hw07

November 27, 2018

1 Homework 7: Percentiles, Bootstrap, A/B Testing

1.1 Due Sunday November 25th, 11:59pm

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged.

You should start early so that you have time to get help if you're stuck.

```
In [167]: #:
          import math
          import numpy as np
          from datascience import *
          import matplotlib.pyplot as plt
          plt.style.use('fivethirtyeight')
          %matplotlib inline

          from client.api.notebook import Notebook
          ok = Notebook('hw07.ok')
          _ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Homework 7
OK, version v1.13.11
=====================================================================

Successfully logged in as wec149@ucsd.edu
```

Important: The `ok` tests don't usually tell you that your answer is correct. More often, they help catch careless mistakes. It's up to you to ensure that your answer is correct. If you're not sure, ask someone (not for the answer, but for some guidance about your approach).

Once you're finished, you must do two things:

1.1.1 a. Turn into OK

Select "Save and Checkpoint" in the File menu and then run the submission cell at the end of the notebook. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we'll only grade your final submission.

1.1.2 b. Turn PDF into Gradescope

Select File > Download As > PDF via LaTeX in the File menu. Turn in this PDF file into the respective assignement at https://gradescope.com/. If you submit more than once before the deadline, we will only grade your final submission

1.2 1. Percentiles

The General Definition

> Let p be a number between 0 and 100. The pth percentile of a collection is the smallest value in the collection that is *at least as large* as p% of all the values.

By this definition, any percentile between 0 and 100 can be computed for any collection of values and is always an element of the collection. Suppose there are n elements in the collection. To find the pth percentile:

1. Sort the collection in increasing order.
2. Find p% of n: $\frac{p}{100} * n$. Call that h. If h is an integer, define $k = h$. Otherwise, let k be the smallest integer greater than h.
3. Take the kth element of the sorted collection.

Question 1. Assign the number of elements in `values` to the variable `n`. Define `k` as above -- your answer should be an integer. Assign the 65th percentile of the array `values` to `sixty_fifth_percentile`. You must use the variables provided for you when solving this problem. For this problem only, you may *not* use `percentile()`.

Hint: Using `math.ceil()` will round up a number to the next nearest whole number. `math` has already been imported for you.

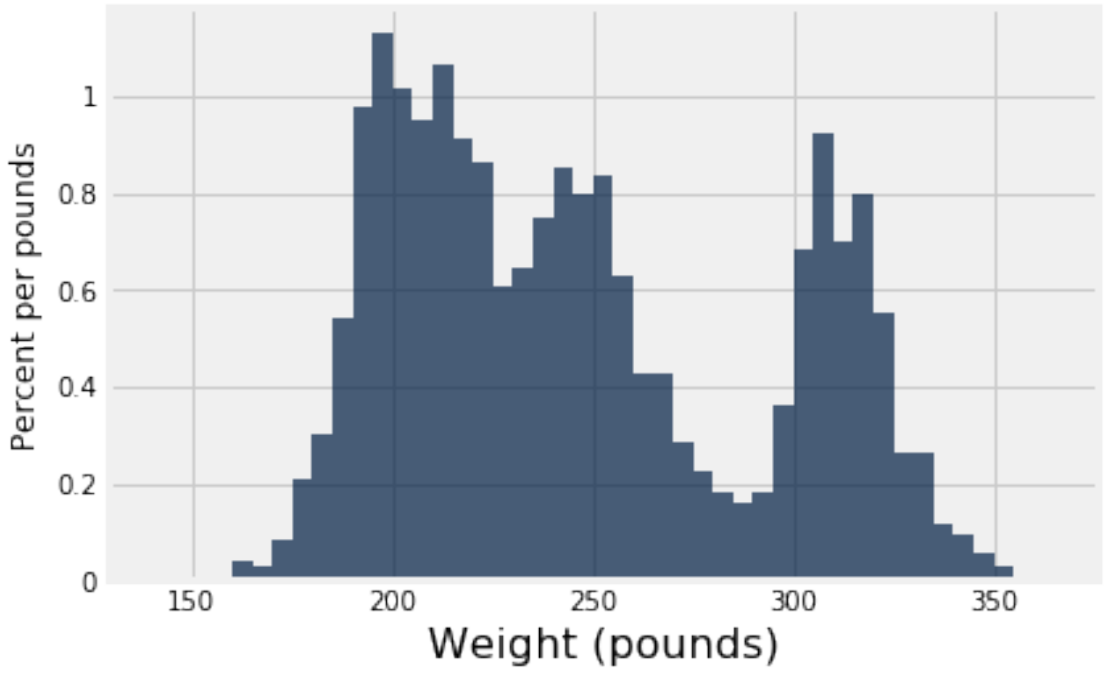
```
In [168]: #: don't change the values in this array!
          values = make_array(34, 65, 103, 73, 4, 32, 45, 87, 99, 54)
          values.sort()  # This line sorts the array
          values

Out[168]: array([  4,  32,  34,  45,  54,  65,  73,  87,  99, 103])

In [169]: n = values.size
          n

Out[169]: 10

In [170]: k = math.ceil(65/100 * n)
          k

Out[170]: 7

In [171]: sixty_fifth_percentile = values.item(k-1)
          sixty_fifth_percentile

Out[171]: 73
```

```
In [172]: _ = ok.grade('q1_1')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 2. The table `nfl_players` contains one row for each player in the National Football League (NFL). The columns include each player's `Number`, `Name`, `Position`, `Age`, `Height` (in inches), `Weight` (in pounds), `College`, and `Team`. Plot a histogram showing each player's weight. Use the bins provided.

Source: http://espn.com and https://www.pro-football-reference.com/players/salary.htm on 2/1/2018.

```
In [173]: #: read in the table
          nfl_players = Table().read_table('nfl_players.csv')
          nfl_bins = np.arange(140, 370, 5)
          # nfl_players

In [174]: # plot your histogram here
          nfl_players.hist('Weight', bins=nfl_bins, unit='pounds')
```



Question 3. Find the absolute difference between the 95th percentile of the weights of both place kickers (PK) and offensive tackles (OT) in the NFL and assign it to `absolute_difference`. You may use `percentile()`.

```
In [175]: place_kickers = nfl_players.where('Position', are.equal_to('PK')).sort('Weight', desce
          # place_percentile = place_kickers.item(math.ceil(95/100 * place_kickers.num_rows))
          place_percentile = percentile(95, place_kickers.column('Weight'))
          offensive_takcles = nfl_players.where('Position', are.equal_to('OT')).sort('Weight', d
          offensive_percentile = percentile(95, offensive_takcles.column('Weight'))
          absolute_difference = abs(place_percentile - offensive_percentile)
          absolute_difference

Out[175]: 116

In [176]: _ = ok.grade('q1_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 4. In an array `quarterback_quartiles` put the values for the first, second, and third quartiles (in that order) of the heights of all quarterbacks (QB) in the NFL. Make sure your values are in the correct order. You may use `percentile()`.

```
In [177]: quarter_backs = nfl_players.where('Position', are.equal_to('QB')).sort('Weight', desce
          first = percentile(25, quarter_backs.column('Weight'))
          second = percentile(50, quarter_backs.column('Weight'))
          third = percentile(75, quarter_backs.column('Weight'))

          quarterback_quartiles = make_array(first, second, third)
          quarterback_quartiles

Out[177]: array([216, 224, 231])

In [178]: _ = ok.grade('q1_4')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
```

```
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 5. Find the the weights of the heaviest players at each position. Assign the 50th percentile of these weights to `fiftieth_percentile`. You may use `percentile()`.

```
In [179]: # place_kickers = nfl_players.where('Position', are.equal_to('PK')).sort('Weight', des
          # offensive_takcles = nfl_players.where('Position', are.equal_to('OT')).sort('Weight',
          # quarter_backs = nfl_players.where('Position', are.equal_to('QB')).sort('Weight', des
          everyone = nfl_players.group('Position', np.max).sort('Weight amax', descending=False)

          fiftieth_percentile = percentile(50, everyone)
          fiftieth_percentile

Out[179]: 265

In [180]: _ = ok.grade('q1_5')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6. Shaun surveyed his class to find the total number of siblings each of his classmates has. You can see his findings below in the table `siblings`. For instance, 2 people have 0 siblings, 4 have 1 sibling, and so on. If one of his classmates, Jake, has some number of siblings that falls in the 75th percentile of Shaun's data, how many siblings does Jake have? Assign your answer to the value `jake_siblings`. You may use `percentile()`.

```
In [181]: #: load the data
          siblings = Table().read_table('siblings.csv')
          siblings

Out[181]: Siblings | Frequency
          0        | 2
          1        | 4
          2        | 5
          3        | 4
          4        | 3
          5        | 1
          8        | 1
```

```
In [182]: whole_class = make_array(0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 8)
          jake_siblings = percentile(75, whole_class)
          jake_siblings

Out[182]: 3

In [183]: _ = ok.grade('q1_6')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

1.3 2. Thanksgiving Dinner

Suppose you're hosting a Thanksgiving dinner and are responsible for cooking a delicious turkey. When you get home from the grocery store, you notice that the turkey you purchased is a bit on the smaller side. You wonder whether turkeys have always been this small. You decide to investigate.

Ideally, you would want to figure out the exact mean weight of all turkeys that were sold for Thanksgiving this year. However, around 46 million turkeys across the world are consumed each Thanksgiving. Thus, it's simply not feasible to obtain the mean weight of *all* turkeys (i.e. the mean weight of the population).

Question 1. Complete the statement below by filling in the blanks.

Therefore, you want to collect a sample of turkeys to obtain a `population` statistic to estimate the `unknown (weight)` parameter.

You head back to the grocery store and record the weights of all whole turkeys that are in stock. While you're at it, you also decide to record the weights of all whole chickens that are in stock. When you arrive back home, you decide to use this data as your sample.

Question 2. Your data is recorded in a CSV file called `stock.csv`. Read this file into a table named `stock`.

```
In [184]: stock = Table.read_table('stock.csv')
          stock

Out[184]: Fowl    | Weight (lb)
          Turkey  | 7.07681
          Turkey  | 14.1794
          Chicken | 3.74413
          Turkey  | 18.915
          Chicken | 3.02597
          Chicken | 2.32612
          Chicken | 2.25437
```

```
          Turkey  | 12.9614
          Chicken | 2.92249
          Chicken | 2.1849
          ... (135 rows omitted)

In [185]: _ = ok.grade('q2_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 3. For now, you only care about the weights of the turkeys. Create a new table with the rows of `stock` where the value of `Fowl` is "Turkey". Assign this new table to `turkeys`.

```
In [186]: turkeys = stock.where('Fowl', are.equal_to('Turkey'))
          turkeys

Out[186]: Fowl   | Weight (lb)
          Turkey | 7.07681
          Turkey | 14.1794
          Turkey | 18.915
          Turkey | 12.9614
          Turkey | 13.7908
          Turkey | 16.2523
          Turkey | 22.8031
          Turkey | 9.31993
          Turkey | 21.1311
          Turkey | 20.9763
          ... (52 rows omitted)

In [187]: _ = ok.grade('q2_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 4. Calculate the mean weight of `turkeys` and assign it to `turkey_mean`.

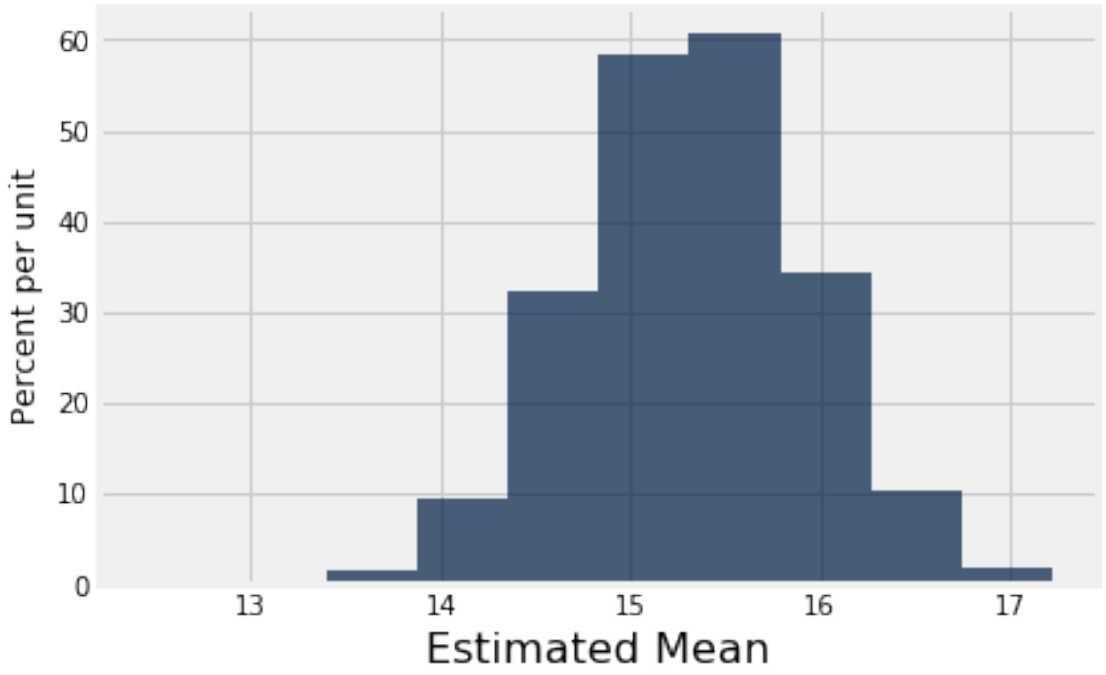
```
In [188]: turkey_mean = np.mean(turkeys.column('Weight (lb)'))
          turkey_mean

Out[188]: 15.309642114028406

In [189]: _ = ok.grade('q2_4')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

You're done! Or are you? You have a single point estimate for the true mean turkey weight. However, you don't know how uncertain your estimate is and you don't know how much these estimates could vary. In other words, you don't have a sense of how good your estimate is. You may have gotten a particular statistic for one sample, but a completely different one for another.

This is where the idea of resampling via the bootstrap comes in. Let's assume that our original sample resembles the population fairly well. We can then resample from our original sample to produce even more estimates, which we can then use to produce an interval estimate for the true mean weight of all turkeys.

Question 5. Fill out the following code to produce 5,000 bootstrapped estimates for the *mean* weight of turkeys. Store your 5,000 estimates in the `turkey_means` array.

```
In [190]: turkey_means = make_array()
          for i in range(5000):
              resample = turkeys.sample(with_replacement=True)
              resample_mean = np.mean(resample.column('Weight (lb)'))
              turkey_means = np.append(turkey_means, resample_mean)

          turkey_means

Out[190]: array([15.76264232, 16.19138488, 14.93124912, ..., 15.82285974,
                 14.94584709, 15.07813662])

In [191]: #: This cell displays a histogram of turkey_means
          Table().with_column('Estimated Mean', turkey_means).hist()
```




```
In [192]: _ = ok.grade('q2_5')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 6. Using the array `means`, compute an approximate 95% confidence interval for the true mean weight of turkeys. (Compute the lower and upper ends of the interval, named `lower_bound` and `upper_bound`, respectively.)

Hint: Use `percentile()`.

```
In [193]: lower_bound = percentile(2.5, turkey_means)
          lower_bound

Out[193]: 14.149131430655984

In [194]: upper_bound = percentile(97.5, turkey_means)
          upper_bound
```

```
Out[194]: 16.509760417974054

In [195]: #: the confidence interval
          print("Bootstrapped 95% confidence interval for the true mean weight of turkeys: [{:f}

Bootstrapped 95% confidence interval for the true mean weight of turkeys: [14.149131, 16.509760]


In [196]: _ = ok.grade('q2_6')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 7. Which of the following would make the histogram narrower? Assign either 1 or 2 to `q2_7`. 1. Increasing the number of resamples (repetitions of bootstrap). 2. Starting with a larger original sample size.

```
In [197]: q2_7 = 2
          q2_7

Out[197]: 2

In [198]: _ = ok.grade('q2_7')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

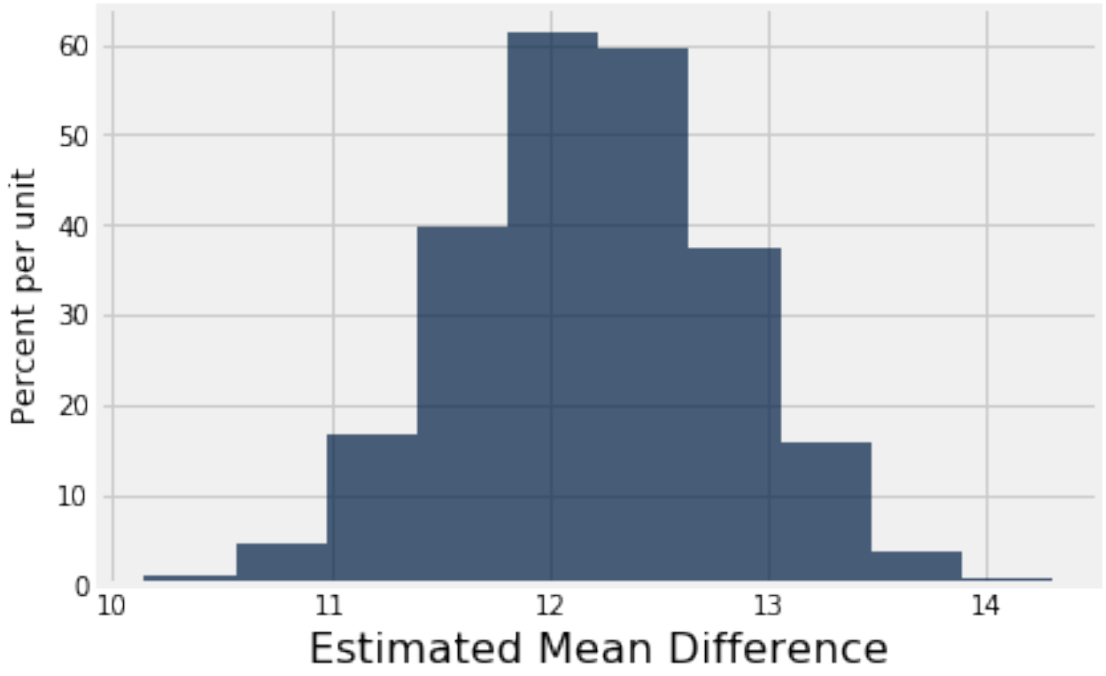
Question 8. Suppose you want to find the weight of the heaviest turkey (maximum weight out of the entire population). Would your bootstrap procedure be effective in estimating the weight of the heaviest turkey? Explain your answer below.

The procedure would not be efficient, since by bootstrapping, the results will always not reach the real value of the heaviest turkey's weight. All the results will have less value of the real biggest weight.

Question 9. Suppose you're wondering how heavy the average turkey is compared to the average chicken. Using the same bootstrap procedure, compute an approximate 95% confidence interval for the true mean difference in weight between turkeys and chickens. Store your 5,000 estimates in the `difference_means` array. Use the original `stock` table for this.

$$\text{mean difference} := \text{mean weight of turkey} - \text{mean weight of chicken}$$

```
In [199]: # You may need to add lines for additional code!
          chickens = stock.where('Fowl', are.equal_to('Chicken'))

          difference_means = make_array()
          for i in range(5000):
              resample_turkey = turkeys.sample(with_replacement=True)
              resample_mean_turkey = np.mean(resample_turkey.column('Weight (lb)'))
              resample_chicken = chickens.sample(with_replacement=True)
              resample_mean_chicken = np.mean(resample_chicken.column('Weight (lb)'))

              difference_means = np.append(difference_means, resample_mean_turkey - resample_mea

          difference_means

Out[199]: array([10.33003024, 11.48559581, 12.63271859, ..., 11.75118935,
                 11.78688656, 12.64114872])

In [200]: #: This cell displays a histogram of difference_means
          Table().with_column('Estimated Mean Difference', difference_means).hist()
```



```
In [201]: _ = ok.grade('q2_9')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 10. Compute the 95% confidence interval for the mean difference in weights of turkeys and chickens. Assign the left and right endpoints to `left_endpoint` and `right_endpoint` respectively.

```
In [202]: left_endpoint = percentile(2.5, difference_means)
          left_endpoint

Out[202]: 10.997345540765405

In [203]: right_endpoint = percentile(97.5, difference_means)
          right_endpoint

Out[203]: 13.398248730349017

In [204]: #: the confidence interval
          print("Bootstrapped 95% confidence interval for the mean difference in weights of turk

Bootstrapped 95% confidence interval for the mean difference in weights of turkeys and chickens:


In [205]: _ = ok.grade('q2_10')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 11: Based on your histogram and confidence interval, would you say with high probability that the mean turkey is heavier than the mean chicken? Explain your answer.

Since in all the results of our 5000 resamples result in the turkey being at least 10.9 pounds (confidence level) heavier than the chicken, we can conclude that turkey is definitively heavier than the chicken.

Question 12. Would changing the units of weight from pounds to kilograms change your conclusion? Assign a boolean (`True` if it would and `False` otherwise) to the name `q2_12`.

```
In [206]: q2_12 = False
          q2_12

Out[206]: False

In [207]: _ = ok.grade('q2_12')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

1.4 3. DSC 10 == Easy?

Here is a questionable article claiming that DSC 10 is the easiest class at UCSD. Suppose you're a toiling DSC 10 student who knows from experience that this claim is not true. To refute this claim, you decide to use A/B testing to show that DSC 10 is in fact more difficult than some of the other classes further down this list, in particular, POLI 5D (Data Analytics for the Social Sciences).

Let's make the assumption that the easier class is the class that is associated with higher grades received. You decide to test the following hypothesis:

Null hypothesis: Grades received from DSC 10 and POLI 5D come from the same distribution. **Alternative hypothesis:** Grades received from DSC 10 are typically lower than those from POLI 5D.

To find evidence against the article's claim, you turn to the Course and Professor Evaluations (CAPE) for historical grades data.

There are a few assumptions we will need to make: - We will disregard P/NP grades, i.e., we will only look at letter grades. - Grades are only shown as unsigned letter grades (e.g. no A+ or A-, only A). Thus, the granularity of our analysis is limited to unsigned letter grades.

Below are the grades given for DSC 10 in Spring 2018:

Below are the grades given for POLI 5D in Winter 2017:

The following cell converts each table into a list of letter grades.

```
In [208]: #: Just run this cell; do not change it
          dsc10_grades  = 33*['A'] + 46*['B'] + 13*['C'] + 2*['D'] + 8*['F']
          poli5d_grades = 32*['A'] + 16*['B'] +  9*['C'] + 3*['D'] + 0*['F']
          print(dsc10_grades)
          print(poli5d_grades)

['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
```

Question 1. Create a new table called `dsc10` with two columns: `Course` and `Letter Grade`. `Course` will contain the string "DSC 10" for all rows and `Letter Grade` will contain the grades from `dsc10_grades`.

Hint: Use list multiplication: `3 * ['element'] = ['element', 'element', 'element']`.

```
In [209]: dsc10 = Table().with_column('Course', len(dsc10_grades)*['DSC 10'], 'Letter Grade', ds
          dsc10

Out[209]: Course | Letter Grade
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          ... (92 rows omitted)

In [210]: _ = ok.grade('q3_1')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 2. Similarly, create a new table called `poli5d` with two columns: `Course` and `Letter Grade`. `Course` will contain the string "POLI 5D" for all rows and `Letter Grade` will contain the grades from `poli5d_grades`.

```
In [211]: poli5d = Table().with_column('Course', len(poli5d_grades)*['POLI 5D'], 'Letter Grade',
          poli5d

Out[211]: Course  | Letter Grade
          POLI 5D | A
          POLI 5D | A
          POLI 5D | A
          POLI 5D | A
          POLI 5D | A
          POLI 5D | A
          POLI 5D | A
```

```
          POLI 5D | A
          POLI 5D | A
          POLI 5D | A
          ... (50 rows omitted)

In [212]: _ = ok.grade('q3_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 3. Create a new table `letter_grades` and append to it both the `poli5d` table and the `dsc10` table.

Hint: `original_tbl.append(new_tbl)` appends `new_tbl` to `old_tbl`, but `old_table` must have all of the columns of `new_tbl`. Try using `letter_grades = Table(['Course', 'Letter Grade'])` to create an empty table with the necessary columns.

Warning: You should be able to run the cell below twice and get the same number of rows each time! If not, you might not be creating a *new* table.

```
In [213]: letter_grades = Table(['Course', 'Letter Grade']).append(dsc10).append(poli5d)
          letter_grades

Out[213]: Course | Letter Grade
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          DSC 10 | A
          ... (152 rows omitted)

In [214]: _ = ok.grade('q3_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
```

```
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Note that `letter_grades` now consists of all 102 rows for DSC 10 followed by all 60 rows for POLI 5D (or *vice versa*, depending on your particular solution). In other words, the `Course` column is not randomly shuffled; all rows for DSC 10 come before all rows for POLI 5D. In your upcoming permutation test, you will shuffle the `Letter Grade` column but will keep `Course` in the same order.

Question 4. Is it a problem that the `Course` column will never be shuffled in your permutation test? Explain your answer below.

Yes. Because the 'Course' column is not shuffled along with the 'Letter Grade' column, each letter grade in the 'Letter Grade' column will not match its correspoding course in the 'Course' column. So the results of the experiments would be useless, since the data got mixed up and no meaningful results would come out

There is one last thing we need to take care of before we can start our permutation test. As mentioned earlier, grades are only shown as letter grades, not numerical grades. However, working with numerical grades rather than letter grades will allow us to use the permutation test more effectively. This is because we can take the mean of numerical grades while we cannot with letter grades.

Therefore, we will map the letter grades in our data set to placeholder numerical grades according to the following mapping:

Letter Grade	Placeholder Numerical Grade
A	4.0
B	3.0
C	2.0
D	1.0
F	0.0

Note that these placeholder numerical grades are just approximations of the actual numerical grades received. For example, a student under the "A" category may have actually received an A-, which corresponds to 3.7 grade points, rather than a 4.0. Nevertheless, we will make do with what we have.

Question 5. Define a function `letter_to_numerical` that takes in a letter grade as a string as input, and returns the corresponding placeholder numerical grade according to the mapping above.

```
In [215]: # write your function here
          def letter_to_numerical(letter_grade):
              if letter_grade == 'A':
                  return 4.0
              elif letter_grade == 'B':
                  return 3.0
              elif letter_grade == 'C':
```

```
            return 2.0
        elif letter_grade == 'D':
            return 1.0
        elif letter_grade == 'F':
            return 0.0
```

```
In [216]: _ = ok.grade('q3_5')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6. Using your function `letter_to_numerical`, create a new table `numerical_grades` with two columns: `Course` and `Numerical Grade`. `Course` will contain the same values as in the `letter_grades` table and `Numerical Grade` will contain corresponding placeholder numerical grades.

```
In [217]: numberss = letter_grades.apply(letter_to_numerical, 'Letter Grade')
          numerical_grades = letter_grades.select('Course').with_column('Numerical Grade', numbe
          numerical_grades
```

```
Out[217]: Course | Numerical Grade
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          DSC 10 | 4
          ... (152 rows omitted)
```

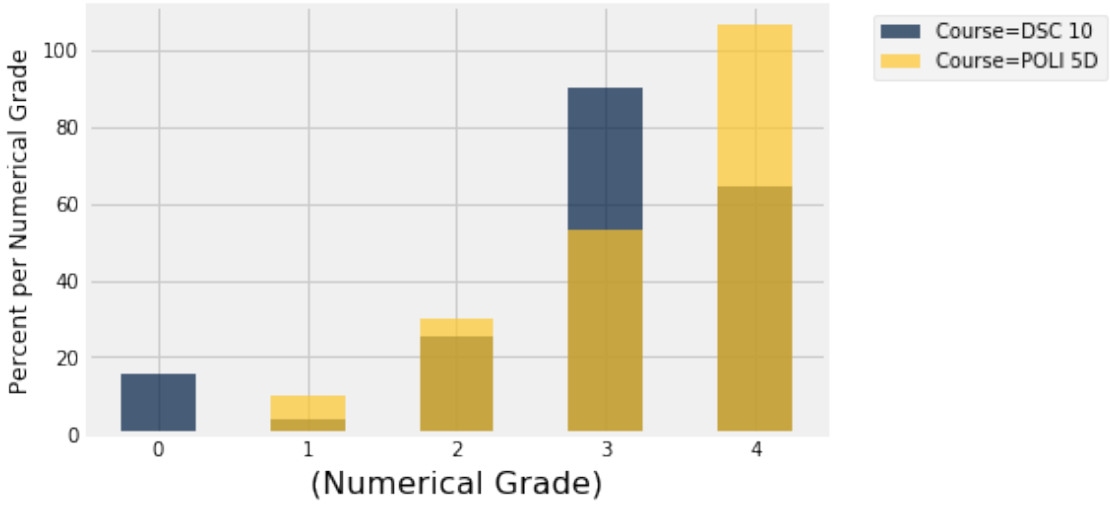
```
In [218]: _ = ok.grade('q3_6')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
```

```
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 7. Run the below cell to display a histogram of `numerical_grades`. Why are there gaps in-between the bars? Explain your answer below.

```
In [219]: #: Just run this cell; do not change it
          numerical_grades.hist('Numerical Grade', unit='Numerical Grade', group='Course', bins=
```



Because in the bins it is specified to hve a 0.5 gap between each bar

Question 8. Using `letter_grades`, calculate the difference between the mean DSC 10 grade and the mean POLI 5D grade. Assign your answer to `observed_difference`.

$$\text{mean difference} := \text{mean DSC 10 grade} - \text{mean POLI 5D grade}$$

```
In [220]: dsc_mean = np.mean(numerical_grades.where('Course', are.equal_to('DSC 10')).column('Nu
          poli_mean = np.mean(numerical_grades.where('Course', are.equal_to('POLI 5D')).column('
          observed_difference = dsc_mean - poli_mean
          observed_difference

Out[220]: -0.361764705882353

In [221]: _ = ok.grade('q3_8')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
```

```
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 9. Interpret in words the number you obtained for `observed_difference`. Explain your answer below. Be as specific as possible.

observed_difference represents the mean of the grades from dsc 10 minus the mean of the grades from poli 5d. Being it a negative number, it tells us that the mean of the grades of dsc 10 is lower than the mean of the grades from poli 5d. This means that the average grade of dsc 10 is lower than the average grade of poli 5d. Suggesting that dsc 10 has a lower average grade than poli 5d.

Question 10. Use a permutation test to calculate 5,000 differences using random permutations of the data. Store your 5,000 differences in the `differences` array.

```
In [222]: differences = make_array()
          for i in range(5000):
              shuffled_grades = numerical_grades.sample(with_replacement=False).column('Numerica
              shuffled = numerical_grades.with_column('Shuffled Numerical Grade', shuffled_grade
              group_means = shuffled.group('Course', np.mean).column('Shuffled Numerical Grade m

              difference = group_means.item(1) - group_means.item(0)
              differences = np.append(differences, difference)

          differences

Out[222]: array([-0.11470588,  0.04411765,  0.09705882, ..., -0.16764706,
                  0.04411765, -0.16764706])

In [223]: _ = ok.grade('q3_10')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

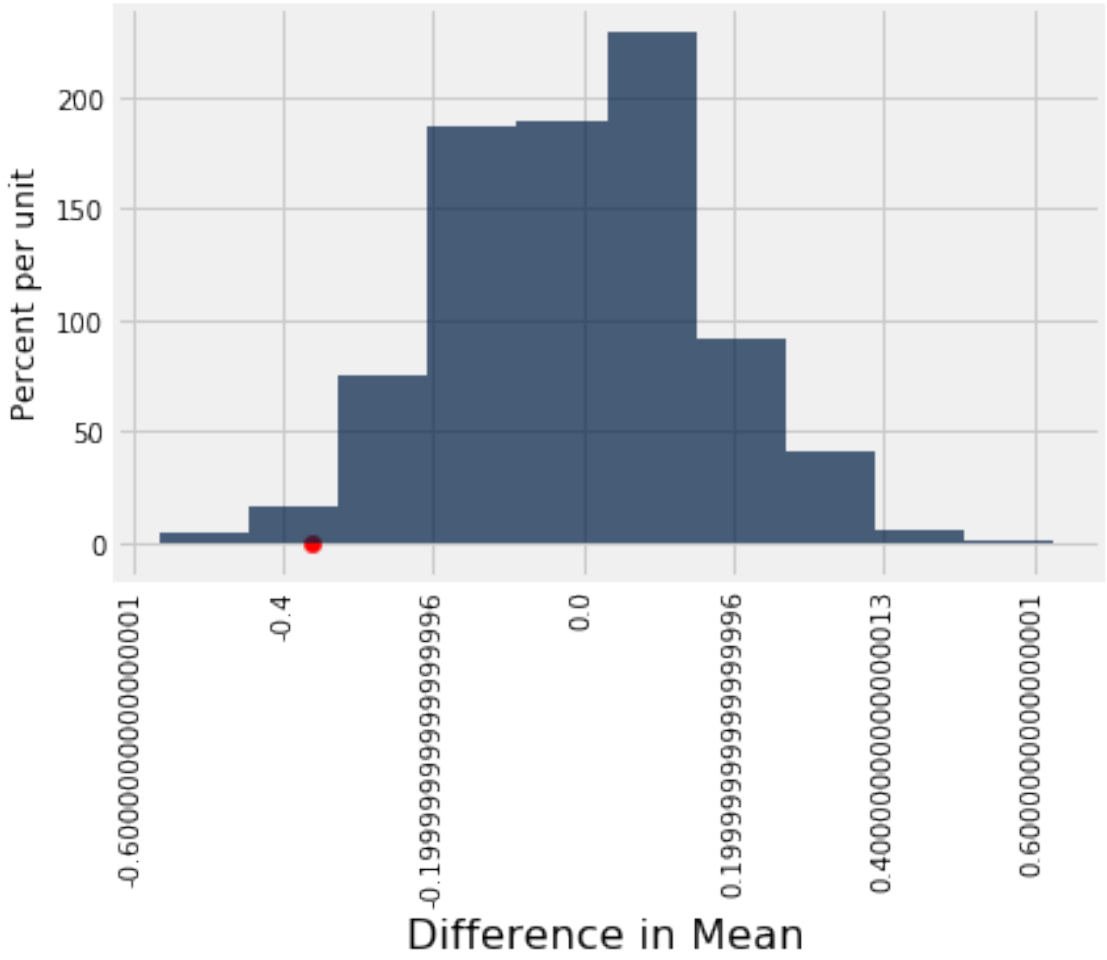
Question 11. Which of the follow choices best describes the purpose of the permutation test with regards to A/B testing? Assign either 1, 2, or 3 to `q3_11`. 1. The permutation test generates a null distribution which we can use in testing our hypothesis. 2. The permutation test mitigates noise in our data by generating new permutations of the data. 3. The permutation test is a special case of the bootstrap and allows us to produce interval estimates.

```
In [224]: q3_11 = 2
          q3_11
```

```
Out[224]: 2
```

The following cell plots your observed difference and a histogram of your null distribution produced under your permutation tests.

```
In [225]: #: Just run this cell; do not change it
          Table().with_column('Difference in Mean', differences).hist()
          plt.scatter(observed_difference, 0, color='red', s=40);
```




```
In [226]: _ = ok.grade('q3_11')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 12. Compute a p-value for the hypothesis. That is, under the null hypothesis, compute the probability that we would have obtained a difference equal to or lower than `observed_difference` by chance alone. Assign your answer to `p_val`.

```
In [227]: p_val = np.count_nonzero(differences <= observed_difference) / 5000
          p_val

Out[227]: 0.0178

In [228]: _ = ok.grade('q3_12')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 13. Do you reject or fail to reject the null hypothesis at the 0.05 significance level? What conclusion can you make with regards to DSC 10 and POLI 5D? Explain your answer below. Be as specific as possible.

At the 0.05 significance level, we reject the null hypothesis in favor to the alternative hypothesis. We can conclude that grades received from DSC 10 are typically lower than those from POLI 5D.

Question 14. Suppose you are interested in two values, value A and value B. Suppose your null hypothesis is as follows:

Null hypothesis: In the population, value A is equal to value B.

Suppose your observed difference (value A − value B) lies in the far left tail of the null distribution.

Which alternative hypothesis will most likely result in a p-value that is NOT significant (a p-value NOT less than 0.05)? Assign either 1, 2, or 3 to `q3_14`.

1. **Alternative hypothesis:** There is a difference between value A and value B.
2. **Alternative hypothesis:** Value A is greater than value B.
3. **Alternative hypothesis:** Value B is greater than value A.

```
In [229]: q3_14 = 1
          q3_14
```

```
Out[229]: 1

In [230]: _ = ok.grade('q3_14')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

To submit:

1. Select `Kernel -> Restart & Run All` to ensure that you have executed all cells, including the test cells.
2. Read through the notebook to make sure everything is fine and all tests passed.
3. Submit using the cell below.
4. Save PDF and submit to gradescope

```
In [231]: #: Run all tests at once
          import os
          _ = [ok.grade(q[:-3]) for q in os.listdir('tests') if q.startswith('q')]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
```

```
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
```

```
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 2
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Running tests

--------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

--------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

--------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

--------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

--------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

--------------------------------------------------------------------
```

```
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
```

```
[ooooooooook] 100.0% passed
```

1.5 Before submitting, select "Kernel" -> "Restart & Run All" from the menu!

Then make sure that all of your cells ran without error.

```
In [232]: #: Submit your notebook
          _ = ok.submit()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

```
Saving notebook... Saved 'hw07.ipynb'.
Submit... 100% complete
Submission successful for user: wec149@ucsd.edu
URL: https://okpy.org/ucsd/dsc10/fa18/hw07/submissions/VP4Jkz
```

1.6 Don't forget to submit to both OK and Gradescope!