Final Exam - DSC 10, Winter 2023

Instructions:

- This exam consists of 16 questions, worth a total of 164 points.
- Write your PID in the top right corner of each page in the space provided.
- Please write **clearly** in the provided answer boxes; we will not grade work that appears elsewhere. Completely fill in bubbles and square boxes; if we cannot tell which option(s) you selected, you may lose points.

  ◯ A bubble means that you should only **select one choice**.

  ☐ A square box means you should **select all that apply**.
- Solve all problems using **the methods of this course**.
- You may refer to the Reference Sheet that was provided to you. No calculators, computers, notes, or other aids may be used.

Full Name:

PID:

Seat you are in:

By signing below, you are agreeing that you will behave honestly and fairly during and after this exam.

Signature:

Version A

Please do not open your exam until instructed to do so, but you may read the data description attached to the Reference Sheet.

Make sure you have read the data description attached to the Reference Sheet!

Question 1 (8 pts)

Let's start by correcting the data in the `"Rating"` column. All the values in this column are strings of length 4. In addition, some strings use commas in place of a dot to represent a decimal point. Select the options below that evaluate to a Series containing the values in the `"Rating"` column, appropriately changed to floats.

Note: The Series method `.str.replace` uses the string method `.replace` on every string in a Series.

☐ `float(games.get("Rating").str.replace(",", "."))`

☐ `games.get("Rating").str.replace(",", ".").apply(float)`

☐ `games.get("Rating").str.replace(",", "").apply(int)/100`

☐ `games.get("Rating").str.replace(",", "").apply(float)`

Important! For the rest of this exam, we will assume that the values in the `"Rating"` column have been correctly changed to floats.

Question 2 (8 pts)

You are unsure whether it would make sense to use `"BGG Rank"` as the index of the `games` DataFrame, because you are unsure whether this column can have duplicate values. Perhaps, for example, two games are tied and both have a rank of 6.

Select all of the expressions below that evaluate to True when the`"BGG Rank"` column could be used as the index (no duplicates), and False when it could not be used as the index (duplicates). In other words, these are the expressions that can be used to detect the presence of duplicate values.

☐ `(games.get("BGG Rank") - np.arange(games.shape[0])).max() == 1`

☐ `games.groupby("BGG Rank").count().get("Name").max() == 1`

☐ `games.shape[0] - len(np.unique(games.get("BGG Rank"))) == 0`

☐ `games.get("BGG Rank").max() - games.shape[0] == 0`

Note: We will not set the index of `games`, instead we'll leave it with the default index.

PID: _____

Question 3 (14 pts)

Notice that `"Strategy Games"` and `"Thematic Games"` are two of the possible domains, and that a game can belong to multiple domains.

Define the variables `strategy` and `thematic` follows.

```
strategy = games.get("Domains").str.contains("Strategy Games")
thematic = games.get("Domains").str.contains("Thematic Games")
```

a) What is the data type of `strategy`?

◯ `bool` ◯ `str` ◯ `Series` ◯ `DataFrame`

b) Suppose we randomly select one of the `"Strategy Games"` from the `games` DataFrame. What is the probability that the randomly selected game is **not** also one of the `"Thematic Games"`? In the space below, write a single line of Python code that evaluates to this probability, using the variables `strategy` and `thematic` in your solution.

Note: For this question and others that require one line of code, it's fine if you need to write your solution on multiple lines, as long as it is just one line of Python code. Please do write on multiple lines to make sure your answer fits within the box provided.

c) Many of the games in the `games` DataFrame belong to more than one domain. We want to identify the number of games that belong to only one domain. Select all of the options below that would correctly calculate the number of games that belong to only one domain.

Hint: Don't make any assumptions about the possible domains.

☐ `(games.get("Domains").str.split(" ").apply(len) == 2).sum()`

☐ `(games.get("Domains").apply(len) == 1).sum()`

☐
```
(games[games.get("Domains").str.split(",").apply(len) == 1]
.groupby("Domains").count().get("Name").sum())
```

☐ `games[games.get("Domains").str.split(",").apply(len) == 1].shape[0]`

Question 4 (8 pts)

We want to create a bootstrapped 95% confidence interval for the median `"Complexity"` of all cooperative games, given a sample of 100 cooperative games.

Suppose `coop_sample` is a DataFrame containing 100 rows of `games`, all of which are cooperative games. We'll calculate the endpoints `left` and `right` of our bootstrapped 95% confidence interval as follows.

```
medians = np.array([])
for i in np.arange(10000):
    resample = coop_sample.sample(100, replace=True)
    median = np.median(resample.get("Complexity"))
    medians = np.append(medians, median)
left = np.percentile(medians, 2.5)
right = np.percentile(medians, 97.5)
```

a) Now consider the interval defined by the endpoints `left_2` and `right_2`, calculated as follows.

```
medians_2 = np.array([])
for i in np.arange(10000):
    shuffle = coop_sample.assign(shuffle=
              np.random.permutation(coop_sample.get("Complexity")))
    resample_2 = shuffle.sample(100, replace=True)
    median_2 = np.median(resample_2.get("shuffle"))
    medians_2 = np.append(medians_2, median_2)
left_2 = np.percentile(medians_2, 2.5)
right_2 = np.percentile(medians_2, 97.5)
```

Which interval should be wider, `[left, right]` or `[left_2, right_2]`?

◯ `[left, right]` ◯ `[left_2, right_2]` ◯ Both about the same.

b) Now consider the interval defined by the endpoints `left_3` and `right_3`, calculated as follows.

```
medians_3 = np.array([])
for i in np.arange(10000):
    resample_3 = (coop_sample.sample(100, replace=True)
                             .sample(100, replace=True))
    median_3 = np.median(resample_3.get("Complexity"))
    medians_3 = np.append(medians_3, median_3)
left_3 = np.percentile(medians_3, 2.5)
right_3 = np.percentile(medians_3, 97.5)
```

Which interval should be wider, `[left, right]` or `[left_3, right_3]`?

◯ `[left, right]` ◯ `[left_3, right_3]` ◯ Both about the same.

Question 5 (12 pts)

As in the previous question, let `coop_sample` be a sample of 100 rows of `games`, all corresponding to cooperative games.

Define `samp` and `resamp` as follows.

```
samp = coop_sample.get("Complexity")
resamp = coop_sample.sample(100, replace=True).get("Complexity")
```

a) Which of the following statements **could** evaluate to `True`? Select all that are possible.

- ☐ `len(samp.unique()) < len(resamp.unique())`
- ☐ `len(samp.unique()) == len(resamp.unique())`
- ☐ `len(samp.unique()) > len(resamp.unique())`

b) Which of the following statements **could** evaluate to `True`? Select all that are possible.

- ☐ `np.count_nonzero(samp == 1) < np.count_nonzero(resamp == 1)`
- ☐ `np.count_nonzero(samp == 1) == np.count_nonzero(resamp == 1)`
- ☐ `np.count_nonzero(samp == 1) > np.count_nonzero(resamp == 1)`

c) Which of the following statements **could** evaluate to `True`? Select all that are possible.

- ☐ `samp.min() < resamp.min()`
- ☐ `samp.min() == resamp.min()`
- ☐ `samp.min() > resamp.min()`

d) Which of the following statements **could** evaluate to `True`? Select all that are possible.

- ☐ `np.std(samp) < np.std(resamp)`
- ☐ `np.std(samp) == np.std(resamp)`
- ☐ `np.std(samp) > np.std(resamp)`

Question 6 (8 pts)

Choose the best tool to answer each of the following questions. Note the following:

- By "hypothesis testing," we mean "standard" hypothesis testing, i.e. hypothesis testing that **doesn't** involve permutation testing or bootstrapping.
- By "bootstrapping," we mean bootstrapping that **doesn't** involve hypothesis testing.

a) Are strategy games rated higher than non-strategy games?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

b) What is the mean complexity of all games?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

c) Are there an equal number of cooperative and non-cooperative games?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

d) Are games with more than one domain more complex than games with one domain?

◯ Hypothesis testing ◯ Permutation testing ◯ Bootstrapping

PID: _____

Question 7 (8 pts)

We use the regression line to predict a game's `"Rating"` based on its `"Complexity"`. We find that for the game *Wingspan*, which has a `"Complexity"` that is 2 points higher than the average, the predicted `"Rating"` is 3 points higher than the average.

a) What can you conclude about the correlation coefficient r?

- ◯ $r < 0$
- ◯ $r = 0$
- ◯ $r > 0$
- ◯ We cannot make any conclusions about the value of r based on this information alone.

b) What can you conclude about the standard deviations of `"Complexity"` and `"Rating"`?

- ◯ SD of `"Complexity"` < SD of `"Rating"`
- ◯ SD of `"Complexity"` = SD of `"Rating"`
- ◯ SD of `"Complexity"` > SD of `"Rating"`
- ◯ We cannot make any conclusions about the relationship between these two standard deviations based on this information alone.

Question 8 (4 pts)

Suppose that for children's games, `"Play Time"` and `"Rating"` are negatively linearly associated due to children having short attention spans. Suppose that for children's games, the standard deviation of `"Play Time"` is twice the standard deviation of `"Rating"`, and the average `"Play Time"` is 10 minutes.

We use linear regression to predict the `"Rating"` of a children's game based on its `"Play Time"`. The regression line predicts that *Don't Break the Ice*, a children's game with a `"Play Time"` of 8 minutes will have a `"Rating"` of 4. Which of the following could be the average `"Rating"` for children's games?

◯ 2 ◯ 2.8 ◯ 3.1 ◯ 4

Question 9 (15 pts)

The function `perm_test` should take three inputs:

1. `df`, a DataFrame.
2. `labels`, a string. The name of a column in `df` that contains two distinct values, which signify the groups in a permutation test.
3. `data`, a string. The name of a column in `df` that contains numerical data.

The function should return an array of 1000 simulated differences of group means, under the assumption of the null hypothesis in a permutation test, namely that data in both groups come from the same population.

The smaller of the two group labels should be first in the subtraction. For example, if the two values in the `labels` column are `"dice game"` and `"card game"`, we would compute the difference as the mean of the `"card game"` group minus the mean of the `"dice game"` group, because `"card game"` comes before `"dice game"` alphabetically. Note that **groupby** orders the rows in ascending order by default.

An incorrect implementation of `perm_test` is provided below.

```
1 def perm_test(df, labels, data):
2     diffs = np.array([])
3     for i in np.arange(1000):
4         df.assign(shuffled=np.random.permutation(df.get(data)))
5         means = df.groupby(labels).mean().get(data)
6         diff = means.iloc[0] - means.iloc[1]
7         diffs = np.append(diffs, diff)
8         return diffs
```

Three lines of code above are incorrect. Your job is to identify which lines of code are incorrect, and describe briefly in English how you would fix them. You don't need to explain why the current code is wrong, just how to fix it.

PID: _____

a) The first line that is incorrect is line number [ ].

Explain in one sentence how to change this line. Do not write code.

b) The second line that is incorrect is line number [ ].

Explain in one sentence how to change this line. Do not write code.

c) The third line that is incorrect is line number [ ].

Explain in one sentence how to change this line. Do not write code.

Suppose you've fixed all the issues with this function, as you described above. Now, you want to use this corrected function to run a permutation test with the following hypotheses:

- **Null Hypothesis**: The play time of children's dice-rolling games and the play time of children's non-dice-rolling games come from the same distribution.
- **Alternative Hypothesis**: The play time of children's dice-rolling games is lower, on average, than the play time of children's non-dice-rolling games.

For this permutation test, consider a children's game to be a game that has `"Children's Games"` as part of the `"Domains"` column. A dice-rolling game is one that has `"Dice Rolling"` as part of the `"Mechanics"` column, and a non-dice-rolling game is one that does not have `"Dice Rolling"` as part of the `"Mechanics"` column.

The DataFrame `with_dice` is defined as follows.

```
with_dice = games.assign(isDice = games.get("Mechanics")
                                        .str.contains("Dice Rolling"))
```

d) Write one line of code that creates an array called `simulated_diffs` containing 1000 simulated differences in group means for this permutation test. You should call your `perm_test` function here!

e) Suppose we've stored the observed value of the test statistic for this permutation test in the variable `obs_diff`. Fill in the blank below to find the p-value for this permutation test.

```
p_value = np.count_nonzero(simulated_diffs ____ obs_diff)/1000
```

What goes in the blank?

◯ `<` ◯ `<=` ◯ `>` ◯ `>=`

PID: _____

Question 10 (8 pts)

It's your first time playing a new game called *Brunch Menu*. The deck contains 96 cards, and each player will be dealt a hand of 9 cards. The goal of the game is to avoid having certain cards, called *Rotten Egg* cards, which come with a penalty at the end of the game. But you're not sure how many of the 96 cards in the game are *Rotten Egg* cards. So you decide to use the Central Limit Theorem to estimate the proportion of *Rotten Egg* cards in the deck based on the 9 random cards you are dealt in your hand.

a) You are dealt 3 *Rotten Egg* cards in your hand of 9 cards. You then construct a CLT-based 95% confidence interval for the proportion of *Rotten Egg* cards in the deck based on this sample. Approximately, how wide is your confidence interval?

Choose the closest answer, and use the following facts:

- The standard deviation of a collection of 0s and 1s is

$$\sqrt{(\text{Prop. of 0s}) \times (\text{Prop. of 1s})}.$$

- $\sqrt{18}$ is about $\frac{17}{4}$.

◯ $\frac{17}{9}$ ◯ $\frac{17}{27}$ ◯ $\frac{17}{81}$ ◯ $\frac{17}{96}$

b) Which of the following are limitations of trying to use the Central Limit Theorem for this particular application? Select all that apply.

☐ The CLT is for large random samples, and our sample was not very large.

☐ The CLT is for random samples drawn with replacement, and our sample was drawn without replacement.

☐ The CLT is for normally distributed data, and our data may not have been normally distributed.

☐ The CLT is for sample means and sums, not sample proportions.

Question 11 (9 pts)

In recent years, there has been an explosion of board games that teach computer programming skills, including *CoderMindz*, *Robot Turtles*, and *Code Monkey Island*. Many such games were made possible by Kickstarter crowdfunding campaigns.

Suppose that in one such game, players must prove their understanding of functions and conditional statements by answering questions about the function `wham`, defined below. Like players of this game, you'll also need to answer questions about this function.

```
1  def wham(a,b):
2      if a < b:
3          return a + 2
4      if a + 2 == b:
5          print(a + 3)
6          return b + 1
7      elif a - 1 > b:
8          print(a)
9          return a + 2
10     else:
11         return a + 1
```

a) What is printed when we run `print(wham(6, 4))`?

b) Give an example of a pair of integers `a` and `b` such that `wham(a, b)` returns `a + 1`.

`a` = _____ , `b` = _____

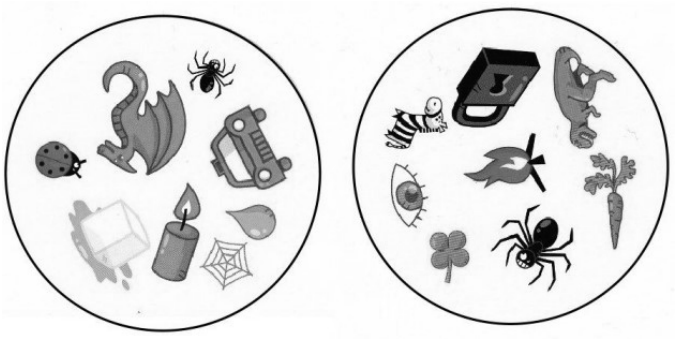
c) Which of the following lines of code will never be executed, for any input?

◯ 3 ◯ 6 ◯ 9 ◯ 11

PID: _____

Question 12 (8 pts)

In the game *Spot It*, players race to identify an object that appears on two different cards. Each card contains images of eight objects, and exactly one object is common to both cards.



a) Suppose the objects appearing on each card are stored in an array, and our task is to find the object that appears on both cards. Complete the function `find_match` that takes as input two arrays of 8 objects each, with one object in common, and returns the name of the object in both arrays.

For example, suppose we have two arrays defined as follows.

```
objects1 = np.array(["dragon", "spider", "car", "water droplet",
                     "spiderweb", "candle", "ice cube", "ladybug"])
objects2 = np.array(["zebra", "lock", "dinosaur", "eye", "fire",
                     "shamrock", "spider", "carrot"])
```

Then `find_match(objects1, objects2)` should evaluate to `"spider"`.

Your function must include a `for` loop, and it must take **at most three lines of code** (not counting the line with `def`).

```
def find_match(array1, array2):
    for obj in array1:
        if obj in array2:
            return obj
```

b) Now suppose the objects appearing on each card are stored in a DataFrame with 8 rows and one column called `"object"`. Complete the function `find_match_again` that takes as input two such DataFrames with one object in common and returns the name of the object in both DataFrames.

Your function may not call the previous function `find_match`, and it must take **exactly one line of code** (not counting the line with `def`).

```
def find_match_again(df1, df2):
    return df1.merge(df2, on="item").get("item").iloc[0]
```

Question 13 (16 pts)

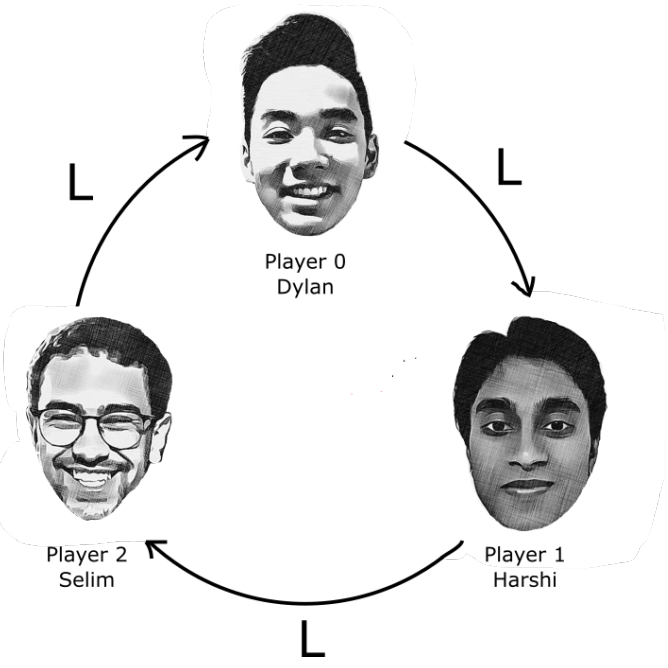
Dylan, Harshi, and Selim are playing a variant of a dice game called *Left, Center, Right (LCR)* in which there are 9 chips (tokens) and 9 dice. Each player starts off with 3 chips. Each die has the following six sides: L, C, R, Dot, Dot, Dot.

During a given player's turn, they must roll a number of dice equal to the number of chips they currently have. Each die determines what to do with one chip:

1. L means give the chip to the player on their left.
2. R means give the chip to the player on their right.
3. C means put the chip in the center of the table. This chip is now out of play.
4. Dot means do nothing (or keep the chip).

Since the number of dice rolled is the same as the number of chips the player has, the dice rolls determine exactly what to do with each chip. There is no strategy at all in this simple game.

Dylan will take his turn first (we'll call him Player 0), then at the end of his turn, he'll pass the dice to his left and play will continue clockwise around the table. Harshi (Player 1) will go next, then Selim (Player 2), then back to Dylan, and so on.



Note that if someone has no chips when it's their turn, they are still in the game and they still take their turn, they just roll 0 dice because they have 0 chips. The game ends when only one person is holding chips, and that person is the winner. If 300 turns have been taken (100 turns each), the game will end and we'll declare it a tie.

PID: _____

The function `simulate_lcr` below simulates one full game of *Left, Center, Right* and returns the number of turns taken in that game. Some parts of the code are not provided. You will need to fill in the code for the parts marked with a blank. The parts marked with `...` are not provided, but you don't need to fill them in because they are very similar to other parts that you do need to complete.

Hint: Recall that in Python, the % operator gives the remainder upon division. For example 12 % 5 is 2.

```
def simulate_lcr():
    # stores the number of chips for players 0, 1, 2 (in that order)
    player_chips = np.array([3,3,3])

    # maximum of 300 turns allotted for the game
    for i in np.arange(300):

        # which player's turn it is currently (0, 1, or 2)
        current_player = __(a)__

        # stores what the player rolled on their turn
        roll = np.random.choice(["L", "C", "R", "Dot", "Dot", "Dot"],
                                __(b)__)

        # count the number of instances of L, C, and R
        L_count = __(c)__
        C_count = ...
        R_count = ...

        if current_player == 0:
            # update player_chips based on what player 0 rolled
            player_chips = player_chips + np.array(__(d)__)
        elif current_player == 1:
            # update player_chips based on what player 1 rolled
            player_chips = player_chips + ...
        else:
            # update player_chips based on what player 2 rolled
            player_chips = player_chips + ...

        # if the game is over, return the number of turns played
        if __(e)__:
            return __(f)__

    # if no one wins after 300 turns, return 300
    return 300
```

a) What goes in blank (a)?

b) What goes in blank (b)?

c) What goes in blank (c)?

d) What goes in blank (d)?

e) What goes in blank (e)?

f) What goes in blank (f)?

PID: _____

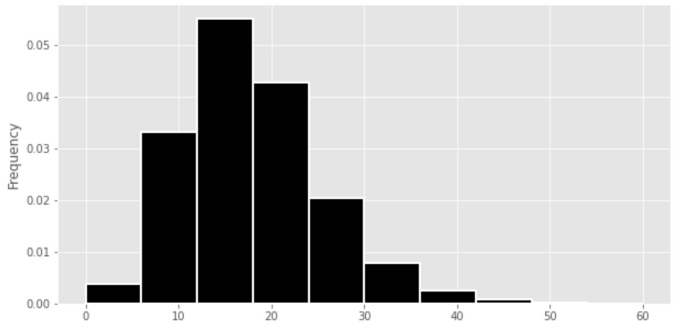
Question 14 (14 pts)

Suppose the function `simulate_lcr` from the last question has been correctly implemented, and we want to use it to see how many turns a game of *Left, Center, Right* usually takes.

Note: You can answer this question even if you couldn't answer the previous one.

a) Consider the code and histogram below.

```
turns = np.array([])
for i in np.arange(10000):
    turns = np.append(turns, simulate_lcr())
(bpd.DataFrame().assign(turns=turns)
                .plot(kind="hist", density = True,
                      ec="w", bins = np.arange(0, 66, 6)))
```



(i) Does this histogram show a probability distribution or an empirical distribution?

◯ Probability distribution ◯ Empirical distribution

(ii) What is the probability of a game of *Left, Center, Right* lasting 30 turns or more? Choose the closest answer below.

◯ 0.01 ◯ 0.06 ◯ 0.10 ◯ 0.60

b) Suppose a player with n chips takes their turn. What is the probability that they will have to put at least one chip into the center? Give your answer as a mathematical expression involving n.

c) Suppose a player with n chips takes their turn. What is the probability that they will end their turn with n chips? Give your answer as a mathematical expression involving n.

Question 15 (12 pts)

At a recent game night, you played several new board games and liked them so much that you now want to buy copies for yourself.

The DataFrame `stores` is shown below in full. Each row represents a game you want to buy and a local board game store where that game is available for purchase. If a game is not available at a certain store, there will be no row corresponding to that store and that game.

	Store	Game
0	Off the Shelf	Ticket to Ride
1	Game Empire	Ticket to Ride
2	Bards & Cards	Hanabi
3	Off the Shelf	Hanabi
4	Game Empire	Hanabi
5	Brute Force	Ingenious
6	Off the Shelf	Ingenious
7	Bards & Cards	Exploding Kittens
8	Off the Shelf	Exploding Kittens
9	Game Empire	Exploding Kittens

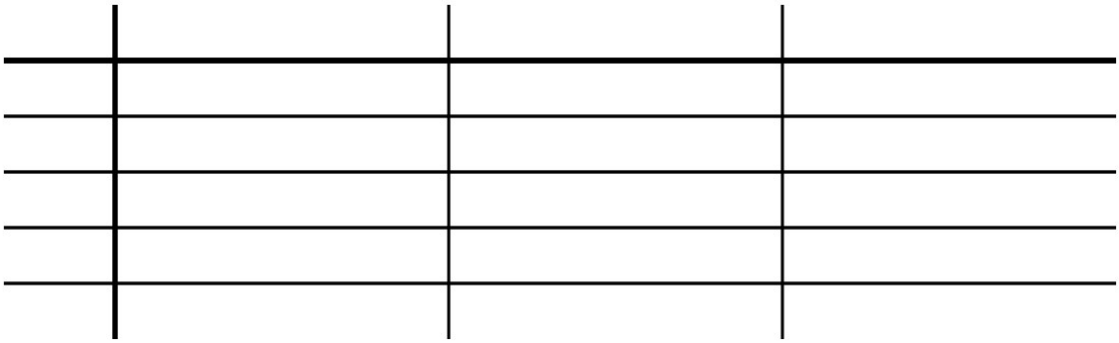
a) The DataFrame `prices` has **five rows**. Below we merge `stores` with `prices` and display the output in full.

```
merged = stores.merge(prices, on="Game")
merged
```

	Store	Game	Price
0	Off the Shelf	Ticket to Ride	47.99
1	Game Empire	Ticket to Ride	47.99
2	Bards & Cards	Hanabi	10.99
3	Off the Shelf	Hanabi	10.99
4	Game Empire	Hanabi	10.99
5	Brute Force	Ingenious	29.99
6	Off the Shelf	Ingenious	29.99
7	Bards & Cards	Exploding Kittens	19.99
8	Off the Shelf	Exploding Kittens	19.99
9	Game Empire	Exploding Kittens	19.99

In the space below, specify what the DataFrame `prices` **could** look like. The column labels should go in the top row, and the row labels (index) should go in the leftmost row. You may not need to use all the columns provided, but you are told that `prices` has five rows, so you should use all rows provided.

Note: There are several correct answers to this question.



b) Suppose `merged` now contains all the available games and their corresponding prices at each store (in particular, a given game is sold for the same price at all stores). You want to buy as many games as possible but you only want to go to one store. Which store should you go to maximize the number of games you can buy?

Fill in the blanks so that `where_to_go` evaluates to the name of the store you should buy your games from.

```
where_to_go = (merged.groupby("Store").__(a)__
                     .sort_values(by="Price", ascending=False)
                     .__(b)__)
```

(i) What goes in blank (a)?

◯ `min()` ◯ `max()` ◯ `count()` ◯ `mean()`

(ii) What goes in blank (b)?

c) Suppose you go to the store `where_to_go` and buy one copy of each of the available games that you enjoyed at game night. How much money will you spend? Write one line of code that evaluates to the answer, using the `merged` DataFrame and no others.

Question 16 (12 pts)

We collect data on the play times of 100 games of *Chutes and Ladders* (sometimes known as *Snakes and Ladders*) and want to use this data to perform a hypothesis test.

a) Which of the following pairs of hypotheses can we test using this data?

◯ **Null Hypothesis**: In a random sample of *Chutes and Ladders* games, the average play time is 30 minutes.
Alternative Hypothesis: In a random sample of *Chutes and Ladders* games, the average play time is not 30 minutes.

◯ **Null Hypothesis**: In a random sample of *Chutes and Ladders* games, the average play time is not 30 minutes.
Alternative Hypothesis: In a random sample of *Chutes and Ladders* games, the average play time is 30 minutes.

◯ **Null Hypothesis**: A game of *Chutes and Ladders* takes, on average, 30 minutes to play.
Alternative Hypothesis: A game of *Chutes and Ladders* does not take, on average, 30 minutes to play.

◯ **Null Hypothesis**: A game of *Chutes and Ladders* does not take, on average, 30 minutes to play.
Alternative Hypothesis: A game of *Chutes and Ladders* takes, on average, 30 minutes to play.

b) We use our collected data to construct a 95% CLT-based confidence interval for the average play time of a game of *Chutes and Ladders*. This 95% confidence interval is $[26.47, 28.47]$. For the 100 games for which we collected data, what is the mean and standard deviation of the play times?

mean = $\boxed{}$, SD = $\boxed{}$

c) Does the CLT say that the distribution of play times of the 100 games is roughly normal?

◯ Yes ◯ No

d) Of the two hypotheses you selected in part (a), which one is better supported by the data?

◯ Null Hypothesis ◯ Alternative Hypothesis

PID: _____

Feel free to draw us a picture or tell us about your fondest memory from DSC 10 this quarter.

Before turning in your exam, please make sure that your PID is on every page.

Congratulations on finishing DSC 10!

Games

The DataFrame `games` contains information about a sample of popular games, including board games, dice games, and card games. The data comes from Board Game Geek, a popular website and vibrant online community for game enthusiasts.

The columns of `games` are as follows.

- `"Name" (str)`: The name of the game.
- `"Mechanics" (str)`: A sequence of descriptors for how the game is played. A game can have several descriptors, each of which is separated by a comma.
- `"Domains" (str)`: A sequence of domains to which the game belongs. A game can belong to multiple domains.
- `"Play Time" (int)`: The average play time of the game, in minutes, as suggested by the game's creators.
- `"Complexity" (float)`: The average complexity of the game, on a scale of 1 to 5 points, as reported by Board Game Geek community members.
- `"Rating" (str)`: The average rating of the game, on a scale of 1 to 10 points, as rated by Board Game Geek community members. Note that while this data should be numerical, it is actually stored as a string, because some entries use a comma in place of a decimal point. For example 8,79 actually represents the number 8.79.
- `"BGG Rank" (int)`: The rank of the game in Board Game Geek's database. The formula for how this rank is calculated is not publicly known, but it likely includes many factors, such as `"Rating"`, number of registered owners of the game, and number of reviews.

The first few rows of `games` are shown below (though `games` has many more rows than pictured here).

	Name	Mechanics	Domains	Play Time	Complexity	Rating	BGG Rank
0	Gloomhaven	Action Queue, Action Retrieval,...	Strategy Games, Thematic Games	120	3.86	8,79	1
1	Pandemic Legacy: Season 1	Action Points, Cooperative Game...	Strategy Games, Thematic Games	60	2.84	8.61	2
2	Brass: Birmingham	Hand Management, Network and Ro...	Strategy Games	120	3.91	8.66	3
3	Terraforming Mars	Card Drafting, End Game Bonuses...	Strategy Games	120	3.24	8,43	4

Throughout this exam, we will refer to `games` repeatedly. Assume that we have already run `import babypandas as bpd` and `import numpy as np`.