

CS 220 - Fall 2023  
Instructors: Mike Doescher, Gurmail Singh, and Cole Nelson

Final Exam — 12%

(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 08:50 AM (Mike)
  - 002 - MWF 11:00 AM (Mike)
  - 003 - MWF 01:20 PM (Gurmail)
  - 004 - MWF 03:30 PM (Gurmail)
  - 005 - MWF 08:50 AM (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 this exam and your note sheet in addition to your filled-in scantron.

---

# Pandas

Consider the DataFrame below that stores employee data for a company:

```
import pandas as pd

employee_data = {
    "Employee_ID": [101, 102, 103, 104, 105],
    "Name": ["Alice", "Bob", "Charlie", "David", "Emma"],
    "Department": ["HR", "Engineering", "Finance", "Engineering", "HR"],
    "Salary": [60000, 80000, 75000, 90000, 65000],
    "Years_Worked": [2, 5, 3, 7, 1]
}

employees = pd.DataFrame(employee_data)
```

1. Which of the following code snippets will output the salary of the employee with the highest salary?
  - A. `employees["Salary"].max()`
  - B. `employees["Salary"].idxmax()`
  - C. `employees[employees["Salary"].max()]["Salary"]`
  - D. `employees.loc[employees["Salary"].idxmax(), "Salary"]`
  - E. Both A and D are correct
2. If the company wants to update its DataFrame to change its department values from "Engineering" to "Tech" and rename all other departments to "Business", which of the following code snippets will accomplish this?
  - A. `employees["Department"] = "Tech" if employees["Department"] == "Tech" else "Business"`
  - B. `employees["Department"].apply(lambda val: "Tech" if val == "Engineering" else ["Business"])`
  - C. `employees["Department"].apply(lambda row: row["Tech"] if row["Departments"] == "Engineering" else "Business")`
  - D. `employees["Department"] = employees["Department"].apply(lambda val: "Tech" if val in ["Engineering"] else "Business")`
  - E. Both B and D are correct

- 
3. The DataFrame below stores movie ratings and reviews and will be referenced in the following questions:

```
import pandas as pd
```

```
movies = pd.DataFrame({
    "Movie_ID": [1, 2, 3, 4, 5],
    "Title": ["The Matrix", "Inception", "Pulp Fiction",
              "The Shawshank Redemption", "The Dark Knight"],
    "Genre": ["Action", "Sci-Fi", "Crime", "Drama", "Action"],
    "Rating": [4.5, 4.8, 4.2, 4.9, 4.7],
    "Year": [1999, 2010, 1994, 1994, 2008],
    "Director": ["Wachowskis", "Christopher Nolan", "Quentin Tarantino",
                 "Frank Darabont", "Christopher Nolan"]
})
```

What will the following code evaluate to?

```
type(movies["Genre"].value_counts())
```

- A. list
  - B. dict
  - C. str
  - D. Series
  - E. DataFrame
4. Which of the following expressions will NOT evaluate to "Quentin Tarantino"?
- A. `list(movies[movies["Title"] == "Pulp Fiction"]["Director"])[0]`
  - B. `movies.loc["Pulp Fiction"]["Director"]`
  - C. `movies.iloc[2]["Director"]`
  - D. `movies[movies["Title"] == "Pulp Fiction"].iloc[0]["Director"]`

- 
5. The following DataFrame has been provided to you and will be used in the following questions:

```
import pandas as pd
df = pd.DataFrame([
    {"year":1994.0},
    {"year":None},
    {"year":2004.0}
])
```

Which of the following code snippets will create a new column `decade` in the DataFrame that represents the decade each year belongs to?

- A. `df["decade"] = df["year"] / 10 * 10`
  - B. `df["decade"] = df["year"].floor(10) * 10`
  - C. `df["decade"] = df["year"] // 10 * 10`
  - D. `df["decade"] = df["year"] * 10 // 10`
  - E. None of the above
6. Which of the following code snippets will return a `pd.Series` object containing all of the years in the DataFrame with non-`NaN` values as ints?
- A. `df["year"].apply(int)`
  - B. `df["year"].totype(int)`
  - C. `int(df["year"])`
  - D. `df["year"].dropna().astype(int)`
  - E. Both B and D are correct

- 
7. Suppose that the DataFrame from the previous problems has been redefined with the code below:

```
df = pd.DataFrame([
    {"year":1994, "decade": 1990},
    {"year":None, "decade": None},
    {"year":2004, "decade": 2000},
    {"year":2008, "decade": 2000}
])
```

Which of the following code snippets will return the number of times the most common decade appears in the DataFrame?

- A. `list(df["decade"])[0]`
- B. `df["decade"].value_counts[0]`
- C. `df["decade"][0]`
- D. `df["decade"].value_counts().iloc[0]`

## Web

8. What is the name of the attribute for the anchor tag that determines where the user should be sent upon clicking the anchor?
- A. `to`
  - B. `link`
  - C. `href`
  - D. `target`
  - E. `goto`

---

9. Consider the following Python code snippet:

```
import requests

def fetch_data(api_url):
    try:
        response = requests.get(api_url)
        status_code = response.status_code
    except requests.HTTPError as e:
        return None
    if status_code == 200:
        return "Success"
    else:
        return "Not 200"

result = fetch_data("https://example.com/abc123")
```

What will be the value of `result` after executing this code if the API server cannot find the requested resource at the url "https://example.com/abc123"?

- A. None
- B. `requests.HTTPError`
- C. 404
- D. Not 200
- E. Success

- 
10. Which code snippet should replace the ??? in the code below to save the HTML content to a local file?

```
import requests

def save_html(url):
    try:
        response = requests.get(url)
        response.raise_for_status()
        file = open("web_page.html", "w", encoding="utf-8")
        ???
        file.close()
    except requests.HTTPError as e:
        print("WARNING! Could not fetch page")
```

- A. `write(response.read())`
- B. `file.read(response.text())`
- C. `file.write(response.text)`
- D. `file.read(response.read)`

11. What should replace the ??? in the code below to correctly save the HTML content as a string to the variable `html_content`?

```
from bs4 import BeautifulSoup
import os

if os.path.exists("web_page.html"):
    file = open("web_page.html")
    html_content = ???
    soup = BeautifulSoup(html_content, "html.parser")
    link = soup.find("a")
    file.close()
```

- A. `str(file)`
- B. `file.json()`
- C. `html(file)`
- D. `file.read()`
- E. `file.html`

---

12. Suppose the file at "https://www.example.com/data.json" contains the following:

```
{"john": 150, "emma": 220, "mike": 180}
```

After executing the below code snippet, which of the following code snippets will evaluate to the integer 220?

```
import requests
response = requests.get("https://www.example.com/data.json")
data_text = response.text
data_json = response.json()
```

- A. `data_text.split(",")[1].split(":")[1]`
- B. `list(data_json.items())[-2]`
- C. `data_text["emma"]`
- D. `data_json["emma"]`



---

## SQL

Assume that a valid connection to a SQL database has been created and the output of the following lines of code is shown in the table below:

```
companies = pd.read_sql("SELECT * FROM companies", conn)
companies
```

	name	industry	headquarters	established
0	Walmart	Retail	USA	1962
1	Samsung Electronics	Electronics	South Korea	1969
2	Toyota	Automotive	Japan	1937
3	Amazon	Retail	USA	1994
4	Mercedes Benz	Automotive	Germany	1926
5	Apple	Electronics	USA	1976
6	Volkswagen	Automotive	Germany	1937

13. Which of the following strings should replace the ??? in the code below to output the companies that were established in 1937?

```
query = "SELECT * FROM companies WHERE ???"
pd.read_sql(query, conn)
```

- A. `established(1937)`
  - B. `established = 1937`
  - C. `established(year) = 1937`
  - D. `established EQUALS 1937`
14. Which query will produce a DataFrame containing the rows with companies headquartered in the USA sorted by the name of the company in alphabetical order?
- A. `SELECT name FROM companies WHERE headquarters = "USA" SORT BY name`
  - B. `SELECT * FROM companies WHERE headquarters = "USA" ORDER BY name ASC`
  - C. `SELECT * FROM companies WHERE companies(headquarters) = "USA" ORDER BY name DESC`
  - D. `SELECT ROWS FROM companies WHERE headquarters = "USA" SORT BY name`
  - E. Both A and D are correct

---

15. Given the following query, how many rows will appear in the resulting DataFrame output?

```
query = """
SELECT headquarters, COUNT(*) AS num_companies
FROM companies
GROUP BY headquarters
HAVING num_companies > 2
ORDER BY num_companies DESC
"""
output = pd.read_sql(query, conn)
```

A. 0    **B. 1**    C. 2    D. 5    E. 7

16. Which SQL query can be used to identify the top-1 US-based retail company that had the earliest established date?

- A. SELECT \* FROM companies WHERE headquarters = "USA" & industry = "Retail" ORDER BY established ASC Limit 1
- B. SELECT \* FROM companies WHERE headquarters = "USA" ORDER BY established DESC Limit 1
- C. **SELECT \* FROM companies WHERE headquarters = "USA" AND industry = "Retail" ORDER BY established ASC Limit 1**
- D. SELECT \* FROM companies WHERE headquarters = "USA" AND industry = "Retail" ORDER BY established ASC

17. Which of the following statements best describes the result of the code below?

```
question_df = pd.read_sql("""
SELECT established, COUNT(*) AS num_companies
FROM companies
GROUP BY established
HAVING num_companies > 1
""", conn)

len(question_df)
```

- A. All companies in the database grouped by their established year
- B. The number of distinct years in the dataset
- C. The number of headquarters where more than one company is established in the same year
- D. **The number of years where more than one company is established in the same year**
- E. The number of distinct established years in the dataset

- 
18. What should replace the ??? in the code below to produce a DataFrame containing only the non-USA based companies?

```
non_usa = pd.read_sql("""  
SELECT *  
FROM companies  
WHERE ???  
""", conn)
```

- A. `headquarters != "USA"`
  - B. `headquarters = ["South Korea", "Japan", "Germany"]`
  - C. `headquarters == "USA" NOT`
  - D. `"USA" NOT headquarters`
  - E. `headquarters NOT "USA"`
19. Please select the answer that correctly describes the differences between the `.loc` and `.iloc` methods in the Pandas library.
- A. `loc` is primarily used with column labels, while `iloc` is used with row labels and integer positions.
  - B. Both `loc` and `iloc` are interchangeable and can be used interchangeably for any indexing operation in a DataFrame.
  - C. **`loc` is primarily used with row and column labels, while `iloc` is used with integer positions.**
  - D. Use `iloc` for both label-based and integer indexing, while `loc` is only used for accessing specific rows in a DataFrame.

## Plotting

20. What type of graph will be displayed after executing the code below?

```
df = pd.DataFrame({"A": [1,2,3,4], "B": [2,7,5,8], "C": [3,6,9,12]})  
df.plot.scatter(x="A", y="C")
```

- A. An error will occur - `df` is not a valid DataFrame
- B. A scatter plot with differently colored dots
- C. **A scatter plot with dots that are all the same color**
- D. An error will occur - columns should not be specified as strings
- E. A scatter plot with B as the y-axis

---

21. Which of the following statements is INCORRECT about plotting using Pandas?

- A. Pandas does not include support for pie charts
- B. Pandas lets you specify the color of the graph
- C. Pandas uses the names of columns as strings to specify the data for each axis
- D. The color argument is not required

22. Assume that the variable `df` stores a valid Pandas DataFrame. Which of the following code snippets will correctly replace the `???` in the code below so that the bar plot is given a label of "Hello World" on its X-axis?

```
ax = df.plot.bar()  
???
```

- A. `ax.xlabel = "Hello World"`
- B. `pd.set_xlabel("Hello World")`
- C. `ax.set_xlabel("Hello World")`
- D. `plt.xlabel = "Hello World"`
- E. None of the above

---

## Stars and Planets

23. The `get_all_paths_in` function shown below aims to get all the paths of files in a directory and the sub-directories it contains. Please select the function body that correctly implements this function when used to replace the ????

```
def get_all_paths_in(directory):
    paths = []
    files = os.listdir(directory)
    for file in files:
        if file.startswith("."):
            continue
        path = os.path.join(directory, file)
        if os.path.isfile(path):
            paths.append(path)
        elif os.path.isdir(path):
            ???
    return sorted(paths, reverse=True)
```

- A. `paths.append(get_all_paths_in(path))`
- B. `paths.extend(get_all_paths_in(path))`
- C. `paths.append(path)`
- D. `paths.extend(path)`

Suppose the variable `stars_dict` is defined as:

```
stars_dict = {
    "11 UMi": {"Stellar Mass": 2.78, "Stellar Age": 1.560},
    "14 And": {"Stellar Mass": 1.78, "Stellar Age": 4.500},
    "CD Cet": {"Stellar Mass": 0.16, "Stellar Age": 3.000},
    "mu2 Sco": {"Stellar Mass": 9.1, "Stellar Age": 0.020},
    "HD 96127": {"Stellar Mass": 12.94, "Stellar Age": 4.067}
}
```

- 
24. A star whose `stellar_mass` is greater than 0.5 but less than 8 will become a Red Giant, while a star whose `stellar_mass` is 8 or greater but less than 12 will become a White Dwarf. Any star with an even greater mass will end up as a Neutron Star. Please select the correct operators to replace the `????` blanks (from top to bottom) in the code below:

```
star_classes = {
    "Red Giant": [],
    "White Dwarf": [],
    "Neutron Star": []
}

for star in stars_dict:
    star_info = stars_dict[star]
    mass = star_info["Stellar Mass"]
    if mass == None:
        continue
    if 0.5 ???? mass ???? 8:
        star_classes["Red Giant"].append(star)
    elif 8 ???? mass ???? 12:
        star_classes["White Dwarf"].append(star)
    elif mass ???? 12:
        star_classes["Neutron Star"].append(star)
```

- A. `<=, <=, <=, <=, >`
  - B. `<, <, <=, <, >=`
  - C. `<=, <, <, <, >`
  - D. `<, <=, <=, <, >=`
25. Assume that the `star_classes` dict from the previous question was successfully created. What will be the output of the following line of code?

```
print(stars_dict[star_classes["Neutron Star"][0]]["Stellar Age"])
```

- A. 1.560
- B. 4.500
- C. 3.000
- D. 4.067
- E. This code will throw an error

---

26. The following functions have been defined for you:

```
import csv
import json
import os

def process_csv(filename):
    with open(filename) as file:
        csv_reader = csv.reader(file)
        list_data = list(csv_reader)
        return list_data

def read_json(path):
    with open(path, encoding="utf-8") as f:
        return json.load(f)

def get_planets_data(planet_file, mapping_file):
    mapping_dict = read_json(mapping_file)
    planets_csv = process_csv(planet_file)
    planets_header = planets_csv[0]
    planets_rows = planets_csv[1:]
    return (planets_header, planets_rows, mapping_dict)
```

In addition, your project file structure looks like this:

```
Root
|
|-- data
|   |-- planets1.csv
|   |-- mappings
|       |-- mapping1.json
|
|-- other_folder
```

Which of the following code snippets correctly calls the `get_planets_data` function and will work on any operating system?

- A. `get_planets_data("planets1.csv", "mapping1.json")`
- B. `get_planets_data(os.path.join("data", "planets1.csv", "data", "mappings", "mapping1.json"))`
- C. `get_planets_data(os.path.join("data", "planets1.csv"), os.path.join("data/mappings", "mapping1.json"))`
- D. `get_planets_data(os.path.join("data", "planets1.csv"), os.path.join("data", "mappings", "mapping1.json"))`
- E. Both C and D

---

# University Rankings

Imagine that the content of the file "2023.html" is in the current directory and has the following contents:

```
<table>
  <thead>
    <tr>
      <th>Column 1</th>
      <th>Column 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>
  </tbody>
</table>
```



- 
27. Which of the following code snippets can be used to replace the `????` in the code below so that the `headers` variable is created correctly? `headers` should contain a list with "Column 1" and "Column 2"

```
from bs4 import BeautifulSoup
import pandas as pd
with open("2023.html", "r", encoding="utf-8") as file:
    html_content = file.read()

soup = BeautifulSoup(html_content, "html.parser")
table = soup.find("table")
headers = ????
```

- A. `[val.get_text().strip() for val in table.find("thead").find_all("th")]`
- B. `[val.strip() for val in table.find("thead").find_all("td")]`
- C. `[val.get_text().strip() for val in table.find_all("tr")]`
- D. `[val.strip() for val in table.find_all("td")]`
- E. `[val.get_text().strip() for val in table.find("tbody").find_all("th")]`

- 
28. Assume that you have a variable `soup` that represents a valid BeautifulSoup HTML parser object, as created in the previous question. Which of the following code snippets correctly replaces the `????` in the code below? {data should contain a list of dictionaries where the keys are the table headers.

```
table = soup.find("table")
headers = ["Column 1", "Column 2"]
data = []
rows = table.find_all("tr")
for row in rows[1:]:
    cols = row.find_all("td")
    row_data = {}
    for i in range(len(headers)):
        ???
    data.append(row_data)
```

```
df = pd.DataFrame(data)
```

- A. `row_data[i] = cols[headers[i]].get_text()`
- B. `row_data[headers[i]] = cols[i].get_text()`
- C. `rows[cols[i]] = headers[i]`
- D. `row_data[cols[i].get_text()] = cols[headers[i]].get_text()`
- E. `row_data[headers] = cols[i]`

29. Assume that you have the same variable `soup` as referenced in the previous 2 questions. Then, what will the following code output?

```
print(soup.find("a"), soup.find_all("a"))
```

- A. None None
- B. "" []
- C. None []
- D. [] []
- E. This code will throw an error

- 
30. What is the primary difference between the `.find()` and `.find_all()` methods in BeautifulSoup?
- A. `.find()` searches only the first row of an HTML table, while `.find_all()` searches the entire document
  - B. `.find()` returns the first match, while `.find_all()` returns a list of all matches
  - C. `.find()` is used for HTML parsing, while `.find_all()` is used for XML parsing
  - D. `.find()` returns a list of all matches, while `.find_all()` returns the first

---

## SQL - Pandas Connection

31. Assume that all necessary imports have already been made. Please select the answer that correctly replaces all of the ??? spaces (in top-to-bottom order) as they appear below to prepare the rankings database:

```
var1 = "rankings.db"  
var2 = "rankings"  
var3 = "rankings.json"
```

```
rankings = pd.read_json(????)  
conn = sqlite3.connect(????)  
rankings.to_sql(????, conn, if_exists="replace", index=False)
```

- A. var1, var2, conn
- B. var 1, var 2, var3
- C. var2, var 1, rankings
- D. var3, rankings, var2
- E. var3, var1, var2

For the following questions, assume that you have an SQL table named `rankings` with the following data:

Year	Institution Name	Country	Fac Student	Cit Per Fac	International
2022	University of Tokyo	Japan	7.2	97.5	99.7
2022	University of Munich	Germany	14.3	97.2	91.5
2022	Kyoto University	Japan	6.5	96.8	98.9
2021	Wisconsin-Madison	USA	15.2	96.1	83.4
⋮	⋮	⋮	⋮	⋮	⋮

---

32. Which of the following SQL queries accesses the institutions with the top 10 highest "Fac Student" scores in Brazil in the year 2023?

- A. `SELECT "Institution Name", International FROM rankings WHERE Year = 2023 AND Country = "Brazil" ORDER BY International ASC LIMIT 10`
- B. `SELECT "Institution Name", International FROM rankings WHERE Country = "Brazil" ORDER BY "Fac Student" DESC`
- C. `SELECT "Fac Student" FROM rankings WHERE Year = 2023 AND Country = "Japan" ORDER BY International DESC LIMIT 10`
- D. `SELECT "Institution Name", "Fac Student" FROM rankings WHERE Year = 2023 AND Country = "Brazil" ORDER BY "Fac Student" DESC LIMIT 10`
- E. None of the above

33. The following query has been defined for you:

```
qry = """SELECT "International", "Cit Per Fac" FROM rankings
WHERE "year" = 2020 AND "Country" = "Germany"
"""
```

Which of the following code snippets will find the correlation between "International" and "Cit Per Fac" for institutions from Germany in the year 2020?

- A. `pd.read(qry, conn).loc["Cit Per Fac"].loc["International Students"].corr()`
- B. `pd.read(qry, conn).loc["International"].corr()`
- C. `pd.read_sql(qry, conn).loc["International Students"].loc["Overall"].corr()`
- D. `pd.read_sql(qry, conn).loc["International"].loc["Overall"].corr()`
- E. `pd.read_sql(qry, conn).corr().loc["International"].loc["Cit Per Fac"]`

- 
34. Which of the following queries will find the top 5 countries with the most institutions in the data for the year 2020?
- A. `""SELECT Country, SUM(*) AS "Num Institutions" FROM rankings WHERE Year = 2020 GROUP BY Country ORDER BY Country DESC, LIMIT 5""`
  - B. `""SELECT Country, COUNT(*) AS "Num Institutions" FROM rankings WHERE Year = 2020 GROUP BY Country ORDER BY "Num Institutions" DESC LIMIT 5""`
  - C. `""SELECT Country, AVG(*) AS "Num Institutions" FROM rankings WHERE Year = 2020 ORDER BY "Num Institutions" DESC GROUP BY Country LIMIT 5""`
  - D. `""SELECT Country, SUM(*) AS "Num Institutions" FROM rankings WHERE Year = 2020 ORDER BY "Num Institutions" DESC GROUP BY "Num Institutions" GROUP BY Country LIMIT 5""`
  - E. None of the above
35. Which of the following SELECT clauses will create a new column called "Cit Per Student" that is the product of the column "Fac Student" multiplied with the column "Cit Per Fac"?
- A. `SELECT ("Fac Student * Cit Per Fac") AS "Cit Per Student"`
  - B. `SELECT "Fac Student" * "Fac Student" AS ("Cit Per Fac")`
  - C. `SELECT COUNT("Fac Student" * "Cit Per Fac") AS "Cit Per Student"`
  - D. `SELECT ("Fac Student" * "Cit Per Fac") AS "Cit Per Student"`
  - E. Both A and D

---

(Blank Page)