COMP 1023 Introduction to Python Programming

Exercises on Pandas

COMP 1023 Teaching Team

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology, Hong Kong SAR, China

Exercise 1

Create a Pandas Series from the following list of temperatures in Celsius: `[20, 25, 30, 35, 40]`. Assign custom index labels: `['Mon', 'Tue', 'Wed', 'Thu', 'Fri']`. Print the Series.

Requirements:

- Use the `pd.Series()` function.
- Assign custom index labels.

Expected output:

```
Mon    20
Tue    25
Wed    30
Thu    35
Fri    40
dtype: int64
```

Exercise 1 Solutions

```python
import pandas as pd

temperatures = [20, 25, 30, 35, 40]
index_labels = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
temperature_series = pd.Series(temperatures, index=index_labels)
print(temperature_series)
```

Exercise 2

Given the Series created in Problem 1, access the temperature for `'Wed'` using label-based indexing and for the 3rd position using position-based indexing. Print both values.

Requirements:

- Use `.loc[]` for label-based indexing.
- Use `.iloc[]` for position-based indexing.

Expected output:

```
Temperature on Wed: 30
Temperature at 3rd position: 30
```

Exercise 2 Solutions

```python
# Assuming temperature_series is defined from Problem 1
temp_wed = temperature_series.loc['Wed']
temp_third = temperature_series.iloc[2]
print(f"Temperature on Wed: {temp_wed}")
print(f"Temperature at 3rd position: {temp_third}")
```

Exercise 3

Create a DataFrame from the following data about students:

- Names: `['Alice', 'Bob', 'Charlie']`
- Ages: `[20, 21, 19]`
- `Grades: [85, 90, 88]`

Then, print the DataFrame.

Requirements:

- Use a dictionary to create the DataFrame.
- Set the index to the names of the students.

Expected output:

```
         Age  Grade
Alice     20     85
Bob       21     90
Charlie   19     88
```

Exercise 3 Solutions

```python
import pandas as pd

data = {
    'Age': [20, 21, 19],
    'Grade': [85, 90, 88]
}
students_df = pd.DataFrame(data, index=['Alice', 'Bob', 'Charlie'])
print(students_df)
```

Exercise 4

Using the DataFrame created in Problem 3, add a new column `'Passed'` that is True if the grade is 88 or higher and False otherwise. Then, group the DataFrame by the `'Passed'` column and calculate the average age for each group.

Requirements:

- Use a boolean condition to create the `'Passed'` column.
- Use `groupby()` and `mean()` to find the average age.

Useful functions:

- `groupby()`: `https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.groupby.html#pandas.DataFrame.groupby`
- `mean()`: `https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.mean.html#pandas.DataFrame.mean`

Expected output:

```
Passed
False    20.0
True     20.0
Name: Age, dtype: float64
```

Exercise 4 Solutions

```python
# Assuming students_df is defined from Problem 3
students_df['Passed'] = students_df['Grade'] >= 88
average_age = students_df.groupby('Passed')['Age'].mean()
print(average_age)
```

Exercise 5

Create a DataFrame with the following data about products:

- Product Names: `['Product A', 'Product B', 'Product C']`
- Prices: `[100, None, 150]`
- Stocks: `[20, 30, None]`

Then, fill the missing values in the `'Prices'` column with the mean price and in the `'Stocks'` column with 0. Print the updated DataFrame.

Requirements:

- Use `fillna()` to handle missing values.
- Use `mean()` to calculate the mean price.

Useful functions:

- `fillna()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.fillna.html#pandas.DataFrame.fillna
- `mean()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.mean.html#pandas.DataFrame.mean

Expected output:

```
     Product  Price  Stock
0  Product A  100.0   20.0
1  Product B  125.0   30.0
2  Product C  150.0    0.0
```

Exercise 5 Solutions

```python
import pandas as pd

data = {
    'Product': ['Product A', 'Product B', 'Product C'],
    'Price': [100, None, 150],
    'Stock': [20, 30, None]
}
products_df = pd.DataFrame(data)

# Fill missing values
products_df['Price'] = products_df['Price'].fillna(products_df['Price'].mean())
products_df['Stock'] = products_df['Stock'].fillna(0)

print(products_df)
```

Exercise 6

You are given a dataset containing sales information for a small retail store. The dataset includes the columns:

- `Product`: The name of the product sold.
- `Category`: The category of the product (e.g., 'Electronics', 'Clothing', 'Groceries').
- `Price`: The price of the product.
- `Quantity Sold`: The number of units sold.
- `Date`: The date of the sale.

You need to perform the following tasks:

1. Create a DataFrame from the following data:
   - Products: `['Laptop', 'Shirt', 'Apple', 'Headphones', 'Pants']`
   - Categories: `['Electronics', 'Clothing', 'Groceries', 'Electronics', 'Clothing']`
   - Prices: `[1000, 20, 1, 150, 30]`
   - Quantities Sold: `[5, 10, 50, 7, 8]`
   - Dates: `['2023-01-01', '2023-01-02', '2023-01-01', '2023-01-03', '2023-01-02']`
2. Calculate the total sales for each product by multiplying `Price` and `Quantity Sold`. Create a new column called `Total Sales`.
3. Group the DataFrame by `Category` and calculate the total sales for each category.
4. Identify the product with the highest total sales and print its details.
5. Handle any missing values in the `Quantity Sold` column by filling them with the mean quantity sold.

Requirements:

- Use Pandas for all operations.
- Ensure the output is clear and formatted properly.

You are given the following skeleton code to start with.

```
import pandas as pd

# Step 1: Create a DataFrame
# Step 2: Calculate total sales and create a new column
# Step 3: Group by Category and calculate total sales for each category
# Step 4: Identify the product with the highest total sales
# (For this example, let's assume we introduce a missing value for demonstration)
sales_df.loc[2, 'Quantity Sold'] = None  # Introduce a missing value for testing
# Step 5: Handle missing values in Quantity Sold

# Print results
print("Sales DataFrame:")
print(sales_df)
print("\nTotal Sales by Category:")
print(category_sales)
print("\nProduct with Highest Total Sales:")
print(highest_sales_product)
```

The following functions will be useful for this exercise:

- `groupby()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.groupby.html#pandas.DataFrame.groupby
- `sum()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.sum.html#pandas.DataFrame.sum
- `reset_index()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.reset_index.html#pandas.DataFrame.reset_index
- `idxmax()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.idxmax.html#pandas.DataFrame.idxmax
- `fillna()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.fillna.html#pandas.DataFrame.fillna
- `mean()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.mean.html#pandas.DataFrame.mean

Expected output:

```
Sales DataFrame:
      Product     Category  Price  Quantity Sold        Date  Total Sales
0      Laptop  Electronics   1000            5.0  2023-01-01         5000
1       Shirt     Clothing     20           10.0  2023-01-02          200
2       Apple    Groceries      1            7.5  2023-01-01           50
3  Headphones  Electronics    150            7.0  2023-01-03         1050
4       Pants     Clothing     30            8.0  2023-01-02          240

Total Sales by Category:
      Category  Total Sales
0     Clothing          440
1  Electronics         6050
2    Groceries           50

Product with Highest Total Sales:
Product               Laptop
Category          Electronics
Price                    1000
Quantity Sold               5
Date               2023-01-01
Total Sales              5000
Name: 0, dtype: object
```

Exercise 6 Solutions

```python
import pandas as pd

# Step 1: Create a DataFrame
data = {
    'Product': ['Laptop', 'Shirt', 'Apple', 'Headphones', 'Pants'],
    'Category': ['Electronics', 'Clothing', 'Groceries', 'Electronics', 'Clothing'],
    'Price': [1000, 20, 1, 150, 30],
    'Quantity Sold': [5, 10, 50, 7, 8],
    'Date': ['2023-01-01', '2023-01-02', '2023-01-01', '2023-01-03', '2023-01-02']
}

sales_df = pd.DataFrame(data)

# Step 2: Calculate total sales and create a new column
sales_df['Total Sales'] = sales_df['Price'] * sales_df['Quantity Sold']

# Step 3: Group by Category and calculate total sales for each category
category_sales = sales_df.groupby('Category')['Total Sales'].sum().reset_index()
```

```python
# Step 4: Identify the product with the highest total sales
highest_sales_product = sales_df.loc[sales_df['Total Sales'].idxmax()]

# Step 5: Handle missing values in Quantity Sold
# (For this example, let's assume we introduce a missing value for demonstration)
sales_df.loc[2, 'Quantity Sold'] = None  # Introduce a missing value for testing
sales_df['Quantity Sold'] = sales_df['Quantity Sold'].fillna(sales_df['Quantity Sold'].mean())

# Print results
print("Sales DataFrame:")
print(sales_df)
print("\nTotal Sales by Category:")
print(category_sales)
print("\nProduct with Highest Total Sales:")
print(highest_sales_product)
```

Exercise 7

You are given a dataset containing information about employees in a company. The dataset includes the following columns:

- `Name`: The name of the employee.
- `Department`: The department where the employee works (e.g., 'HR', 'IT', 'Finance').
- `Salary`: The salary of the employee.
- `Experience`: The years of experience the employee has.
- `Joining Date`: The date the employee joined the company.

You need to perform the following tasks:

1. Create a DataFrame from the following data:
   - Names: `['Alice', 'Bob', 'Charlie', 'Diana', 'Eve']`
   - Department: `['HR', 'IT', 'Finance', 'IT', 'HR']`
   - Salary: `[60000, 75000, 80000, 90000, 50000]`
   - Experience: `[2, 5, 7, 4, 3]`
   - Joining Date: `['2020-01-15', '2019-03-22', '2018-07-01', '2021-05-30', '2020-11-10']`
2. Calculate the average salary for each department and create a new DataFrame to store these results.
3. Identify the employee with the highest salary and print their details.
4. Handle any missing values in the `Experience` column by filling them with the median experience.
5. Sort the original DataFrame by `Experience` in descending order.

Requirements:

- Use Pandas for all operations.
- Ensure the output is clear and formatted properly.

You are given the following skeleton code to start with.

```
import pandas as pd

# Step 1: Create a DataFrame
# Step 2: Calculate average salary for each department
# Step 3: Identify the employee with the highest salary
# (For this example, let's assume we introduce a missing value for demonstration)
employees_df.loc[2, 'Experience'] = None  # Introduce a missing value for testing
# Step 4: Handle missing values in Experience
# Step 5: Sort the DataFrame by Experience in descending order

# Print results
print("Employees DataFrame:")
print(employees_df)
print("\nAverage Salary by Department:")
print(average_salary)
print("\nEmployee with Highest Salary:")
print(highest_salary_employee)
print("\nSorted Employees by Experience:")
print(sorted_employees)
```

The following functions will be useful for this exercise:

- `groupby()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.groupby.html#pandas.DataFrame.groupby
- `mean()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.mean.html#pandas.DataFrame.mean
- `idxmax()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.idxmax.html#pandas.DataFrame.idxmax
- `fillna()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.fillna.html#pandas.DataFrame.fillna
- `median()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.median.html#pandas.DataFrame.median
- `sort_values()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.sort_values.html#pandas.DataFrame.sort_values

Expected output:

```
Employees DataFrame:
      Name Department  Salary  Experience Joining Date
0    Alice         HR   60000         2.0   2020-01-15
1      Bob         IT   75000         5.0   2019-03-22
2  Charlie    Finance   80000         3.5   2018-07-01
3    Diana         IT   90000         4.0   2021-05-30
4      Eve         HR   50000         3.0   2020-11-10

Average Salary by Department:
  Department   Salary
0    Finance  80000.0
1         HR  55000.0
2         IT  82500.0

Employee with Highest Salary:
Name                          Diana
Department                       IT
Salary                        90000
Experience                        4
Joining Date    2021-05-30 00:00:00
Name: 3, dtype: object

Sorted Employees by Experience:
      Name Department  Salary  Experience Joining Date
1      Bob         IT   75000         5.0   2019-03-22
3    Diana         IT   90000         4.0   2021-05-30
2  Charlie    Finance   80000         3.5   2018-07-01
4      Eve         HR   50000         3.0   2020-11-10
0    Alice         HR   60000         2.0   2020-01-15
```

Exercise 7 Solutions

```python
import pandas as pd

# Step 1: Create a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Diana', 'Eve'],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'Salary': [60000, 75000, 80000, 90000, 50000],
    'Experience': [2, 5, 7, 4, 3],
    'Joining Date': ['2020-01-15', '2019-03-22', '2018-07-01', '2021-05-30', '2020-11-10']
}

employees_df = pd.DataFrame(data)

# Step 2: Calculate average salary for each department
average_salary = employees_df.groupby('Department')['Salary'].mean().reset_index()

# Step 3: Identify the employee with the highest salary
highest_salary_employee = employees_df.loc[employees_df['Salary'].idxmax()]
```

```python
# Step 4: Handle missing values in Experience
# (For this example, let's assume we introduce a missing value for demonstration)
employees_df.loc[2, 'Experience'] = None  # Introduce a missing value for testing
employees_df['Experience'] = employees_df['Experience'].fillna(employees_df['Experience'].median())

# Step 5: Sort the DataFrame by Experience in descending order
sorted_employees = employees_df.sort_values(by='Experience', ascending=False)

# Print results
print("Employees DataFrame:")
print(employees_df)
print("\nAverage Salary by Department:")
print(average_salary)
print("\nEmployee with Highest Salary:")
print(highest_salary_employee)
print("\nSorted Employees by Experience:")
print(sorted_employees)
```

Exercise 8

You are given a CSV file named `employees.csv` with the following structure:

```
Name,Department,Salary,Experience,Joining Date
Alice,HR,60000,2,2020-01-15
Bob,IT,75000,5,2019-03-22
Charlie,Finance,80000,7,2018-07-01
Diana,IT,90000,4,2021-05-30
Eve,HR,50000,3,2020-11-10
```

You need to perform the following tasks:

1. Load the data from the `employees.csv` file into a Pandas DataFrame.
2. Add a new column named `Passed` that is True if the salary is greater than or equal to 70000 and False otherwise.
3. Group the DataFrame by the Passed column and calculate the average salary for each group.
4. Write the modified DataFrame, including the new `Passed` column, back to a new CSV file named `employees_updated.csv`.

Requirements:

- Use `pd.read_csv()` to load the data.
- Use boolean conditions to create the `Passed` column.
- Use `groupby()` and `mean()` to find the average salary.
- Print the average salary by `Passed` group.
- Use `to_csv()` to save the updated DataFrame.

Expected output:

```
Passed
False    55000.000000
True     81666.666667
Name: Salary, dtype: float64
```

Expected output CSV file `employees_updated.csv`

```
Name,Department,Salary,Experience,Joining Date,Passed
Alice,HR,60000,2,2020-01-15,False
Bob,IT,75000,5,2019-03-22,True
Charlie,Finance,80000,7,2018-07-01,True
Diana,IT,90000,4,2021-05-30,True
Eve,HR,50000,3,2020-11-10,False
```

The following functions will be useful for this exercise:

- `read_csv()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.read_csv.html#pandas.read_csv
- `groupby()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.groupby.html#pandas.DataFrame.groupby
- `mean()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.mean.html#pandas.DataFrame.mean
- `to_csv()`: https://pandas.pydata.org/pandas-docs/version/2.3.1/reference/api/pandas.DataFrame.to_csv.html#pandas.DataFrame.to_csv

Exercise 8 Solutions

```python
import pandas as pd

# Step 1: Load the data from the CSV file
employees_df = pd.read_csv('employees.csv')

# Step 2: Add a new column 'Passed'
employees_df['Passed'] = employees_df['Salary'] >= 70000

# Step 3: Group by 'Passed' and calculate the average salary for each group
average_salary = employees_df.groupby('Passed')['Salary'].mean()

# Step 4: Print the average salary by Passed group
print(average_salary)

# Step 5: Write the modified DataFrame to a new CSV file
employees_df.to_csv('employees_updated.csv', index=False)
```