

**INSTRUCTIONS**

You have 1 hour and 50 minutes to complete the exam. The exam is worth a total of 80 points.

- The exam is closed book, closed notes, closed computer/calculator, except for the provided cheat sheet.
- Mark your answers on the exam itself in the spaces provided.
- If you need to use the restroom, bring your phone and exam to the front of the room.

For questions with **circular bubbles**, you should fill in exactly *one* choice.

- You must choose either this option
- Or this one, but not both!

**Preliminaries**

You can complete these questions before the exam starts.

- (a) What is your full name?

- (b) What is your student ID number?

- (c) Who is sitting to your left? (Write *no one* if no one is next to you.)

- (d) Who is sitting to your right? (Write *no one* if no one is next to you.)

- (e) **UC Berkeley Honor Code**

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. By signing my name below, I affirm that all of my answers are my own work, and that I have used no external resources (other than the Data 6 cheat sheet) during this exam.

- i. (2.0 pt) **Sign your name in the space provided.**

**1. (9.0 points) Tado's True/False**

For each of the following statements, indicate whether it is true or false.

(a) (1.0 pt) In Data 6, a 'variable' and a 'name' are the same thing.

True

False

(b) (1.0 pt) The code `true = 6 == True` will cause an error.

True

False

(c) (1.0 pt) Unlike arrays, Python dictionary values do not have an index.

True

False

(d) (1.0 pt) Changing your phone's privacy settings will completely prevent any app from learning private information about you.

True

False

(e) (1.0 pt) A histogram is the best choice for visualizing a categorical variable.

True

False

(f) (1.0 pt) `sum(np.arange(5))` evaluates to 10.

True

False

(g) (1.0 pt) There are two different kinds of loops in Python.

True

False

(h) (1.0 pt) Data science is *only* about manipulating data.

True

False

(i) (1.0 pt) `print("hello")` will return "hello".

True

False

**2. (10.0 points) Sunya's Survey**

Sunya is an avid environmentalist and is doing a research project investigating the impacts of carbon emissions and air pollution in communities across the Bay Area.

- (a) (1.0 pt) Sunya starts with data from 50 different carbon sensor stations scattered throughout the Bay Area, each of which is associated with a particular Bay Area city or town. In order to get a general sense of carbon emissions trends, Sunya wants to create a table showing the average carbon emission rates for the four main regions in the Bay: San Francisco, South Bay, East Bay, and North Bay. To do this she needs to \_\_\_\_\_ her data.

- aggregate  
 disaggregate

- (b) (2.0 pt) On second thought, Sunya realizes that it might be more useful to analyze pollution data on a neighborhood-by-neighborhood basis. **Can she do this *only* with the data she has currently? Why or why not?**

No.

- (c) (3.0 pt) While developing her research report, Sunya creates a scatter plot map to show emissions levels by neighborhood. When her research advisor reviews her research paper draft, she suggests using a choropleth map instead. **Which map should Sunya use? Name one advantage of using this type of map over the other type.**

- (d) (4.0 pt) Even with all of the quantitative data that Sunya has collected, she still believes that her research is missing something. To augment her analysis, Sunya decides to collect qualitative data on the impacts of pollution in the form of a survey. **Provide two ethical considerations that Sunya should keep in mind while writing and fielding her survey to Bay Area residents.**

**3. (11.0 points) Josh's Job**

The year is 2025. Josh is the senior data scientist at Snoopchat, Inc. The fate of the entire company now rests on his shoulders as he tries to #MakeSnoopchatPopularAgain.

In this question, you'll be working with the following `snoopchat` table, in which each row represents the past one week summary of a Snoopchat user's engagement with the application:

User ID	Snoops Sent	Snoops Received	Stories Posted
75682867	44	46	4
66755036	16	17	4
66882282	25	25	5
31081788	35	32	2
23315092	11	12	4

... (245 rows omitted)

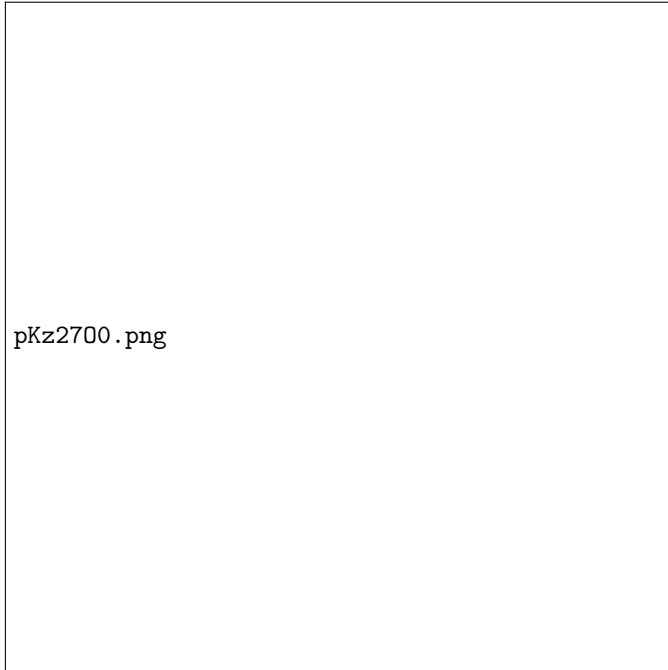
**(a) (2.0 points) Variable Types**

Look at the "User ID" column from the `snoopchat` table above. Notice that each unique Snoopchat user is represented by an eight digit number. These IDs are randomly generated when a user makes an account.

- i. (2.0 pt)** Which of the following most closely represents the **variable type** of the "User ID" column?
- Numerical Discrete
  - Numerical Continuous
  - Categorical Nominal
  - Categorical Ordinal
  - None of the Above

**(b) (4.0 points) Sent vs. Received**

Josh wonders whether there is a relationship between the number of Snoops sent vs. the number of Snoops received by any given user. Using the `snoopchat` table, **write a line of code that generates the scatter plot below** comparing the "Snoops Sent" (x-axis) to the "Snoops Received" (y-axis).



----- (-----, -----)  
(a) (b) (c) (d)

i. (1.0 pt) Fill in blank (a).

`snoopchat`

ii. (1.0 pt) Fill in blank (b).

`scatter`

iii. (1.0 pt) Fill in blank (c).

`"Snoops Sent"`

iv. (1.0 pt) Fill in blank (d).

`"Snoops Received"`

## (c) (5.0 points) Total Engagement

Add a new column, “Total Engagement” to the snoopchat table which represents the total number of interactions each user had on Snoopchat over the past week. We define the total number of interactions to be:

*Total Number of Interactions = Snoops Sent + Snoops Received + Stories Posted*

totals = \_\_\_\_\_  
(a)

snoopchat = snoopchat.\_\_\_\_\_("Total Engagement", \_\_\_\_\_)  
(b) (c)

i. (3.0 pt) Fill in blank (a).

```
snoopchat.column("Snoops Sent") + snoopchat.column("Snoops Received") +
snoopchat.column("Stories Posted")
```

ii. (1.0 pt) Fill in blank (b).

```
with_column
```

iii. (1.0 pt) Fill in blank (c).

```
totals
```

#### 4. (22.0 points) Weichert's Waffle Palace

Frustrated by the lack of quality waffle establishments on the West Coast, James decides to start his own chain restaurant serving a variety of delicious waffles. *Weichert's Waffle Palace* develops a cult following in the Bay Area and expands to three locations: Berkeley, Palo Alto, and San Jose.

In this question, you'll first start by working with the `wwp` table containing information about waffle sales across all three locations of *Weichert's Waffle Palace*. Each row in the `wwp` table contains information about **one order**. The `wwp` table contains three columns:

- "Location" — Where the order occurred
- "Waffle" — The type of waffle ordered
- "Day" — The day of the week when the order occurred

Location	Waffle	Day
San Jose	Deluxe	Sun
Berkeley	Plain	Sun
San Jose	Cinnamon	Sun
Berkeley	Deluxe	Sun
Berkeley	Cinnamon	Sun
Palo Alto	Cinnamon	Sun
San Jose	Blueberry	Sun

(... 334 rows omitted)

##### (a) (3.0 points) Which Waffle Store?

Fearing that his Berkeley location is selling fewer Cinnamon Waffles than the other locations, James asks you to create a new table called `waffle_counts` where *each row represents a unique location* and where *each column represents a unique waffle type*. **Fill in the blanks below to create the table.**

The `waffle_counts` table should contain information about the total number of waffles ordered for each (Location, Waffle) combination. Your table should have three rows and four columns.

```
waffle_counts = wwp._____ (_____, _____)
```

(a)                      (b)                      (c)

i. (1.0 pt) Fill in blank (a).

- group
- join
- pivot
- select
- None of the Above

ii. (1.0 pt) Fill in blank (b).

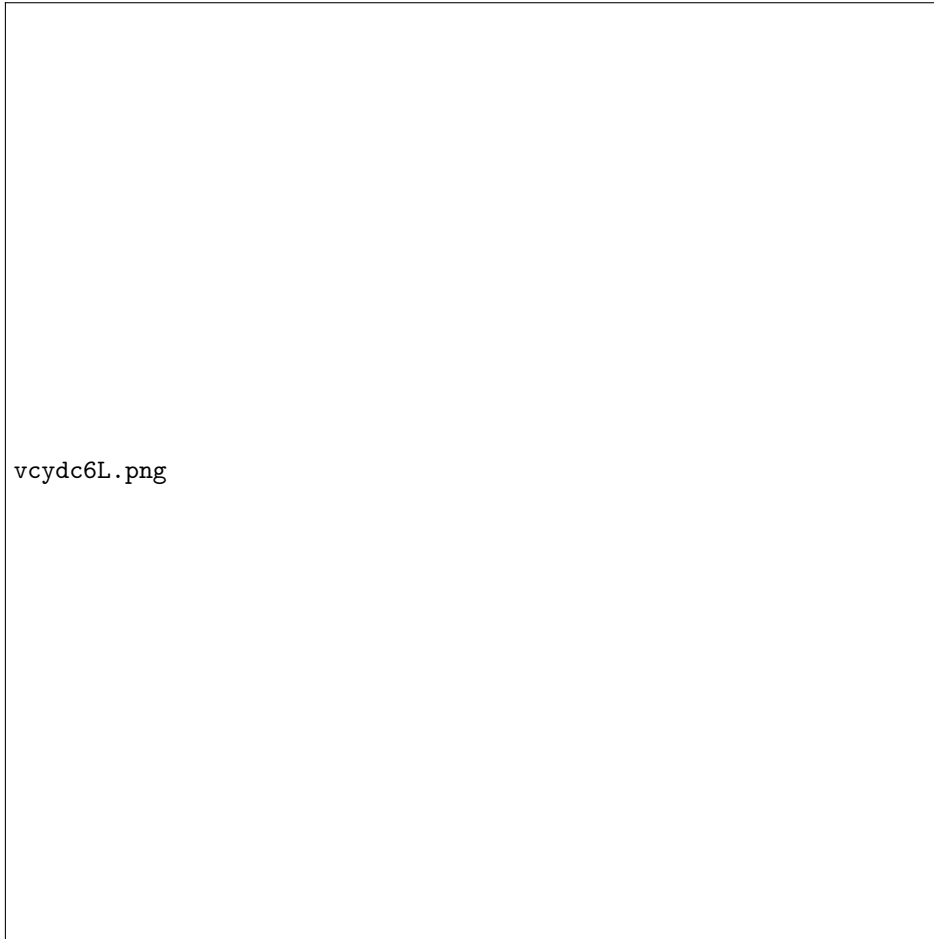
"Waffle"

iii. (1.0 pt) Fill in blank (c).

"Location"

**(b) (7.0 points) Cinnamon Waffles**

Thinking about it again, James believes that it would be more effective to create a visualization to prove that Berkeley is selling fewer **Cinnamon Waffles** than the other locations. **Fill in the following code to reproduce the following horizontal bar chart:**



```

only_cinnamon = wwp.where(_____, _____)
                        (a)         (b)
locs_sorted_by_count = only_cinnamon._____("Location")
                        (c)
                        .sort(_____, _____)
                        (d)         (e)
locs_sorted_by_count._____(_____)
                        (f)         (g)

```

i. (1.0 pt) Fill in blank (a).

"Waffle"

ii. (1.0 pt) Fill in blank (b).

"Cinnamon"



iii. (1.0 pt) Fill in blank (c).

- group
- join
- pivot
- select
- None of the Above

iv. (1.0 pt) Fill in blank (d).

```
"count"
```

v. (1.0 pt) Fill in blank (e).

```
descending=True
```

vi. (1.0 pt) Fill in blank (f).

- scatter
- barh
- hist
- group
- None of the Above

vii. (1.0 pt) Fill in blank (g).

```
"Location"
```

**(c) (4.0 points) Waffle Prices**

You find the pricing information for each type of waffle sold by *Weichert's Waffle Palace* and store it in the following prices table:

Name	Price
Plain	4.99
Blueberry	5.49
Cinnamon	5.49
Deluxe	6.99

Using one of the table methods from the Python Reference Sheet, create a new table, `wwp_with_prices`, which contains both the information from the `wwp` table and the prices table. The first few rows of your `wwp_with_prices` table should look like this:

Name	Price	Location	Day
Blueberry	5.49	San Jose	Sun
Blueberry	5.49	Berkeley	Sat
Blueberry	5.49	San Jose	Sat

```
wwp_with_prices = prices._____ (_____, _____, _____)
                        (a)         (b)         (c)         (d)
```

`wwp_with_prices`

i. (1.0 pt) Fill in blank (a).

`join`

ii. (1.0 pt) Fill in blank (b).

`"Name"`

iii. (1.0 pt) Fill in blank (c).

`wwp`

iv. (1.0 pt) Fill in blank (d).

`"Waffle"`

**(d) (8.0 points) Weekend Waffles**

Which *Weichert's Waffle Palace* location makes **the most money** on the weekends (i.e., on "Sat" and "Sun")? **Fill in the following code** so that `top_seller` is assigned to the **name of the location** that makes the most revenue on the weekends. Recall that one row of the `wwp` table represents one order.

```
weekend = wwp_with_prices.where(_____,
                                (a)
                                are.contained_in(make_array(_____, _____)))
                                (b)          (c)

totals = weekend._____("Location", _____)
                                (d)          (e)

winner = totals._____("Price sum", descending=True).column(_____.item(0)
                                (f)          (g)
```

i. (1.0 pt) Fill in blank (a).

"Day"

ii. (1.0 pt) Fill in blank (b).

"Sat"

iii. (1.0 pt) Fill in blank (c).

"Sun"

iv. (1.0 pt) Fill in blank (d).

group

v. (2.0 pt) Fill in blank (e).

sum

vi. (1.0 pt) Fill in blank (f).

sort

vii. (1.0 pt) Fill in blank (g).

"Location"

### 5. (26.0 points) Sandra's Sneaky Sequences

In this question, you'll work with **iteration** and **conditionals**. Sandra is a huge *Diary of a Wimpy Kid* fan; she never fails to buy the new book when it's released.

#### (a) (8.0 points) Attention Seekers

Fill in the following code blanks so that `main_characters` is assigned to an array of the two main characters of Jeff Kinney's 2007 masterpiece, *Diary of a Wimpy Kid*: **greg** and **rowley**.

```
characters = make_array("manny", "greg", "fregley", "rowley",
                       "chirag", "holly", "rodrick")
main_characters = make_array()

for character in _____:
    (a)
    if _____ == _____ or _____ == _____:
        (b)           (c)           (d)           (e)
        _____ = np.append(_____, _____)
        (f)           (g)           (h)
```

i. (1.0 pt) Fill in blank (a).

`characters`

ii. (1.0 pt) Fill in blank (b).

`character`

iii. (1.0 pt) Fill in blank (c).

`"greg"`

iv. (1.0 pt) Fill in blank (d).

`character`

v. (1.0 pt) Fill in blank (e).

`"rowley"`

vi. (1.0 pt) Fill in blank (f).

`main_characters`

vii. (1.0 pt) Fill in blank (g).

```
main_characters
```

viii. (1.0 pt) Fill in blank (h).

```
character
```

**(b) (12.0 points) Sibling Rivalry**

Sandra wants to know how many siblings **Greg Heffley** has in the novel. She thinks it's two, but she wants you to confirm. Below, **implement the following code so that `num_siblings` is assigned to the number of siblings that Greg has.** Assume that in the *Wimpy Kid* universe, if two characters have the same **last name**, they are considered siblings.

To help, we've provided two arrays called `first_names` and `last_names` representing the characters' first and last names, respectively. You can assume that the first name at index 0 in `first_names` corresponds to the last name at index 0 in `last_names`, and so on.

```

first_names = make_array("manny", "fregley", "greg", "rowley",
                        "chirag", "holly", "rodrick")
last_names = make_array("heffley", "N/A", "heffley", "jefferson",
                        "gupta", "hills", "heffley")

num_total_names = _____(first_names)
                        (a)

num_siblings = _____
                (b)

for i in np.arange(_____):
                    (c)
    first = first_names._____ (_____ )
                        (d)         (e)
    last = last_names._____ (_____ )
                        (f)         (g)
    if _____ != "greg" and _____ == _____:
        (h)         (i)         (j)
        _____ = _____ + 1
        (k)         (l)

```

i. (1.0 pt) Fill in blank (a).

len

ii. (1.0 pt) Fill in blank (b).

0

iii. (1.0 pt) Fill in blank (c).

num\_total\_names

iv. (1.0 pt) Fill in blank (d).

item

v. (1.0 pt) Fill in blank (e).

```
i
```

vi. (1.0 pt) Fill in blank (f).

```
item
```

vii. (1.0 pt) Fill in blank (g).

```
i
```

viii. (1.0 pt) Fill in blank (h).

```
first
```

ix. (1.0 pt) Fill in blank (i).

```
last
```

x. (1.0 pt) Fill in blank (j).

```
"heffley"
```

xi. (1.0 pt) Fill in blank (k).

```
num_siblings
```

xii. (1.0 pt) Fill in blank (l).

```
num_siblings
```

(c) (6.0 points) **The Cheese Touch!**

For each of the following code blocks, **what would be printed out if the code were to be executed?** Assume that before each block is executed, the following `mystery` function is defined as follows:

```
def mystery(name):  
    if len(name) <= 4:  
        print("He doesn't have the cheese touch...")  
    elif name == "fregley":  
        print("He's got the cheese touch!")  
    elif len(name) == 7:  
        print("Rodrick Rules!")  
    else:  
        print("Zoo wee mama!")
```

i. (2.0 pt)

```
mystery("rowley")
```

**Zoo wee mama!**

ii. (2.0 pt)

```
mystery("fregley")
```

**He's got the cheese touch!**

iii. (2.0 pt)

```
mystery("greg")
```

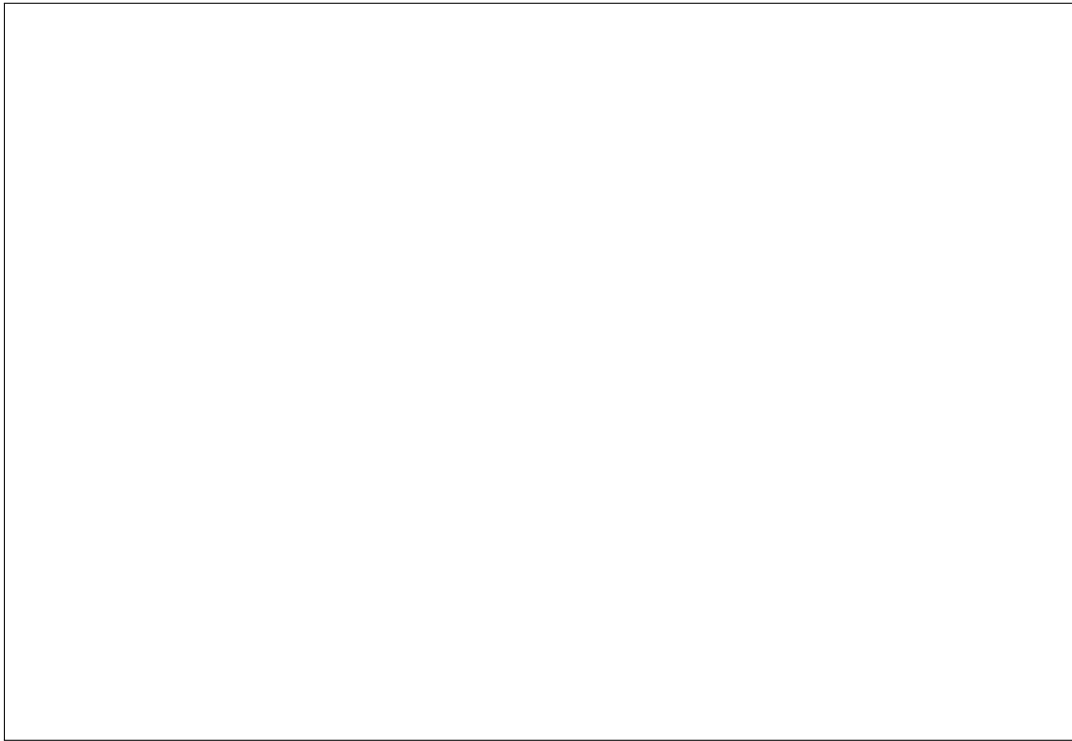
**He doesn't have the cheese touch...**



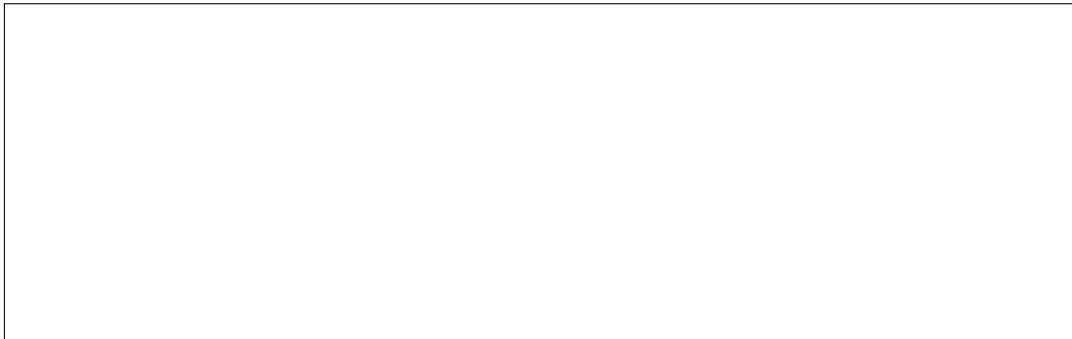
**6. Leanne's Lucky Bonus Questions**

These questions are extra bonus questions, and do not have anything to do with the course content.

(a) (0.0 pt) Draw us a picture :)



(b) (0.0 pt) How was your overall experience in Data 6?



(c) (0.0 pt) What is your favorite type of waffle?



(d) (0.25 pt) What is Will's favorite color?

