CSCI 443: LECTURE 17
HW4 REVIEW

Professor David Harrison

OFFICE HOURS

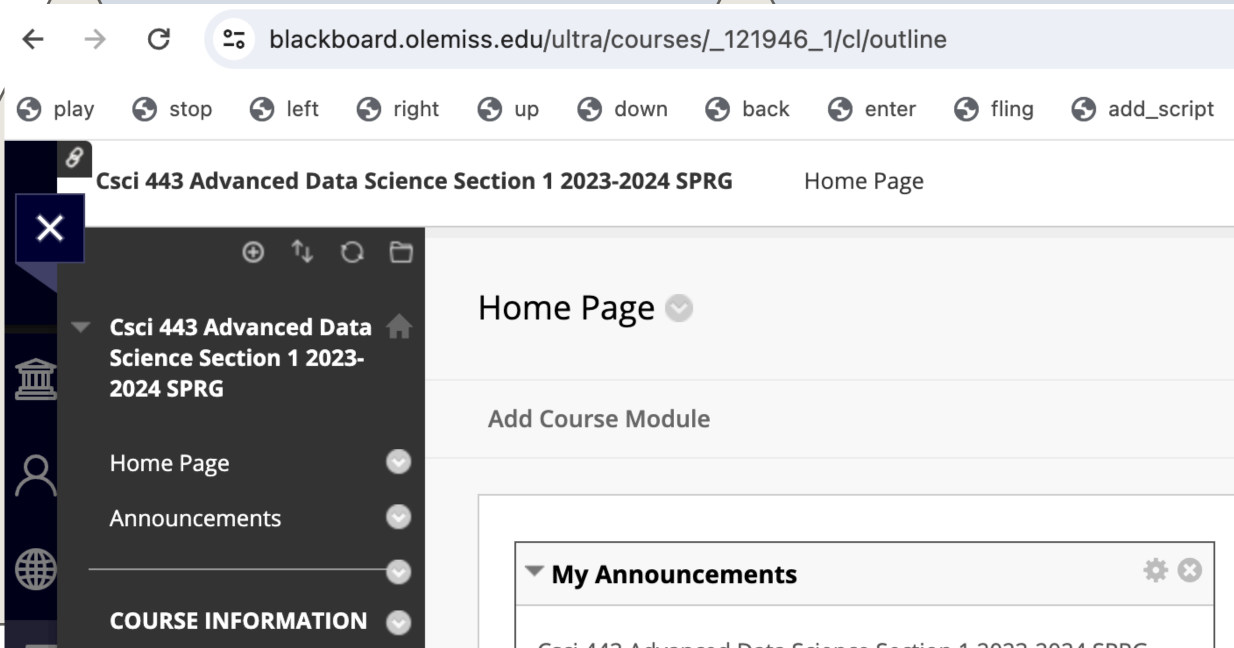
Tuesday	4:00-5:00 PM
Wednesday	12:30-2:30 PM

BLACKBOARD & GITHUB

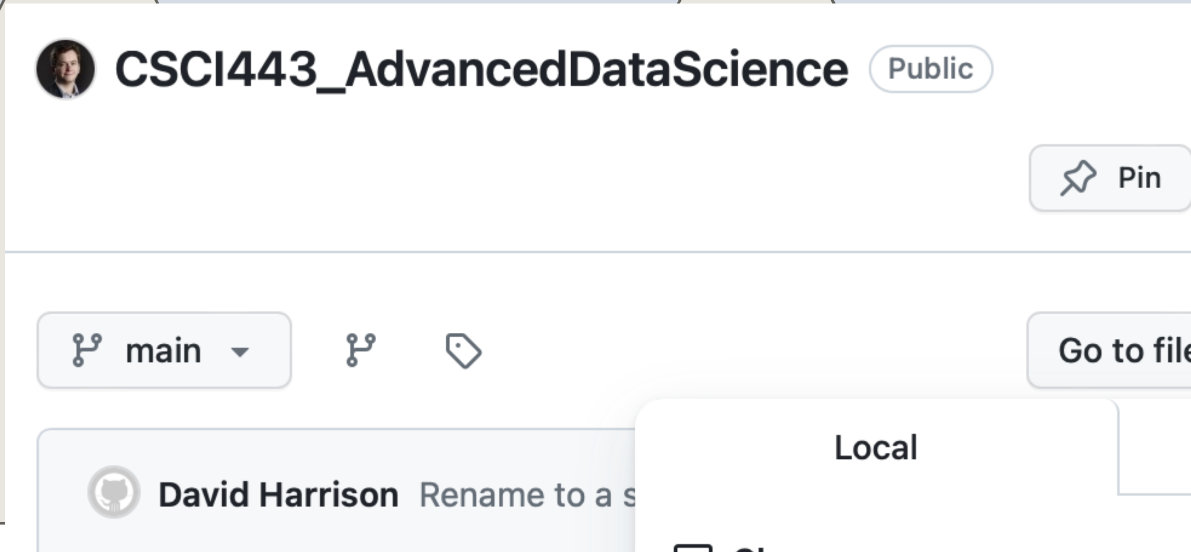
Slides up, handwritten notes AND a jupyter notebook for lecture 16 are on blackboard and in GitHub.

The project is at

https://github.com/dosirrah/CSCI443_AdvancedDataScience

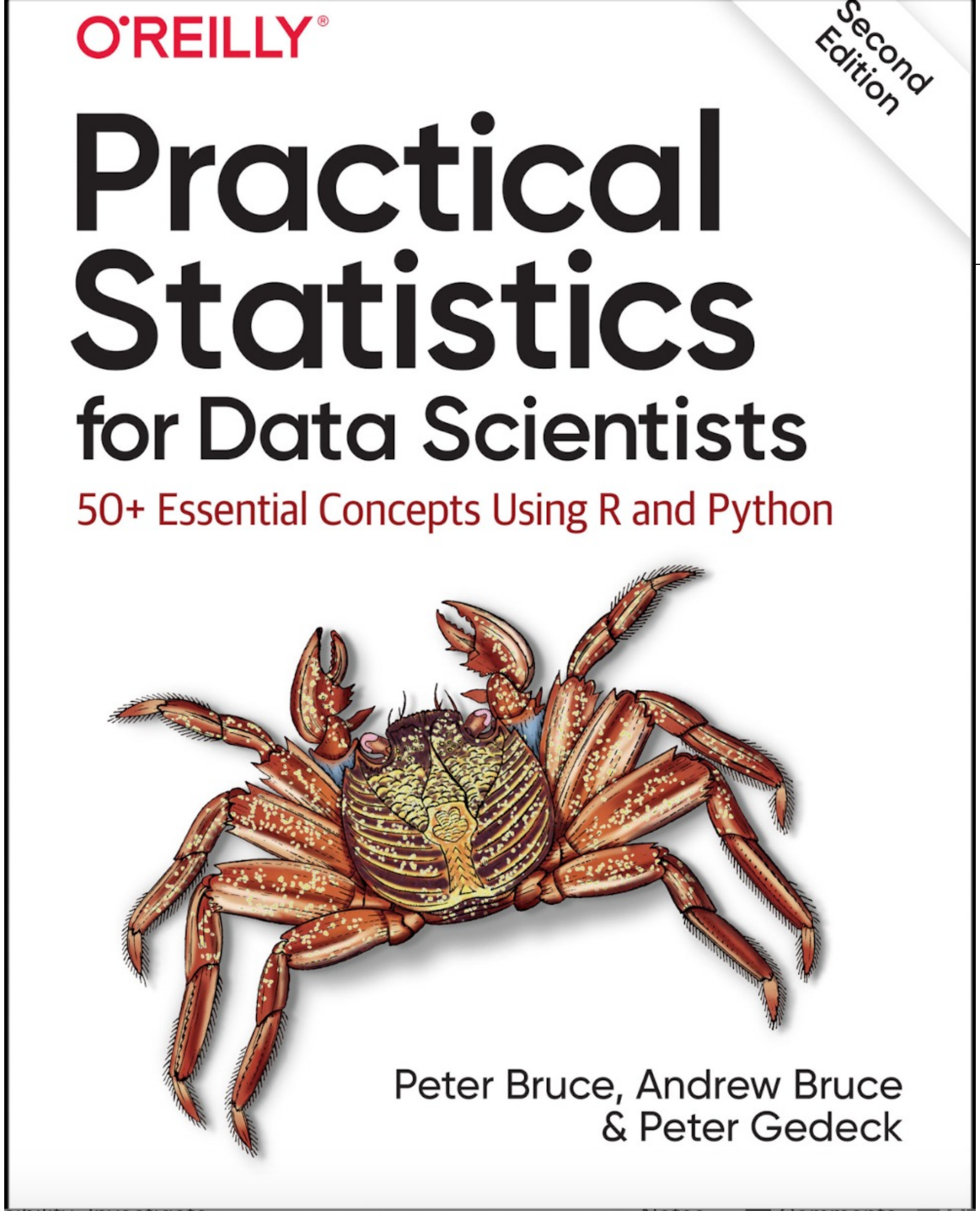


blackboard.olemiss.edu/ultra/courses/_121946_1/cl/outline



READ ABOUT

- chapter 3: experiments, hypothesis testing
  - Degrees of Freedom
  - Permutation Tests
  - ANOVA
  - Chi-square
  - Multi-arm bandit



O'REILLY®

Second Edition

Practical Statistics for Data Scientists

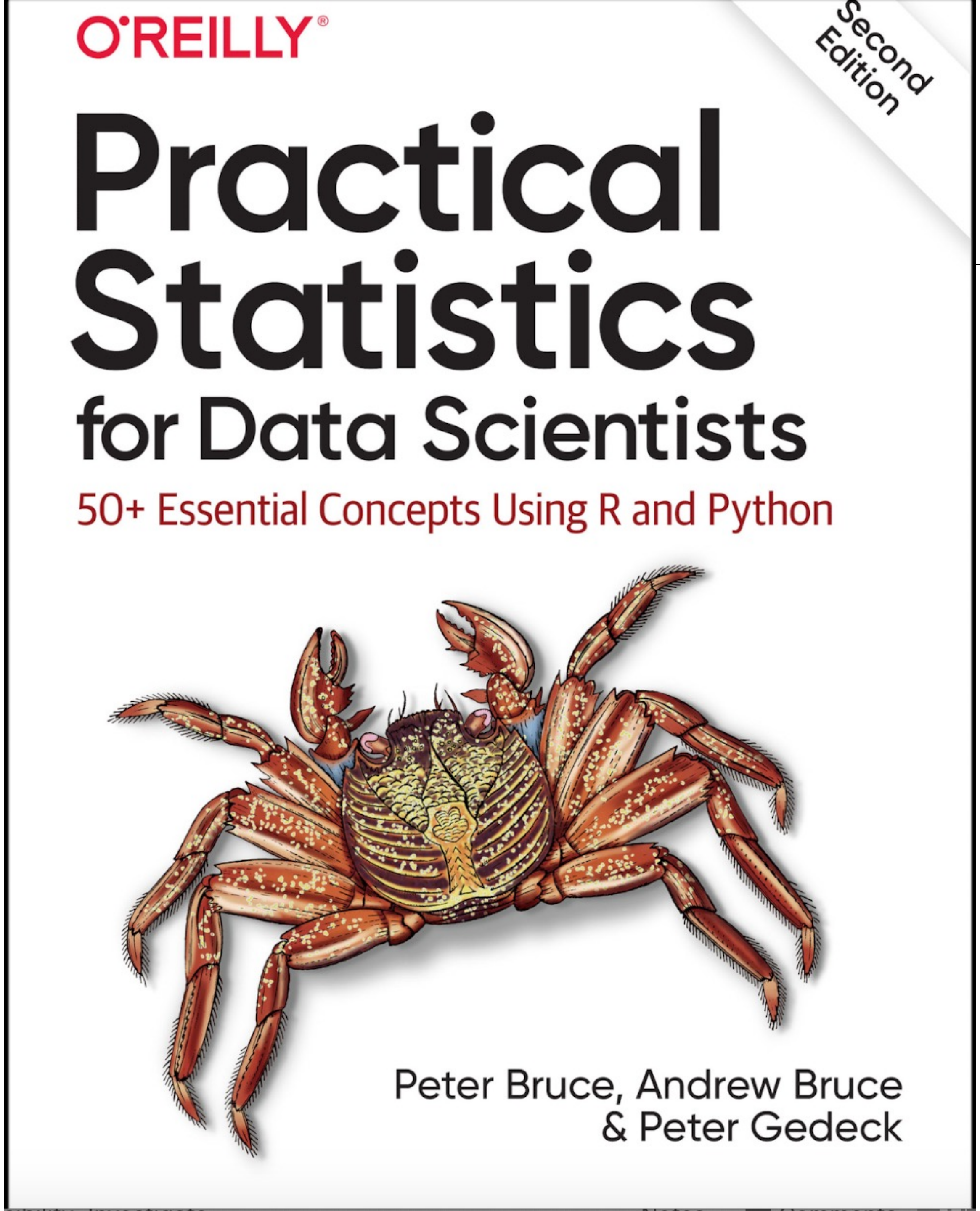
50+ Essential Concepts Using R and Python

Peter Bruce, Andrew Bruce & Peter Gedeck

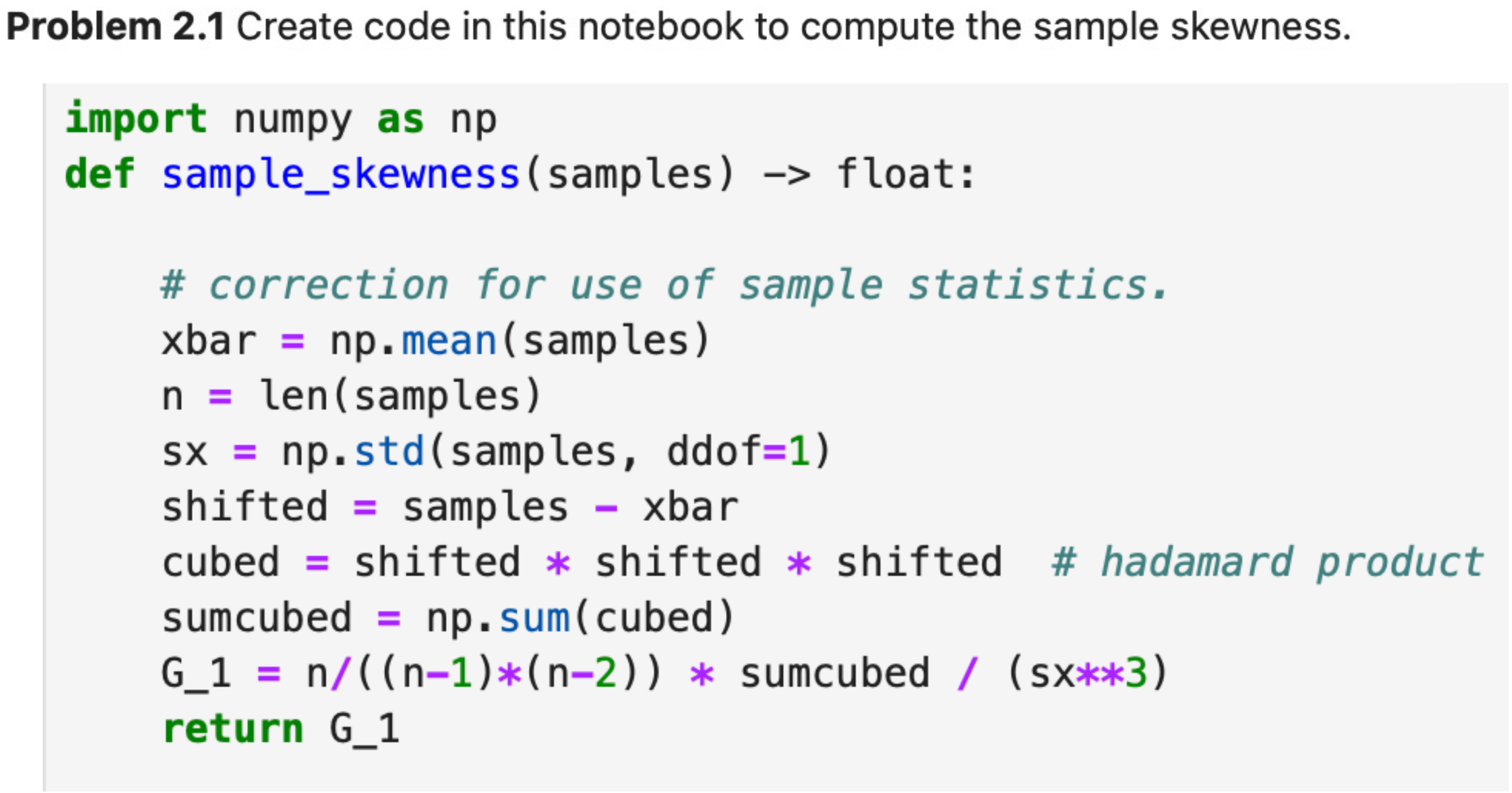
THINGS I WANT TO COVER TODAY

- HW4
- Holdouts and cross-validation in the context of multiple hypotheses

- Permutation tests for hypothesis testing.
- ANOVA


O'REILLY
Second Edition
Practical Statistics for Data Scientists
50+ Essential Concepts Using R and Python
Peter Bruce, Andrew Bruce & Peter Gedeck

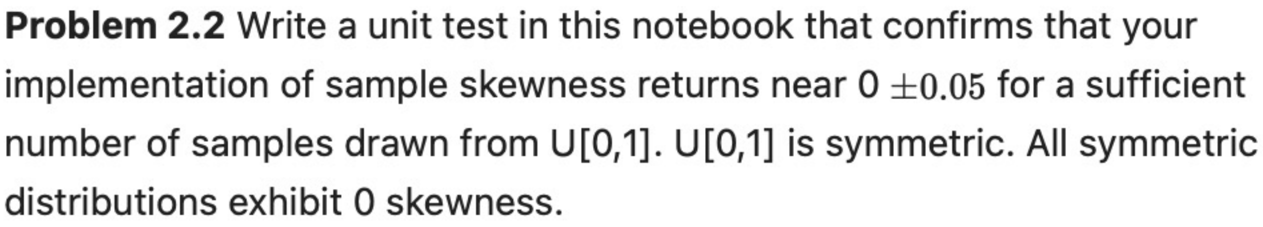
HW4 PROBLEM 2.1



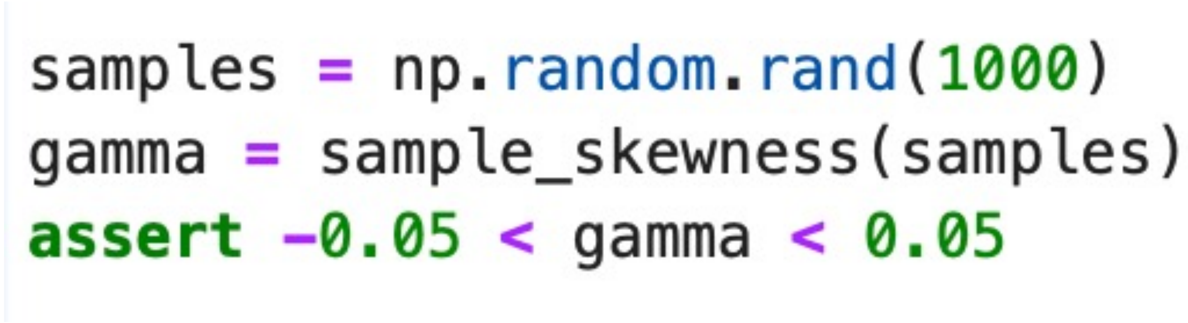
Problem 2.1 Create code in this notebook to compute the sample skewness.

```
import numpy as np
def sample_skewness(samples) -> float:

    # correction for use of sample statistics.
    xbar = np.mean(samples)
    n = len(samples)
    sx = np.std(samples, ddof=1)
    shifted = samples - xbar
    cubed = shifted * shifted * shifted  # hadamard product
    sumcubed = np.sum(cubed)
    G_1 = n/((n-1)*(n-2)) * sumcubed / (sx**3)
    return G_1
```

HW4 PROBLEM 2.2

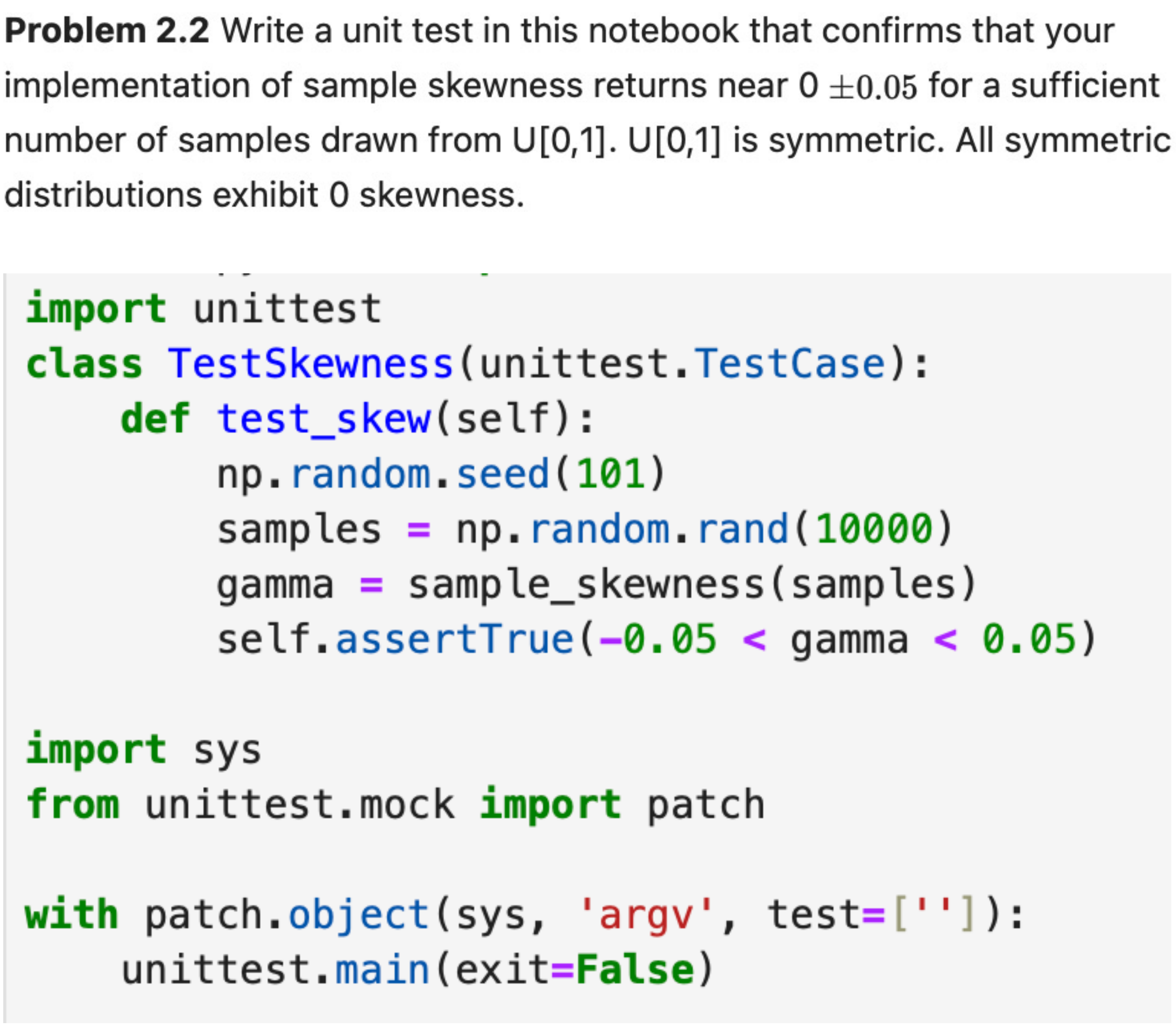


Problem 2.2 Write a unit test in this notebook that confirms that your implementation of sample skewness returns near 0 ± 0.05 for a sufficient number of samples drawn from U[0,1]. U[0,1] is symmetric. All symmetric distributions exhibit 0 skewness.



```
samples = np.random.rand(1000)
gamma = sample_skewness(samples)
assert -0.05 < gamma < 0.05
```

HW4 PROBLEM 2.2

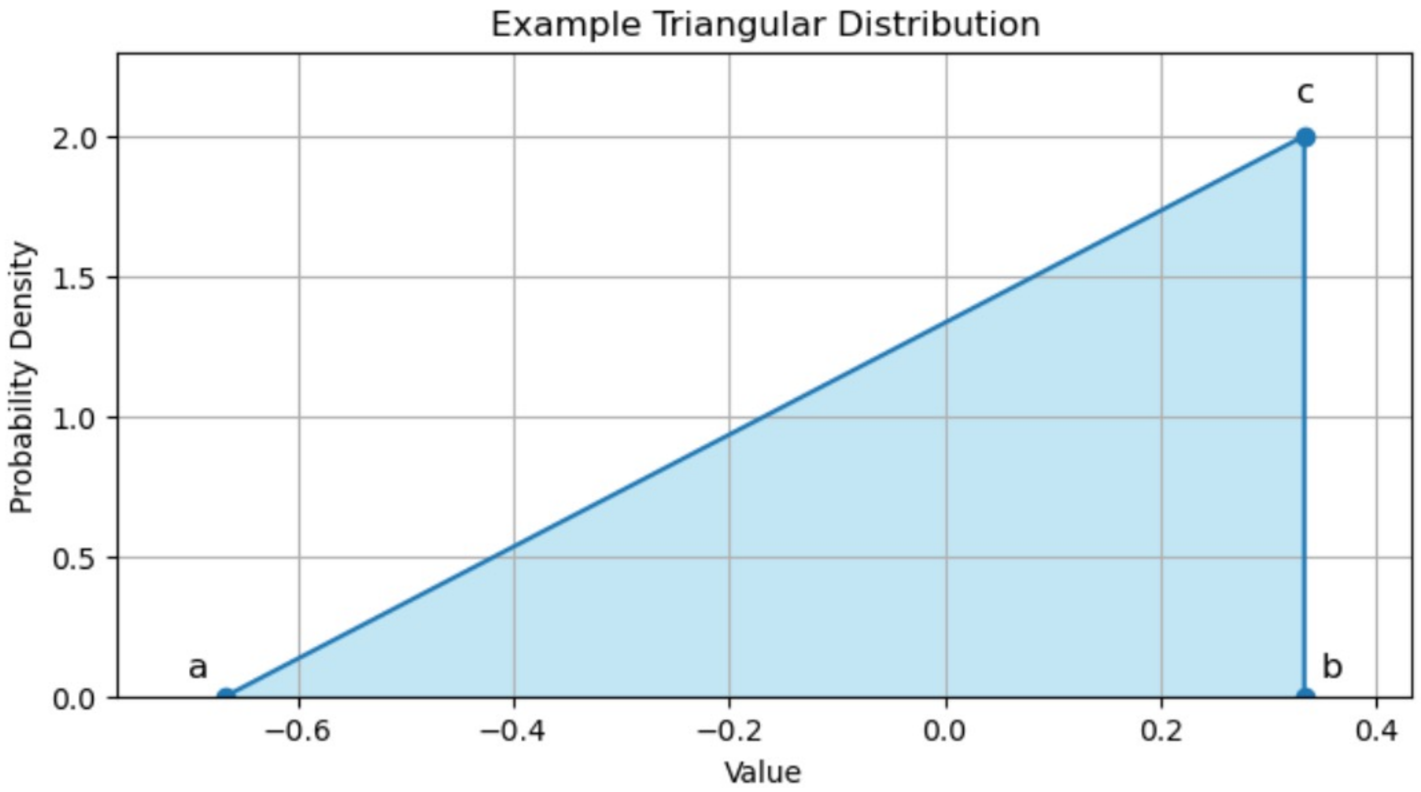


Problem 2.2 Write a unit test in this notebook that confirms that your implementation of sample skewness returns near 0 ± 0.05 for a sufficient number of samples drawn from U[0,1]. U[0,1] is symmetric. All symmetric distributions exhibit 0 skewness.

```
import unittest
class TestSkewness(unittest.TestCase):
    def test_skew(self):
        np.random.seed(101)
        samples = np.random.rand(10000)
        gamma = sample_skewness(samples)
        self.assertTrue(-0.05 < gamma < 0.05)

import sys
from unittest.mock import patch

with patch.object(sys, 'argv', test=['']):
    unittest.main(exit=False)
```

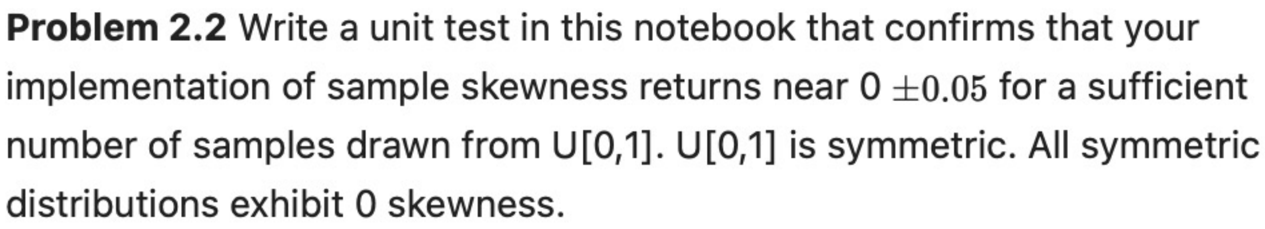
HW4 PROBLEM 2.4

Problem 2.2 Write a unit test in this notebook that confirms that your implementation of sample skewness returns near 0 ± 0.05 for a sufficient number of samples drawn from U[0,1]. U[0,1] is symmetric. All symmetric distributions exhibit 0 skewness.

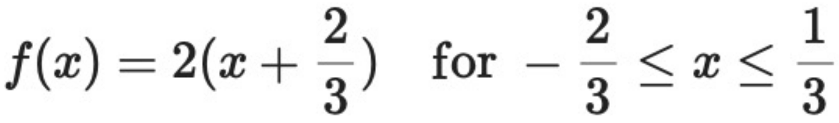


$$f(x) = 2(x + \frac{2}{3}) \quad \text{for } -\frac{2}{3} \le x \le \frac{1}{3}$$

HW4 PROBLEM 2.4



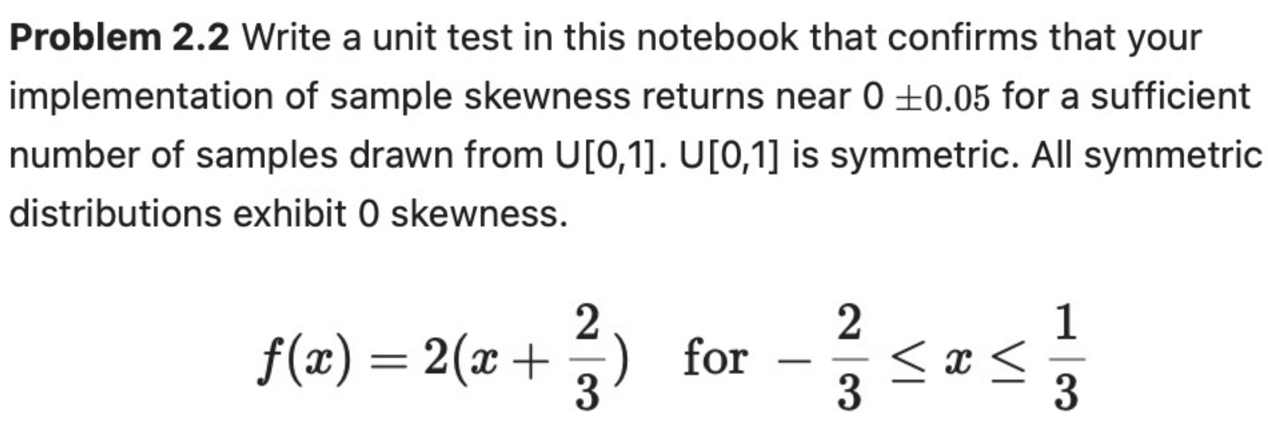
Problem 2.2 Write a unit test in this notebook that confirms that your implementation of sample skewness returns near 0 ± 0.05 for a sufficient number of samples drawn from U[0,1]. U[0,1] is symmetric. All symmetric distributions exhibit 0 skewness.



$$f(x) = 2(x + \frac{2}{3}) \quad \text{for } -\frac{2}{3} \leq x \leq \frac{1}{3}$$

See hw4_answers notebook

HW4 PROBLEM 2.4

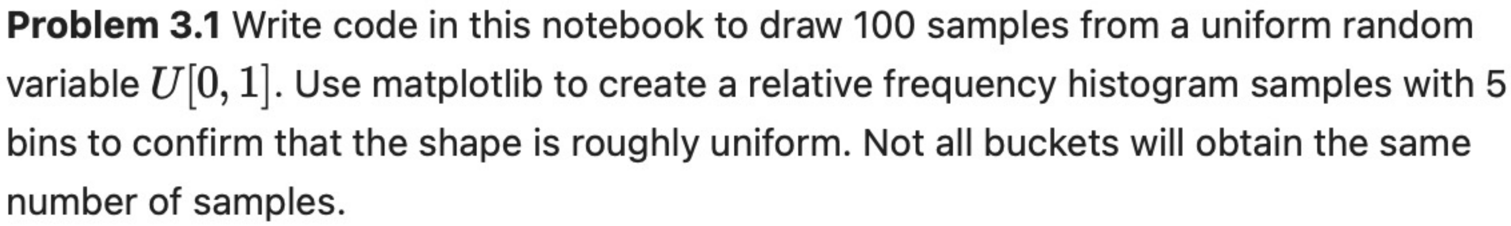


Problem 2.2 Write a unit test in this notebook that confirms that your implementation of sample skewness returns near 0 ± 0.05 for a sufficient number of samples drawn from U[0,1]. U[0,1] is symmetric. All symmetric distributions exhibit 0 skewness.

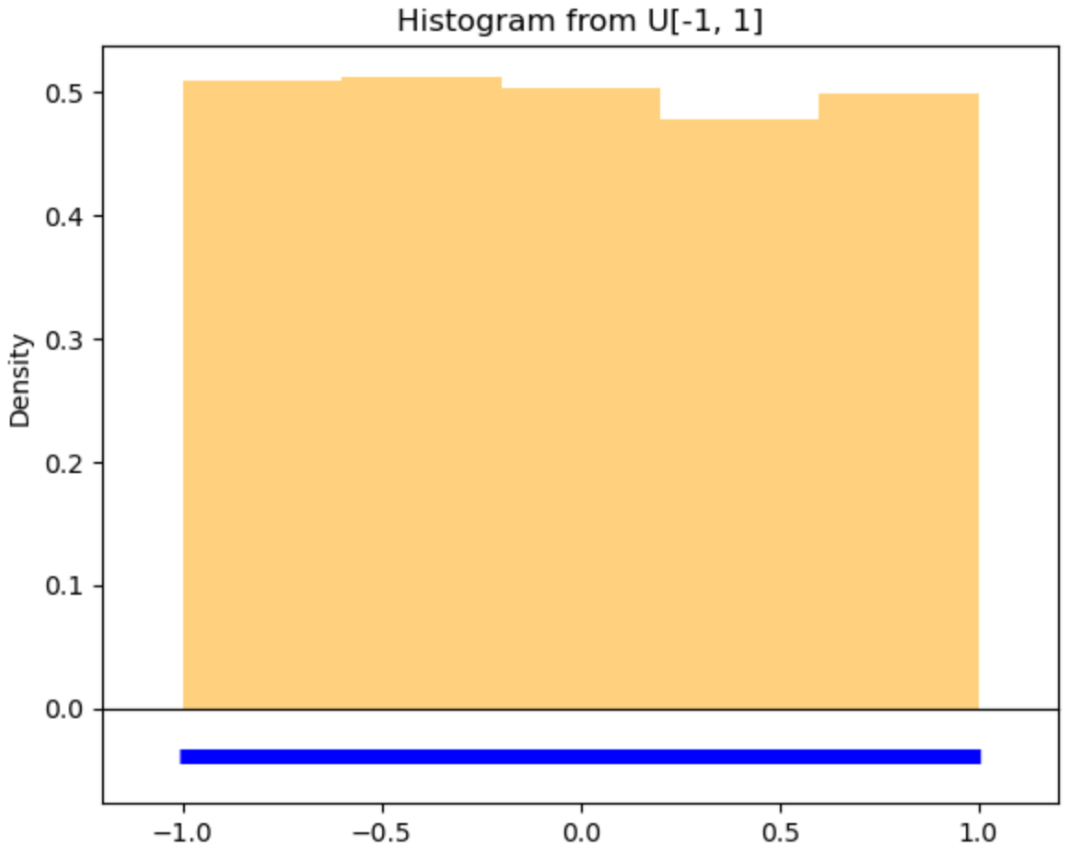
$$f(x) = 2(x + \frac{2}{3}) \quad \text{for} \ -\frac{2}{3} \le x \le \frac{1}{3}$$

See hw4_answers notebook. I wrote the answer on the board.

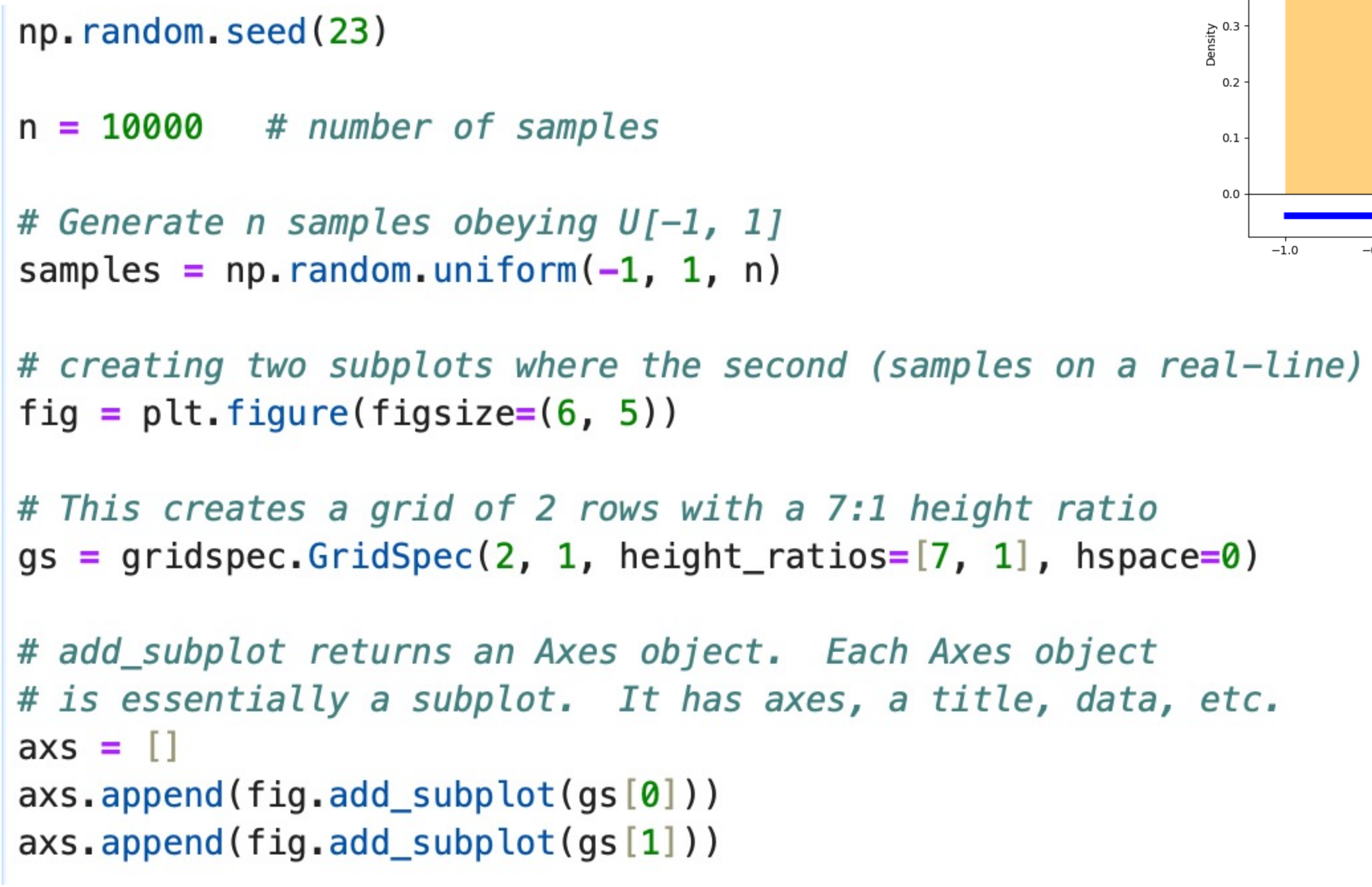
HW4 PROBLEM 3.1



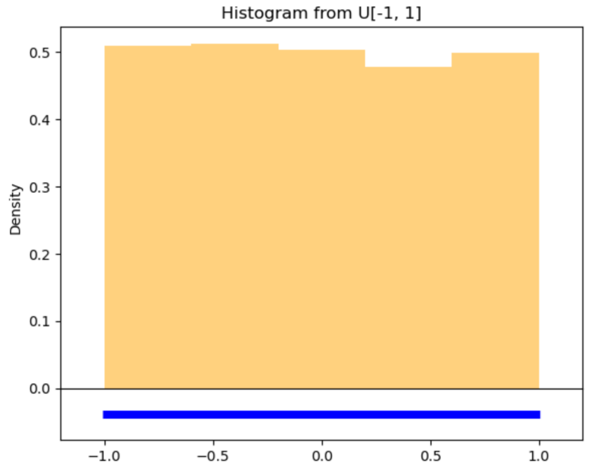
Problem 3.1 Write code in this notebook to draw 100 samples from a uniform random variable $U[0, 1]$. Use matplotlib to create a relative frequency histogram samples with 5 bins to confirm that the shape is roughly uniform. Not all buckets will obtain the same number of samples.



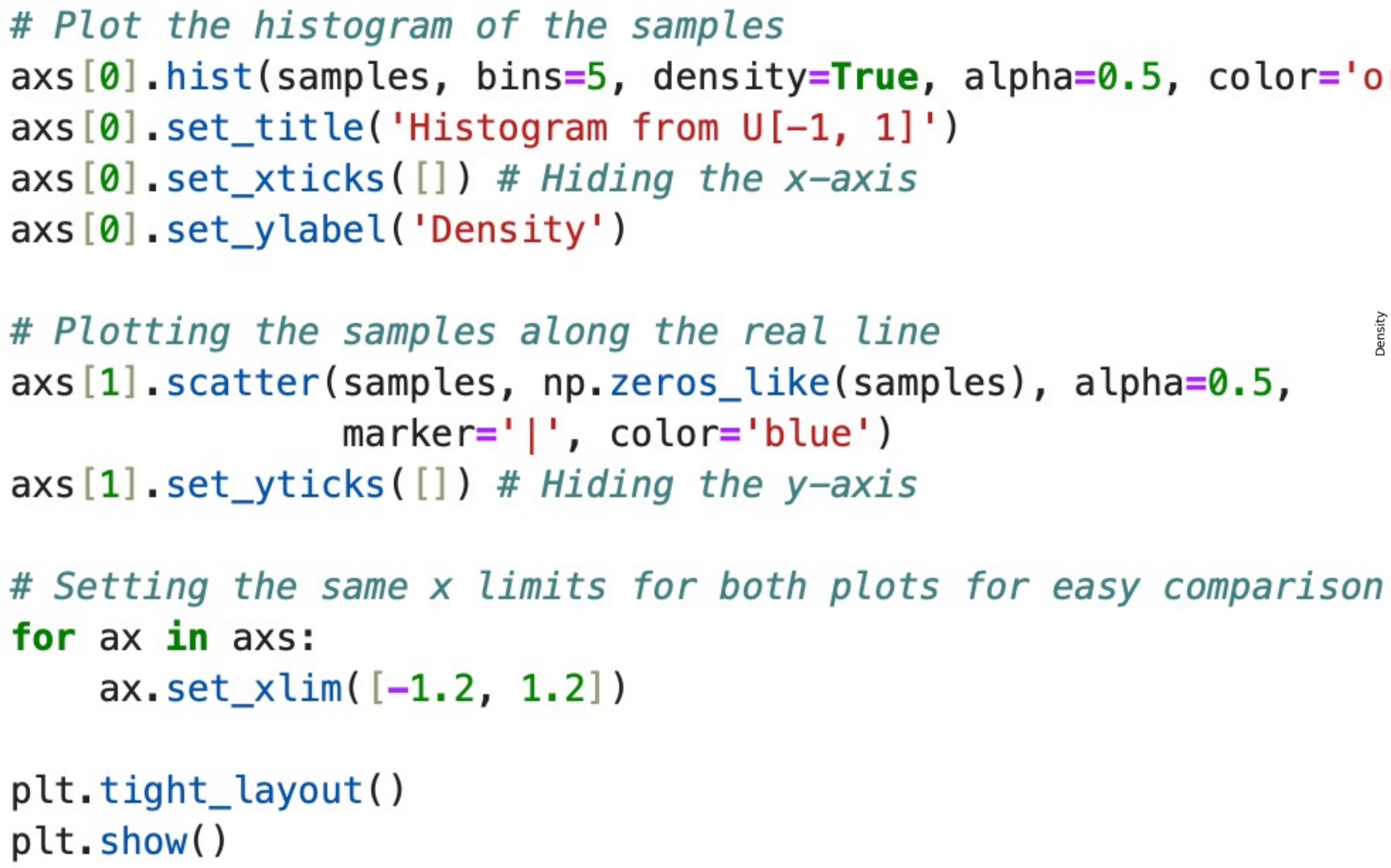
HW4 PROBLEM 3.1(CONT.)



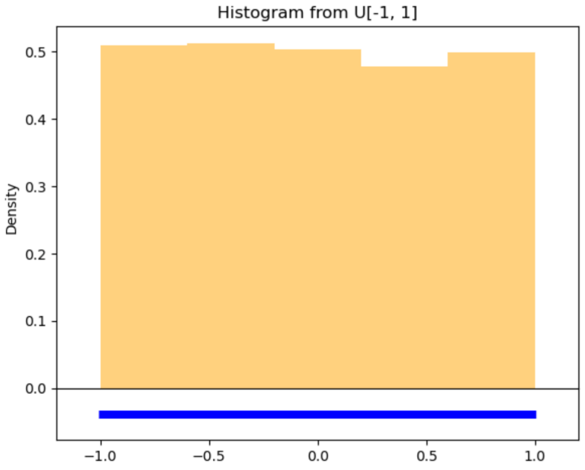
```
np.random.seed(23)

n = 10000   # number of samples

# Generate n samples obeying U[-1, 1]
samples = np.random.uniform(-1, 1, n)

# creating two subplots where the second (samples on a real-line)
fig = plt.figure(figsize=(6, 5))

# This creates a grid of 2 rows with a 7:1 height ratio
gs = gridspec.GridSpec(2, 1, height_ratios=[7, 1], hspace=0)

# add_subplot returns an Axes object.  Each Axes object
# is essentially a subplot.  It has axes, a title, data, etc.
axs = []
axs.append(fig.add_subplot(gs[0]))
axs.append(fig.add_subplot(gs[1]))
```



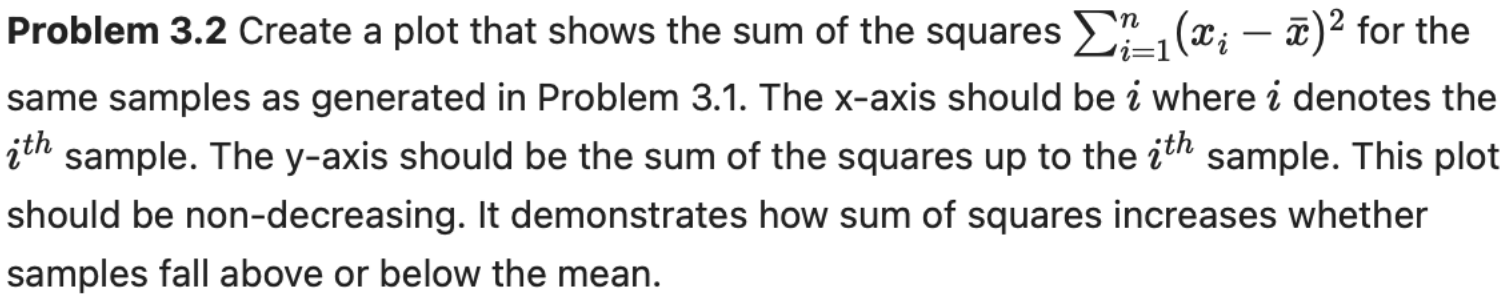
HW4 PROBLEM 3.1(CONT.)



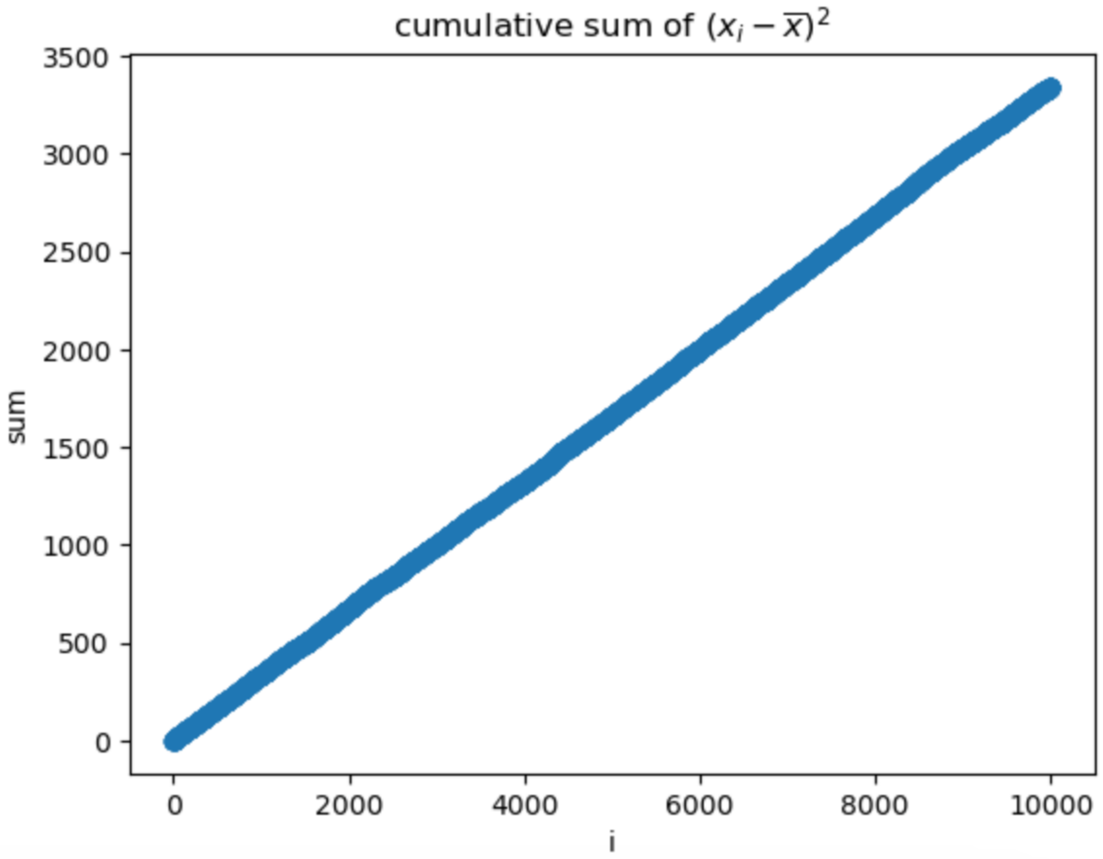
```python
# Plot the histogram of the samples
axs[0].hist(samples, bins=5, density=True, alpha=0.5, color='orange')
axs[0].set_title('Histogram from U[-1, 1]')
axs[0].set_xticks([]) # Hiding the x-axis
axs[0].set_ylabel('Density')

# Plotting the samples along the real line
axs[1].scatter(samples, np.zeros_like(samples), alpha=0.5,
               marker='|', color='blue')
axs[1].set_yticks([]) # Hiding the y-axis

# Setting the same x limits for both plots for easy comparison
for ax in axs:
    ax.set_xlim([-1.2, 1.2])

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



HW4 PROBLEM 3.2

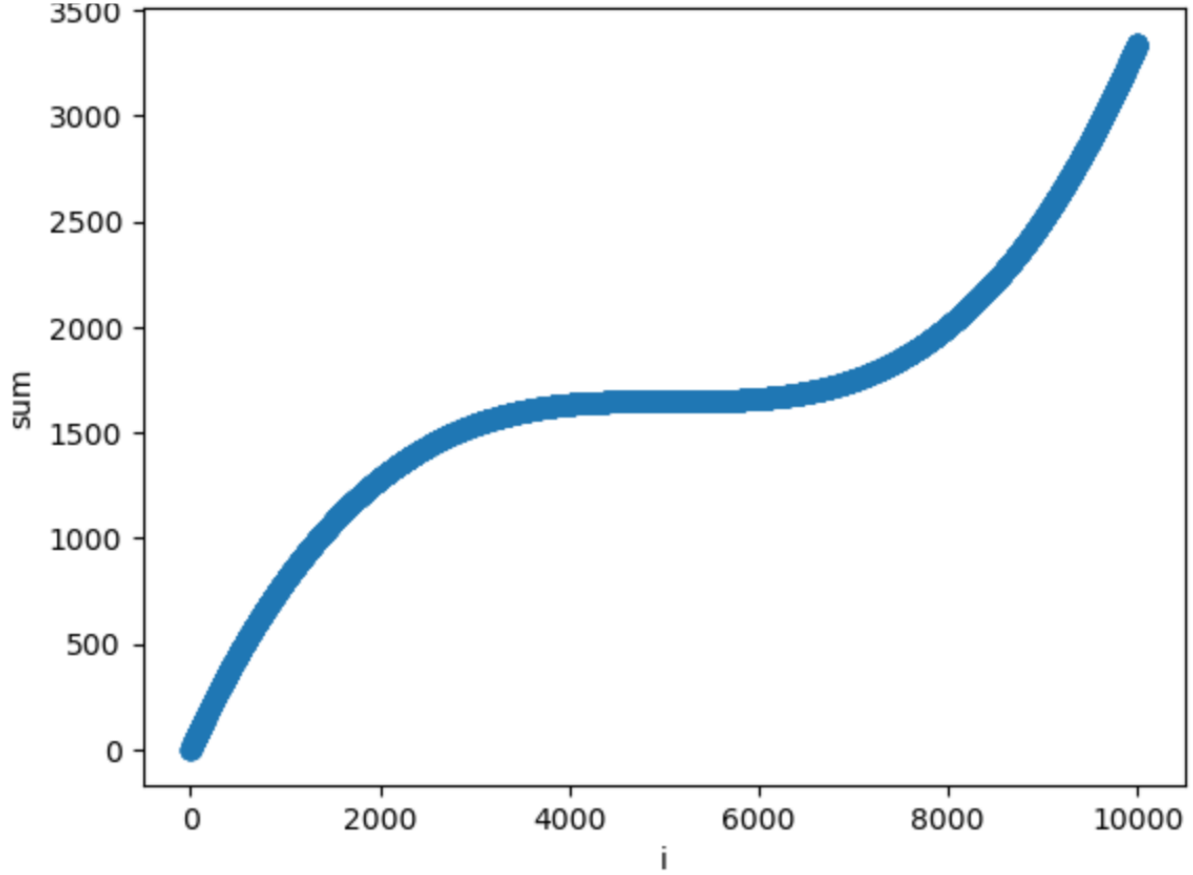


Problem 3.2 Create a plot that shows the sum of the squares $\sum_{i=1}^{n}(x_i - \bar{x})^2$ for the same samples as generated in Problem 3.1. The x-axis should be i where i denotes the i^{th} sample. The y-axis should be the sum of the squares up to the i^{th} sample. This plot should be non-decreasing. It demonstrates how sum of squares increases whether samples fall above or below the mean.



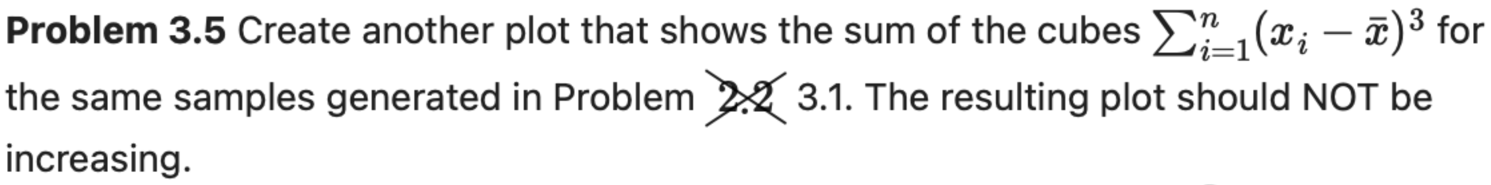
HW4 PROBLEM 3.3 AND 3.4

Problem 3.3 Order the samples from Problem ~~2.5~~ 3.1 from smallest to largest and create another plot.

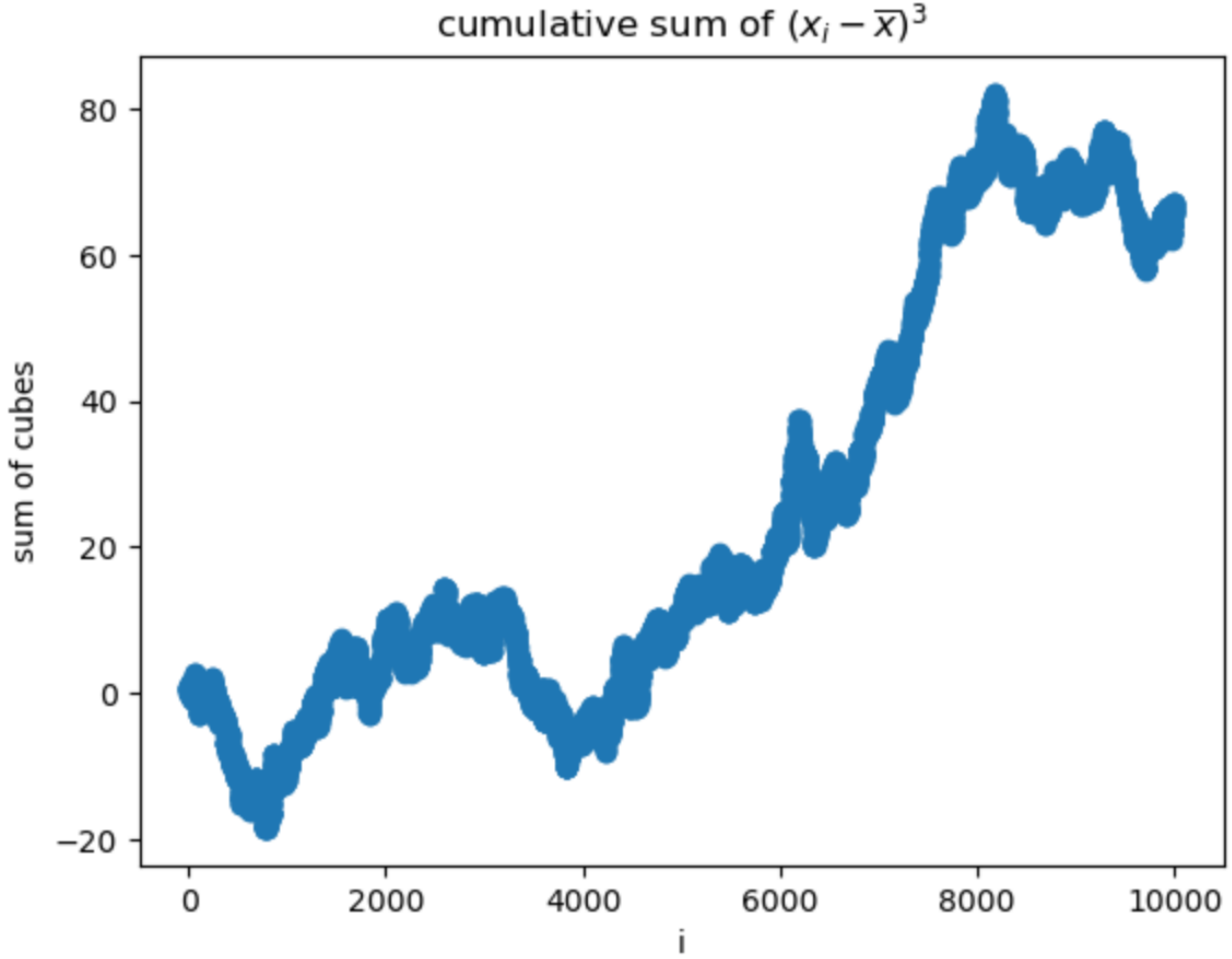


Problem 3.4 When the samples are ordered, what happens? Explain the shape. What does the shape of the curve tell us about the contribution to the variance of points that are farther from the mean?

HW4 PROBLEM 3.5

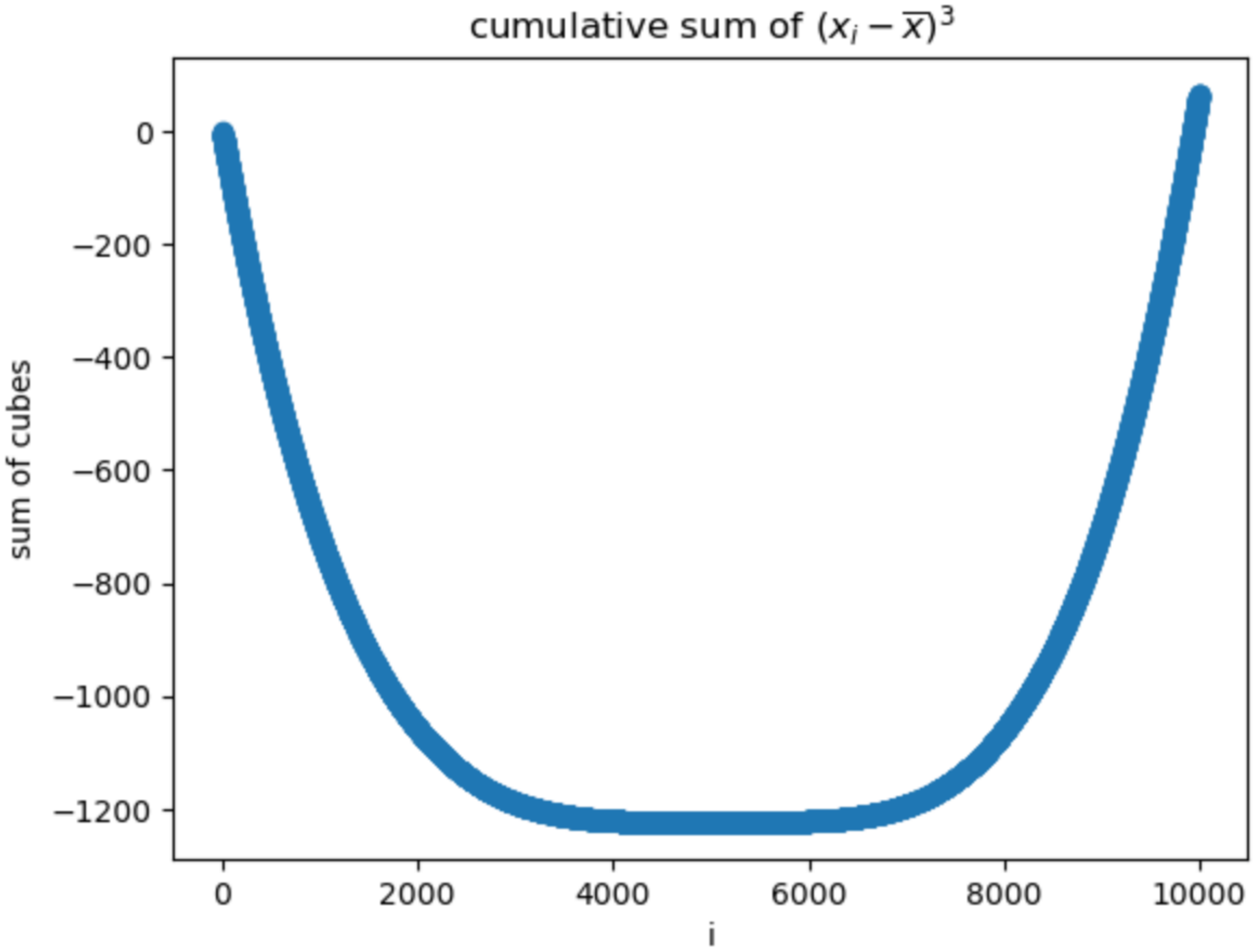


Problem 3.5 Create another plot that shows the sum of the cubes $\sum_{i=1}^{n}(x_i - \bar{x})^3$ for the same samples generated in Problem ~~2.2~~ 3.1. The resulting plot should NOT be increasing.



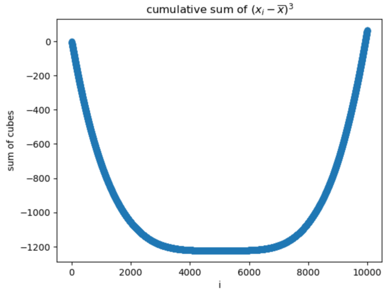
HW4 PROBLEM 3.6

Problem 3.6 Order the samples from smallest to largest and create the same plot again but with the ordered samples.



HW4 PROBLEM 3.6

Problem 3.6 Order the samples from smallest to largest and create the same plot again but with the ordered samples.

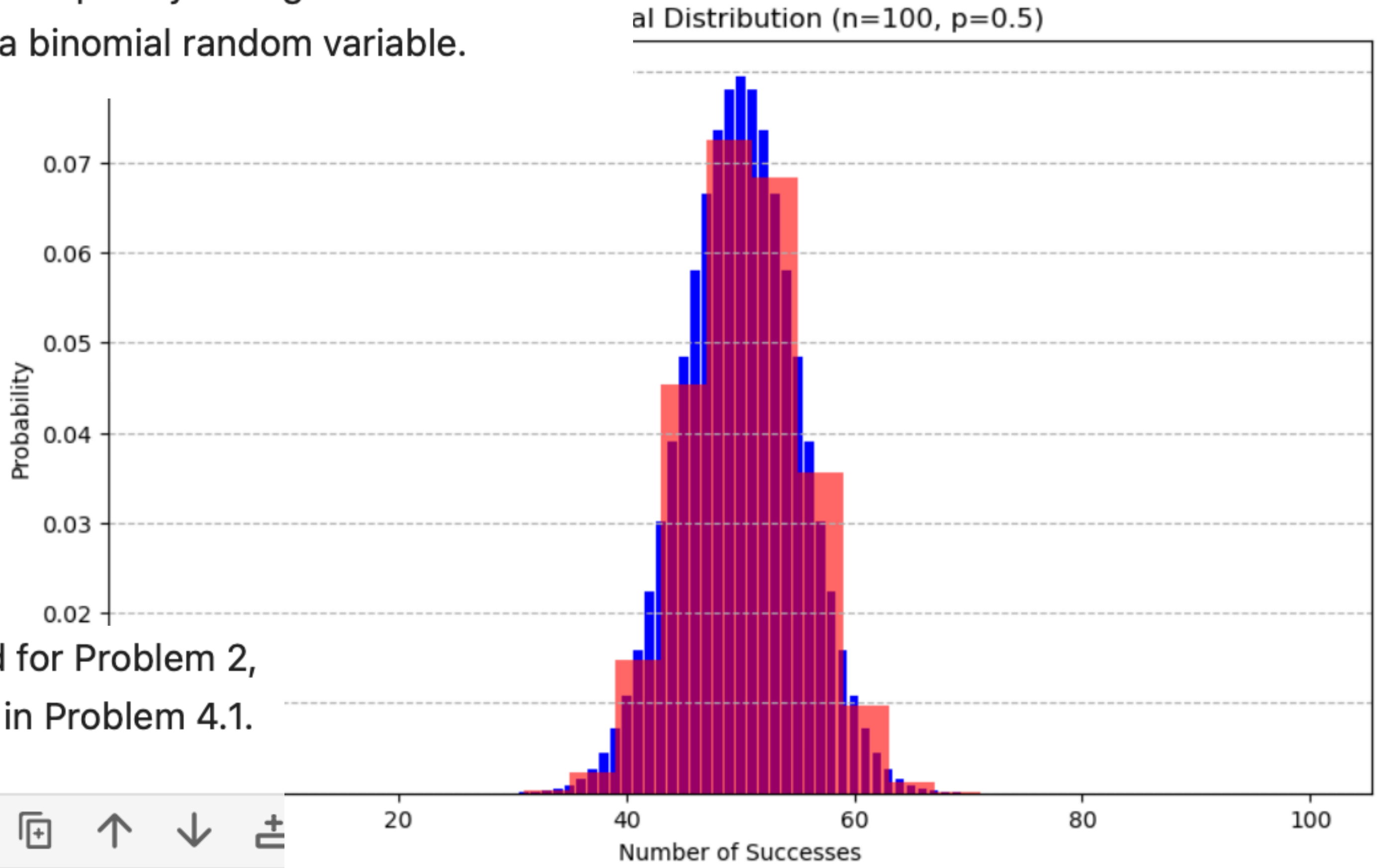


(c) How does this plot differ from the plot for cumulative sum of squares? Note that the function is no longer non-decreasing. How does the shape of the curve affect the contribution of the samples to the left of the mean vs. to the right of the mean for an unskewed distribution?

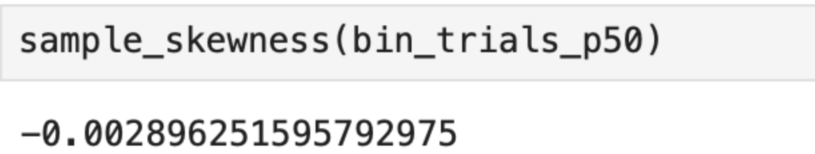
(d) How does the shape of this curve affect the contribution of samples farther from the mean than nearer to the mean? What happens if more samples are near the mean on one-side of the distribution as would occur with a skewed distribution?

HW4 PROBLEM 4.1 & 4.4

Problem 4.1 Use matplotlib to create a single plot containing the PMF of a Binomial random variable for $p = 0.5$. A trial of a binomial random variable involves perfoming n Bernoulli trials. Perform 1000 or 10000 (see note 1) binomial trials each of $n = 100$ Bernoulli trials with $p = 0.5$. Perform each Bernoulli trial using a random number generator. Plot relative frequency histogram of the Bernoulli trials on the same plot as the PMF of a binomial random variable.



Problem 4.4 Using your function implemented for Problem 2, compute the sample skewness of the samples in Problem 4.1.

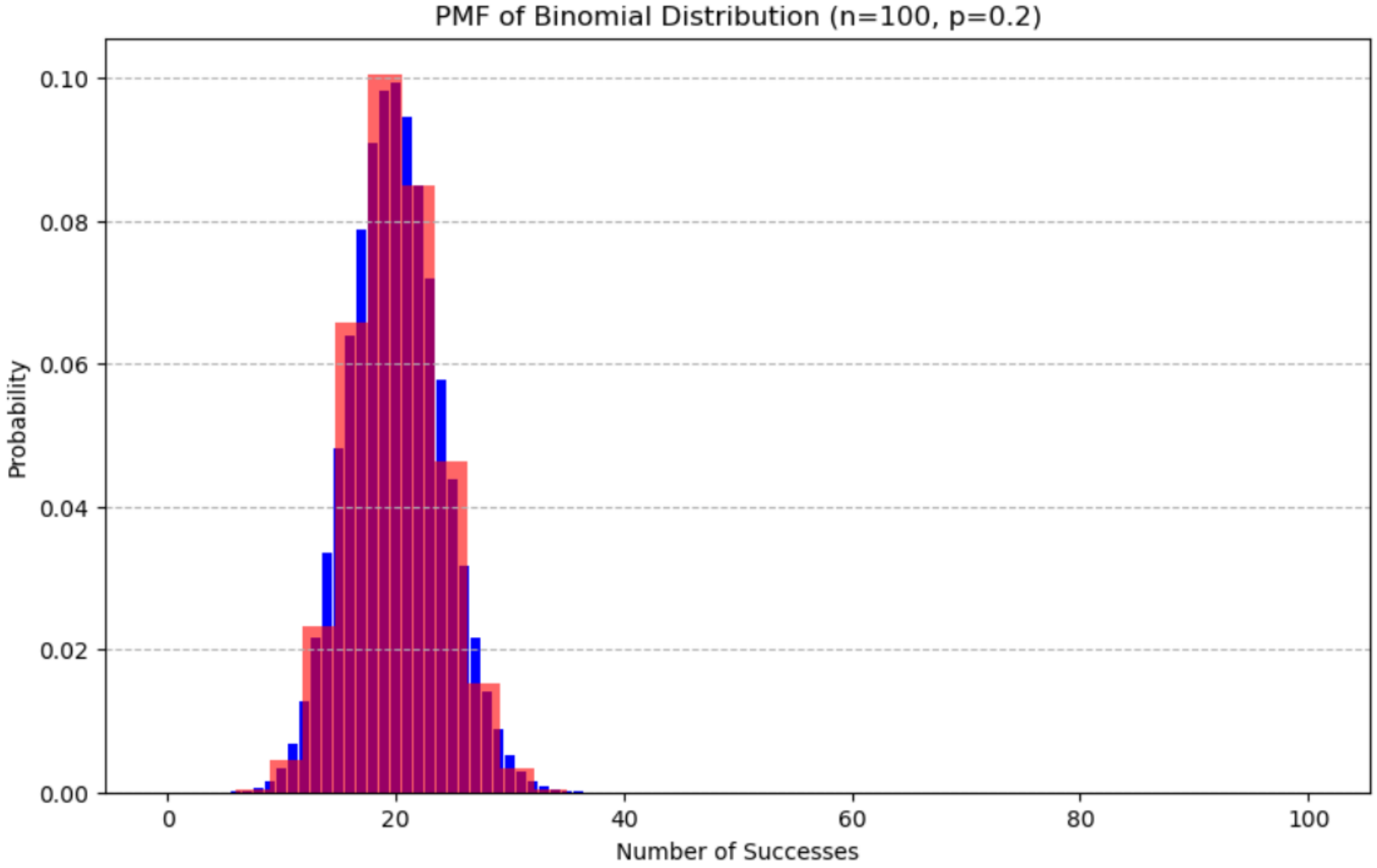


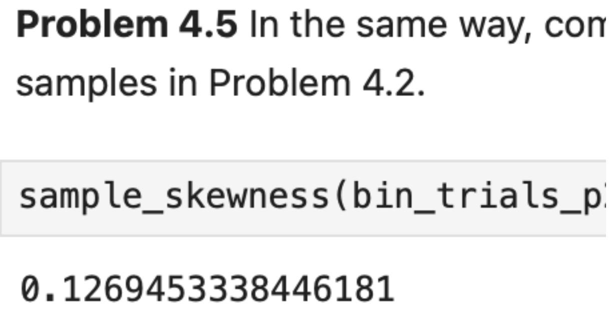
```
sample_skewness(bin_trials_p50)
```

```
-0.002896251595792975
```

HW4 PROBLEM 4.2 & 4.5

Problem 4.2 Repeat Problem 4.1 for $p = 0.2$.





Problem 4.5 In the same way, com

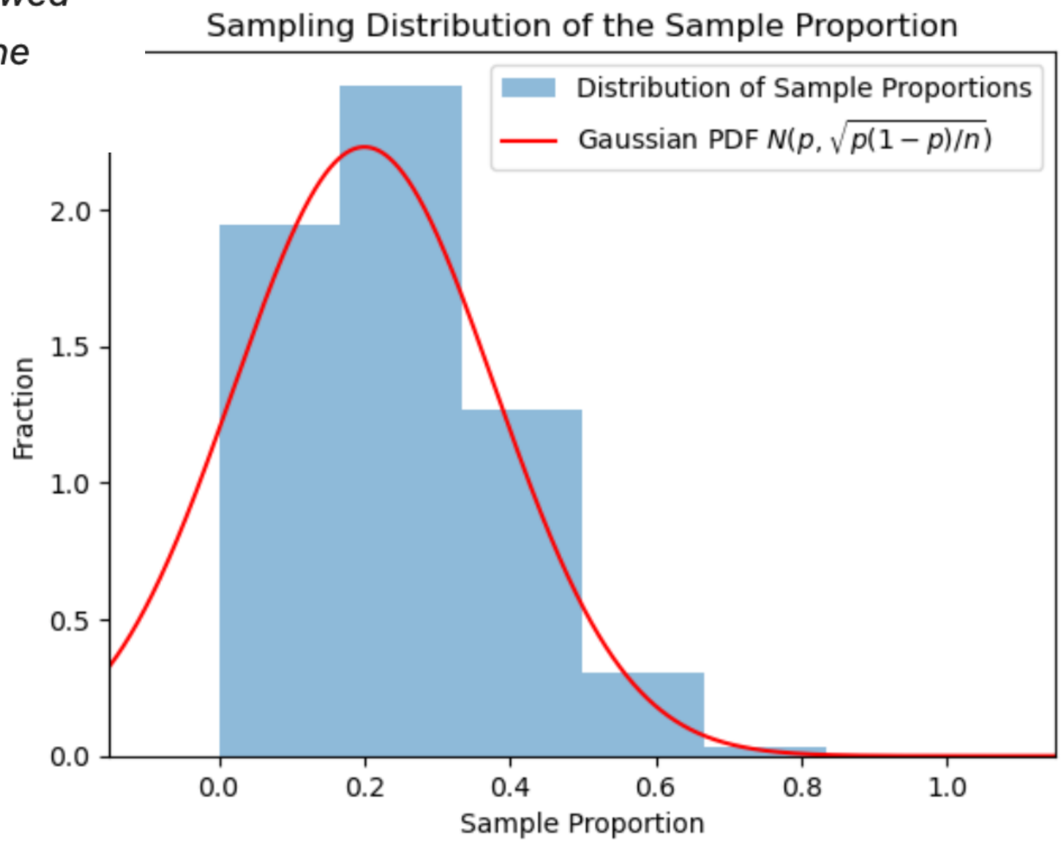
samples in Problem 4.2.

```
sample_skewness(bin_trials_p2
```

```
0.1269453338446181
```

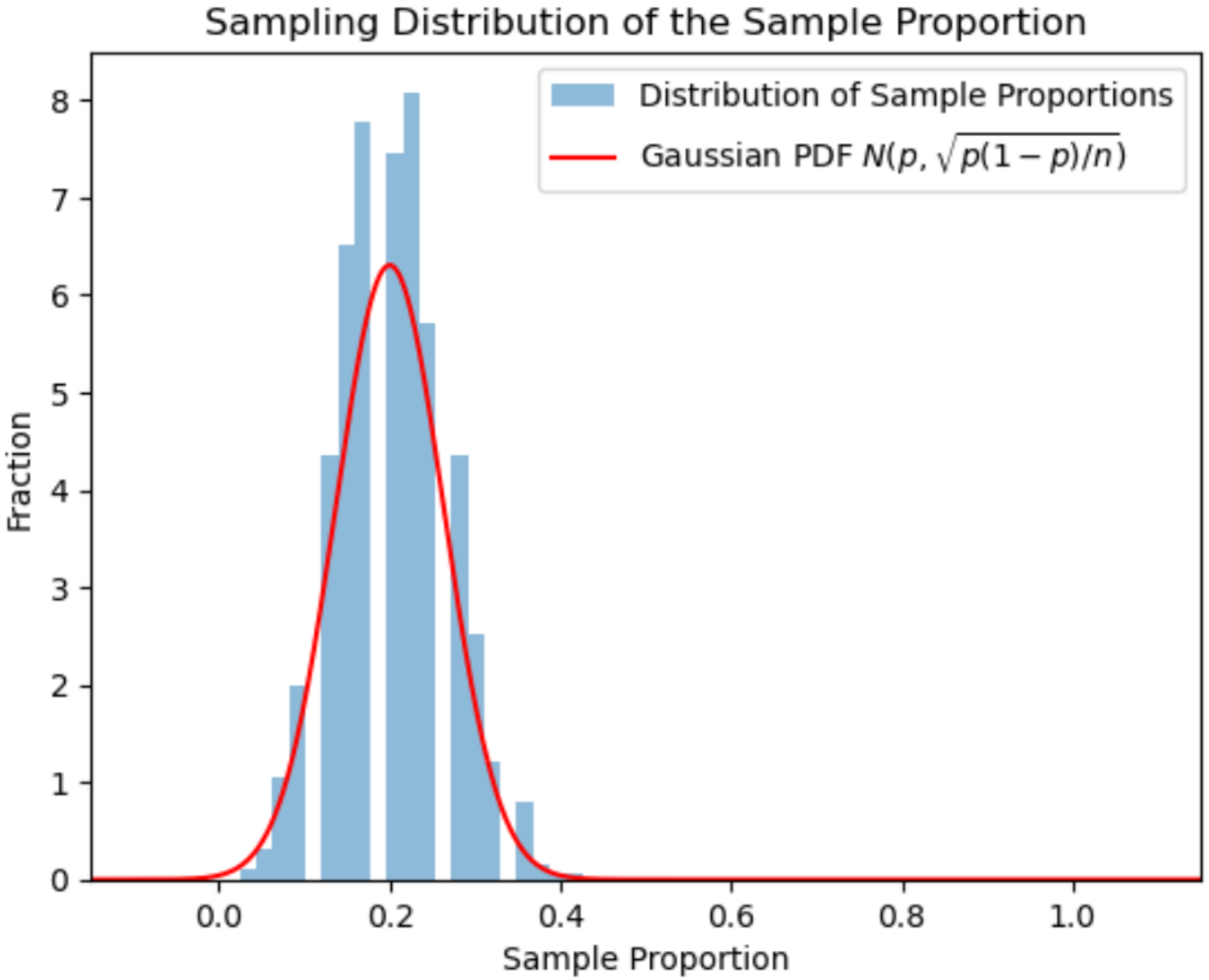
HW4 PROBLEM 4.7

(Revised wording) *For a binomial distribution with* $n = 5$ *and* $p = 0.2$*, simulate drawing 1000 samples of* $X \sim Bin(n, p)$ *and computing the sample proportion . The distribution of the sample proportion is the sampling distribution of* p*. Plot this sampling distribution. On the same plot place the PDF of a Gaussian random variable* $N(p, \sqrt{p(1-p)/n}\,)$*. How does the Gaussian PDF relate to computing confidence intervals? Is the sampling distribution skewed or symmetric? How does the Gaussian distribution compare to the distribution of the sample proportion?*



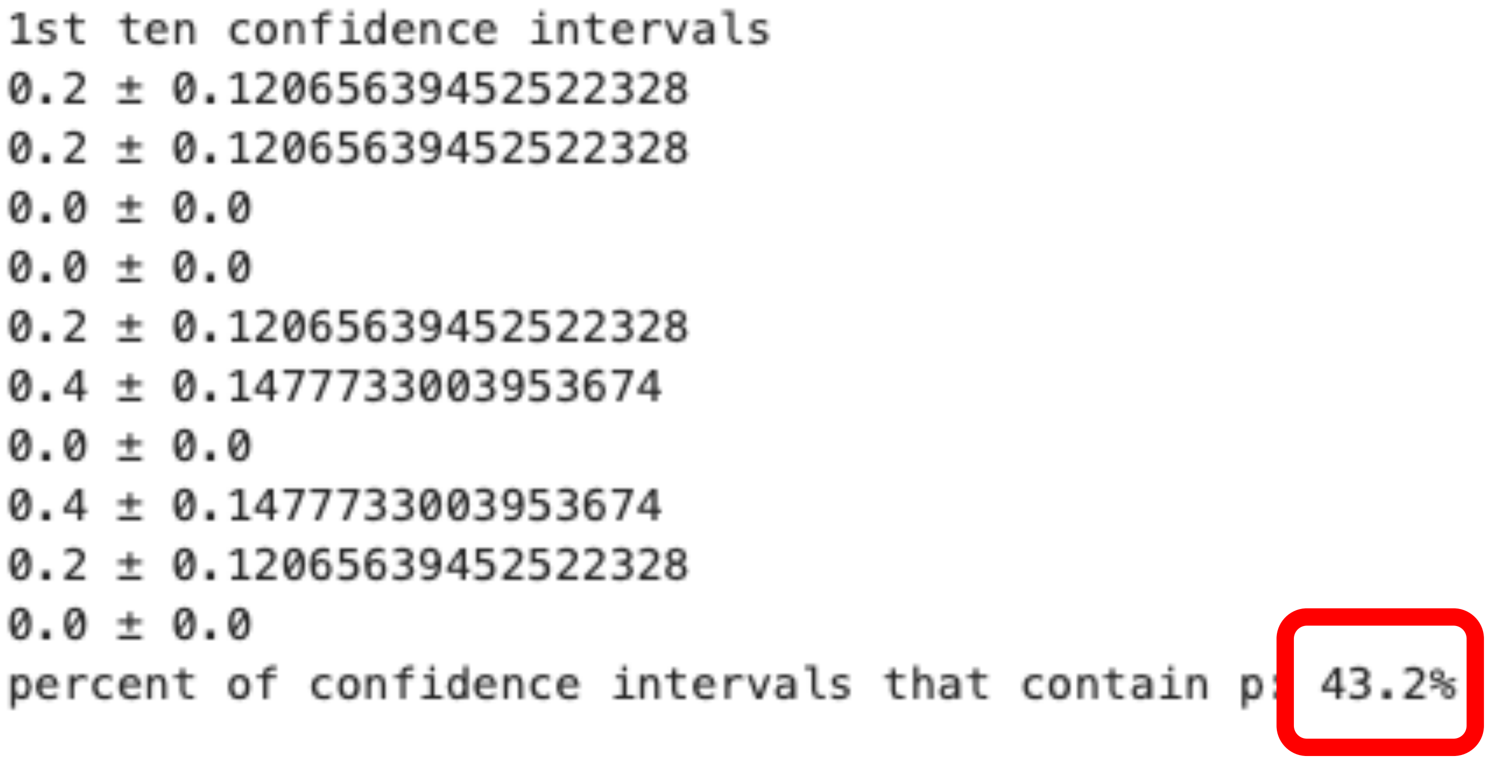
HW4 PROBLEM 4.8

Problem 4.8 For a binomial distribution with $n = 40$ and $p = 0.2$, simulate drawing 1000 sample sets each of size 40. Plot the sample distribution of the sample proportion. How does increasing n affect the skewness of the sampling distribution?



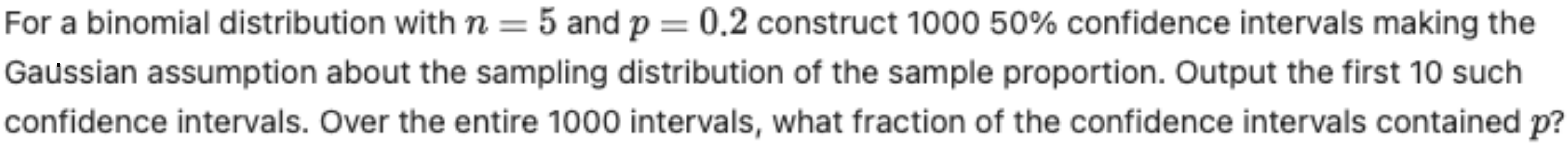
HW4 PROBLEM 4.9

For a binomial distribution with $n = 5$ and $p = 0.2$ construct 1000 50% confidence intervals making the Gaussian assumption about the sampling distribution of the sample proportion. Output the first 10 such confidence intervals. Over the entire 1000 intervals, what fraction of the confidence intervals contained p?

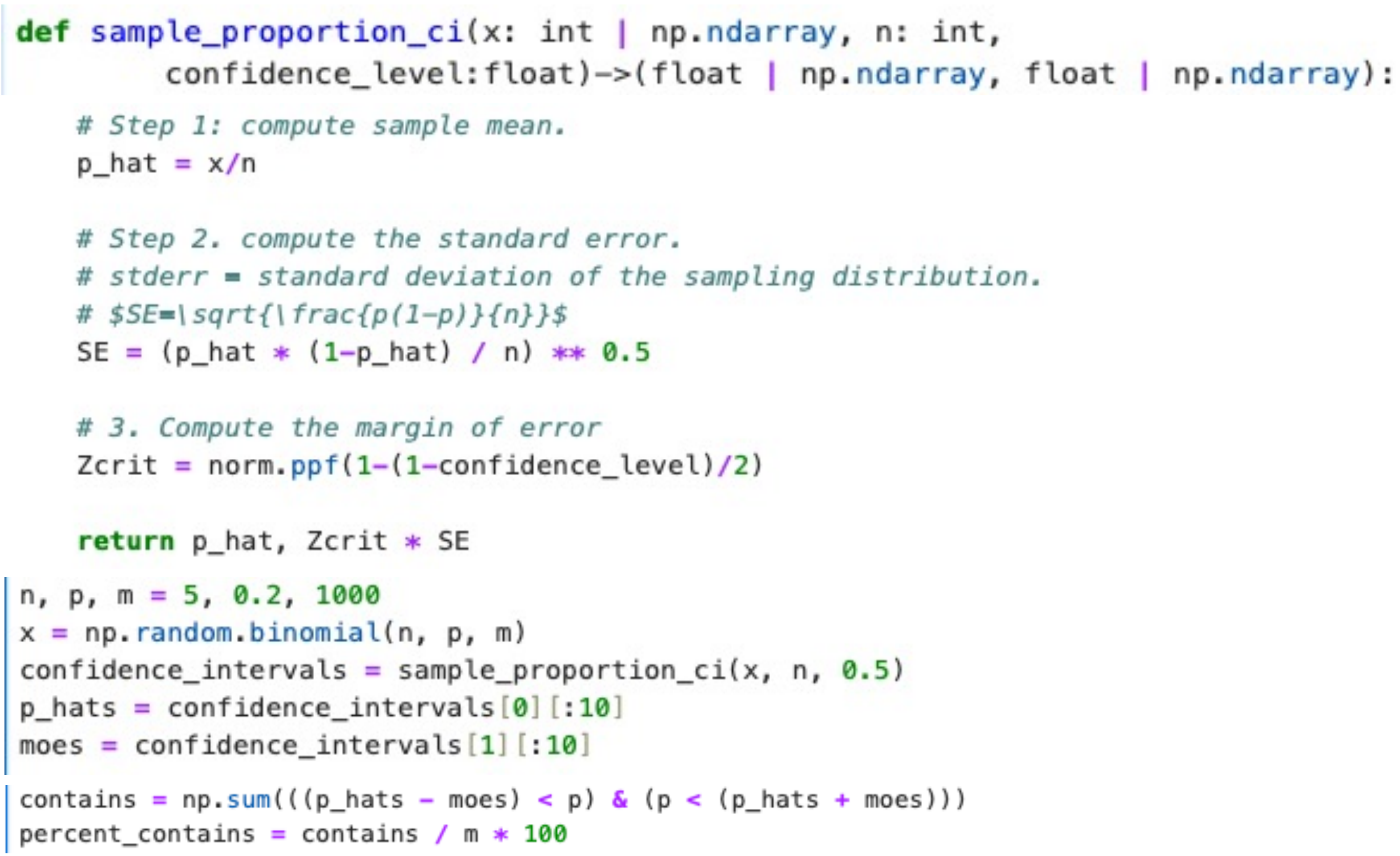


```
1st ten confidence intervals
0.2 ± 0.1206563945252328
0.2 ± 0.1206563945252328
0.0 ± 0.0
0.0 ± 0.0
0.2 ± 0.1206563945252328
0.4 ± 0.1477733003953674
0.0 ± 0.0
0.4 ± 0.1477733003953674
0.2 ± 0.1206563945252328
0.0 ± 0.0
percent of confidence intervals that contain p: 43.2%
```

HW4 PROBLEM 4.9



For a binomial distribution with $n = 5$ and $p = 0.2$ construct 1000 50% confidence intervals making the Gaussian assumption about the sampling distribution of the sample proportion. Output the first 10 such confidence intervals. Over the entire 1000 intervals, what fraction of the confidence intervals contained p?

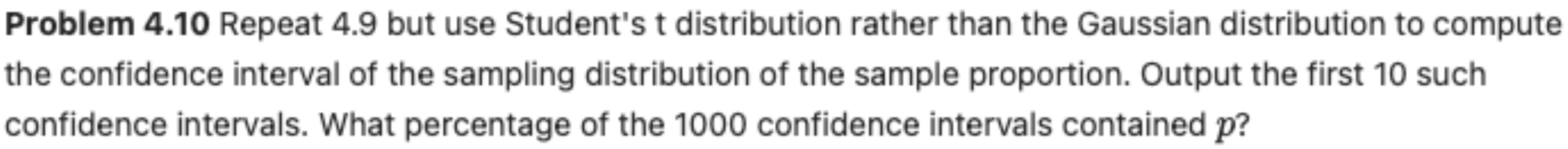


```
def sample_proportion_ci(x: int | np.ndarray, n: int,
        confidence_level:float)->(float | np.ndarray, float | np.ndarray):
    # Step 1: compute sample mean.
    p_hat = x/n

    # Step 2. compute the standard error.
    # stderr = standard deviation of the sampling distribution.
    # $SE=\sqrt{\frac{p(1-p)}{n}}$
    SE = (p_hat * (1-p_hat) / n) ** 0.5

    # 3. Compute the margin of error
    Zcrit = norm.ppf(1-(1-confidence_level)/2)

    return p_hat, Zcrit * SE
```

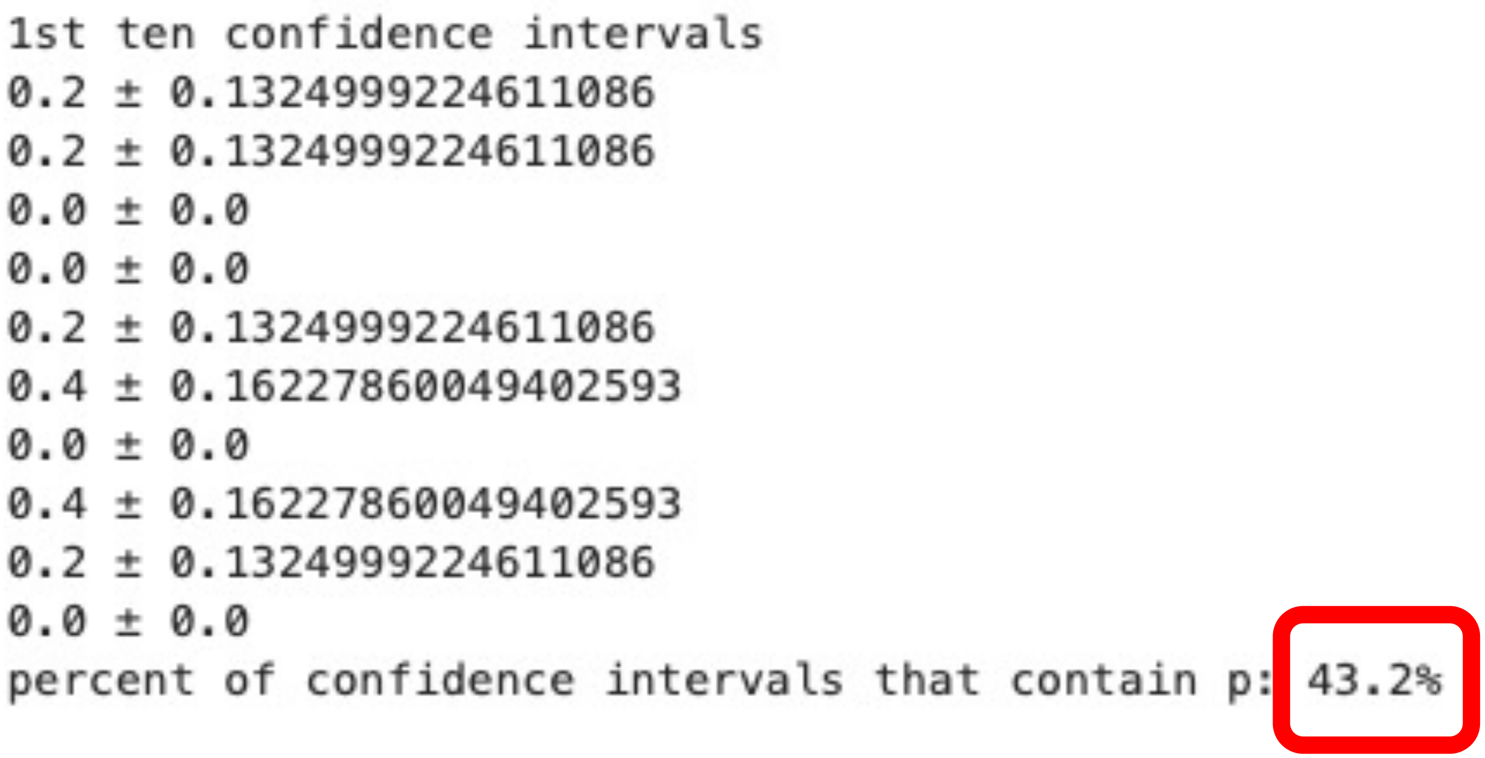
```
n, p, m = 5, 0.2, 1000
x = np.random.binomial(n, p, m)
confidence_intervals = sample_proportion_ci(x, n, 0.5)
p_hats = confidence_intervals[0][:10]
moes = confidence_intervals[1][:10]
```

```
contains = np.sum(((p_hats - moes) < p) & (p < (p_hats + moes)))
percent_contains = contains / m * 100
```

HW4 PROBLEM 4.10

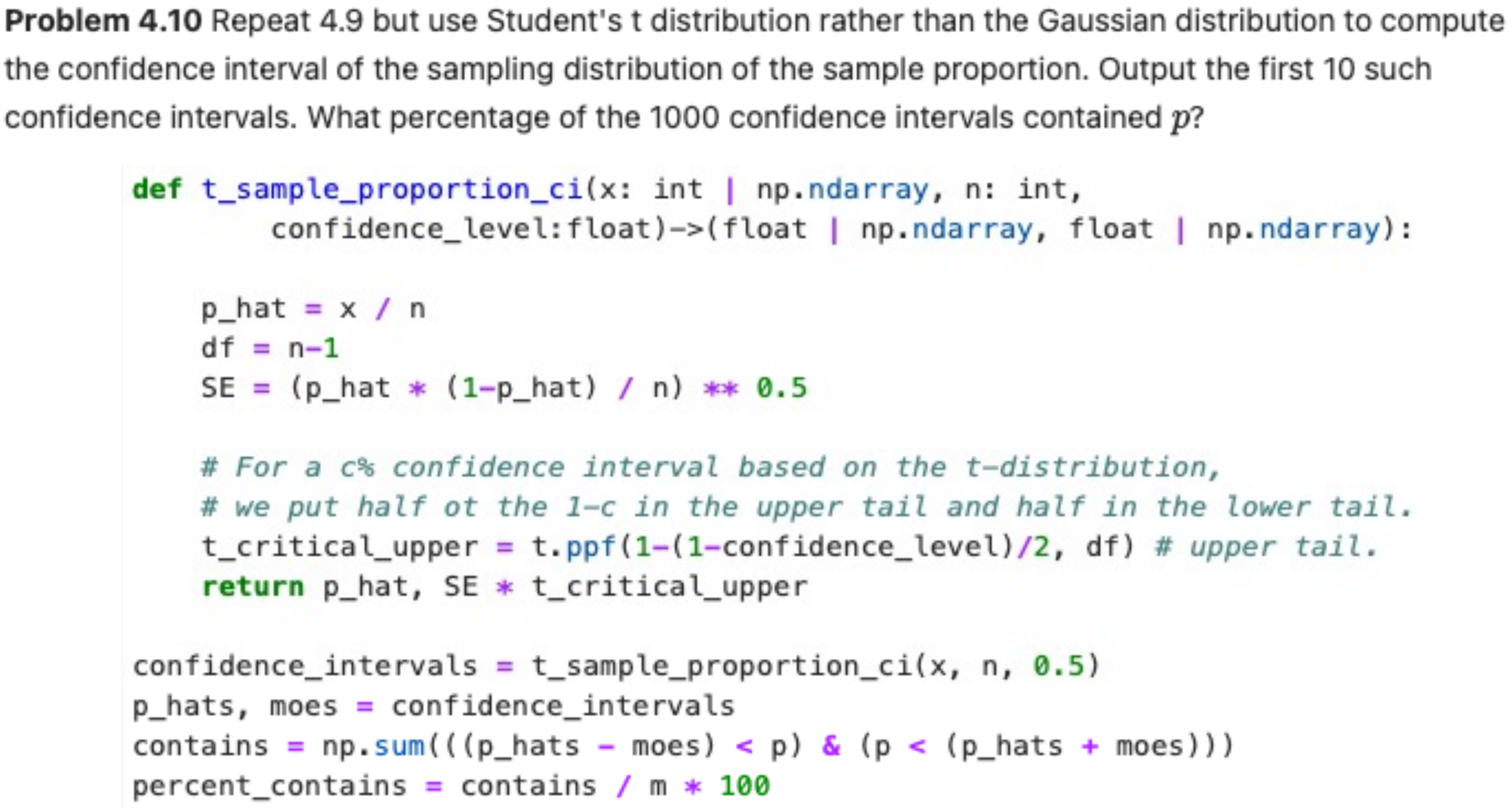


Problem 4.10 Repeat 4.9 but use Student's t distribution rather than the Gaussian distribution to compute the confidence interval of the sampling distribution of the sample proportion. Output the first 10 such confidence intervals. What percentage of the 1000 confidence intervals contained p?



```
1st ten confidence intervals
0.2 ± 0.1324999224611086
0.2 ± 0.1324999224611086
0.0 ± 0.0
0.0 ± 0.0
0.2 ± 0.1324999224611086
0.4 ± 0.16227860049402593
0.0 ± 0.0
0.4 ± 0.16227860049402593
0.2 ± 0.1324999224611086
0.0 ± 0.0
percent of confidence intervals that contain p: 43.2%
```

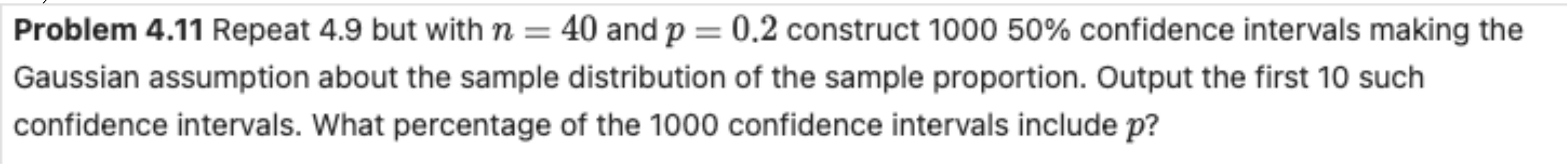
HW4 PROBLEM 4.10



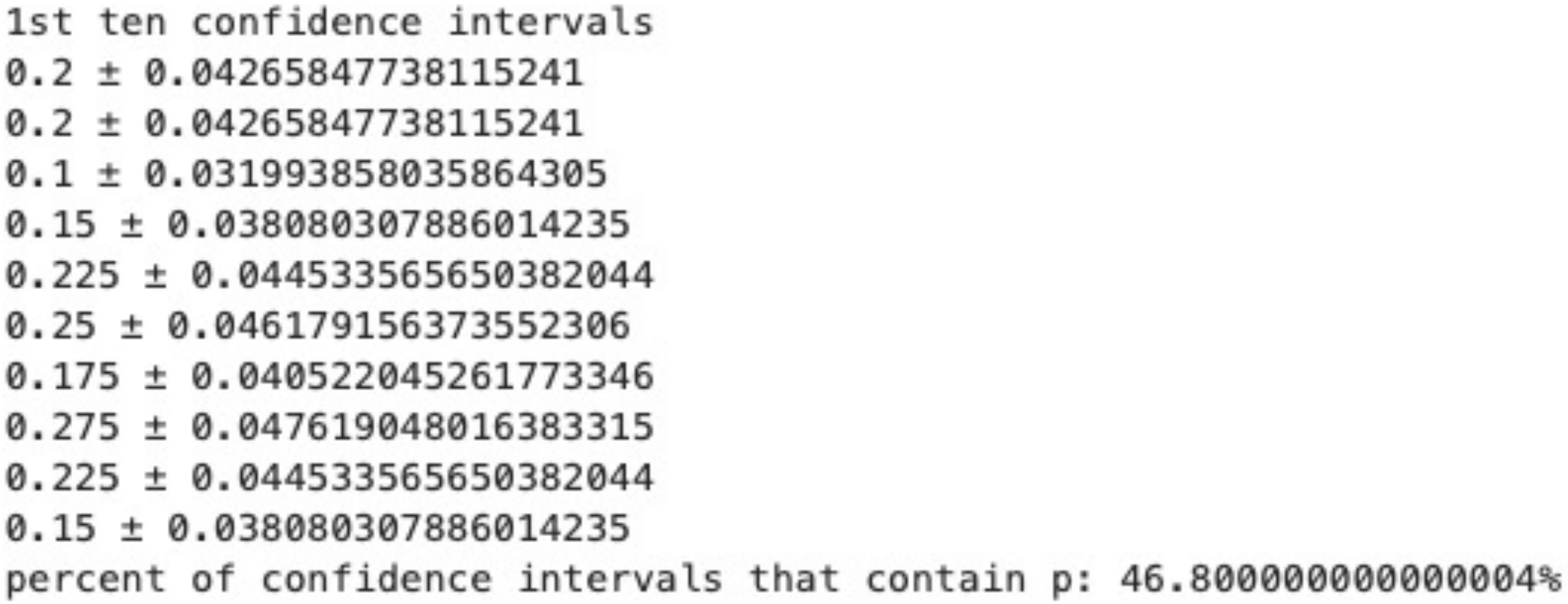
Problem 4.10 Repeat 4.9 but use Student's t distribution rather than the Gaussian distribution to compute the confidence interval of the sampling distribution of the sample proportion. Output the first 10 such confidence intervals. What percentage of the 1000 confidence intervals contained p?

```
def t_sample_proportion_ci(x: int | np.ndarray, n: int,
        confidence_level:float)->(float | np.ndarray, float | np.ndarray):

    p_hat = x / n
    df = n-1
    SE = (p_hat * (1-p_hat) / n) ** 0.5

    # For a c% confidence interval based on the t-distribution,
    # we put half ot the 1-c in the upper tail and half in the lower tail.
    t_critical_upper = t.ppf(1-(1-confidence_level)/2, df) # upper tail.
    return p_hat, SE * t_critical_upper

confidence_intervals = t_sample_proportion_ci(x, n, 0.5)
p_hats, moes = confidence_intervals
contains = np.sum(((p_hats - moes) < p) & (p < (p_hats + moes)))
percent_contains = contains / m * 100
```

HW4 PROBLEM 4.11

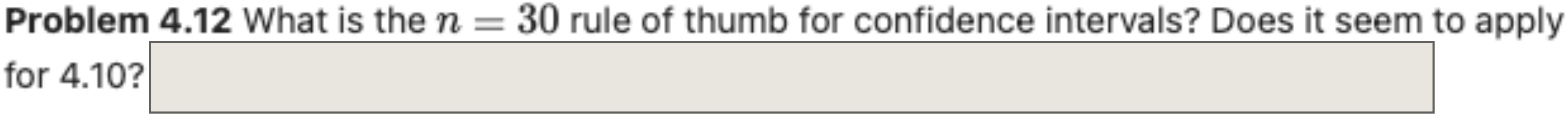


Problem 4.11 Repeat 4.9 but with $n = 40$ and $p = 0.2$ construct 1000 50% confidence intervals making the Gaussian assumption about the sample distribution of the sample proportion. Output the first 10 such confidence intervals. What percentage of the 1000 confidence intervals include p?



```
1st ten confidence intervals
0.2 ± 0.04265847738115241
0.2 ± 0.04265847738115241
0.1 ± 0.031993858035864305
0.15 ± 0.038080307886014235
0.225 ± 0.044533565650382044
0.25 ± 0.046179156373552306
0.175 ± 0.040522045261773346
0.275 ± 0.047619048016383315
0.225 ± 0.044533565650382044
0.15 ± 0.038080307886014235
percent of confidence intervals that contain p: 46.800000000000004%
```

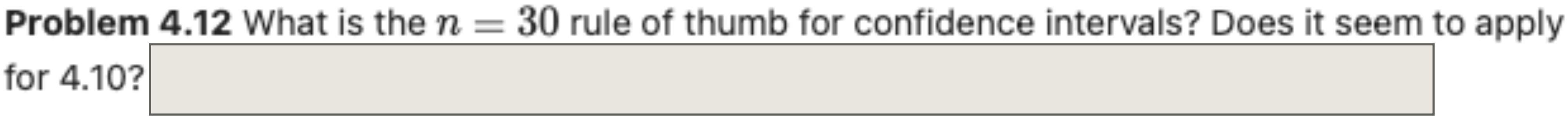
HW4 PROBLEM 4.12



Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10?

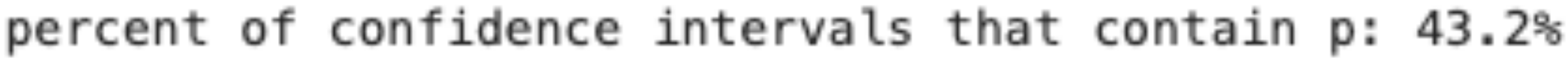
In 4.10, n=5 so the t-distribution is used.

HW4 PROBLEM 4.12



Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10?

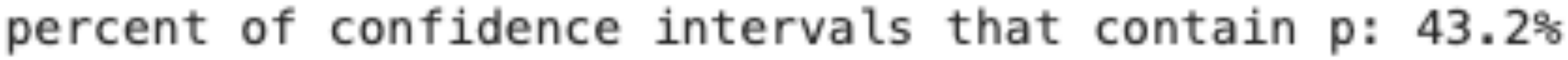
In 4.10, n=5 so the t-distribution is used.



```
percent of confidence intervals that contain p: 43.2%
```

This is a bit low. I repeated this with different seeds and it consistently came out low.

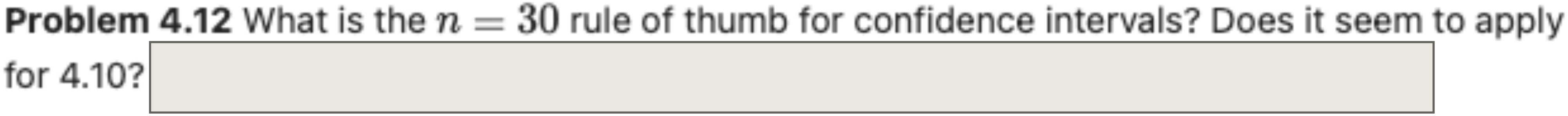
In 4.9, n=5 but we used a Gaussian distribution to compute CI.



```
percent of confidence intervals that contain p: 43.2%
```

No difference! What is going on?

HW4 PROBLEM 4.12

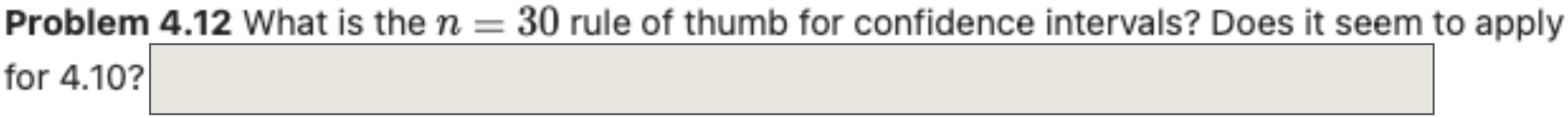


Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10?

T-distribution is derived from small n when the underlying distribution is Gaussian. It works when the distribution is reasonably Gaussian-like.

A binomial with p=20% is a asymmetric.

HW4 PROBLEM 4.12



Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10?

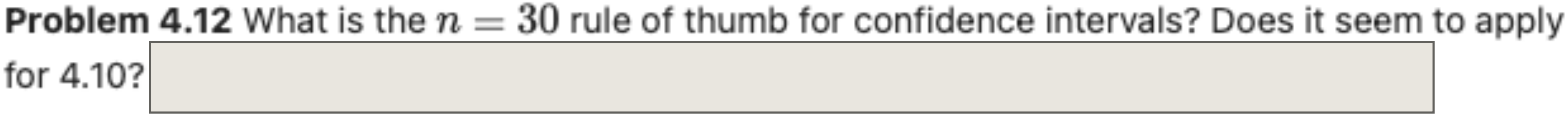
T-distribution is derived from small n when the underlying distribution is Gaussian. It works when the distribution is reasonably Gaussian-like.

A binomial with p=20% is a asymmetric.

And sample proportion can only take on discrete values that are farther apart than the margin of error using either Gaussian or t-distribution.

Be wary of small n.

HW4 PROBLEM 4.12



Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10?

Sufficient *n* is dependent on the shape of the underlying distribution. A distribution that is skewed may require larger *n*. A separate rule of thumb is sometimes used for binomial distributions to account for asymmetry.

Rule of Thumb for binomial distributions: It is considered reasonable to compute CI assuming Gaussian when

$$np > 5$$

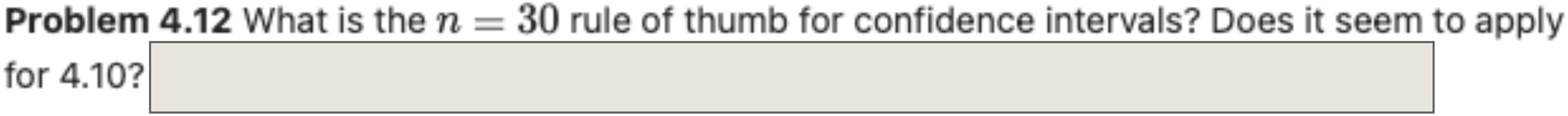
and

$$n(1-p) > 5$$

For problems 4.9 and 4.10

$n = 5$ and $p = 0.2$, $np = 1$.

HW4 PROBLEM 4.12



Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10?

For problems 4.9 and 4.10

$n = 5$ and $p = 0.2$, $np = 1$.

np << 5, so Gaussian won’t work.

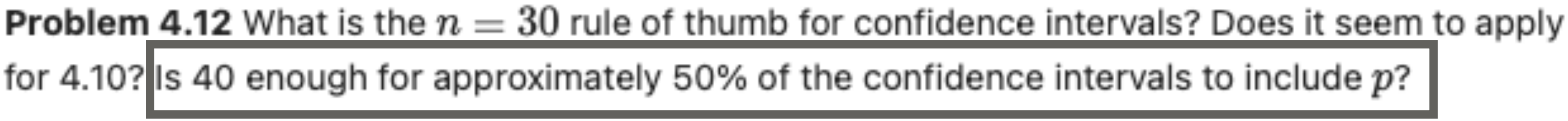
In 4.10 we used t-distribution, but this didn’t work either.

T-distribution typically isn’t used with binomial distributions at all.

If *np* is too small neither Gaussian or t-distribution works. If *np* is large enough, just use Gaussian.

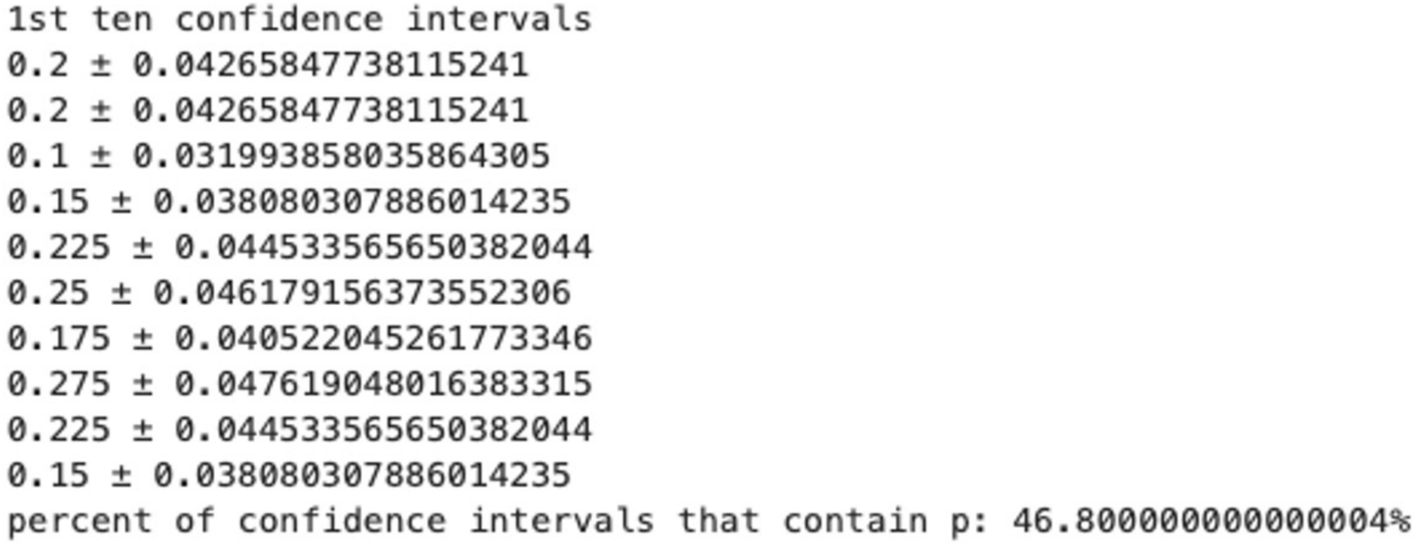
If *np* is small. Use exact methods (which you can look up).

HW4 PROBLEM 4.12



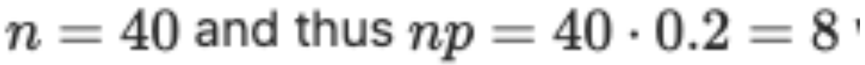
Problem 4.12 What is the $n = 30$ rule of thumb for confidence intervals? Does it seem to apply for 4.10? Is 40 enough for approximately 50% of the confidence intervals to include p?

In problem 4.11 we had



```
1st ten confidence intervals
0.2 ± 0.04265847738115241
0.2 ± 0.04265847738115241
0.1 ± 0.031993858035864305
0.15 ± 0.038080307886014235
0.225 ± 0.044533565650382044
0.25 ± 0.046179156373552306
0.175 ± 0.040522045261773346
0.275 ± 0.047619048016383315
0.225 ± 0.044533565650382044
0.15 ± 0.038080307886014235
percent of confidence intervals that contain p: 46.800000000000004%
```

Closer. If you repeat a few times with different seeds, I sometimes got closer to 50%, but I never reached 50%.



$n = 40$ and thus $np = 40 \cdot 0.2 = 8$

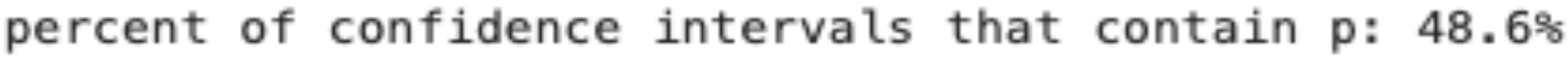
8 is only slightly higher than 5, so maybe 40 isn't quite enough.

HW4 PROBLEM 4.12 (BEYOND ASKED)

In problems 4.11 and 4.12, n=40.

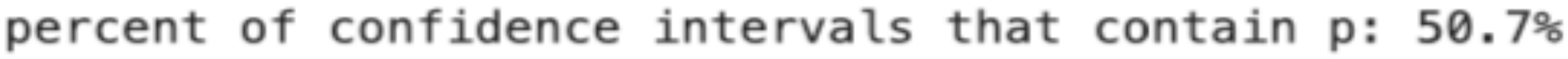
This was not asked, but if we increase to n=100

np = 100 * 0.2 = 20 >> 5



```
percent of confidence intervals that contain p: 48.6%
```

At n=1000,



```
percent of confidence intervals that contain p: 50.7%
```

At n=1000, with different seeds, it sometimes is above and sometimes below 50%.

THANK YOU

David Harrison

Harrison@cs.olemiss.edu