Fall 2024
Course Name: Data 6
Instructor: Tsang
Final

Print Your Name: ____

Print Your Student ID: ____________________________

You have 110 minutes. There are 5 questions of varying credit (100 points total).

Question:	1	2	3	4	5	Total
Points:	25	9	48	12	6	100

○ For questions with circular bubbles, you may select only one choice.

☐ For questions with square checkboxes, you may select one or more choices.

Anything you write outside the answer boxes or anything you *cross out* will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation.

You are disallowed from using the following: ternary operators, lambdas, list comprehensions, and any libraries/functions that are not built in/on the reference card.

You may only write one statement per line (i.e. you may not use semicolons).

Question 1: Potpourri

Question 1.1 (4 points)
Match each of the following situations with the best sampling method to use in that situation. Your options are: simple random sampling, stratified random sampling, quota sampling, and snowball sampling. Each sampling method will be used exactly once.

Situation	Best Sampling Method
You have a full, accurate roster of your population. This roster contains full demographic data as well, which is what you want to collect information on.	Stratified Random Sampling
You are studying a rare medical disease and want to contact people who have this disease. However, you aren't sure where to find these individuals.	Snowball Sampling
You want to quickly gather some data that is guaranteed to have individuals from each subgroup you want to study, but don't have a full, accurate roster to choose from.	Quota Sampling
You have a full, accurate roster of your population, but don't care as much about collecting information across certain subgroups in your population.	Simple Random Sampling

Question 1.2 (2 points)
What does the concept of "representation" mean in the context of data?

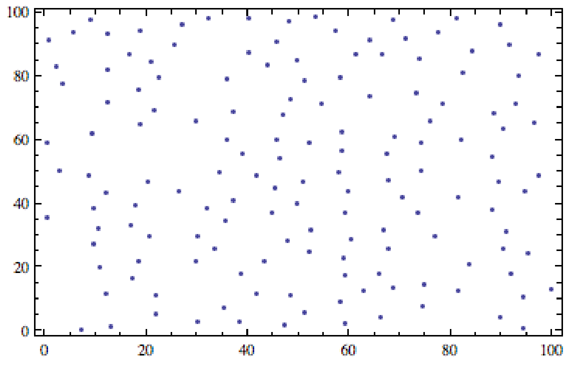
- ○ The act of directly recording all aspects of reality.
- [✓ ● The process of making something stand for a phenomenon in the world.]
- ○ A method to allow arbitrary observations to define phenomena.
- ○ A way to summarize large datasets for ease of use.

Question 1.3 (2 points)
When we say that representation is "selective", what could it apply to? Select all that apply.

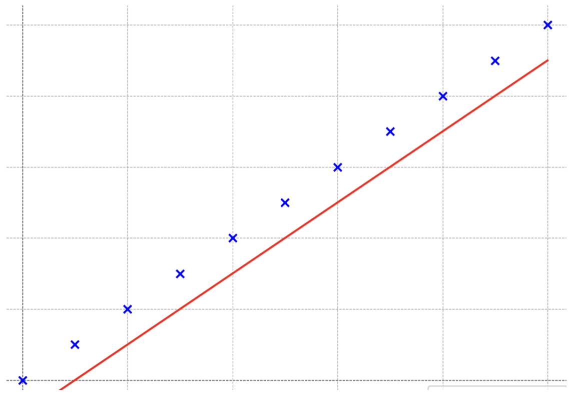
- ☑ ~~We are being "selective" to a fault; datasets often become a series of arbitrary observations when we are too selective.~~
- ☐ We are being "selective" about what data to collect in the first place.
- ☐ We are being "selective" about the methods to collect certain types of data
- ☐ We are being "selective" about what aspects of the data to showcase.

Question 1.4 (3 points)
Which of the following plots best represents Simpson's paradox?

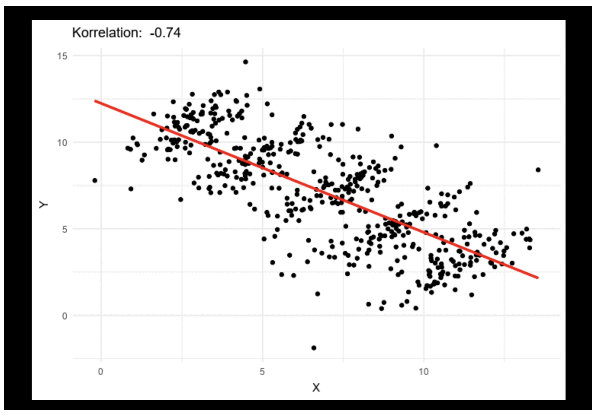
PLOT A



PLOT B



PLOT C



PLOT A
PLOT B
[✓ PLOT C]

Question 1.5 (2 points)
Write a Python expression that evaluates to 0 if n is even, and 1 if n is odd. You may only use arithmetic operators (+, -, *, /, //, %), n, and integers.

```python
n % 2
```

Question 1.6 (2 points)
Write a Python expression that evaluates to 1 if n is even, and 0 if n is odd. You may only use arithmetic operators (+, -, *, /, //, %), n, and integers.

```python
1 - n % 2
```

Question 1.7 (2 points)
Write a Python expression that **evaluates** to the last two digits of n. For example, if n is 100, this expression should evaluate to 0. If n is 12189, the expression should evaluate to 89. You may only use arithmetic operators (+, -, *, /, //, %), n, and integers.

```python
n % 100
```

Question 1.8 (2 points)
Write a Python expression that **removes** the last two digits of n. For example, if n is 100, this expression should evaluate to 1. If n is 12189, the expression should evaluate to 121. You may only use arithmetic operators (+, -, *, /, //, %), n, and integers.

```python
n // 100
```

Question 1.9 (2 points)
Is this an associative or causal relationship: As ice cream sales increase, we also observe shark attacks increase as well.

[✓ Associative]

Causal

Question 1.10 (2 points)
With the relationship above, it is possible that summer months impact both ice cream sales and shark attacks. This is considered a…

Options: Confounding variable, theory, concept, causal hypothesis, associative hypothesis, exploratory research question, unit of analysis, scientific method, aggregation, disaggregation, internal validity, external validity, generalizability, categorical variable, numerical variable

Confounding Variable

Question 1.11 (2 points)

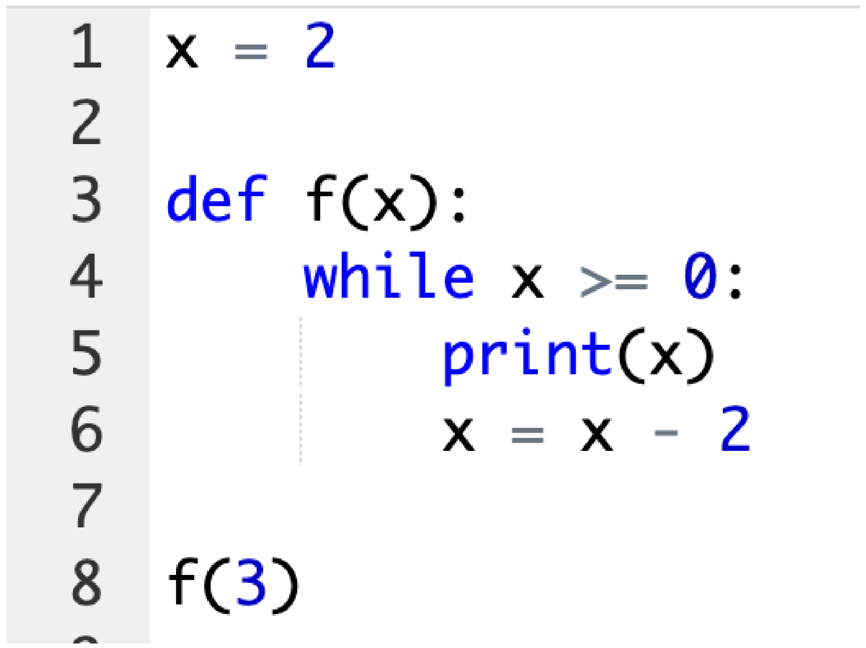
True or False: It is easier to go from aggregated data to disaggregated data than from disaggregated data to aggregated data.

True

[✓ False]

Question 2: Potpourri

Suppose we have the following Python program:

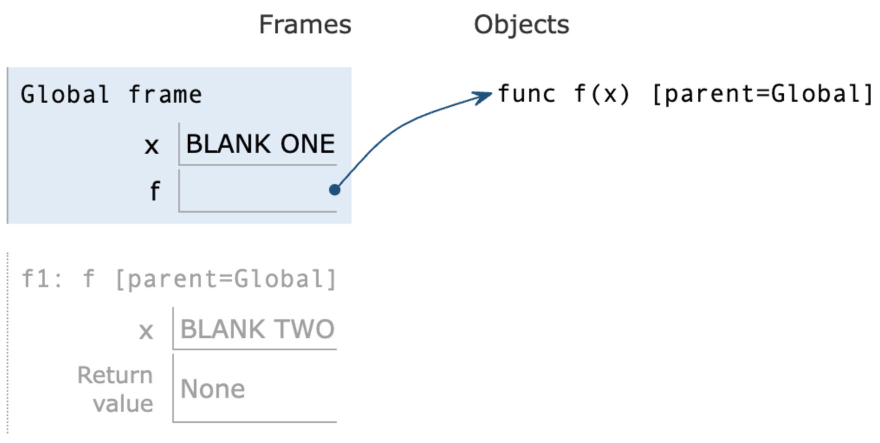


```python
x = 2

def f(x):
    while x >= 0:
        print(x)
        x = x - 2

f(3)
```

Assume we run the code, and this is the state of the environment diagram at the end of the program.



Question 2.1 (2 points)

What should go in BLANK ONE?

- -2
- -1
- 0
- 1
- [✓ 2]
- 3

Question 2.2 (3 points)

What should go in BLANK TWO?

- -2
- [✓ -1]
- 0
- 1
- 2
- 3

Question 2.3 (4 points)

What is printed out?

```
3
1
```

Question 3: RPS!

Let’s simulate a game of rock, paper, scissors! Here’s how the game works:

There are two players: Player 1 (P1) and Player 2 (P2).

Each game, a player will select one choice out of rock (“R”), paper (“P”), and scissors (“S”). Rock beats scissors, scissors beats paper, and paper beats rock.

If both players make the same selection, then the game ends in a tie.

Question 3.1 (6 points) Start by implementing `winner`, which takes in the choice by player 1 and player 2, and returns the following:

- If player 1 wins, return 1
- If player 2 wins, return 2
- If the game ends in a tie, return 0.

Python

```python
def winner(p1, p2):
      """
      Returns the winner of a game.
      "P" represents Paper
      "S" represents Scissors
      "R" represents Rock
      Inputs: Two Strings
      >>> winner("P", "S")
      2
      >>> winner("S", "P")
      1
      >>> winner("R", "P")
      2
      >>> winner("R", "S")
      1
      >>> winner("R", "R")
      0
      """
      #Write your code here
      if p1 == "P" and p2 == "S":
            return 2
      elif p1 == "S" and p2 == "R":
            return 2
      elif p1 == "R" and p2 == "P":
            return 2
      elif p1 == p2:
            return 0
      else:
            return 1
```

Question 3.2 (2 points) Next, let's develop a strategy for this game. All strategies will contain a numpy array of past `opponent_choices`, with the most recent decisions coming last. Let's continue by implementing `simple_strategy`, which always chooses `choice`, regardless of opponent actions.

```python
def simple_strategy(choice, opponent_choices):
    """
    Returns choice, regardless of the opponent's choice.
    Inputs:
    choice: a String
    opponent_choices: a NumPy array
    >>> simple_strategy("R", make_array())
    "R"
    >>> simple_strategy("R", make_array("R", "R"))
    "R"
    >>> simple_strategy("P", make_array("S", "P"))
    "P"
    """
    return choice
```

Question 3.3 (8 points) This simple strategy is probably too simple. Fill in the implementation of `smarter_strategy`, which follows these guidelines:

- If there are less than 2 opponent_choices that we know of, default to the `simple_strategy`.
- Otherwise, return the move that would beat the most recent opponent move (e.g. if the most recent opponent move is "R", we should choose "P", with <u>one exception</u>:
- If the most recent 2 opponent choices are the same, then repeat the opponent choice. For example, if the most recent opponent choices are "R" and "R", we should choose "R".

Python
```python
def smarter_strategy(choice, opponent_choices):
    """
    See problem description for more details.
    >>> smarter_strategy("R", make_array())
    "R"
    >>> smarter_strategy("R", make_array("R", "R"))
    "R"
    >>> smarter_strategy("P", make_array("S", "P"))
    "S" #Since the recent opponent choice is "P"
    """
    if len(opponent_choices) < 2:

        return simple_strategy(choice) #DO NOT JUST WRITE choice

    elif opponent_choices.item(-1) == opponent_choices.item(-2):

        return opponent_choices.item(-1)
    else:

        return {"R": "P", "P":"S", "S":"R"}[opponent_choices[-1]]
```

Question 3.4 (12 points) Now that we have a strategy, let’s run a simulation! Implement `simulation,` which takes in P1’s strategy, as well as the number of simulations to run (N). For each game that is played, we add a row to a Table containing P1’s choice, P2’s choice, and the result of the game (1 if P1 wins, 2 if P2 wins, and 0 otherwise).

For `simulation,` assume that P2’s strategy is to pick a random option from the three options of [”R”, ”P”, ”S”]. In order to track the simulation, it will also take in a seed in order for the outcomes to be deterministic (see next page for code).

Python

```python
def simulation(p1_strategy, choice, n, seed):
    """
    Runs n games of Rock, Paper, Scissors.
    Adds every game result to a Table that is returned.
    Inputs:
    p1_strategy: a strategy function for P1
    choice: P1's default choice
    n: number of simulations to run
    seed: random seed for reproducibility
    """
    np.random.seed(seed) #Set the seed
    outcomes = Table.with_columns("P1_choice", make_array(),
                                  "P2_choice", make_array(),
                                  "Result", make_array())
    opponent_choices = make_array()

    for i in np.arange(n):
        # P2 chooses randomly

        p2_choice = np.random.choice(["R", "P", "S"])
        # P1 chooses based on the strategy

        p1_choice = p1_strategy(choice, opponent_choices)
        # Determine the result

        game_result = winner(p1_choice, p2_choice)
        # Update the table

        outcomes = outcomes.with_row([p1_choice, p2_choice,
game_result])

        # Update opponent_choices

        opponent_choices = np.append(opponent_choices, p2_choice)
    return outcomes
```

Question 3.5 (2 points) True or False: Calling `simulation` twice with the same parameters is guaranteed to result in the same exact table.

[✓ True]

False

Suppose we run 1000 simulations, and get the following results in `tbl`:

`tbl`

P1_choice	P2_choice	Result
"R"	"S"	1
"R"	"R"	0
"P"	"S"	2
"R"	"P"	2
"S"	"P"	1
"P"	"P"	0
"P"	"S"	2

. . . (993 rows omitted)

Question 3.6 (3 points) Match each variable to its corresponding variable type. You may pick between: ordinal, nominal, discrete, and continuous.

Variable	Variable Type
`P1_choice`	Nominal
`P2_choice`	Nominal
`Result`	Nominal

Question 3.7 (3 points) Write an expression that creates a pivot table where the **columns** are the unique choices from P1, and the **rows** comprise the unique choices from P2. The value at that row and column is the number of times that particular combination of choices have been made. Assume that this table is called `tbl`.

```Python
tbl.pivot("P1_choice", "P2_choice")
```

Question 3.8 (4 points) Write an expression that creates a horizontal bar chart, where there is one bar for each type of choice ("R", "P", "S"), and the length of the bar represents the number of times that P2 made that choice. (Hint: just using barh will create a bar for every row, which likely isn't what you want).

```Python

tbl.group("P2_choice").barh("P2_choice")
```

Question 3.9 (4 points) Briefly explain why a line plot or scatter plot is an inappropriate visualization to use for `tbl`.

There is no numerical data to represent.

Question 3.10 (4 points) Let's add a new column to `tbl` called `Score`, which calculates the running score of Player 1 as the simulation runs. If Player 1 wins, we add 1 to the score. If Player 1 loses, we subtract 1 from the score. In other words, the updated `tbl` should look like this:

P1_choice	P2_choice	Result	Score
"R"	"S"	1	1
"R"	"R"	0	1
"P"	"S"	2	0
"R"	"P"	2	-1
"S"	"P"	1	0
"P"	"P"	0	0
"P"	"S"	2	-1

. . . (993 rows omitted)

You may use the following function, `convert`, which turns the `Result` into Player 1's score change.

```python
def convert(result):
      if result == 1:
            return 1
      elif result == 2:
            return -1
      else:
            return 0
```

Write an expression that adds this new `Score` column to `tbl`.

```python
```

```
score_changes = tbl.apply(convert, "Result")

running_score = np.cumsum(score_changes)

tbl.with_column("Score", running_score)
```

Question 4: Drawing Triangles!

Question 4.1 (6 points) We're back to drawing triangles! Fill in the implementation of `draw_triangle`, which prints out a left-aligned triangle with height `h`. See the doctests for details.

```python
def draw_triangle(h):
    """Prints a triangle with height h.
    h is guaranteed to be greater than or equal to 1.
    >>> draw_triangle(1)
    *
    >>> draw_triangle(2)
    *
    **
    >>> draw_triangle(5)
    *
    **
    ***
    ****
    *****
    """
    for layer in np.arange(1, h + 1):
        line = ""
        for i in np.arange(layer):
            line = line + "*"
        print(line)
```

Question 4.2 (6 points) Let's move onto drawing a cooler triangle. We're back to drawing triangles! Fill in the implementation of `draw_cooler_triangle`, which prints out a **center**-aligned triangle with height `h`. Since this triangle is center aligned, each layer of the triangle increases by 2 instead of 2. See the doctests for details.

Python

```python
def draw_cooler_triangle(h):
    """Prints a cooler triangle with height h.
    h is guaranteed to be greater than or equal to 1.
    >>> draw_cooler_triangle(1)
    *
    >>> draw_cooler_triangle(2)
     *
    ***
    >>> draw_cooler_triangle(3)
      *
     ***
    *****
    """
    for layer in np.arange(1, h + 1):
        line = ""
        for i in np.arange(1, h + layer):
            if i <= h - layer:
                line = line + " "
            else:
                line = line + "*"
        print(line)
```

Question 5: API Requests

Suppose we receive the following `response` from the OpenAI API after prompting it.

```python
response =

{
  "id": "chatcmpl-123",
  "object": "chat.completion",
  "created": 1677652288,
  "model": "gpt-4o-mini",
  "system_fingerprint": "fp_44709d6fcb",
  "choices": [{
    "index": 0,
    "message": {
      "role": "assistant",
      "content": "\n\nHello there, how may I assist you today?",
    },
    "logprobs": null,
    "finish_reason": "stop"
  }],
  ...
}
```

Question 5.1 (3 points) Write a Python expression that evaluates to the model we are using (in this case, "gpt-4o-mini") using `response` (which is a dictionary).

```python

response["model"]
```

Question 5.2 (3 points) Write a Python expression that evaluates to the value associated with `finish_reason` (in this case, "stop") using `response` (which is a dictionary).

```python
response["choices"][0]["finish_reason"]
```