(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 09:55 AM (Louis)
    004 - MWF 01:20 PM (Louis)
4. Under *F* of SPECIAL CODES, write **C** and fill in bubble **8**

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.

General Part 1

1. Consider the following code:

```
counter = int(32.7 // 8) ** 2

nums=[]

for i in range(10,counter):
    nums.append(i % 3)

if counter > 10 and nums[-1] != 0:
    print(nums[-2:])
elif counter < 10 or nums[-1] != 1:
    print(nums[:-3])
else:
    print(nums)
```

What is printed?

     A. [0, 1]

     B. [2, 0]

     C. [1, 2, 0]

     D. [1, 2, 0, 1, 2, 0]

2. Consider the following code:

```
def find_first_vowel(str):
    for i in range(len(str)):
        if str[i] in ["a","e","i","o","u"]:
            return i
    return -1


name=input("What is your full name?")
age=input("How old are you?")
first_name = name.split()[0]
print("The first vowel in {} is at {}, and your age doubled is {}"\
.format(first_name,find_first_vowel(first_name),age*2))
```

What is printed if the user enters `Adele Adkins` and `36` when prompted? Be careful.

    A. The first vowel in Adele is at 1, and your age doubled is 72

    B. The first vowel in Adele is at 2, and your age doubled is 3636

    C. The first vowel in Adele is at 0, and your age doubled is 72

    D. The first vowel in Adele is at 0, and your age doubled is 3636

Use the following code to answer the next three questions.

```
students = [
    { "Name":"Sally Smith", "Scores": [100, 95, 98], "Grade": "C"},
    { "Name":"Fred Flanders", "Scores": [62, 75, 81]},
    { "Name":"Molly Malone", "Scores": [91, 84]}]

def modify_score(student, which_score=0, new_value=100):
    student["Scores"][which_score] = new_value
    return student

def calculate_grade(student):
    total = 0
    count = 0
    for score in student["Scores"]:
        total += score
        count += 1
    percent = total / count
    if percent > 90:
        return "A"
    elif percent > 80:
        return "B"
    elif percent > 70:
        return "C"
    elif percent > 60:
        return "D"
```

3. What will be printed upon execution of
   `print(calculate_grade(modify_score(students[-1],-1)))`?

   A. A

   B. B

   C. C

   D. D

4. What does the following print:

```
students.append(students[0])
modify_score(students[-1], new_value=0)
print(calculate_grade(students[0]))
```

    A. A

    B. B

    C. C

    D. D

5. Which line of code should be used in place of ??? in order to correctly print the average score of all students whose list of scores has length greater than 2?

```
sum = 0
count = 0

???
    if len(student["Scores"]) > 2:
        for score in student["Scores"]:
            sum += score
            count += 1
print(sum / count)
```

    A. while count < len(students):

    B. for student in len(students):

    C. for student in range(len(students)):

    D. for student in students:

General Part 2

6. Consider the following Python code:

```
import copy

x = [[1, 2], {"a": [3, 4]}]
y = copy.copy(x)
z = copy.deepcopy(x)
y[1]["a"][0] = 99
z[0][1] = 100
print(x, y, z)
```

What will be printed by the `print(x, y, z)` statement?

    A. `[[1, 2], {"a": [3, 4]}] [[1, 2], {"a": [99, 4]}] [[1, 2], {"a": [3, 4]}]`

    B. `[[1, 2], {"a": [99, 4]}] [[1, 2], {"a": [99, 4]}] [[1, 2], {"a": [3, 4]}]`

    C. `[[1, 2], {"a": [3, 4]}] [[1, 2], {"a": [3, 4]}] [[1, 100], {"a": [3, 4]}]`

    D. `[[1, 2], {"a": [99, 4]}] [[1, 2], {"a": [99, 4]}] [[1, 100], {"a": [3, 4]}]`

7. You have the following recursive function defined:

```
def mystery(nums):
    if len(nums) <= 1:
        return nums
    pivot = nums[0]
    left = [x for x in nums[1:] if x < pivot]
    right = [x for x in nums[1:] if x >= pivot]
    return mystery(left) + [pivot] + mystery(right)
```

What does the function call `mystery([4, 2, 2, 5, 1])` return?

    A. $[1, 2, 2, 4, 5]$

    B. $[4, 2, 2, 5, 1]$

    C. $[5, 4, 2, 2, 1]$

    D. $[2, 1, 4, 2, 5]$

8. Consider the following Python code:

```python
def safe_apply(func, x):
    try:
        return func(x)
    except ZeroDivisionError:
        return "ZeroDiv"
    except TypeError:
        return "TypeErr"
    except Exception as e:
        return "OtherErr: {}".format(e)

def incr(x): return x + 1
def boom(x): return 1 / x

print(safe_apply(boom, incr(-1)))
```

What is printed?

    A. 0

    B. ZeroDiv

    C. TypeErr

    D. OtherErr: division by zero

9. Consider the following code:

```
import os

def count_files(directory, file_extension=None):
    try:
        total = 0
        for item in os.listdir(directory):
            path = os.path.join(directory, item)
            if os.path.isdir(path):
                total += count_files(path, file_extension)  # Recursive call
            elif os.path.isfile(path):
                if file_extension == None or path.endswith(file_extension):
                    total += 1
        return total
    except Exception as e:
        return "Error: {}".format(e)
```

What will `count_files(directory, ".py")` return if the directory contains:

- A folder with 3 `.py` files

- 2 `.txt` files

- Another folder with 1 `.py` file and 1 `.txt` file

    A. 3
    B. 4
    C. 5
    D. Error

General Part 3

Use the following code to answer the next two questions.

```python
import pandas as pd

course_data = {
    "Course_ID": ["CS 220", "MATH 240", "CHEM 103", "ME 361"],
    "Title": ["Data Science Programming I", "Intro to Discrete", "Gen Chem I",
    "Thermodynamics"],
    "Instructor": ["Dr. Smith", "Dr. Johnson", "Dr. Patel", "Dr. Nguyen"],
    "Credits": [3, 4, 4, 3],
    "Students_Enrolled": [120, 85, 100, 60],
    "Prerequisites": [[], ["Algebra"], ["Algebra"], ["Gen Chem I"]],
    "Online_Available": [True, False, True, False]
}

courses = pd.DataFrame(course_data)
```

10. Which of the following options filters `courses` to only those that can be taken online? (i.e. where `Online_Available` is True)

    A. `courses[courses["Online_Available"] == True]`

    B. `courses.loc[courses["Online_Available"]]`

    C. `courses.filter("Online_Available" == True)`

    D. A and B

    E. A and C

11. If the university wants to create a new column, `Difficulty`, where courses with `Credits` greater than 4 are labeled as "Hard" and all others are labeled as "Easy", which of the following code snippets will achieve this?

    A. `courses["Difficulty"] = courses["Credits"].replace(4, "Hard")`

    B. `courses["Difficulty"] = courses["Credits"].apply(lambda x: "Hard" if x >= 4 else "Easy")`

    C. `courses["Difficulty"] = "Easy"`

    D. `courses["Difficulty"] = courses["Credits"].substitute({4: "Hard", 3: "Easy"})`

    E. None of the above

Use the following code to answer the next two questions.

```python
import pandas as pd

df = pd.DataFrame([
    {"student_id": 101, "name": "Alice", "major": "Computer Science",
        "gpa": 3.9, "graduation_year": 2025},
    {"student_id": 102, "name": "Bob", "major": "Mathematics",
        "gpa": 3.5, "graduation_year": 2024},
    {"student_id": 103, "name": "Charlie", "major": "Physics",
        "gpa": 3.8, "graduation_year": 2023},
    {"student_id": 104, "name": "David", "major": "Biology",
        "gpa": 3.2, "graduation_year": 2025},
    {"student_id": 105, "name": "Emma", "major": "Chemistry",
        "gpa": 3.7, "graduation_year": 2024},
    {"student_id": 106, "name": "Frank", "major": "Computer Science",
        "gpa": 3.4, "graduation_year": 2023}
])
```

12. Which of the following code snippets will get Bob's gpa?

     A. `df.loc[0, "gpa"]`

     B. `df.iloc[0, "gpa"]`

     C. `df.loc[1, "gpa"]`

     D. `df.iloc[1, "gpa"]`

     E. `None of the above`

13. Which of the following code snippets will return a new DataFrame containing only students whose GPA is greater than or equal to 3.5?

     A. `df[df["gpa"] >= 3.5]`

     B. `df[df["gpa"].apply(lambda df:  df["gpa"] >= 3.5)]`

     C. `df.iloc[df["gpa"] >= 3.5]`

     D. A and C

     E. None of the above

14. Which of the following options checks the status of the HTTP request trying to get a non-existent file and throws the appropriate exception if the file is not returned successfully?

```
import requests

url = "https://cs220.cs.wisc.edu/hello.txtttttt"
r = requests.get(url)
```

      A. `r.assert_no_errors()`

      B. `assert r.status_code != 200`

      C. `r.raise_for_status()`

      D. `assert r.status_code == 404`

      E. None of the above

15. What is the value of `result` after running the following code?

```
from bs4 import BeautifulSoup

html_string = "<b>To Do List</b><ul><li>Eat Healthy</li>
               <li>Sleep <b>More</b></li><li>Exercise</li></ul>"
bs_obj = BeautifulSoup(html_string, "html.parser")
result = bs_obj.find("li")
```

      A. `<li>Eat Healthy</li>`

      B. `Eat Healthy`

      C. `[<li>Eat Healthy</li>, <li>Sleep <b>More</b></li>, <li>Exercise</li>]`

      D. `Eat Healthy, Sleep <b>More</b>, Exercise`

      E. None of the above

16. Assume that the file at `"https://cs220.cs.wisc.edu/num_students.json"` contains the following:
`{"Section 001":  200, "Section 002":  300, "Section 003":  125}`
After executing the following code snippet, which of the following will produce the output 300 as an integer?

```
import requests
r = requests.get("https://cs220.cs.wisc.edu/num_students.json")
contents_text = r.text
contents_json = r.json()
```

    A. `list(contents_json.values())[-1]`

    B. `contents_text["Section 002"]`

    C. `list(contents_json.items())[-2]`

    D. `contents_text.split(",")[1].split(":")[1]`

    E. `contents_json["Section 002"]`

17. Assume that the dataframe `df` contains the data below:

team_num	department	employee_count	total_salary	avg_salary	projects_completed
5	Engineering	25	2,000,000	80,000	150
6	Sales	15	1,000,000	66,667	75
1	HR	10	700,000	70,000	50
3	Marketing	20	1,400,000	70,000	100
2	Support	30	1,800,000	60,000	200
4	Engineering	10	1,100,000	65,000	215
9	HR	5	200,000	40,000	55

Which of the following SQL queries is logically equivalent to the pandas command `df.groupby("department").sum()["total_salary"]`?

    A. `SELECT department, AVG(total_salary) FROM df GROUP BY department`

    B. `SELECT department, COUNT(total_salary) FROM df GROUP BY total_salary`

    C. `SELECT department, SUM(total_salary) FROM df GROUP BY department`

    D. `SELECT department FROM df WHERE SUM(total_salary)`

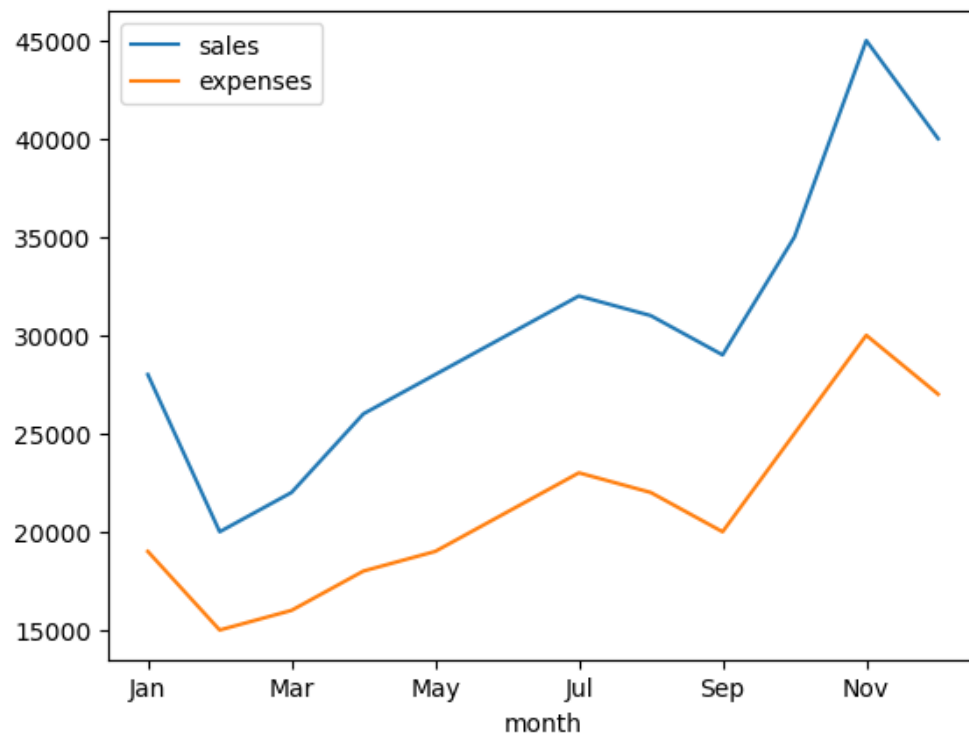   E. `SELECT df.groupby("department").sum("total_salary")`

18. Assume that there are some missing values in the `projects_completed` column. Which of the following code snippets will modify `df` by filling all the missing values in the `projects_completed` column with the value 0?

   A. `df["projects_completed"]=df["projects_completed"].dropna(0)`

   B. `df["projects_completed"]=df["projects_completed"].replace(np.NaN, 0)`

   C. `df["projects_completed"].apply(0)`

   D. `df["projects_completed"].na_fill(0)`

   E. `A and B`

19. Use the following code to answer this question:

```
df = pd.DataFrame({
    "month": ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
              "Sep", "Oct", "Nov", "Dec"],
    "sales": [28000, 20000, 22000, 26000, 28000, 30000, 32000, 31000,
              29000, 35000, 45000, 40000],
    "expenses": [19000, 15000, 16000, 18000, 19000, 21000, 23000,
              22000, 20000, 25000, 30000, 27000]
})
```

Which code snippet produces the following plot?

   A. `df.plot.scatter(x="month", y=["sales", "expenses"])`

   B. `df.plot.bar(x="month", y=["sales", "expenses"])`

   C. `df.plot.line(x="month", y=["sales", "expenses"])`

   D. `df.plot()`

   E. `C and D`

20. Given the `DataFrame` `trees_df` with columns `"age"`, `"height"`, and `"diameter"`, representing various attributes of trees, which of the following lines of code correctly scales the size of markers in a scatter plot to be proportional to the `"diameter"` column, with the size scaled up by a factor of 30, and plots the relationship between `"age"` and `"height"`?

   A. `trees_df.plot.scatter(x="age", y="height", s="diameter"*30)`

   B. `trees_df.plot.scatter(x="age", y="height", s=trees_df["diameter"]*30)`

   C. `trees_df.plot.scatter(x="age", y="height", marker=trees_df["diameter"]*30)`

   D. `trees_df.plot.scatter(x="age", y="height", s=trees_df["diameter"], scale=30)`

For the next six questions, consider a SQLite database named `bus.db` with the following tables:

- `routes` **table:**

    - `route_short_name`: Short identifier for each route (e.g., "A", "B").
    - `route_service_name`: Description of the route's service area.
    - `route_desc`: Additional details about the route.
    - `route_url`: URL for more information about the route.
    - `route_color`: Color code representing the route.

- `stops` **table:**

    - `stop_id`: Unique identifier for each stop.
    - `stop_name`: Name of the bus stop.
    - `stop_lat` and `stop_lon`: Latitude and longitude of the stop.
    - `jurisdiction_id`: Identifier of the governing area for the stop.
    - `wheelchair`: Indicates if the stop is wheelchair accessible (1 = Yes, 0 = No).

21. You need to find the name of the route (`route_short_name`) that appears in the **last row** of the `routes` table when the results are sorted alphabetically by the `route_short_name`. Below is a partially completed code snippet. Fill in the missing query to get the correct result.

```
# Assume conn is the connection object
qry = """???"""
df = pd.read_sql(qry, conn)
last_route_name = df.iloc[0][0]  # Retrieves the result
```

    A. SELECT route_short_name FROM routes ORDER BY route_short_name DESC LIMIT 1

    B. SELECT route_short_name FROM routes ORDER BY route_short_name ASC

    C. SELECT route_short_name FROM routes WHERE route_short_name = 'Z' ORDER BY route_short_name DESC

    D. SELECT route_short_name FROM routes ORDER BY route_short_name LIMIT 1

22. Databases are often preferred over other data storage methods, such as CSV files or JSON files. Which of the following is a key feature of databases that provides an advantage in terms of data structure and integrity?

    A. They can store data in a structured table format, ensuring named columns and rows.

    B. They allow dynamic resizing of data when rows or columns are added.

    C. They ensure all values in a column adhere to a strict data type (e.g., integers, strings).

    D. They can handle null values and missing data without crashing.

    E. All of the above.

23. Write a query to calculate the total number of bus stops (`stop_id`) and the average latitude (`stop_lat`) of wheelchair-accessible stops for each jurisdiction. The query must also only include jurisdictions that have more than 5 wheelchair-accessible stops. Which query correctly achieves this?

A. ```
SELECT jurisdiction_id, COUNT(stop_id) AS stop_count, SUM(stop_lat)
AS avg_lat
FROM stops
WHERE wheelchair = 1
GROUP BY jurisdiction_id
HAVING stop_count > 5
```

B. ```
SELECT jurisdiction_id, COUNT(stop_id) AS stop_count, AVG(stop_lat)
AS avg_lat
FROM stops
WHERE wheelchair = 1
GROUP BY jurisdiction_id
HAVING stop_count > 5
```

C. ```
SELECT jurisdiction_id, COUNT(stop_id) AS stop_count, AVG(stop_lat)
AS avg_lat
FROM stops
WHERE wheelchair = 1 AND avg_lat > 5
GROUP BY jurisdiction_id
```

D. ```
SELECT jurisdiction_id, COUNT(stop_id) AS stop_count, AVG(stop_lat)
AS avg_lat
FROM stops
GROUP BY jurisdiction_id
HAVING stop_count > 5 AND wheelchair = 1
```

24. Which query calculates the total number of unique routes (`route_short_name`) and finds the count of route descriptions (`route_desc`) for each `route_color` in the `routes` table. The query should sort the results by `route_color` in alphabetical order?

A. `SELECT route_color, COUNT(DISTINCT route_short_name) AS`
`unique_routes, COUNT(route_desc) AS desc_count`
`FROM routes`
`GROUP BY route_color`
`ORDER BY route_color ASC`

B. `SELECT route_color, COUNT(DISTINCT route_short_name) AS`
`unique_routes, COUNT(route_desc) AS desc_count`
`FROM routes`
`GROUP BY route_color`
`HAVING unique_routes > 2`

C. `SELECT route_color, COUNT(route_short_name) AS unique_routes,`
`COUNT(route_desc) AS desc_count`
`FROM routes`
`GROUP BY route_color`
`ORDER BY route_color DESC`

D. `SELECT route_color, COUNT(DISTINCT route_short_name) AS unique_routes`
`FROM routes`
`GROUP BY route_color`
`ORDER BY unique_routes DESC`

The next two questions will follow the below `stops` table:

stop_id	stop_name	stop_lat	stop_lon	wheelchair
1	W Johnson at N Lake	10	-20	1
2	University at N Lake	15	-10	1
3	E Washington at S Second	5	-5	1
4	E Washington at Aberg	20	-15	0
5	E Washington at S Sixth	0	-25	1

25. You want to calculate the total number of stops `total_stops` (equivalently `COUNT(*)`) and the number of wheelchair-accessible stops `wheelchair_stops` (equivalently `COUNT(*) WHERE wheelchair = 1`) in the `stops` table. Which of the following is the correct equivalent Pandas code for this query?

   A.
   ```
   total_stops = len(stops)
   wheelchair_stops = len(stops[stops["wheelchair"] == 1])
   ```

   B.
   ```
   total_stops = stops["stop_id"].count()
   wheelchair_stops = stops["wheelchair"].mean()
   ```

   C.
   ```
   total_stops = stops["stop_id"].count()
   wheelchair_stops = stops[stops["wheelchair"] ==
   1]["stop_id"].value_counts().iloc[0]
   ```

   D.
   ```
   total_stops = len(stops)
   wheelchair_stops = stops["wheelchair"].nunique()
   ```

26. The following SQL query is executed:

   ```
   SELECT stop_name, (stop_lat * -1) + stop_lon AS adjusted_lat_lon
   FROM stops
   WHERE wheelchair = 1 AND stop_lat > 5
   ORDER BY adjusted_lat_lon ASC
   LIMIT 2
   ```

   What are the results of the above query?

   A.

stop_name	adjusted_lat_lon
W Johnson at N Lake	**-30**
University at N Lake	**-25**

   B.

stop_name	adjusted_lat_lon
W Johnson at N Lake	-30
E Washington at S Sixth	-25

   C.

stop_name	adjusted_lat_lon
E Washington at Aberg	-35
W Johnson at N Lake	-30

   D.

stop_name	adjusted_lat_lon
E Washington at S Second	-10
University at N Lake	-25

Movies

Given the `bucketize` function, which groups movies by a specified category (e.g., `"year"`, `"director"`, `"genres"`) and returns a dictionary where the keys are unique values of that category, and the values are lists of movies corresponding to each key:

```python
def bucketize(movie_list, category):
    buckets = {}
    for movie in movie_list:
        category_value = movie[category]
        if (type(category_value)) == list:
            for value in category_value:
                if value not in buckets:
                    buckets[value] = []
                buckets[value].append(movie)
        else:
            if category_value not in buckets:
                buckets[category_value] = []
            buckets[category_value].append(movie)
    return buckets
```

And `movies` data structure, which is a list of dictionaries representing movies in the IMDb dataset:

```python
[{"title": "Dune: Part Two",
"year": 2024,
"duration": 166,
"genres": ["Action", "Adventure", "Drama"],
"rating": 9.0,
"directors": ["Denis Villeneuve"],
"cast": ["Timothée Chalamet", "Zendaya", "Rebecca Ferguson"]},
{"title": "Barbie",
"year": 2023,
"duration": 114,
"genres": ["Adventure", "Comedy", "Fantasy"],
"rating": 6.9,
"directors": ["Greta Gerwig"],
"cast": ["Margot Robbie", "Ryan Gosling", "Issa Rae"]},
...]
```

27. Which of the following correctly describes the output of `bucketize(movies, "genres")` when applied to the `movies` data?

    A. A dictionary where each key is a unique genre, and each value is a list of all movies that belong to that genre.

    B. A dictionary where each key is a unique genre, and each value is the total number of movies in that genre.

    C. A dictionary where each key is a tuple of genres, and each value is a list of movies containing only that genre.

    D. A dictionary where each key is a list of genres, and each value is a list of movies in that category.

28. `year_buckets` is the result of `bucketize(movies, "year")`. Which of the following correctly sorts `year_buckets` as a list of tuples by the **number of movies in each year**, in **decreasing order**?

    A. `sorted(year_buckets.items(), key=lambda x:len(x[1]), reverse=True)`

    B. `sorted(year_buckets.keys(), key=lambda year:len(year_buckets[year]), reverse=True)`

    C. `sorted(year_buckets, key=lambda x:len(year_buckets[x]), reverse=True)`

    D. `year_buckets.sort(key=lambda x:len(year_buckets[x]), reverse=True)`

29. `director_buckets` data structure is the result of `bucketize(movies, "directors")`. Which of the following retrieves the **average IMDb rating of movies directed by Christopher Nolan**?

    A. `sum([movie["rating"] for movie in movies if "Christopher Nolan" in movie["directors"]]) / len(movies)`

    B. `sum([movie["rating"] for movie in director_buckets["Christopher Nolan"]]) / len(director_buckets["Christopher Nolan"])`

    C. `len([movie["rating"] for movie in director_buckets["Christopher Nolan"]]) / sum([movie["rating"] for movie in director_buckets["Christopher Nolan"]])`

    D. `sum([movie["rating"] for movie in director_buckets["Christopher Nolan"]]) / len(movies)`

Stars and Planets

30. Consider the following code, where `Star` is a named tuple with the attributes:

    - name

    - spectral_type

    - stellar_mass

    - stellar_radius

    We have a list of star objects as shown below:

    ```python
    from collections import namedtuple

    Star = namedtuple("Star", ["name", "spectral_type",
                               "stellar_mass", "stellar_radius"])

    stars = [
        Star("Sirius A", "A1V", 2.02, 1.711),
        Star("Proxima Centauri", "M5.5Ve", 0.122, 0.1542),
        Star("Betelgeuse", "M1-2Ia-Iab", 20.0, 887),
        Star("Rigel", "B8Ia", 21.0, 78.9)
    ]
    ```

    We want to calculate the average stellar radius of all stars that have a stellar mass greater than 1. Which of the following list comprehensions correctly filters and calculates the average stellar radius?

    A. `[star.stellar_radius for star in stars if star.stellar_mass > 1]`

    B. `[star.stellar_radius for star in stars if star.stellar_mass > 1] / len(stars)`

    C. `sum([star.stellar_radius for star in stars if star.stellar_mass > 1]) / len([star for star in stars if star.stellar_mass > 1])`

    D. `sum([s.stellar_radius for s in stars if s.stellar_mass > 1]) / len([s.stellar_radius for s in stars if s.stellar_mass > 1])`

    E. Both C and D

31. The following function aims to collect all `.json` files from a given directory and its subdirectories. Which of the following correctly fills in the missing parts `????` to achieve this functionality?

```python
import os

def get_json_files(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) and path.endswith(".json"):
            paths.append(path)
        elif os.path.isdir(path):
            ????
    return sorted(paths, reverse=True)
```

What should replace the `????` to complete the function?

    A. `paths.append(get_json_files(path))`

    B. `paths.extend(get_json_files(path))`

    C. `paths += get_json_files(path)`

    D. `paths.extend([get_json_files(path)])`

    E. `Both B and C`

Rankings

32. In P12 and P13, we analysed World University Rankings data. For the following four questions, assume that you have a pandas DataFrame named `rankings` with the following data:

Year	World Rank	Institution	Country	National Rank	Education Rank	Faculty Rank	Research Rank
2023	28	University of Wisconsin–Madison	USA	20	36	30	41
2023	29	Imperial College London	United Kingdom	4	68	55	18
2022	38	Sorbonne University	France	3	89	92	18
2021	2	Massachusetts Institute of Technology	USA	2	4	2	8
2021	3	Stanford University	USA	3	10	3	2
2021	13	University of Tokyo	Japan	1	39	107	27

Which of the following options filters `rankings` to only display the rows where `Country` is `USA` or `United Kingdom`?

   A. `rankings[rankings["Country"].isin("USA", "United Kingdom")]`

   B. `rankings[rankings["Country"] == "USA" && rankings["Country"] == "United Kingdom"]`

   C. `rankings[rankings["Country"].match("USA", "United Kingdom")]`

   D. `rankings[(rankings["Country"] == "United Kingdom") or (rankings["Country"] == "USA")]`

   E. `rankings[(rankings["Country"] == "USA") | (rankings["Country"] == "United Kingdom")]`

33. Which expression answers this question: What is the name of the university with the worst (i.e. Sorbonne University) `World Rank` in `rankings`?

   A. `rankings["World Rank"].min()`

   B. `rankings[rankings["World Rank"] == rankings["World Rank"].min()]["Institution"].iloc[0]`

   C. `rankings["World Rank"].max()`

   D. `rankings[rankings["World Rank"] == rankings["World Rank"].max()]["Institution"].iloc[0]`

   E. None of the above

34. Let's assume the following code is run for the next question:

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

Which of the following options finds the average faculty rank for all universities with a national rank less than or equal to 2 for the year 2021?

A. `pd.read_sql("SELECT AVG('Faculty Rank') FROM rankings WHERE 'National Rank' <= 2 AND 'Year' = 2021", conn)`

B. `pd.read_sql("SELECT AVG('Faculty Rank') FROM rankings WHERE 'National Rank' = 2 AND 'Year' = 2021", conn)`

C. `pd.read_sql("SELECT AVG('Faculty Rank') FROM rankings WHERE 'National Rank' < 2 AND 'Year' = 2021", conn)`

D. `pd.read_sql("SELECT 'Faculty Rank' FROM rankings WHERE 'National Rank' <= 2 AND 'Year' = 2021", conn)`

E. `pd.read_sql("SELECT AVG('Faculty Rank') FROM rankings WHERE 'Year' = 2021", conn)`

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

```python
pd.read_sql("""
    SELECT `Institution`, AVG(`Research Rank`) AS avg_research_rank
    FROM rankings
    WHERE ABS(`National Rank` - `World Rank`) <= 10
    GROUP BY `Institution`
    HAVING AVG(`Research Rank`) > 50
    ORDER BY avg_research_rank DESC
""", conn)
```

- A. The query returns a list of institutions where the absolute difference between the National Rank and World Rank is less than or equal to 10, and the average Research Rank for each institution is greater than 7. The results are sorted in descending order of the average Research Rank.
- B. The query returns a list of institutions with an average research rank of more than 7, ordered by the highest national rank.
- C. The query returns a list of institutions with an average research rank of more than 50, but only if the difference between National Rank and World Rank is greater than 7, ordered by the lowest research rank.
- D. The query returns all institutions, with the average Research Rank displayed for each institution, where the difference between National Rank and World Rank is less than or equal to 10. The results are sorted in ascending order of the average Research Rank.
- E. The query returns a list of institutions where the absolute difference between the National Rank and World Rank is greater than 7, with the average Research Rank being less than 50. The results are sorted by the Institution name.

(Blank Page)