

CS 220 - Spring 2022

Instructors: Meenakshi Syamkumar, Andrew Kuemmel, Cole Nelson

Final — 10%

(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:
001 - MWF 11:00am (Meena)
002 - MWF 1:20pm (Meena)
003 - MWF 8:50am (Andy)
004 - MWF 9:55am (Cole)
 4. Under *F* of SPECIAL CODES, write **A** and fill in bubble **6**
-

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 note sheet. 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 (including smart watches) 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)

Web

1. Suppose the file at "https://www.msyalamkumar.com/scores.json" contains the following:

```
{"alice":100,"bob":200,"cindy":300}
```

After executing the below code snippet, which one of the following will not produce the output 200?

```
import requests
r = requests.get("https://www.msyalamkumar.com/scores.json")
resp_text = r.text
resp_json = r.json()
```

- A. `resp_text.split(",") [1].split(":") [1]`
 - B. `list(resp_json.items()) [-2] [-1]`
 - C. `resp_text["bob"]`
 - D. `resp_json["bob"]`
-

Assume that these are the contents of the file "http://seasons.com/spring.html":

```
<h1>Spring season</h1>
<ul>
    <li><a href="flowers.html">rain <i>wonderful rain</i></a></li>
    <li>puddles</li>
</ul>
```

Assume this code is being used to create a local copy of `spring.html` (part of the code is hidden by `????`):

```
import requests

def download(url):
    try:
        r = requests.get(url)      # line 1
        r.raise_for_status()      # line 2
        f = open("spring.html", 'w', encoding='utf-8')
        ????(r.text)            # line 3
        f.close()
    except requests.exceptions.HTTPError as e:
        print("WARNING! Could not fetch page")

download("http://seasons.com/spring.html")
```

-
2. What should replace `????` so that the code works as expected?
 - A. `read`
 - B. `write`
 - C. `f.read`
 - D. `f.write`
 3. Which of the following options is equivalent to line 2?
 - A. `assert 200`
 - B. `assert r.status_code == 200`
 - C. `assert r.text == 200`
 - D. `assert r.json() == 200`
-

Consider the below code snippet that is trying to parse the content of local file `spring.html`:

```
from bs4 import BeautifulSoup
import os

if os.path.exists("spring.html"):
    f = open("spring.html")
    html = ??????
    doc = BeautifulSoup(html, "html.parser")
    link = doc.find("a")
    f.close()
```

4. What should replace `????` so that the code works?
 - A. `str(f)`
 - B. `f.html()`
 - C. `f.json()`
 - D. `f.read()`
 - E. `html(f)`
5. What is the output of `print(link.get_text())`?
 - A. `["flowers.html", "rain wonderful rain"]`
 - B. `rain wonderful rain`
 - C. `flowers.html`
 - D. `rain <i>wonderful rain</i>`
6. What is the output of `len(doc.find_all("li"))`?
 - A. 0
 - B. 1
 - C. 2
 - D. `TypeError`

**Blank Page: this page is intentionally blank,
please turn over for next section questions**

Cars (Pandas)

Consider the following code:

```
import pandas as pd
cars = pd.DataFrame([
    ["Hyundai", "Elantra", 2022, "From $20,200", 357],
    ["Hyundai", "Sonata", 2020, "From $23,600", 238],
    ["Toyota", "Corolla", 2022, "From $22,350", 762],
    ["Ford", "Edge", 2021, "From $36,145", 63]],
    columns = ["Make", "Model", "Release Year",
               "Cost", "Units sold (in mil)"],
    index = ["1st", "2nd", "3rd", "4th"]
)
```

7. What is the data type of `cars["Release Year"]`?
A. str B. int C. Series D. list E. DataFrame
8. What is the ROWS x COLS dimension for the `cars` DataFrame?
A. 4 x 5 B. 4 x 6 C. 5 x 4 D. 5 x 5 E. 5 x 6
9. How can we get all of the data on the Hyundai Sonata?
 - A. `cars[(cars["Make"] == "Hyundai") and (cars["Model"] == "Sonata")]`
 - B. `cars[(cars["Make"] == "Hyundai") & (cars["Model"] == "Sonata")]`
 - C. `cars[cars["Make"] == "Hyundai" & cars["Model"] == "Sonata"]`
 - D. `cars[(cars["Make"] == "Hyundai") or (cars["Model"] == "Sonata")]`
 - E. `cars[(cars["Make"] == "Hyundai") | (cars["Model"] == "Sonata")]`
10. What is the output of the following code?
`cars["Release Year"].iloc[1:]`
 - A. [2020, 2022, 2021]
 - B. [2022, 2020, 2022, 2021]
 - C. `Series({0: 2020, 1:2022, 2:2021})`
 - D. `Series({"1st":2022, "2nd":2020, "3rd":2022, "4th":2021})`
 - E. `Series({"2nd":2020, "3rd":2022, "4th":2021})`

-
11. Find the total Units sold (in mil) for cars released in the year 2022.
- A. `cars["Units sold (in mil)"] [cars["Release Year"] == "2022"].sum()`
 - B. `cars["Units sold (in mil)"].sum()`
 - C. `sum(cars["2022"] ["Units sold (in mil)"])`
 - D. `cars[cars["Release Year"] == 2022] ["Units sold (in mil)"].sum()`
12. After this data was taken, Hyundai sold 20 million more Elantras and 15 million more Sonatas. Which of the following expressions would **NOT** produce the correct total Units sold (in mil) for each Model of Hyundai cars?
- A. `cars[cars["Make"] ["Units sold (in mil)"] == "Hyundai" + Series([20, 15])]`
 - B. `cars[cars["Make"] == "Hyundai"] ["Units sold (in mil)"] + pd.Series([20, 15], index = ["1st", "2nd"])`
 - C. `cars.iloc[:2] ["Units sold (in mil)"] + pd.Series({"1st": 20, "2nd": 15})`
 - D. `cars["Units sold (in mil)"] [cars["Make"] == "Hyundai"] + pd.Series({"1st": 20, "2nd": 15})`
13. Which expression answers this question: What is the most recent release year of a Hyundai car?
- A. `cars["Hyundai"] ["Release Year"].max()`
 - B. `cars[cars["Make"] == "Hyundai"] ["Release Year"].max()`
 - C. `cars[cars["Make"] == "Hyundai"] ["Release Year"].sort_values() [0]`
 - D. `cars[cars["Make"] == "Hyundai"].iloc["Release Year"].max()`

UFO sightings (Database)

Consider the following code, and the corresponding output:

```
import sqlite3
import pandas as pd

conn = sqlite3.connect("ufo_sightings.db")
ufo_sightings = pd.read_sql("SELECT * from ufo_sightings", conn)
ufo_sightings
```

	date	city	state	shape	duration
0	4/27/04	san marcos	TX	cylinder	2700
1	8/7/07	milwaukee	WI	triangle	600
2	2/14/10	lexington	NC	oval	30
3	2/14/10	madison	WI	circle	20
4	10/2/99	norwalk	CT	disk	1200
5	2/14/10	san antonio	TX	other	30

14. What is the type of `ufo_sightings`?
A. Series B. Dataframe C. List of lists D. List of dicts E. sqlite_master
15. How many rows of data would the following SQL query generate?

```
SELECT state, SUM(duration) as total_duration FROM ufo_sightings
WHERE duration >= 30 GROUP BY state HAVING total_duration > 600
```


A. 1 B. 2 C. 3 D. 4 E. 6
16. Which SQL query will return the city with the smallest UFO sighting duration?
A. `SELECT MIN(duration) FROM ufo_sightings`
B. `SELECT city FROM ufo_sightings ORDER BY duration DESC LIMIT 1`
C. `SELECT city FROM ufo_sightings ORDER BY duration LIMIT 1`
D. `SELECT * FROM ufo_sightings LIMIT 1`

17. What SQL query best translates the below Pandas code?

```
ufo_sightings[ufo_sightings["duration"] < 1000]["duration"].max()
```

- A. SELECT duration FROM ufo_sightings WHERE duration < 1000 ORDER BY duration LIMIT 1
- B. SELECT MAX(duration) FROM ufo_sightings HAVING duration < 1000
- C. SELECT MAX(duration) FROM ufo_sightings IF duration < 1000
- D. SELECT MAX(duration) FROM ufo_sightings WHERE duration < 1000

18. Which UFO shape does **NOT** feature in the results of the following SQL query?

```
SELECT shape FROM ufo_sightings  
WHERE state = 'TX' OR state = 'WI' OR state = 'NC'
```

- A. disk
- B. cylinder
- C. triangle
- D. oval
- E. circle

19. Which line of Python code will produce the closest results to the following query?

```
SELECT * FROM ufo_sightings WHERE duration > 100 ORDER BY state ASC
```

- A. ufo_sightings[ufo_sightings["duration"] > 100].sort_index()
- B. ufo_sightings[ufo_sightings["duration"] > 100].sort_index(by = "state")
- C. ufo_sightings[ufo_sightings["duration"] > 100].sort_values(by = "state")
- D. ufo_sightings[ufo_sightings["duration"] > 100].sort_values(by = "state", ascending = False)

20. Which SQL query answers the following question: Which dates have had more than 2 UFO sightings for a duration less than 1000?

- A. SELECT date, COUNT(*) as sight_count FROM ufo_sightings WHERE sight_count > 2 GROUP BY date HAVING duration < 1000
- B. SELECT date, COUNT(*) as sight_count FROM ufo_sightings WHERE duration < 1000 GROUP BY date HAVING sight_count > 2
- C. SELECT date, COUNT(*) as sight_count FROM ufo_sightings WHERE duration < 1000 AND sight_count > 2 GROUP BY date
- D. SELECT date, COUNT(*) as sight_count FROM ufo_sightings GROUP BY date HAVING duration < 1000 AND sight_count > 2

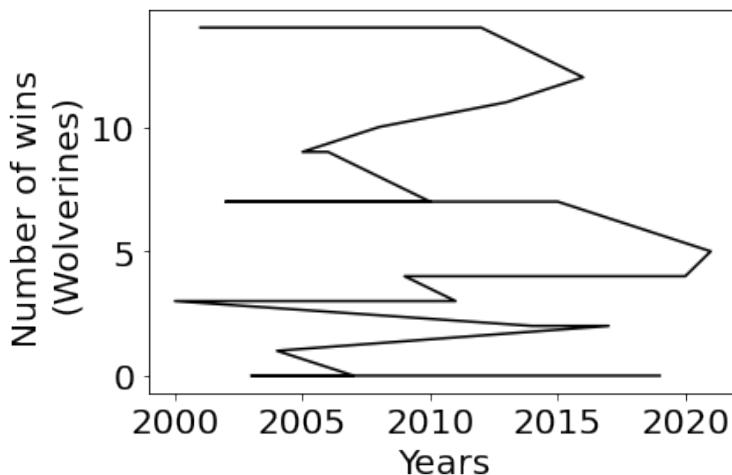
Plotting

Assume the following:

- football DataFrame contains the columns `team_name`, `year` (ranges from 2000 to 2022), and `wins`
- unique values stored within `team_name` column are Badgers, Wildcats, and Wolverines

21. Which of the following should replace `wolverines_df` in line 2 to create a correct line plot to visualize Wolverines' wins?

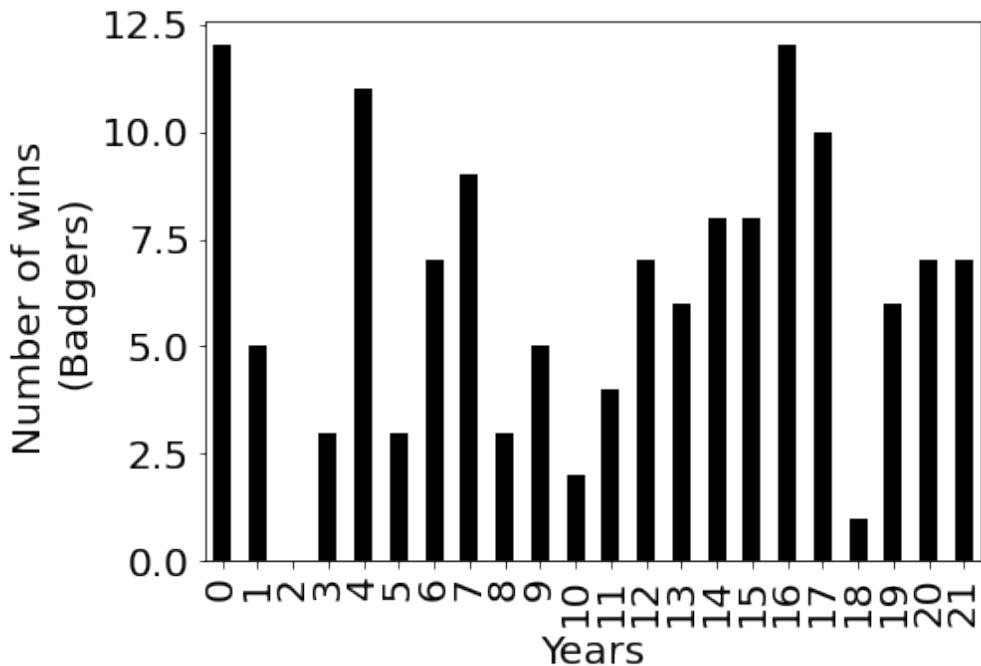
```
wolverines_df = football[football["team_name"] == 'Wolverines'] # line 1  
wolverines_df.plot.line(x = "year", y = "wins") # line 2
```



- A. `wolverines_df.sort_values(by = "wins")`
- B. `wolverines_df.set_index("year")`
- C. `wolverines_df.set_index("wins")`
- D. `wolverines_df.sort_values(by = "year")`

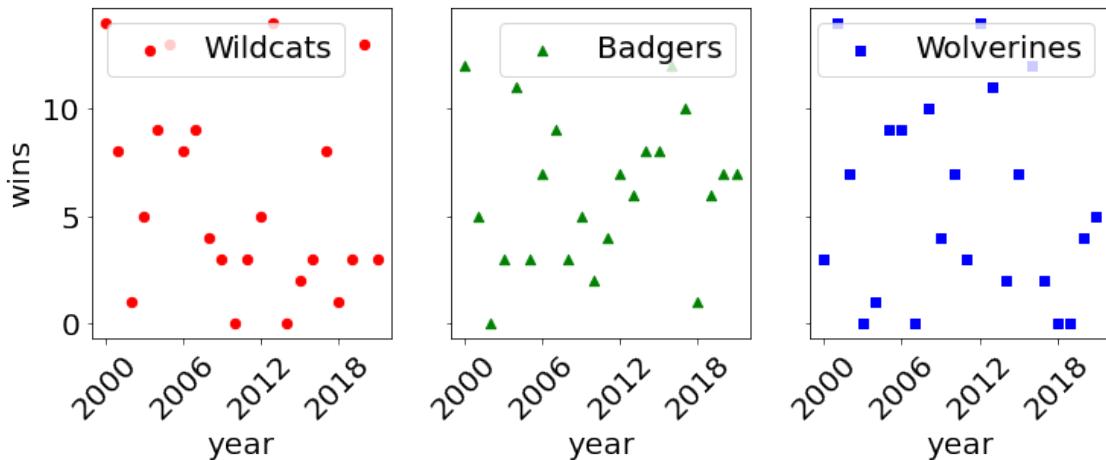
-
22. Which of the following should replace `badgers_df["wins"]` in line 2 to fix the issue with the below bar plot's x-axis?

```
badgers_df = football[football["team_name"] == 'Badgers'] # line 1  
ax = badgers_df["wins"].plot.bar() # line 2  
ax.set_ylabel("Number of wins\n(Badgers)") # line 3  
ax.set_xlabel("Years") # line 4
```



- A. `badgers_df.sort_values(by = "wins")["year"]`
- B. `badgers_df.set_index("year")["wins"]`
- C. `badgers_df.set_index("wins")["year"]`
- D. `badgers_df.sort_values(by = "year")["wins"]`

23. What arguments should be passed to the `plt.subplots` function to produce the below scatter plot?



- A. `ncols = 3, nrows = 3`
- B. `nrows = 3, sharey = True`
- C. `ncols = 3, nrows = 3, sharey = True`
- D. `ncols = 3, sharey = True`

24. How can we set the font size of all the plots to 20?

- A. `ax.set_font_size(20)`
 - B. `matplotlib.rcParams["font.size"] = 20`
 - C. `ax["font.size"] = 20`
 - D. `matplotlib.fontsize = "20"`
25. Suppose `s` is a Series, when we invoke `s.plot.line()`, which of the following in the Series represents the x-axis?
- A. integer positions
 - B. values
 - C. indexes
 - D. columns
26. Which of the following parameter-argument pair enables us to set logarithmic scale to y-axis?
- A. `logy = True`
 - B. `logx = True`
 - C. `logy = False`
 - D. `logx = False`

Review

27. The below function is supposed to calculate the median of a list. Fill in the blanks in the below function definition to calculate the first middle and second middle, in order to compute median of a list.

```
def median(items):
    items.sort()
    length = len(items)
    if length % 2 == 1:
        median = items[length // 2]
        return median
    else:
        firstMiddle = # Blank1 #
        secondMiddle = # Blank2 #
        median = (firstMiddle + secondMiddle) / 2
        return median
```

- A. items[(len(items) // 2) - 1], items[len(items) // 2 + 1]
 - B. items[(len(items) // 2)], items[(len(items) // 2) + 1]
 - C. items[(len(items) // 2)], items[len(items) // 2 + 1]
 - D. items[(len(items) // 2) - 1], items[len(items) // 2]
28. What is the value of total after executing the following snippet of code?

```
total = 0
for i in range(5000):
    if i % 2 == 0:
        continue
    elif i > 5:
        break
    total += i
```

- A. 0
- B. 4
- C. 9
- D. 10

29. What does the function call `mystery("hello")` evaluate to?

```
def mystery(s = "abcd"):
    for i in range(len(s)):
        if s[i] == s[-1]:
            return True
    return False
```

- A. True B. False C. Runtime error D. Syntax error

30. What is the output of the following code snippet?

```
result = 10
if result == 10:
    print("Condition 1")
elif result < 50:
    print("Condition 2")
else:
    print("Condition 3")
```

- A. Condition 1 B. Condition 2 C. Condition 3 D. Condition 1, Condition 2

31. What is the output for following code?

```
names = {1: "computer sciences", 2: "psychology", 3: "agriculture"}
sorted(names.items(), key = lambda x: x[1])
```

- A. `[(1, "computer sciences"), (2, "psychology"), (3, "agriculture")]`
B. `[(1, "computer sciences"), (3, "agriculture"), (2, "psychology")]`
C. `[(2, "psychology"), (3, "agriculture"), (1, "computer sciences")]`
D. `[(3, "agriculture"), (1, "computer sciences"), (2, "psychology")]`

32. What is the value of `list_a` after this code is run?

```
list_a = ["computer sciences", "psychology", "agriculture"]
list_b = list_a
list_a[2] = "psychology"
list_b[0] = ["psychology"]
```

- A. `["psychology", "psychology", "psychology"]`
B. `[["psychology"], "psychology", "psychology"]`
C. `["computer sciences", "psychology", "psychology"]`
D. `["computer sciences", "psychology", ["psychology"]]`

33. What is the value of `list_a` after this code is run?

```
import copy
list_a = ["computer sciences", "psychology", "agriculture"]
list_b = copy.deepcopy(list_a)
list_a[2] = "psychology"
list_b[0] = ["psychology"]
```

- A. ["psychology", "psychology", "psychology"]
- B. [[["psychology"]], "psychology", "psychology"]]
- C. ["computer sciences", "psychology", "psychology"]
- D. ["computer sciences", "psychology", ["psychology"]]

34. What is the value of `output` after this code is run?

```
my_list = [1, 2, 3]
my_dict = {1: "a", 2: "b"}
output = []
for i in my_list:
    try:
        output.append(my_dict[i])
    except:
        output.append(0)
```

- A. [0, "a", "b"]
- B. [3, "a", "b"]
- C. ["a", "b", 3]
- D. ["a", "b", 0]

35. What does the function call `some_func("apple")` evaluate to?

```
def some_func(my_string):
    if len(my_string) < 8:
        return my_string + some_func(my_string + my_string)
    return my_string
```

- A. "apple"
- B. "appleapple"
- C. "appleappleapple"
- D. "appleappleappleapple"
- E. "banana"

(Blank Page)

(Blank Page: you may tear this off for scratch work, but hand it in with your exam)

(Blank Page: you may tear this off for scratch work, but hand it in with your exam)