DATA 6 Final Exam, Summer 2024

Instructions
You have 1 hours and 50 minutes to complete the exam.

- The exam is closed book, closed notes, closed computer, closed calculator, except the provided reference sheet.
- Mark your answers on the exam itself in the spaces provided. We will not grade answers written on scratch paper or outside the designated answer spaces.
- If you need to use the restroom, bring your phone, exam, and student ID to the front of the room.
- The test is designed to be completed using methods we have learned in this class. We reserve the right to deduct or not score answers using methods out of scope.

For multiple choice questions with circles, you should select exactly one choice. You should indicate your selection by **completely** filling in the circle.

◯ You must choose either this option

◯ Or this one, but not both!

For multiple choice questions with square checkboxes, you may select multiple choices. You should indicate your selection by **completely** filling in the box.

☐ You could select this choice.

☐ You could select this one, too!

Please write your initials at the top of every page as you are taking the exam.

Question	Points	Score
1	22	
2	9	
3	21	
4	20	
5	10	
6	6	
Total:	88	

Name: _____

TA's name: _____

Student ID: _____

Name of person to your left: _____

Name of person to your right: _____

1. **True or False**

(a) (2 points) All of the work on this exam is your own.

◯ True

◯ False

(b) (2 points) In the Data 6 context, a *variable* is the same thing as a *name* in Python.

◯ True

◯ False

(c) (2 points) The assignment statement $21 = 9 + 10$ errors

◯ True

◯ False

(d) (2 points) When defining a function, you **must** have a `return` call

◯ True

◯ False

(e) (2 points) We can add multiple columns to our table, by using `tbl.with_column(...)`

◯ True

◯ False

(f) (2 points) `np.randint(1,11)` returns a **true** random number.

◯ True

◯ False

(g) (2 points) When trying to study the effect of tutoring on student performance, Fred randomly asked students who are in office hours for Data 6. Fred's study is susceptible to selection bias.

◯ True

◯ False

(h) (2 points) A dictionary in Python can have duplicate keys.

◯ True

◯ False

(i) (2 points) The expression `1.0 + 2` results in a value of type float.

◯ True

◯ False

(j) (2 points) All hypothesis must be testable.

◯ True

◯ False

(k) (2 points) According to Professor Carson, data is a representation of the world and therefore its selection is better thought of as perspective than as bias.

◯ True

◯ False

2. **Gotta Catch Them All**

Su Min is a Pokemon trainer who is building out her pokedex. She organizes her pokedex as a Python dictionary called `pokedex`, where each pokemon's name is a key and the values are dictionaries with the pokemon's *nickname*, *types*, and *total* as keys and the pokemon's nickname as a string, types as an array, and total power as an int being the corresponding values. You can see a few members of Su Min's pokedex below.

```
pokedex = {
"rowlet": {"nickname" : "life partner", "types": make_array("flying", "grass") , "total":
    320},
"tinkaton" : {"nickname" : "kristen", "types": make_array("fairy", "steel") , "total":
    506}, ... ,
"growlithe": {"nickname" : "dana", "types": make_array("fire"), "total": 350}
}
```

(a) (3 points) Su Min just caught a new ***luxray***. That pokemon has types ***electric*** and power ***523***. Su Min gave the nickname ***ethan***. Add it to Su Min's pokedex.

(b) Su Min wants to have an array that contains all the **different** types in her pokedex. Help Su Min by filling in the blanks.

```
all_types = ___(a)____
for pokemon in ___(b)___:
    for type in ___(c)___:
        if type not in all_types:
            all_types = ___(d)____
```

i. (1 point) Fill in blank (a).

ii. (2 points) Fill in blank (b).

iii. (2 points) Fill in blank (c).

iv. (1 point) Fill in blank (d).

3. **Road to 100**

Edwin and his friend Sam have been sharing their daily push-up counts via text messages every day since October 20th, 2023. This has been ongoing for 287 days at the time of making this question. Now, we want to analyze their data using two tables: `edwins_pushups` and `sams_pushups`.

Table 1: `edwins_pushups`

Day	Edwin's Push-ups
1	40
2	49
3	51
4	52
5	56

`...(301 rows omitted)`

Table 2: `sams_pushups`

Day	Sam's Push-ups
1	40
2	40
3	42
4	35
4	35

`...(293 rows omitted)`

The tables contain the following columns:

- `Day`: an `int`, the days since October 20th
- `Edwin's Push-ups`: a `string`, the number of push-ups by Edwin
- `Sam's Push-ups`: a `string`, the number of push-ups by Sam

(a) Edwin downloaded the push-up data from his text messages, resulting in all push-up counts being strings. He wants to convert this data to integers. Edwin needs your help in defining `convert_to_int`, which takes in a string and returns it as an integer. From there, he wants to define `edwins_pushups_int_array` and `sams_pushups_int_array`, so he can update the values in both tables.

```
def convert_to_int(string):
    return ___(a)___

edwins_pushups_int_array = edwins_pushups.___(b)___(___(c)___, "Edwin's Push-ups")
edwins_pushups = edwins_pushups.with_column("Edwin's Push-ups", edwins_pushups_int_array )

sams_pushups_int_array = sams_pushups.___(b)___(___(c)___, "Sam's Push-ups")
sams_pushups = sams_pushups.with_column("Sam's Push-ups", sams_pushups_int_array)
```

i. (2 points) Fill in blank (a).

ii. (2 points) Fill in blank (b).

iii. (1 point) Fill in blank (c).

(b) (2 points) Sam wants to find the most number of push-ups he has done in one go. Select all that apply.

- ☐ `max(sams_pushups.column("Sam's Push-ups"))`
- ☐ `min(sams_pushups.column("Sam's Push-ups"))`
- ☐ `sams_pushups.sort("Sam's Push-ups").column("Sam's Push-ups").item(0)`
- ☐ `sams_pushups.num_rows`
- ☐ None of the above

(c) Edwin and Sam sometimes do push-ups multiple times per day. They're now interested in their total push-ups for each day. Make `edwins_pushups_grouped` and `sams_pushups_grouped` a **table with one row per day** and **each day has the total number of pushups**

```
edwins_pushups_grouped = edwins_pushups.___(a)___(___(b)___,___(c)___)
sams_pushups_grouped = sams_pushups.___(a)___(___(b)___,___(c)___)
```

i. (1 point) Fill in blank (a).

- ◯ `group`
- ◯ `where`
- ◯ `sort`
- ◯ `pivot`
- ◯ None of the above

ii. (1 point) Fill in blank (b).

iii. (2 points) Fill in blank (c).

(d) Sam now wants to **join** both tables in order to do further analysis. Fill in the code below such that `combined_pushups` results in the following table.

Table 3: `combined_pushups`

Day	Edwin's Push-ups sum	Sam's Push-ups sum
1	40	40
2	49	40
3	51	42
4	52	105
5	56	10

`...(282 rows omitted)`

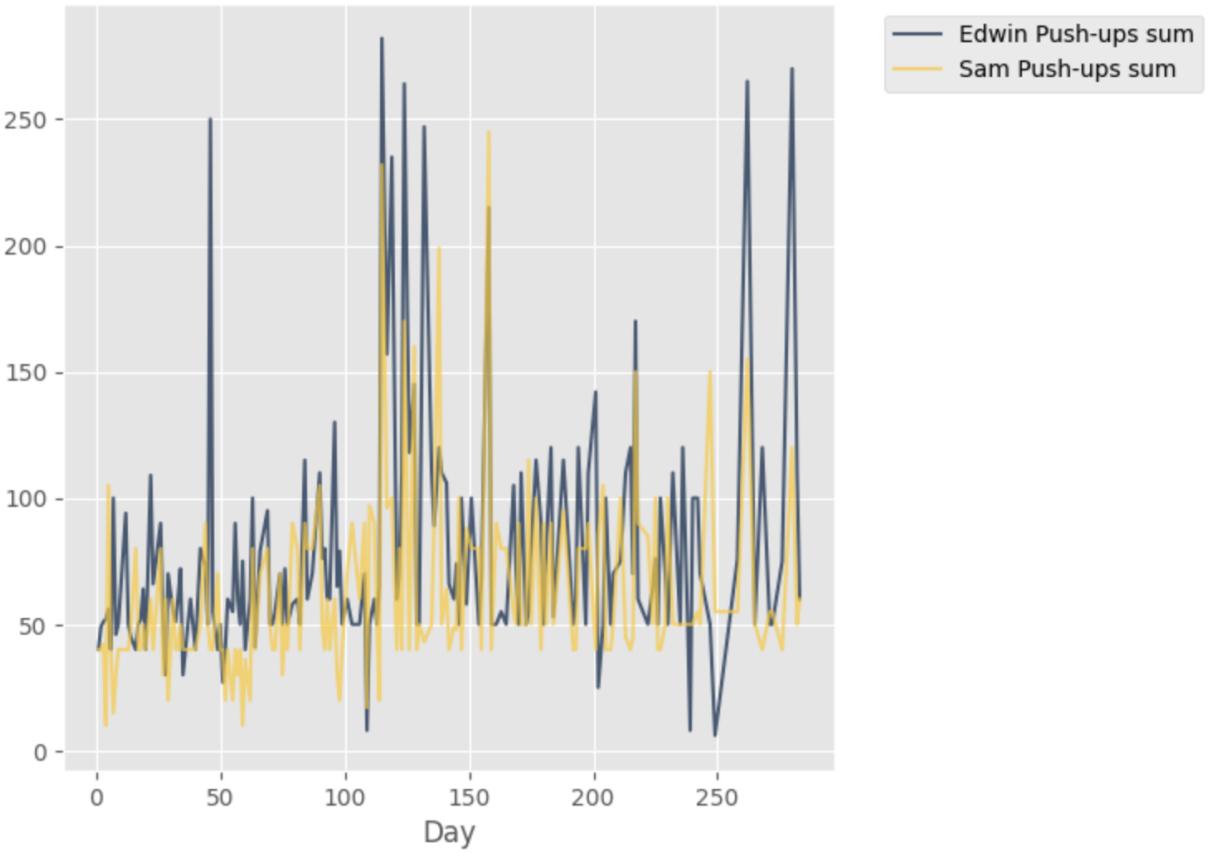
```
combined_pushups = edwins_pushups_grouped.___(a)___(___(b)___,___(c)___)
```

i. (1 point) Fill in blank (a).

ii. (1 point) Fill in blank (b).

iii. (1 point) Fill in blank (c).

(e) Edwin wants to visualize Sam's and his push-ups over time. Using the `combined_pushups` table, write a line of code that generates the **line plot** below.




```
_______(a)_______._____(b)_____(______(c)______)
```

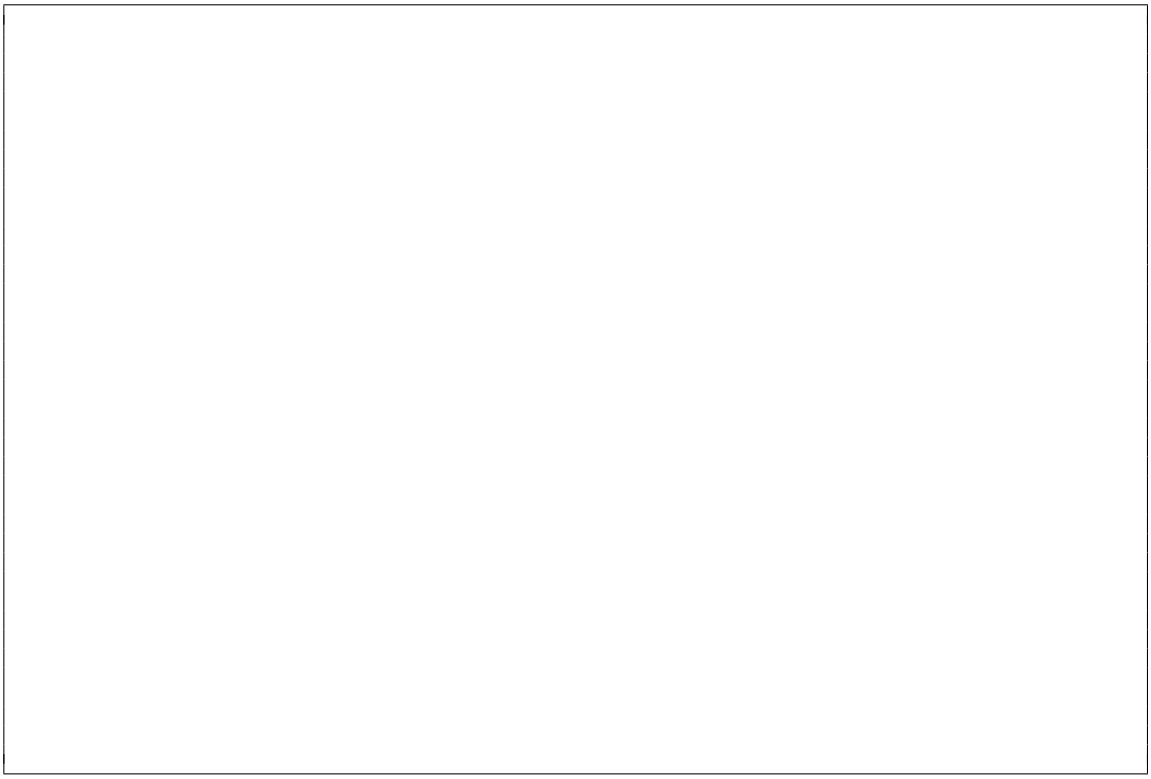
i. (1 point) Fill in blank (a).

ii. (1 point) Fill in blank (b).

iii. (1 point) Fill in blank (c).

(f) (4 points) Edwin initially filtered all text messages to leave only push-up counts. Recall from Discussion 2 that we lose a lot of information and context when data cleaning. What stories are we potentially missing when we clean this dataset? Refer to some text messages below. (Write 2-3 sentences below)

Message Date	Text
2024-02-19 23:43:24	The road to 100 🤜
2024-02-24 23:17:24	Road to 100
2024-02-24 23:23:36	Emphasized “Road to 100”
2024-04-21 23:04:49	78, and then lots of mini sets to get to 100
2024-04-21 23:05:56	Funny story: A student org was doing donate $1 and they’ ...



4. **Apartment Hunting**

Atticus is apartment hunting in Chicago and has stored all his potential apartments in a dataset to help him organize his search. The dataset, named `apartments`, consists of data from 40 apartments Atticus found on apartments.com. The dataset includes the following columns:

- `Address`: a string, the address of the apartment.
- `Beds`: a int, the number of bedrooms in the apartment.
- `Baths`: a float, the number of bathrooms in the apartment.
- `Cost`: an integer, the monthly cost for rent + utilities
- `Furnished`: a boolean, True if the apartment comes furnished otherwise False.

Table 4: `apartments`

Address	Beds	Baths	Cost	Furnished
1367 E. 53rd Street	1	1	1250	False
1330 E. 53rd Street	2	2	2398	True
847 E. 65th Street	4	2	1800	False

... (37 Rows Omitted)

Throughout this problem you'll be updating *good_housing* until you've helped Atticus find his perfect place.

(a) Atticus doesn't want to share a bathroom with more than 1 more other person. Add a column `"Bed:Bath"` which contains the ratio of Bedrooms to Bathrooms as a float. Then filter the table to only include apartments where the ratio is **no** higher than 2.0.

```
good_housing = apartments.___(a)____("Bed:Bath", ____(b)____)
good_housing = good_housing.____(c)____("Bed:Bath", ___(d)___)
```

i. (1 point) Fill in blank (a).

ii. (1 point) Fill in blank (b).

iii. (1 point) Fill in blank (c).

iv. (1 point) Fill in blank (d).

(b) (2 points) In order to make the move to Chicago go smoother, Atticus wants only `Furnished` rooms. Write code below, updating `good_housing` so it only contains Furnished apartments.
As a reminder the `good_housing` table contains the columns `Address`, `Beds`, `Baths`, `Cost`, `Furnished`, `Bed:Bath`

(c) (2 points) After visiting Chicago Atticus discovered his new favorite cafe called Cafe 53. Atticus wants to live on 53rd street so he can go there every day. Write code below updating `good_housing` so it only includes apartments on *"E. 53rd Street"*.

(d) Having decided on his favorite apartment, Atticus wants to make sure he's not overpaying. To do so, help Atticus make a new table where each row is the number of bedrooms an apartment has, each column is the number of bathrooms, and each cell is the average cost for an apartment with that many bedrooms and bathrooms.

```
good_housing.___(a)____(___(b)___, ___(c)____, ____(d)____, ____(e)___)
```

i. (1 point) Fill in blank (a).

ii. (1 point) Fill in blank (b).

iii. (1 point) Fill in blank (c).

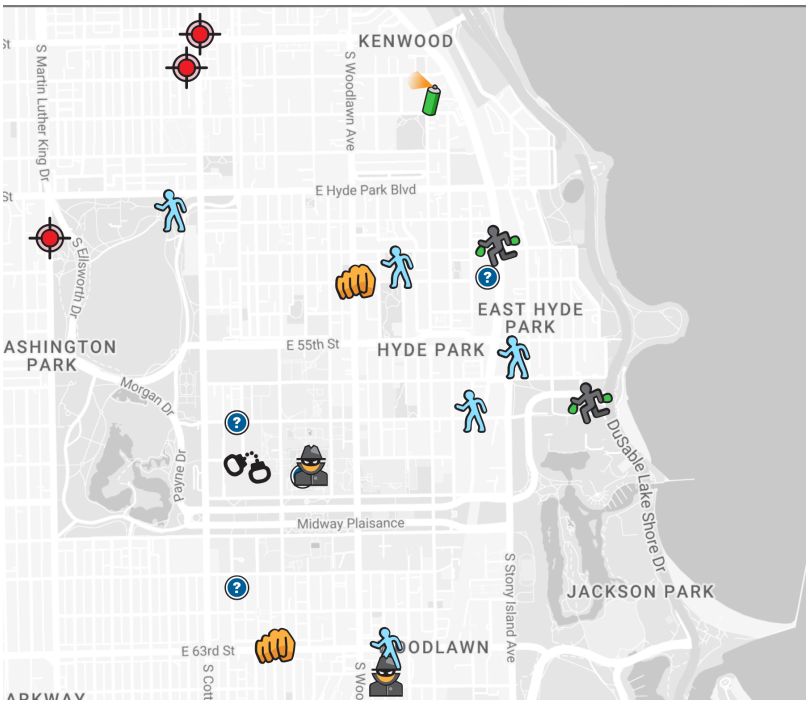
iv. (1 point) Fill in blank (d).

v. (1 point) Fill in blank (e).

(e) (2 points) In hindsight Atticus is concerned about how much more expensive the furnished apartments might be compared to unfurnished apartments. Which of the following plots would best help Atticus compare the median cost between furnished and unfurnished apartments.

- ◯ Histogram
- ◯ Line Plot
- ◯ Scatter Plot
- ◯ Bar Chart

(f) Before Atticus signs the lease he decides to take a look at crime in the area and found the image below.



i. (3 points) In the text box below list each variable encoded in the image along with a description of what that variable represents.

ii. (2 points) What type of visualization is the image above?

- ◯ Choropleth Map
- ◯ Scatterplot
- ◯ Scatterplot Map
- ◯ None of the Above

5. **What's for Breakfast?**

Kenneth and Elizabeth don't want to spend time choosing what to eat for breakfast at Croads, so they want to build a function that will randomly select **two** breakfast items for them each day. They do have a couple criteria for their breakfast though.

- If the function says to have bagels, they also must add cream cheese to their breakfast that day. Cream Cheese won't count as one of their two items in this case.
- If the function says to have cereal, they also must add milk to their breakfast that day. Milk won't count as one of their two items in this case.
- They don't want **two** of the same thing. If the first **two** breakfast items are the same they should simulate another set of random breakfast items.

```
choices = make_array("eggs", "toast", "cereal", "muffins", "bagels", "grits")
random_breakfast = ____(a)_____
while ___(b)___ == ____(c)____:
    random_breakfast = ___(a)___
if ____(d)____:
        random_breakfast = np.append(random_breakfast, "cream cheese")
if ____(e)_____:
    random_breakfast = np.append(random_breakfast, "milk")
return random_breakfast
```

(a) Help them finish the function so they can choose their most important meal of the day!

i. (2 points) Fill in blank (a).

ii. (1 point) Fill in blank (b).

iii. (1 point) Fill in blank (c).

iv. (1 point) Fill in blank (d).

v. (1 point) Fill in blank (e).

(b) Kenneth and Elizabeth simulated an entire semesters worth of breakfast's and stored them in the table `breakfast` which has one column called *Item*. They are worried they'll have too many of the same item throughout the semester. Fill in the blanks below to create a table with 1 row which has two columns: the name of the most common item and the number of times their simulation chose that item for them.

```
breakfast.____(a)____(____(b)_____).____(c)___("count", ____(d)___).take(0)
```

i. (1 point) Fill in blank (a).

ii. (1 point) Fill in blank (b).

iii. (1 point) Fill in blank (c).

iv. (1 point) Fill in blank (d).

6. **Mystery Function**

(a) The Data 6 staff defined a function called `mystery_function`, but they forgot what it does. Help them figure it out!

```
def mystery_function(string):
    result = ""
    indices = np.arange(-1, -(len(string)+1), -1)
    for i in indices:
        result = result + string[i]
    return result
```

i. (2 points) What does `mystery_function("hi")` output?

- ◯ 'hi'
- ◯ 'ih'
- ◯ Error
- ◯ None of the above

ii. (2 points) What does `mystery_function("tacocat")` output?

- ◯ 'cattaco'
- ◯ 'tacocat'
- ◯ Error
- ◯ None of the above

iii. (2 points) Write 1-2 sentences on what `mystery_function` does.

7. **Bonus Extra Credit**

(1 point) What is the value of $\frac{1}{2}!$ in math.

8. **Assumptions**

(a) (0 points) If you felt any question required additional assumptions, please write them here. Be warned: We will only consider these assumptions if the question indeed required additional information.

(b) (0 points) Draw us a picture!