = 1300
σ = 200


Z = (X - μ)/σ

print(Z)
```

```
-1.605
```

Calculate the probability using the CDF of the standard normal distributuion

```
probability = 1 - norm.cdf(Z)
roundTo4 = round(probability, 4)
print("Probability: ", roundTo4)
```

```
Probability:  0.9458
```

2. SVGA monitors manufactured by TSI Electronics have life spans that have normal distribution with a variance of 1,960,000 and a mean life span of 11,000 hours. If a SVGA monitor is selected at random, find the probability that the life span of the monitor will be more than 8340 hours. (Round your answer to 4 decimal places)

Given values are: mean = 11000, variance = √1960000 ≈ 1400 X = 8340,

Z = (X - μ)/σ

```
import math
μ = 1100
σ = 1400

X = 8340

Z = (X - μ)/σ

print(Z)

probabilityN = 1- norm.cdf(Z)
roundTo4 = round(probabilityN, 4)
print("Probability: ", roundTo4)
```

```
5.171428571428572
Probability:  0.0
```

3. Suppose the mean income of firms in the industry for a year is 80 million dollars with a standard deviation of 3 million dollars. If incomes for the industry are distributed normally, what is the probability that a randomly selected firm will earn between 83 and 85 million dollars? (Round your answer to 4 decimal places)

Standard Deviation = 3 million Mean = 80 million First we need to calculate the Z-scores for the values 83 and 85 using the same formula as we used above. Z = (X - μ)/σ

```
std_d = 3
mean = 80
# we have 2 X values, those are 85 and 83
z1 = (83 - mean)/std_d
z2 = (85 - mean)/std_d
probability_1 = norm.cdf(z1)
probability_2 = norm.cdf(z2)

# now calculate the probability of being between 85 and 83
probability_between = probability_2 - probability_1

# round the result to 4 decimal places
rounded_4 = round(probability_between, 4)

#display the result
print("Probability is: ", rounded_4)
```

```
Probability is:  0.1109
```

4. Suppose GRE Verbal scores are normally distributed with a mean of 456 and a standard deviation of 123. A university plans to offer tutoring jobs to students whose scores are in the top 14%. What is the minimum score required for the job offer? Round your answer to the nearest whole number, if necessary.

mean = 456 std_dev = 123 percentile = 0.86 (100% - 86% = 14%)

now, we use Z-score formula to find the 86th percentile

Z = (X - μ)/σ

where, X is the value we want to find the Z-score for μ is Mean value of the distribution, and σ is the standard deviation

```
from scipy.stats import norm
mean = 456
std_dev = 123
percentile = 0.86
# find the Z score corresponding to the 86th percentile (Top 14 percent)

z_score = norm.ppf(percentile)

#calculate the minimum score required for the job offer
mini_score = mean + z_score * std_dev

#round it to nearest whole number
rounded_whole = round(mini_score)
print("Required minimum score for job is: ", rounded_whole)
```

```
Required minimum score for job is:  589
```

5. The lengths of nails produced in a factory are normally distributed with a mean of 6.13 centimeters and a standard deviation of 0.06 centimeters. Find the two lengths that separate the top 7% and the bottom 7%. These lengths could serve as limits used to identify which nails should be rejected. Round your answer to the nearest hundredth, if necessary.

Here, most essential part is to find the two lenghts top 7, and bottom 7th. 7th percentile (bottom), and 93rd percentile(top)

```
from scipy.stats import norm
# provided values
```

```
mean = 6.13
std_deviation = 0.06
percent_bottom = 0.07
percent_top = 0.93

#find the Z-scores corresponding to the 7th and 93rd percentile

z_7th = norm.ppf(percent_bottom)
z_93rd = norm.ppf(percent_top)

# Now, calculate the lengths that separate the bottom 7% and top 93%
length_at_7th = mean + z_7th * std_deviation
lenght_at_93rd = mean + z_93rd * std_deviation

# round them to the nearest 100th
rounded_100_for7th = round(length_at_7th, 2)
rounded_100_for93rd = round(lenght_at_93rd, 2)

# display
print("Length at 7th percentile in centimeters is: ", rounded_100_for7th)
print("Length at 93rd percentile in centimeters is: ", rounded_100_for93rd)
```

```
Length at 7th percentile in centimeters is:  6.04
Length at 93rd percentile in centimeters is:  6.22
```

6. An English professor assigns letter grades on a test according to the following scheme. A: Top 13% of scores B: Scores below the top 13% and above the bottom 55% C: Scores below the top 45% and above the bottom 20% D: Scores below the top 80% and above the bottom 9% F: Bottom 9% of scores Scores on the test are normally distributed with a mean of 78.8 and a standard deviation of 9.8. Find the numerical limits for a C grade. Round your answers to the nearest whole number, if necessary.

$Z = (X - \mu)/\sigma$

we need to find the Z-scores corresponding to the 45th and 80th percentile

```
from scipy.stats import norm

#provided values
mean = 78.8
std_dev = 9.8
percentile_45th = 0.45
percentile_80th = 0.80

#find the Z-zcores
zScore_45 = mean + percentile_45th * std_dev
zScore_80 = mean + percentile_80th * std_dev

#round them to nearest whole number
rounded_whole_45 = round(zScore_45)
rounded_whole_80 = round(zScore_80)

#display results
print("Numerical limit for grade C for 45th percentile is: ",rounded_whole_45)
print("Numerical limit for grade C for 80th percentile is: ",rounded_whole_80)
```

```
Numerical limit for grade C for 45th percentile is:  83
Numerical limit for grade C for 80th percentile is:  87
```

7. Suppose ACT Composite scores are normally distributed with a mean of 21.2
and a standard deviation of 5.4. A university plans to admit students whose scores are in the top 45%. What is the minimum score required for admission? Round your answer to the nearest tenth, if necessary.

We can use the Z-score formula to find the Z-score associated with the 55th percentile.

```
from scipy.stats import norm

#provided values
mean = 21.2
std_dev = 5.4
percentile_55th = 0.55     #100% - 45% = 55%

#find the Z-score corresponding to the 55th percentile
z_55th_percentile = norm.ppf(percentile_55th)

# calculate the minimmum score required for the admssion
min_score = mean + z_55th_percentile * std_dev

#round it to nearest 10th
rounded_min = round(min_score,1)

#display
print("Minimum score required for admission is: ", rounded_min)
```

```
Minimum score required for admission is:  21.9
```

8. Consider the probability that less than 11 out of 151 students will not graduate on time. Assume the probability that a given student will not graduate on time is 9%. Approximate the probability using the normal distribution. (Round your answer to 4 decimal places.)

First we find the mean and standard deviation using the normal approximation then we use cumulative distribution function cdf

```
from scipy.stats import norm

#provided values
n = 151 # number of students
p = 0.09 # probability of students not graduating on time
X = 11  # number of students not graduating on time

# Find the mean and standard deviation
mean = n * p
std_dev = (n * p * (1 - p)) ** 0.5

probability = norm.cdf(X, mean, std_dev)

#round the probability to 4th decimal places
rounded_p = round(probability, 4)

#display the result
print("Approximate probability that less than 11 out of 151 students will not graduate on time is: ", rounded_
```

```
Approximate probability that less than 11 out of 151 students will not graduate on time is:  0.2307
```

9. The mean lifetime of a tire is 48 months with a standard deviation of 7. If 147 tires are sampled, what is the probability that the mean of the sample would be greater than 48.83 months? (Round your answer to 4 decimal places)

using central limit theorem,

```
from scipy.stats import norm
#provided values
mean_popu = 48
std_dev_popu = 7
sample_size = 147
sample_mean = 48.83
```

```
#find standard deviation of the sampling distribution of the sample mean
std_dev_sample_mean = std_dev_popu / (sample_size ** 0.5)

Z = (sample_mean - mean_popu) / std_dev_sample_mean
# use the Cumulative distribution function
probability = 1 - norm.cdf(Z)
# round the probability to 4 decimal places

rounded_prob = round(probability,4)

#display the result
print("Probability that the mean of the sample would be greater than 48.83 months: ", rounded_prob)
```

```
Probability that the mean of the sample would be greater than 48.83 months:  0.0753
```

10. The quality control manager at a computer manufacturing company believes that the mean life of a computer is 91 months, with a standard deviation of 10. If he is correct, what is the probability that the mean of a sample of 68 computers would be greater than 93.54 months? (Round your answer to 4 decimal places)

```
from scipy.stats import norm
# provided values
mean_population = 91
std_dev_population = 10
sample_size = 68
sample_mean = 93.54

#calculate the standard deviation of the saampling distribution of the sample mean
std_dev_sample_mean1 = std_dev_population / (sample_size ** 0.5)

# find the Z-score
Z = (sample_mean - mean_population) / std_dev_sample_mean1

# use cumulative distribution function cdf
probability = 1 - norm.cdf(Z)
# round it to 4 decimal places
rounded4 = round(probability,4)
#display result
print("Probability that the mean of a 68 computers would be greater than 93.54 months: ", rounded4)
```

```
Probability that the mean of a 68 computers would be greater than 93.54 months:  0.0181
```

11. A director of reservations believes that 7% of the ticketed passengers are no-shows. If the director is right, what is the probability that the proportion of no-shows in a sample of 540 ticketed passengers would differ from the population proportion by less than 3%? (Round your answer to 4 decimal places)

Double-click (or enter) to edit

```
from scipy.stats import norm
#provided values
population_p = 0.07
sample_n = 540
error_margin  = 0.03

# calculate standard deviation
std_dev_samProportion = (population_p * (1 - population_p) / sample_n) ** 0.5
# calculate the Z-zcores for upper and lower bounds
Z_upper = (population_p + error_margin) / std_dev_samProportion
Z_lower = (population_p - error_margin) / std_dev_samProportion
```

```
#probability
probability = norm.cdf(Z_upper) - norm.cdf(Z_lower)

#round the probability to 4 decimal places
rounded_to_4 = round(probability, 4)

#display the result
print("Probability that the proportion of no-shows in a sample of 540 ticketed passengers would differ from tl
```

```
Probability that the proportion of no-shows in a sample of 540 ticketed passengers would differ from the population by less than 3% is:
```

12. A bottle maker believes that 23% of his bottles are defective. If the bottle maker is accurate, what is the probability that the proportion of defective bottles in a sample of 602 bottles would differ from the population proportion by greater than 4%? (Round your answer to 4 decimal places)

```
from scipy.stats import norm
#provided values
p_pop = 0.23
n_sample = 602
margin_error = 0.04

#calculate the standard deviation
std_dev_Prop_sample = (p_pop * (1 - p_pop) / n_sample) ** 0.5
# calculate the Z scores for the upper and lower bounds
Z_upper = (p_pop + margin_error) / std_dev_Prop_sample
Z_lower = (p_pop - margin_error) / std_dev_Prop_sample
# calculate the probability using cumulative distribution function
probability = norm.cdf(Z_lower) + (1 - norm.cdf(Z_upper))
#round the probability to 4 decimal places
rounded_to4 = round(probability, 4)
#display the results
print("Probability that the proportion of defective bottles in a sample of 602 bottles would difer from the pc
```

```
Probability that the proportion of defective bottles in a sample of 602 bottles would difer from the population by greater than 4% is:
```

13. A research company desires to know the mean consumption of beef per week among males over age 48. Suppose a sample of size 208 is drawn with x ̄ = 3.9. Assume ® = 0.8 . Construct the 80% confidence interval for the mean number of lb. of beef per week among males over 48. (Round your answers to 1 decimal place)

we can use the formula for the confidence interval for the population mean

```
from scipy.stats import norm
import scipy.stats as stats
#provided values
sample_mean = 3.9
pop_std_dev = 0.8
sample_size = 208
confidence_level = 0.80

#calculate the critical Z score for the given confidence level
Z = stats.norm.ppf((1 + confidence_level) / 2)
#calculate margin of error
error_margin2 = Z * (pop_std_dev / (sample_size ** 0.5))
#calculate the lower and upper bounds of the confidence interval
upper_b = sample_mean + error_margin2
lower_b = sample_mean - error_margin2
#round the bounds to 1 decimal place
rounded_to_lower = round(lower_b,1)
```

```
rounded_to_upper = round(upper_b, 1)
print("Lower bound is: ", rounded_to_lower)
print("Upper bound is: ", rounded_to_upper)
```

```
Lower bound is:  3.8
Upper bound is:  4.0
```

14. An economist wants to estimate the mean per capita income (in thousands of dollars) in a major city in California. Suppose a sample of size 7472 is drawn with x̄ = 16.6. Assume ® = 11 . Construct the 98% confidence interval for the mean per capita income. (Round your answers to 1 decimal place)

```
from scipy.stats import norm
import scipy.stats as stats

# given values
sample_mean = 16.6
pop_std_dev = 11
samp_size = 7472
confidence_level = 0.98
# calculate the critical Z values for the given confidence level
Z = stats.norm.ppf((1 + confidence_level) / 2)

#calculate the margin of error
error_margin3 = Z * (pop_std_dev / (samp_size ** 0.5))

#calculate the lower and upper bounds of the confidence interval
lower_b1 = sample_mean - error_margin3
upper_b1 = sample_mean + error_margin3

#round the bounds to 1 decimal place
round_lower = round(lower_b1,1)
round_upper = round(upper_b1,1)
#display the results
print("Upper bound is: ", round_upper)
print("Lower bound is: ", round_upper)
```

```
Upper bound is:  16.9
Lower bound is:  16.9
```

Double-click (or enter) to edit

16.

```
import math
#given value
sample_data = [383.6, 347.1, 371.9, 347.6, 325.8, 337]
#step 1: calculate the mean
sampleMean = sum(sample_data)/ len(sample_data)
rounded_sample_mean = round(sampleMean,2)
#step 2: calculate the sample standard deviation
sum_squared_diff = sum((x - sampleMean)**2 for x in sample_data)
sample_std_dev = math.sqrt(sum_squared_diff / (len(sample_data) - 1))
round_it = round(sample_mean,2)

# step 3: Find the critical value for a 90% confidence level and 5 degrees of freedom (n - 1)
confidence_level_ = 0.90
degrees_of_freedom = len(sample_data) - 1
critical_value = 2.571
```

```
#step 4: Construct the 90% confidence level
confidence_inter_margin = critical_value * (sample_std_dev / math.sqrt(len(sample_data)))
lower_limit = sample_mean - confidence_inter_margin
upper_limit = sample_mean + confidence_inter_margin
round_lower_limit = round(lower_limit,2)
round_upper_limit = round(upper_limit,2)
# display the results
print("Step 1: Sample mean = ", sample_mean)
print("Step 2: sample standard Deviation = ", sample_std_dev)
print("Step 3: Critical value = ", critical_value)
print("Step 4: 90% Confidence Interval = ", (round_lower_limit,round_upper_limit))
```

```
Step 1: Sample mean =  16.6
Step 2: sample standard Deviation =  21.675854462204402
Step 3: Critical value =  2.571
Step 4: 90% Confidence Interval =  (-6.15, 39.35)
```

17. A random sample of 16 fields of spring wheat has a mean yield of 46.4 bushels per acre and standard deviation of 2.45 bushels per acre. Determine the 80% confidence interval for the true mean yield. Assume the population is normally distributed.

    Step 1. Find the critical value that should be used in constructing the confidence interval. (Round answer to 3 decimal places)

Step 2. Construct the 80% confidence interval. (Round answer to 1 decimal place)

```
import math
# given sample data
sample_mean = 46.4
sample_std_dev = 2.45
sample_size = 16

# Step 1: Find the critical value for an 80% confidence level and 15 degrees of freedom (n - 1)
confidence_level = 0.80
degrees_of_freedom = sample_size - 1
critical_value = 1.753

# Step 2: Construct the 80% confidence interval
confidence_inter_margin = critical_value * (sample_std_dev / math.sqrt(sample_size))
lower_limit = sample_mean - confidence_inter_margin
upper_limit = sample_mean + confidence_inter_margin
rounded_upper = round(upper_limit,1)
rounded_lower = round(lower_limit,1)
#display the results
print("Step 1: Critical value = ", critical_value)
print("Step 2: 80% Confidence Interval = ", (rounded_upper, rounded_lower))
```

```
Step 1: Critical value =  1.753
Step 2: 80% Confidence Interval =  (47.5, 45.3)
```

18. A toy manufacturer wants to know how many new toys children buy each year.
    She thinks the mean is 8 toys per year. Assume a previous study found the standard deviation to be 1.9. How large of a sample would be required in order to estimate the mean number of toys bought per child at the 99% confidence level with an error of at most 0.13 toys? (Round your answer up to the next integer)

```
import math
#given values
mean = 8
std_dev1 = 1.9
confidence_level_1 = 0.99
error_margin_1 = 0.13
# calculate the Z score (99% of confidence level is aproximately 2.576 )
z_score = 2.576
```

```
#Calculate the sample size using the formula
sample_size = math.ceil((z_score * std_dev1 / error_margin_1)** 2)
# display the result
print("The required sample size is: ", sample_size)
```

```
The required sample size is:  1418
```

19. A research scientist wants to know how many times per hour a certain strand of bacteria reproduces. He believes that the mean is 12.6. Assume the variance is known to be 3.61. How large of a sample would be required in order to estimate the mean number of reproductions per hour at the 95% confidence level with an error of at most 0.19 reproductions? (Round your answer up to the next integer)

```
import math
#given data
mean_popu = 12.6
variance_popu = 3.61
confidence_level = 0.95
margin_ofError = 0.19

#step 1: find the Z-score for the desired confidence level
z_score = 1.96 #This value can be found in the standard normal distribution table

#step 2: calculate the sample size
sample_size = (z_score *math.sqrt(variance_popu) / margin_ofError) ** 2
rounded_sample_size = math.ceil(sample_size) # round up to the next integer
#display the result
print("The required smple size is: ", sample_size)
```

```
The required smple size is:  384.1599999999999
```

20. The state education commission wants to estimate the fraction of tenth grade students that have reading skills at or below the eighth grade level.

    Step 1. Suppose a sample of 2089 tenth graders is drawn. Of the students sampled, 1734 read above the eighth grade level. Using the data, estimate the proportion of tenth graders reading at or below the eighth grade level. (Write your answer as a fraction or a decimal number rounded to 3 decimal places)

Step 2. Suppose a sample of 2089 tenth graders is drawn. Of the students sampled, 1734 read above the eighth grade level. Using the data, construct the 98% confidence interval for the population proportion of tenth graders reading at or below the eighth grade level. (Round your answers to 3 decimal places)

```
import math
# given data
student_above_8 = 1734
total_sample_size = 2089
confi_level = 0.98

#Step 1: Calculate the sample proportion
sample_proportion = student_above_8 / total_sample_size
# Display the estimated proportion
print("Estimated proportion of 10th graders reading ar or below the 8th grade level: ", round(sample_proportic

# Step 2: Construct the confidence interval
z_score = 2.33 # For a 96% confidence level
margin_of_error = z_score * math.sqrt((sample_proportion * (1 - sample_proportion)) / total_sample_size)

#Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_proportion - margin_of_error
upper_bound = sample_proportion + margin_of_error

#Display the confidence interval
```

```
#Display the confidence interval
print("98% Confidence Interval is: " , (round(lower_bound,3), round(upper_bound, 3)))
```

```
Estimated proportion of 10th graders reading ar or below the 8th grade level:  0.83
98% Confidence Interval is:  (0.811, 0.849)
```

21. An environmentalist wants to find out the fraction of oil tankers that have spills each month.

    Step 1. Suppose a sample of 474 tankers is drawn. Of these ships, 156 had spills. Using the data, estimate the proportion of oil tankers that had spills. (Write your answer as a fraction or a decimal number rounded to 3 decimal places)

    Step 2. Suppose a sample of 474 tankers is drawn. Of these ships, 156 had spills. Using the data, construct the 95% confidence interval for the population proportion of oil tankers that have spills each month. (Round your answers to 3 decimal places)

```
import math
# Given data
tankers_with_spills = 156
total_sample_size = 474
confidence_level = 0.95
#Step 1: Calculate the sample proportion
sample_proportion = tankers_with_spills / total_sample_size

#Display the estimated proportion
print("Estimated proportion of oil tankers that had spills: ", round(sample_proportion, 3))

#Step 2: Construct the confidence interval
z_score = 1.96 # For a 95% confidence level we can find the value from standard normal distribution table.
margin_of_Error = z_score * math.sqrt((sample_proportion * (1 - sample_proportion)) / total_sample_size)
# calculate the lower and upper bounds of the confidence interval
lower_bound = sample_proportion - margin_of_Error
upper_bound = sample_proportion + margin_of_Error
#Display the confidence interval
print("95% Confidence Interval is: ", (round(lower_bound,3), round(upper_bound, 3)))
```

```
Estimated proportion of oil tankers that had spills:  0.329
95% Confidence Interval is:  (0.287, 0.371)
```