

SRIRAM V – CSE FINAL YEAR

QUESTION

1.createCollection(p_collection_name)

Using Any of the programming language implement below functions

1. **indexData(p_collection_name, p_exclude_column):**
Index the given employee data into the specified collection, excluding the column provided in p_exclude_column.
2. **searchByColumn(p_collection_name, p_column_name, p_column_value):**
Search within the specified collection for records where the column p_column_name matches the value p_column_value.
3. **getEmpCount(p_collection_name)**
4. **delEmpById(p_collection_name, p_employee_id)**
5. **• getDepFacet(p_collection_name):**
Retrieve the count of employees grouped by department from the specified collection.






Step :1

Selfie Pic:





Step :2


First Task Email






First Challenge for Hash Agile Internship/ Freshers Drive Inbox





Recruitment Yesterday
to Recruitment 



Dear Candidate,

We are pleased to inform you that your first programming challenge is attached to this email. Please carefully read the problem statement and submit your solution.

Instructions:

- **Deadline:** You must solve the attached program in Ruby Language and submit your code via the provided Google Form link by **12:00 PM** today.
- **Submission Form:** <https://forms.gle/4xuGnsgdvHyjzozGA>
- **Important:** Submissions received after the deadline will be given lower priority in the selection process.

Guidelines:

- Ensure that your solution is **original** and does not use built-in functions (as specified in the problem statement). We will be using tools like [ZeroGPT](#) to check for plagiarism and any content generated by AI.
- **Please attach a PDF document** that includes:
 - The **problem statement** given
 - The **program code** you have written
 - At least **three sample inputs** along with their respective outputs

We wish you all the best!





Best regards,
HR Team,
Hash Agile Technologies

Disclaimer: This email, sent by HashAgile Technologies Pvt Ltd(HashAgile), contains confidential information intended solely for the designated recipients. Any sharing of the contents of this message with third parties, including attachments, is strictly prohibited without the written consent of the sender. The information contained in this email is proprietary to the company. In the event of receiving this email in error, please notify us by replying to this message and delete it permanently and HashAgile is not responsible for the contents of this message.

HATFD1030

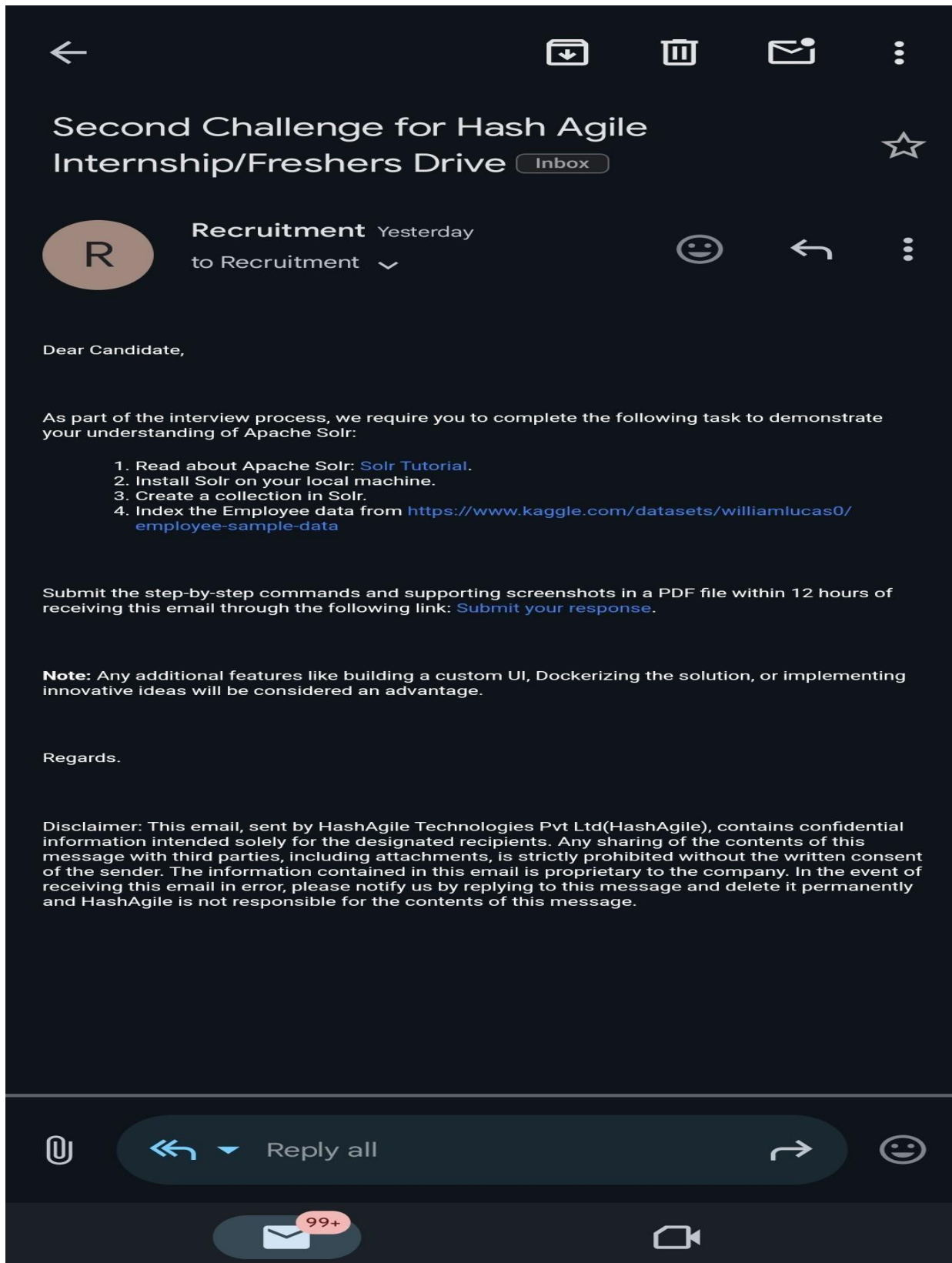
Rotate an Array Right by K Positions
Write a program to rotate an array right by k positions without using any built-in array or rotation functions. For example, rotating `[1, 2, 3, 4, 5]` by 2 would give `[4, 5, 1, 2, 3]`.

Instructions: You should implement the logic manually for rotating the array.

Reply all

Step : 3

Second Task Email



Step :4

GitHub URL for Round 1

<https://github.com/Ebi-75/Hash-Agile-company-interview/blob/7c98ba66ac4c9bcce81c8df12a7ad358565b0da3/rotate%20an%20array%20in%20python%20Round%201.pdf>

Step :5

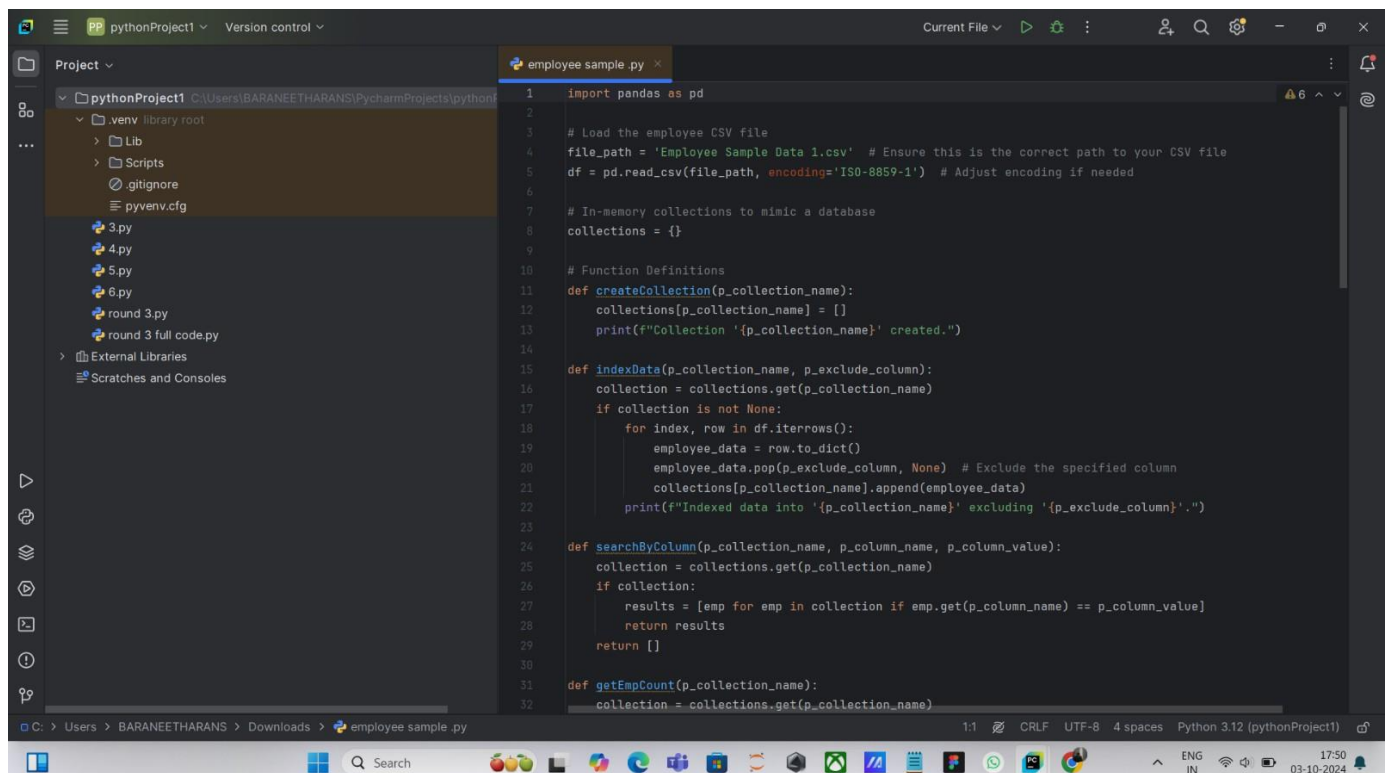
GitHub URL for Assignment :

<https://github.com/Ebi-75/Hash-Agile-company-interview/blob/00424c0cc3fcddbabb41eeaf90d593191f5e7d678/employee%20sample%20.py>

Step :6

Function Execution Results

INPUT



The screenshot shows a PyCharm IDE with a project named 'pythonProject1'. The file explorer on the left shows a directory structure with files like '3.py', '4.py', '5.py', '6.py', 'round 3.py', and 'round 3 full code.py'. The main editor displays a file named 'employee sample.py' with the following Python code:

```
1 import pandas as pd
2
3 # Load the employee CSV file
4 file_path = 'Employee Sample Data 1.csv' # Ensure this is the correct path to your CSV file
5 df = pd.read_csv(file_path, encoding='ISO-8859-1') # Adjust encoding if needed
6
7 # In-memory collections to mimic a database
8 collections = {}
9
10 # Function Definitions
11 def createCollection(p_collection_name):
12     collections[p_collection_name] = []
13     print(f"Collection '{p_collection_name}' created.")
14
15 def indexData(p_collection_name, p_exclude_column):
16     collection = collections.get(p_collection_name)
17     if collection is not None:
18         for index, row in df.iterrows():
19             employee_data = row.to_dict()
20             employee_data.pop(p_exclude_column, None) # Exclude the specified column
21             collections[p_collection_name].append(employee_data)
22         print(f"Indexed data into '{p_collection_name}' excluding '{p_exclude_column}'.")
23
24 def searchByColumn(p_collection_name, p_column_name, p_column_value):
25     collection = collections.get(p_collection_name)
26     if collection:
27         results = [emp for emp in collection if emp.get(p_column_name) == p_column_value]
28         return results
29     return []
30
31 def getEmpCount(p_collection_name):
32     collection = collections.get(p_collection_name)
```

The status bar at the bottom indicates the file encoding is UTF-8, 4 spaces, and the Python version is 3.12 (pythonProject1). The system tray shows the date as 03-10-2024 and time as 17:50.

Output:

```
1 import pandas as pd
2
3 # Load the employee CSV file
4 file_path = 'Employee Sample Data 1.csv' # Ensure this is the correct path to your CSV file
5 df = pd.read_csv(file_path, encoding='ISO-8859-1') # Adjust encoding if needed
6
7 # In-memory collections to mimic a database
8 collections = {}
9
10 # Function Definitions
11 def createCollection(p_collection_name):
12     collections[p_collection_name] = []
13     print(f"Collection '{p_collection_name}' created.")
14
15 def indexData(p_collection_name, p_exclude_column):
16     collection = collections.get(p_collection_name)
17     if collection is not None:
18         for index, row in df.iterrows():
19             employee_data = row.to_dict()
```

Run employee sample

C:\Users\BARANEETHARANS\PycharmProjects\pythonProject1\.venv\Scripts\python.exe "C:\Users\BARANEETHARANS\Downloads\employee sample .py"

Collection 'Hash_YourName' created.
Collection 'Hash_1234' created.
Indexed data into 'Hash_YourName' excluding 'Department'.
Indexed data into 'Hash_1234' excluding 'Gender'.
Employee with ID 'E02003' deleted from 'Hash_YourName'.
Initial Employee Count: 0
Final Employee Count: 1259
IT Employees (Name Collection): []

```
1 import pandas as pd
2
3 # Load the employee CSV file
4 file_path = 'Employee Sample Data 1.csv' # Ensure this is the correct path to your CSV file
5 df = pd.read_csv(file_path, encoding='ISO-8859-1') # Adjust encoding if needed
6
7 # In-memory collections to mimic a database
8 collections = {}
9
10 # Function Definitions
11 def createCollection(p_collection_name):
12     collections[p_collection_name] = []
13     print(f"Collection '{p_collection_name}' created.")
14
15 def indexData(p_collection_name, p_exclude_column):
16     collection = collections.get(p_collection_name)
17     if collection is not None:
18         for index, row in df.iterrows():
19             employee_data = row.to_dict()
```

Run employee sample

C:\Users\BARANEETHARANS\PycharmProjects\pythonProject1\.venv\Scripts\python.exe "C:\Users\BARANEETHARANS\Downloads\employee sample .py"

Collection 'Hash_YourName' created.
Collection 'Hash_1234' created.
Indexed data into 'Hash_YourName' excluding 'Department'.
Indexed data into 'Hash_1234' excluding 'Gender'.
Employee with ID 'E02003' deleted from 'Hash_YourName'.
Initial Employee Count: 0
Final Employee Count: 1259
IT Employees (Name Collection): []

Input:

Program:

```
import pandas as pd

# Load the employee CSV file
file_path = 'Employee Sample Data 1.csv' # Ensure this is the correct path to your CSV file
df = pd.read_csv(file_path, encoding='ISO-8859-1') # Adjust encoding if needed

# In-memory collections to mimic a database
collections = {}

# Function Definitions
def createCollection(p_collection_name):
    collections[p_collection_name] = []
    print(f"Collection '{p_collection_name}' created.")

def indexData(p_collection_name, p_exclude_column):
    collection = collections.get(p_collection_name)
    if collection is not None:
        for index, row in df.iterrows():
            employee_data = row.to_dict()
            employee_data.pop(p_exclude_column, None) # Exclude the specified column
            collections[p_collection_name].append(employee_data)
        print(f"Indexed data into '{p_collection_name}' excluding '{p_exclude_column}'.")

def searchByColumn(p_collection_name, p_column_name, p_column_value):
    collection = collections.get(p_collection_name)
    if collection:
        results = [emp for emp in collection if emp.get(p_column_name) == p_column_value]
        return results
    return []

def getEmpCount(p_collection_name):
    collection = collections.get(p_collection_name)
    if collection is not None:
        return len(collection)
    return 0

def delEmpById(p_collection_name, p_employee_id):
    collection = collections.get(p_collection_name)
    if collection:
        collections[p_collection_name] = [emp for emp in collection if emp.get('Employee ID') != p_employee_id]
        print(f"Employee with ID '{p_employee_id}' deleted from '{p_collection_name}'.")

def getDepFacet(p_collection_name):
    collection = collections.get(p_collection_name)
    if collection:
        dep_count = {}
        for emp in collection:
            department = emp.get('Department')
            if department:
                dep_count[department] = dep_count.get(department, 0) + 1
        return dep_count
    return {}

# Execute the required functions with the dataset

# Replace with your actual name and phone last four digits
v_nameCollection = 'Hash_YourName'
v_phoneCollection = 'Hash_1234'

# 1. Create collections
createCollection(v_nameCollection)
createCollection(v_phoneCollection)

# 2. Get employee count before indexing
initial_count_name = getEmpCount(v_nameCollection)

# 3. Index data into both collections
```



```
indexData(v_nameCollection, 'Department')
indexData(v_phoneCollection, 'Gender')
```

```
# 4. Delete an employee by ID
delEmpById(v_nameCollection, 'E02003')
```

```
# 5. Get employee count after deletion
final_count_name = getEmpCount(v_nameCollection)
```

```
# 6. Search by columns
it_employees_name = searchByColumn(v_nameCollection, 'Department', 'IT')
male_employees_name = searchByColumn(v_nameCollection, 'Gender', 'Male')
it_employees_phone = searchByColumn(v_phoneCollection, 'Department', 'IT')
```

```
# 7. Get department facet
dep_facet_name = getDepFacet(v_nameCollection)
dep_facet_phone = getDepFacet(v_phoneCollection)
```

```
# Collecting results for output
output_results = {
    "Initial Employee Count": initial_count_name,
    "Final Employee Count": final_count_name,
    "IT Employees (Name Collection)": it_employees_name,
    "Male Employees (Name Collection)": male_employees_name,
    "IT Employees (Phone Collection)": it_employees_phone,
    "Department Facet (Name Collection)": dep_facet_name,
    "Department Facet (Phone Collection)": dep_facet_phone
}
```

```
# Print results for documentation
for key, value in output_results.items():
    print(f"{key}: {value}")
```