CS 301 - Spring 2019

Instructor: Tyler Caraza-Harter

Final — 20%

(Last) Surname: _____ (First) Given name: _____

NetID (email): _____ @wisc.edu

Fill in these fields (left to right) on the scantron form (use #2 pencil):

1. LAST NAME (surname) and FIRST NAME (given name), fill in bubbles
2. IDENTIFICATION NUMBER is your Campus ID number, fill in bubbles
3. Under *ABC* of SPECIAL CODES, write your lecture number, fill in bubbles:
   002 - MWF 1:20pm (Tyler afternoon)
   003 - MWF 8:50am (Tyler morning)
4. Under *F* of SPECIAL CODES, write **B** and fill in bubble **7**

If you miss step 4 above (or do it wrong), the system may not grade you against the correct answer key, and your grade will be no better than if you were to randomly guess on each question. So don't forget!

Many of the problems in this exam are related to the course projects, but some questions assume the availability of slightly different functions (e.g., for accessing the data). We won't have any trick questions where we call a function that doesn't exist and you need to notice. Thus, if you see a call to a function we haven't explicitly defined in the problem, assume the function was properly implemented (perhaps immediately before the code snippet we DO show) and is available to you.

You may only reference your notesheet. You may not use books, your neighbors, calculators, or other electronic devices on this exam. Please place your student ID face up on your desk. Turn off and put away portable electronics now.

Use a #2 pencil to mark all answers. When you're done, please hand in these sheets in addition to your filled-in scantron.

(Blank Page)

General

1. What is the general shape of this DataFrame?

```
df = DataFrame({
    "7": [1, 1, 1, 1],
    "8": [4, 4, 4, 1]
})
```

   A. 3 columns, 3 rows
   B. 2 columns, 4 rows
   C. 1 column, 5 rows
   D. 8 columns, 8 rows
   E. 7 columns, 9 rows

2. Which statement(s), if inserted at the indicated location, would change what is printed?

```
import copy
x = [["A", "B"], ["C", "D"]]
y = copy.copy(x)
# INSERT CODE HERE
print(y)
```

   A. x = None
   B. x.append([”E”, ”F”])
   C. x[-1].append(”E”)
   D. B and C
   E. A and B and C

3. What is the range of possible values for total?

```
from numpy.random import choice
rolls = choice(6, size=10)
total = sum(rolls)
```

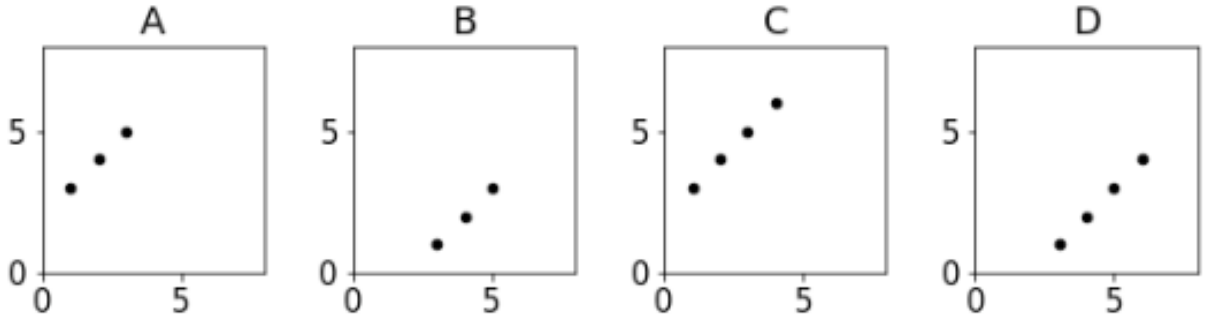
   A. 0 to 5 (inclusive)
   B. 1 to 6 (inclusive)
   C. 0 to 50 (inclusive)
   D. 9 to 54 (inclusive)
   E. 10 to 60 (inclusive)

4. What protocol does a client use to send a POST request for a resource to a server?

   A. HTTP B. HTML C. CSV D. SQL

5. Which plot corresponds to the following code snippet?

```
data = []
for i in range(3):
    data.append({"a": i + 3, "b": i + 1})
DataFrame(data).plot.scatter(x='a', y='b')
```



   A. A **B. B** C. C D. D

6. What is true about frames?

   A. there is always exactly one frame per function definition

   B. every variable is associated with exactly one frame

   C. all of the above

   D. none of the above

7. What numbers get printed, in what order?

```
def f(n):
    return n % 2 == 1

def exclude(nums, fn):
    for num in nums:
        if not fn(num):
            yield num

for val in exclude([1, 2, 3, 4], f):
    print(val)
```

   A. 0 B. 1, 3 **C. 2, 4** D. 1, 2, 3, 4 E. 4, 3, 2, 1

8. What hint will be printed?

```
hint = "trust your instincts"

def show_hint(hint="drink some coffee"):
    print(hint) # what is printed

def ask(q, a, hint="eliminate the worst option"):
    show_hint(hint=hint)
    # more code here...

ask("what do you get from a pampered cow?", "spoiled milk!")
```

A. trust your instincts

B. drink some coffee

C. eliminate the worst option

D. what do you get from a pampered cow?

E. spoiled milk!

9. What is `math.log10(301301301301)`? Yes, you should be able to narrow this down to the right option in your head.

A. 8.479 **B. 11.479** C. 14.479 D. 18.479 E. 23.479

Tweets

Assume the HTML for http://example.com/tweets.html is this:

```
<h2>Today</h2>
<ul>
  <li>Tweet 5: summer break!
  <li>Tweet 4: the semester is over
</ul>
<h1>Yesterday</h1>
<ul>
  <li>Tweet 3: debugging
  <li>Tweet 2: writing some code
  <li>Tweet 1: i'm broadcasting my life on this site
</ul>
```

Also assume this code is being used to scrape that page (part of the code is hidden by `????`):

```
import requests
from bs4 import BeautifulSoup

def scrape():
    try:
        resp = requests.get("http://example.com/tweets.html")
    except:
        print("WARNING!  Could not fetch page")
        return None

    doc = BeautifulSoup(????, "html.parser")
    element = doc.find("ul")
    return len(element.find_all("li"))
```

10. What text on the tweets.html page is larger?

    A. Today **B. Yesterday**

11. How many bulleted lists are on the tweets.html page?

    A. 0 B. 1 **C. 2** D. 3 E. 5

12. What should replace `????` so that the `scrape()` function works?

    A. resp **B. resp.text** C. resp.status_code D. resp.json() E. resp.dump()

13. Assuming no exceptions, what does `scrape()` return?

    A. 0 **B. 2** C. 3 D. 5 E. None

14. If tweets.html is deleted from that website so that the server returns a 404 status, what will `scrape()` do?

    A. prints "WARNING! Could not fetch page"

    B. returns None

    C. both A and B

    D. none of the above

15. What might `os.path.join("full", "1.csv")` return, assuming the following expression may run on either a Mac or Windows computer?

    A. `"full/1.csv"` B. `"full\1.csv"` C. `"full1.csv"` **D. A or B** E. A, B, or C

    Consider the following code that has a non-deterministic bug. If the directory `dir_path` contains at least one file with the json extension, it should the name of one of the JSON files in that directory; otherwise, it should return None. Assume the `sample` directory contains these files: 1.csv, 2.csv, 3.json, and 4.json. Assume `listdir` behaves according to specification.

```
def json_finder(dir_path):
    children = os.listdir(dir_path)
    for name in children:
        if name.endswith(".json"):
            return name
        else:
            return None
```

16. What is a **possible** list that `listdir` might return that would cause `json_finder("sample")` to return an **incorrect** result? **DROPPED FROM EXAM**

    A. `["1.csv", "2.csv"]`

    B. `["1.csv", "2.csv", "1.json", "2.json"]`

    C. `["1.json", "2.json", "1.csv", "2.csv"]`

    D. `["2.json", "1.json", "1.csv", "2.csv"]`

    E. both C and D

17. What is a **possible** list that `listdir` might return that would cause `json_finder("sample")` to return a **correct** result? **DROPPED FROM EXAM**

    A. `["1.csv", "2.csv"]`

    B. `["1.csv", "2.csv", "1.json", "2.json"]`

    C. `["1.json", "2.json", "1.csv", "2.csv"]`

    D. `["2.json", "1.json", "1.csv", "2.csv"]`

    E. both C and D

City

Assume the following code:

```
rows = [
    ["tree", "height", "priority"],
    ["A", "14", "6"],
    ["B", "68", "8"],
]
```

18. What does the following evaluate to? `len(rows[1:])`

    A. 0 B. 1 **C. 2** D. 3 E. 4

19. What would the following print? `print(rows[1][-1] + rows[2][-1])`

    A. 6 B. 8 C. 14 D. 48 **E. 68**

20. What does the following evaluate to? `rows[-1][rows[0].index("tree")]`

    A. `"tree"` B. `"priority"` **C.** `"B"` D. `"8"` E. `["B", "68", "8"]`

Assume the `hydrants` table in the `fire.db` database is as follows:

year	color	style	owner	alt	active
1999	red	K-81	private	1179	0
2000	red	M-3	public	1065	0
2001	green	Pacer	private	1058	1
2010	blue	Pacer	public	1081	1
2014	blue	Pacer	public	1052	1
2018	blue	Pacer	public	1109	1

Also assume `fire_df` contains the same data as the table, meaning it is setup like this:

```
import sqlite3
import pandas as pd
conn = sqlite3.connect("fire.db")
pd.read_sql("SELECT * FROM hydrants", conn)
conn.close()
```

21. Which query will have the most rows in the results?

    A. `SELECT AVG(alt) FROM hydrants GROUP BY year;`

    B. `SELECT AVG(alt) FROM hydrants GROUP BY color;`

    C. `SELECT MIN(alt), MAX(alt) FROM hydrants GROUP BY color;`

    D. `SELECT AVG(alt) FROM hydrants GROUP BY style;`

    E. `SELECT AVG(alt), MIN(alt), MAX(alt) FROM hydrants GROUP BY style;`

22. Which question, when translated to SQL, will require a HAVING?

A. what is the average altitude of all hydrants?

B. what is the average altitude of active hydrants?

C. what colors are used on at least two hydrants?

D. which hydrants have private ownership?

E. how many hydrants have been installed since 2000?

23. Which SQL query is most similar to this Python expression?

```
fire_df[fire_df["style"] == "Pacer"]["alt"].mean()
```

A. `SELECT alt, style = "Pacer" FROM hydrants;`

B. `SELECT AVG(alt) FROM hydrants WHERE style is "Pacer";`

C. `SELECT AVG(alt) FROM hydrants WHERE style = "Pacer";`

D. `SELECT AVG(alt) FROM hydrants WHERE style == "Pacer";`

E. `SELECT AVG(alt) FROM hydrants HAVING style == "Pacer";`

24. Which Python expression is most similar to this SQL query?

```
SELECT color, COUNT(*) FROM hydrants
WHERE year > 2000 GROUP BY color
```

A. `fire_df[fire_df["year"] > 2000].color_counts()`

B. `fire_df["color"].value_counts()[fire_df["year"] > 2000]`

C. `fire_df.set_index("color")[fire_df["year"] > 2000]`

D. `fire_df[fire_df["year"] > 2000].set_index("color")`

E. `fire_df[fire_df["year"] > 2000]["color"].value_counts()`

25. What does the following evaluate to?

```
len(fire_df[(fire_df["color"] == "red") & (fire_df["active"] == 1)])
```

A. 0 B. 1 C. 2 D. 3 E. 6

26. Assuming X and Y are valid, what can `fire_df.loc[X, Y]` evaluate to?

A. cell at row X and column Y

B. cell at row Y and column X

C. rows X and Y

D. rows from X to Y

E. columns from X to Y

Geography

For the following questions, assume the `population` DataFrame looks like the follow (numbers are in millions):

	Alabama	**Arizona**	**Florida**	**Indiana**	**Ohio**	**Wisconsin**
2000	4.4	5.1	15.9	6.0	11.3	5.3
2010	4.7	6.3	18.8	6.4	11.5	5.6
2015	4.8	6.7	19.9	6.5	11.5	5.7

27. Which expression(s) will evaluate to 5.7?

    A. `population["Wisconsin"][2015]`

    B. `population.loc[2015]["Wisconsin"]`

    C. `population.loc[2015, "Wisconsin"]`

    D. `population["Wisconsin"].iloc[-1]`

    E. all of the above

28. What is the type of the values returned by the following? `population.corr()`

    A. `float` B. `list` C. `dict` D. `Series` **E.** `DataFrame`

29. How many lines would be plotted by the following? `population.plot.line()`

    A. 1 B. 3 **C. 6** D. 18 E. 2000

30. What will be the values in the Series created from the following?
    `0.1 + population["Wisconsin"]`

    A. 5.4, 5.7, 5.8

    B. 0.1, 5.3, 5.6, 5.7

    C. 5.3, 5.6, 5.7, 0.1

    D. (5.3, 0.1), (5.6, 0.1), (5.7, 0.1)

    E. None of the above, because a type float my not be added to type Series

31. What will be the values in the Series created from the following?
    `population["Wisconsin"] + Series([5.7, 5.8], index=[2015, 2020])`

    A. 5.3, 5.6, 5.7, 5.8

    B. 5.3, 5.6, 11.4, 5.8

    C. NaN, NaN, 11.4, NaN

    D. NaN, NaN, NaN, NaN

    E. None of the above, because an UnalignedIndex exception will be raised

32. Which plot should **NOT** be used under any circumstances because it it is so misleading? Assume reasonable titles and axes labels (not shown in below code) are added.

    A. population["Wisconsin"].plot.bar()

    B. population.loc[2000].plot.bar()

    C. population.plot.bar(logy=True)

    D. population.sum().plot.bar(logy=True)

    E. population.plot.bar(stacked=True, logy=True)

For the following questions, this code just ran:

```
row = {}
distances = {}
cities = ["Kings Landing", "Winterfell", "Oldtown"]
dist = 1
for c1 in cities:
    for c2 in cities:
        row[c2] = dist # set distance
        dist += 1
    distances[c1] = row

city_series = Series(cities, index=[2, 1, 0])
```

33. How many times did the line with the "set distance" comment execute?

    A. 0 B. 1 C. 3 D. 6 **E. 9**

34. What is `distances["Winterfell"]["Winterfell"]`? Be careful!

    A. 0 B. 1 C. 2 **D. 8** E. 10

35. What is `city_series.iloc[2]`?

    A. `"Kings Landing"` B. `"Winterfell"` **C.** `"Oldtown"` D. -1 E. None

(Blank Page)