Midterm Exam - DSC 10, Fall 2025

Full Name:

PID:

Section: ◯ A ◯ B ◯ C ◯ D

Instructions:

- This exam consists of 8 questions, worth a total of 62 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**.
- For full credit, your solutions must use methods of the course.
- You may use one page of double-sided handwritten notes. Aside from this, you may not refer to any other resources or technology during the exam. No calculators!

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.

Important: Before proceeding, make sure to rip off the last page of this exam packet and read the data description.

Question 1 (10 pts)

Each value in the `"ps"` column is a string formatted like `"('word', 'tag')"`, where the first part is the word and the second part is a tag indicating the word's part of speech (e.g. `"('getting', 'VBG')"`). Note that the word and the part of speech tag are each enclosed in single quotes. To clean the column so that `"ps"` contains **only** the part of speech tag, we run the following line of code.

```
words = words.assign(ps = words.get("ps").apply(fix_ps))
```

After we run this line of code, the first five rows of `words` should appear as follows.

word	freq	length	ps	diff
getting	2	7	VBG	0
apathy	1	6	JJ	10
devoured	2	8	VBD	5
identifiable	2	12	JJ	0
beat	3	4	NN	0

Which of the following options correctly define the function `fix_ps` to produce the desired result? **Select all that apply.**

☐
```
def fix_ps(x):
    return x.split("'")[3]
```

☐
```
def fix_ps(x):
    return x.split(",")[1].strip("')")
```

☐
```
def fix_ps(x):
    return x.split(", ")[1].strip("')")
```

☐
```
def fix_ps(x):
    return x.split(", ")[1].strip(")").strip("'")
```

☐
```
def fix_ps(x):
    return x.split(", ")[1].strip("'").strip(")")
```

☐ None of the above.

Important: For the rest of the exam, we'll assume that the `"ps"` column contains only the part of speech tag.

PID: _____

Question 2 (6 pts)

In the early 1990s, computational linguists developed the Penn Treebank part of speech tagging system. In this system, there are 36 tags that represent different parts of speech. The tags, their meanings, and a few examples are given in the `tags` DataFrame, whose first five rows are shown here.

	tag	meaning	examples
0	JJ	adjective	calm, yellow, curious
1	JJR	comparative adjective	nicer, smaller
2	NN	singular noun	tree, ticket, carpet
3	NNS	plural noun	trees, tickets, carpets
4	VBD	past tense verb	explored, wondered

All of the values in the `"ps"` column of `words` are tags from the Penn Treebank system. Remember, we're assuming that we've already applied the `fix_ps` function from the previous problem.

a) (3 pts) We want to merge `words` with `tags` so that we can see the meaning of each word's part of speech tag more easily. Fill in the blank in the code below to accomplish this.

Note that the index is lost when merging, so in order to keep `"word"` as the index of our result, we have to use `reset_index()` before merging, then set the index to `"word"` again after merging.

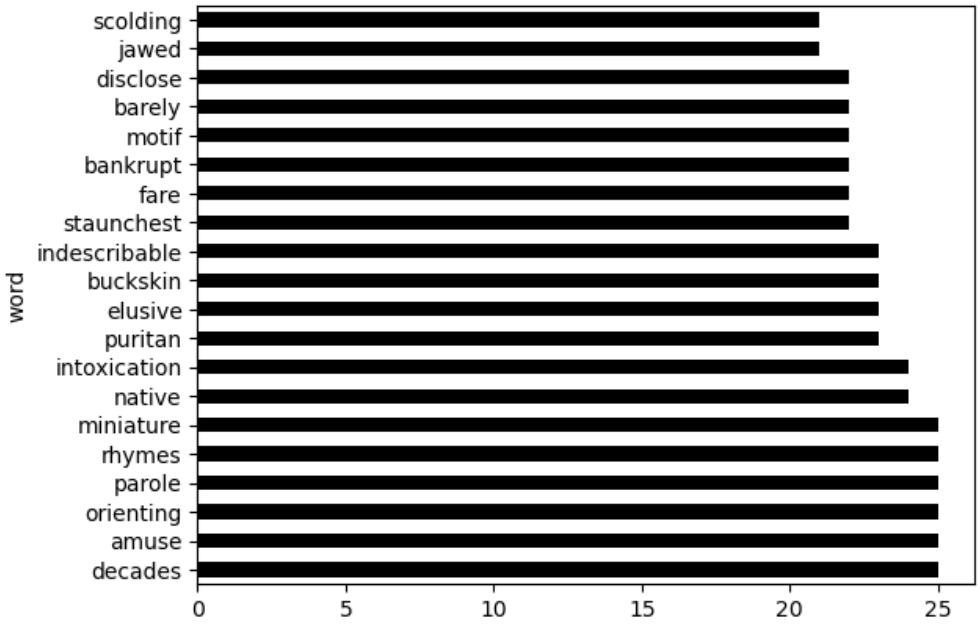
```
merged = words.reset_index().merge(__(a)__).set_index("word")
merged
```

`(a)`:

b) (3 pts) Suppose `words` contains 5357 rows and `tags` contains 36 rows. How many rows does `merged` contain? Give your answer as an integer or a mathematical expression that evaluates to an integer.

Question 3 (6 pts)

Below is a bar chart and the code that produced it.




```
(words.sort_values(by="diff", ascending=False).take(np.arange(20))
      .plot(kind='barh', y="diff"));
```

Determine whether each statement below is true or false.

a) (2 pts) Students found the word "decades" more difficult than any other word.

◯ True ◯ False

b) (2 pts) If we removed `ascending = False` from the code, the bar chart would show the same information, except the bars would appear in the opposite order from top to bottom.

◯ True ◯ False

c) (2 pts) This bar chart shows all words that more than 20 students found difficult. If we changed `np.arange(20)` to `np.arange(15)` in the code, the bar chart would instead show all words that more than 15 students found difficult.

◯ True ◯ False

PID: _____

Question 4 (6 pts)

The function below classifies the length of a word as `"short"`, `"medium"`, or `"long"`. Its input, `x`, represents the length of a word and will always be a positive integer.

```
def classify_length(x):
    if x <= 6:
        return "short"
    elif x <= 10:
        return "medium"
    else:
        return "long"
```

Fill in the blanks of the function below so that `classify_length_2` gives the same output as `classify_length` on all positive integer inputs. Each blank must be filled in with a **single line** of code.

```
def classify_length_2(x):
    if __(a)__:
        return "medium"
    if x >= 7:
        __(b)__
    __(c)__
```

(a):

(b):

(c):

Question 5 (9 pts)

Suppose we have a function `classify_difficulty` that takes as input the number of students that marked a word difficult, and returns a difficulty rating of the word: `"easy"`, `"moderate"`, or `"hard"`. Define the DataFrame `grouped` as follows. Note that this uses the `classify_length` function from the previous problem.

```
grouped = (words.assign( diff_category =
                         words.get("diff").apply(classify_difficulty),
                         length_category =
                         words.get("length").apply(classify_length)
                         )
                .groupby(["diff_category", "length_category"])
                .count()
                .reset_index()
          )
```

Suppose there is **at least one word** with every possible pair of values for `"diff_category"` and `"length_category"`.

a) (3 pts) How many rows does `grouped` have? Give your answer as an integer.

b) (3 pts) Fill in the blank to set `num` to the number of words whose `"diff_category"` is `"hard"` and whose `"length_category"` is `"short"`. Recall that `.groupby` arranges the rows of its output in alphabetical order.

```
num = grouped.get("ps").iloc[__(a)__]
```

(a):

c) (3 pts) We want to determine the proportion of `"short"` words that are `"hard"`. Fill in the blanks in the code below such that `prop` evaluates to this proportion.

```
denom = grouped[__(b)__].get("freq").__(c)__
prop = num / denom
```

(b):

(c):

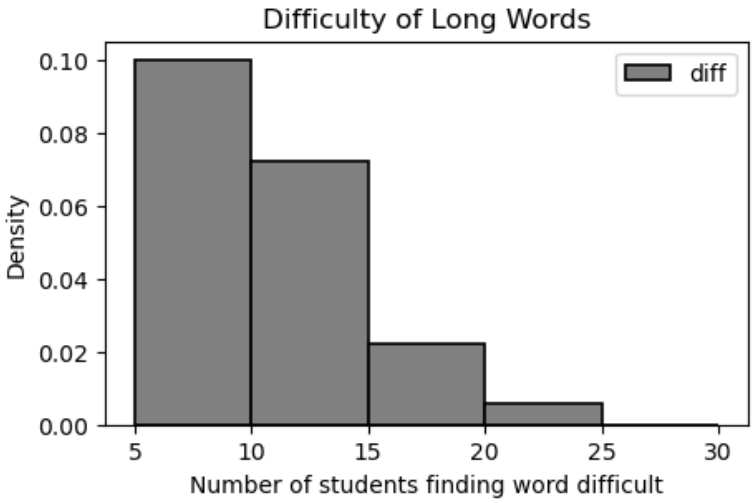
PID: _____

Question 6 (9 pts)

The DataFrame `long` was created by taking the 500 rows of `words` corresponding to words whose `"length"` is more than 10. Of the 500 words in `long`,

- 100 words were marked difficult by 5 or more students, and
- no words were marked difficult by more than 25 students.

The density histogram below was created using the data in the `"diff"` column of `long`. The `bins` argument was set as `bins=np.arange(5, 30, 5)` so the histogram **does not include** words that fewer than 5 students marked difficult.



a) (3 pts) How many words were marked difficult by 10 or more students? Give your answer as an integer.

b) (6 pts) Suppose we regenerated the histogram using the same data, but setting the `bins` argument as `bins=np.arange(0, 30, 5)`.

i) What would be the height of the bar $[5, 10)$? Give your answer as a decimal number.

ii) What would be the height of the bar $[0, 5)$? Give your answer as a decimal number.

Question 7 (8 pts)

Fill in the blanks below to estimate the probability that a randomly generated three-letter lowercase sequence forms a real English word represented in the `words` DataFrame.

```
letters = __(a)__
for letter in "abcdefghijklmnopqrstuvwxyz":
    letters = np.append(letters, letter)

repetitions = 10000
found = 0

for i in np.arange(repetitions):
    seq = "".join(np.random.choice(__(b)__, 3, replace=True))
    if __(c)__:
        found = found + 1

prob_real_word = __(d)__
prob_real_word
```

(a):

(b):

(c):

(d):

PID: _____

Question 8 (8 pts)

Suppose that for each word in the English language, Matt has a 70% chance of finding it difficult. Suppose also that the difficulty of each word is independent of the difficulty of any other word.

For all parts of this problem, give your answer as a decimal number or as an unsimplified mathematical expression.

a) (2 pts) If Matt reads a sentence of 10 words, what is the probability that he finds **every** word difficult?

b) (2 pts) If Matt reads a sentence of 10 words, what is the probability that he finds **at least one** word difficult?

Now suppose that Ryan finds the English language slightly more difficult, as it's not his first language:

- For each word, if Matt finds it difficult, there's a 90% chance that Ryan finds it difficult.
- For each word, if Matt doesn't find it difficult, there's still a 50% chance that Ryan finds it difficult.

c) (2 pts) For any given word, what is the probability that **at least one of Matt and Ryan** finds it difficult?

d) (2 pts) What is the probability that for **every** word in a sentence of 10 words, **exactly one of Matt and Ryan** finds the word difficult?

English Words

The English language is considered a challenging language to learn, largely because there are exceptions to just about every "rule"!

In this exam, we'll work with a data set of English words that appeared in texts used by Indian undergraduate students learning English. The students were asked to read the texts and mark the words they found difficult.

The DataFrame `words` is indexed by `"word"` and contains the following columns:

- `"freq"` (`int`): The frequency of the word, or the number of times it appeared in the texts.
- `"length"` (`int`): The length of the word, or the number of letters it contains.
- `"ps"` (`str`): The part of speech of the word (e.g. noun, verb, adjective). This is formatted as an ordered pair, where the first component is the word, and the second component is a tag indicating the part of speech (e.g. VBD for past tense verb). The tags and their meanings are determined by a system known as the Penn Treebank part of speech tagging system.
- `"diff"` (`int`): The number of students who marked the word difficult.

The first few rows of `words` are shown below, though there are many more rows than pictured.

word	freq	length	ps	diff
getting	2	7	('getting', 'VBG')	0
apathy	1	6	('apathy', 'JJ')	10
devoured	2	8	('devoured', 'VBD')	5
identifiable	2	12	('identifiable', 'JJ')	0
beat	3	4	('beat', 'NN')	0

Assume that we have already run `import babypandas as bpd` and `import numpy as np`.