3.1.1. Numpy array operations

01:46

Write a python program to demonstrate the usage of ndim, shape and size for a Numpy Array. The program should create a NumPy array using the entered elements and display it. Assume all input elements are valid numeric values.

Input Format:

- User inputs the number of rows and columns with space separated values.
- User inputs elements of the array row-wise followed line by line, separated by spaces.

Output Format:

- The created NumPy array based on the input dimensions and elements.
- Dimensions (ndim): Number of dimensions of the array.
- Shape: Tuple representing the shape of the array (number of rows, number of columns).
- Size: Total number of elements in the array.

Note: Use reshape() function to reshape the input array with the specified number of rows and columns.

Sample Test Cases +

Explorer

numpyarr.py

Submit

Debugger

```
1  import numpy as np
2
3  rows, cols = map(int,
   input().split())
4
5  elements = []
6  for _ in range(rows):
7      elements.extend(map(int,
   input().split()))
8
9  array =
   np.array(elements).reshape(rows,
   cols)
10
11 print(array)
12 print(array.ndim)
13 print(array.shape)
14 print(array.size)
15
```

Average time
0.030 s
29.60 ms

Maximum time
0.044 s
44.00 ms

3 out of 3 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 43 ms Debug

Expected output	Actual output
3 4	3 4
1 2 3 4	1 2 3 4
5 6 7 8	5 6 7 8

3.2.1. Numpy: Matrix Operations

06:12

The given code takes two 3×3 matrices, matrix_a, and matrix_b, as input from the user and converts them into NumPy arrays.

Task:

You are required to compute and display the results of the following matrix operations:

1. **Addition** ($\text{matrix\_a} + \text{matrix\_b}$)
2. **Subtraction** ($\text{matrix\_a} - \text{matrix\_b}$)
3. **Element-wise Multiplication** ($\text{matrix\_a} * \text{matrix\_b}$)
4. **Matrix Multiplication** ($\text{matrix\_a} \cdot \text{matrix\_b}$)
5. **Transpose of Matrix A**

Input Format:

- The user will input 3 rows for matrix_a, each containing 3 integers separated by spaces.
- Similarly, the user will input 3 rows for matrix_b, each containing 3 integers separated by spaces.

Output Format:

The program should display the results of the operations in the following order:

1. The result of Addition.
2. The result of Subtraction.
3. The result of Element-wise Multiplication.
4. The result of Matrix Multiplication.
5. The Transpose of Matrix A.

Sample Test Cases +

Explorer

matrixOpe...

Submit

Debugger

```python
import numpy as np

# Input matrices
print("Enter Matrix A:")
matrix_a = np.array([list(map(int, input().split())) for i in range(3)])

print("Enter Matrix B:")
matrix_b = np.array([list(map(int, input().split())) for i in range(3)])

addition = matrix_a + matrix_b
subtraction = matrix_a-matrix_b
elementwise_multiplication = matrix_a*matrix_b
matrix_multilication = np.dot(matrix_a,matrix_b)
transpose_a=matrix_a.T
# Addition
print("Addition (A + B):")
print(addition)
# Subtraction
print("Subtraction (A - B):")
print(subtraction)
# Multiplication (element-wise)
print("Element-wise Multiplication (A * B):")
print(elementwise_multiplication)
# Matrix multiplication (dot product)
print("A dot B:")
print(matrix_multilication)
# Transpose
print("Transpose of A:")
print(transpose_a)
```

Average time **0.060 s** 60.25 ms

Maximum time **0.072 s** 72.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 72 ms Debug

Expected output	Actual output
Enter Matrix A:	Enter Matrix A:
1 2 3	1 2 3
4 5 6	4 5 6

3.2.2. Numpy: Horizontal and Vertical Stacki... 11:33

You are given two arrays `arr1` and `arr2`. You need to perform horizontal and vertical stacking operations on them using NumPy.

- **Horizontal Stacking**: Stack the two matrices horizontally (side by side).
- **Vertical Stacking**: Stack the two matrices vertically (one below the other).

Input Format:

- The program should first prompt the user to input two 3x3 arrays.
- Each array consists of 3 rows, and each row contains 3 space-separated integers.
- The user will input the two arrays row by row.

Output Format:

- The program should display the result of the Horizontal Stack (side-by-side stacking) of the two arrays.
- The program should then display the result of the Vertical Stack (one below the other) of the two arrays.

Sample Test Cases +

Explorer

stacking.py

Submit

Debugger

```python
import numpy as np

# Input matrices
print("Enter Array1:")
arr1 = np.array([list(map(int, input().split())) for i in range(3)])

print("Enter Array2:")
arr2 = np.array([list(map(int, input().split())) for i in range(3)])

# Perform horizontal stacking (hstack)
a=np.hstack((arr1,arr2))
print("Horizontal Stack:")

print(a)
print("Vertical Stack:")
b=np.vstack((arr1,arr2))
print(b)


# Perform vertical stacking (vstack)

```

Average time	Maximum time
0.061 s 60.75 ms	0.069 s 69.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 69 ms Debug

Expected output	Actual output
Enter Array1:	Enter Array1:
1 2 3	1 2 3
4 5 6	4 5 6

CODETANTRA

3.2.3. Numpy: Custom Sequence Generation

00:51

Write a Python program that takes the following inputs from the user:

- Start value: The starting point of the sequence.
- Stop value: The sequence should end before this value.
- Step value: The increment between each number in the sequence.

The program should then generate a sequence using numpy based on these inputs and print the generated sequence.

Input Format:

- The user will input three integer values: start, stop, and step, each on a new line.

Output Format:

- The program should print the generated sequence based on the input values.

Sample Test Cases +

Explorer

customSe...

Submit

Debugger

```
1  import numpy as np
2
3  # Take user input for the start,
   stop, and step of the sequence
4  start = int(input())
5  stop = int(input())
6  step = int(input())
7
8  # Generate the sequence using
   np.arange()
9  a=np.arange(start,stop,step)
10 print(a)
11 # Print the generated sequence
12
```

Average time
0.031 s
31.25 ms

Maximum time
0.036 s
36.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 36 ms Debug

Expected output	Actual output
3	3
10	10
2	2

< Prev Reset Submit Next >

3.2.4. Numpy: Arithmetic and Statistical Ope... 06:16

You are given two arrays A and B. Your task is to complete the function `array_operations`, which will convert these lists into NumPy arrays and perform the following operations:

1. **Arithmetic Operations**:
   - Compute the element-wise sum, difference, and product of the two arrays.
2. **Statistical Operations**:
   - Calculate the mean, median, and standard deviation of array A.
3. **Bitwise Operations**:
   - Perform bitwise AND, bitwise OR, and bitwise XOR on the arrays (ex: A_i OR B_i).

Input Format:
- The first line contains space-separated integers representing the elements of array A.
- The second line contains space-separated integers representing the elements of array B.

Output Format:
- For each operation (arithmetic, statistical, and bitwise), print the results in the specified format as shown in sample test cases.

Sample Test Cases +

Explorer

differentO...

Submit

Debugger

```python
import numpy as np

def array_operations(A, B):

    # Convert A and B to NumPy arrays
    A = np.array(A)
    B = np.array(B)
    # Arithmetic Operations
    sum_result = A+B
    diff_result = A-B
    prod_result = A*B

    # Statistical Operations
    mean_A = np.mean(A)
    median_A = np.median(A)
    std_dev_A = np.std(A)

    # Bitwise Operations
    and_result = A&B
    or_result = A | B
    xor_result = A^B

    # Output results with one space between each element
    print("Element-wise Sum:", ' '.join(map(str, sum_result)))
    print("Element-wise Difference:", ' '.join(map(str, diff_result)))
    print("Element-wise Product:", ' '.join(map(str, prod_result)))

    print(f"Mean of A: {mean_A}")
    print(f"Median of A: {median_A}")
    print(f"Standard Deviation of A: {std_dev_A}")

    print("Bitwise AND:", ' '.join(map(str, and_result)))
    print("Bitwise OR:", ' '.join(map(str, or_result)))
    print("Bitwise XOR:", ' '.join(map(str, xor_result)))

A = list(map(int, input().split())) # Elements of array A
B = list(map(int, input().split())) # Elements of array B
array_operations(A, B)
```

Average time **0.033 s** 32.67 ms

Maximum time **0.043 s** 43.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 43 ms Debug

Expected output	Actual output
1 2 3 4	1 2 3 4
5 6 7 8	5 6 7 8
Element-wise Sum: 6 8 10 12	Element-wise Sum: 6 8 10 12

3.2.5. Numpy: Copying and Viewing Arrays

02:42

The given code takes a list of integers as input and converts it into a NumPy array. Your task is to complete the code by:

- Creating a view of the `original_array` and assigning it to `view_array`.
- Creating a copy of the `original_array` and assigning it to `copy_array`.

After completing these steps, observe how modifying the view affects the `original_array`, while modifying the copy does not.

Input Format:

- A single line of space-separated integers.

Output Format:

- After modifying the view:

```
Original array after modifying view: <original_array>
View array: <view_array>
```

- After modifying the copy:

```
Original array after modifying copy: <original_array>
Copy array: <copy_array>
```

Sample Test Cases +

Explorer

copyAndvi...

Submit

Debugger

```python
import numpy as np

inputlist = list(map(int,input().split(" ")))

# Original array
original_array = np.array(inputlist)

# Create a view
view_array =original_array[0:8]
# Create a copy
copy_array =original_array.copy()

# Modify the view
view_array[0] = 99
print("Original array after modifying view:", original_array)
print("View array:", view_array)

# Modify the copy
copy_array[1] = 88
print("Original array after modifying copy:", original_array)
print("Copy array:", copy_array)

```

Average time
0.019 s
19.25 ms

Maximum time
0.029 s
29.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 29 ms

Debug

Expected output	Actual output
10 20 30 40 50 60 70 80	10 20 30 40 50 60 70 80
Original array after modifying view: [99 20 30 40	Original array after modifying view: [99 20 30 4

CODETANTRA

3.2.6. Numpy: Searching, Sorting, Counting, ... 02:33

The given code in the editor takes a single array, `array1`, as space-separated integers as input from the user.
Additionally, it takes the following inputs:

- `search_value`: The value to search for in the array.
- `count_value`: The value to count its occurrences in the array.
- `broadcast_value`: The value to add for broadcasting across the array.

You need to complete the code to perform the following operations:
1. Searching: Find the indices where `search_value` appears in `array1` and print these indices.
2. Counting: Count how many times `count_value` appears in `array1` and print the count.
3. Broadcasting: Add `broadcast_value` to each element of `array1` using broadcasting, and print the resulting array.
4. Sorting: Sort `array1` in ascending order and print the sorted array.

Input Format:

1. A single line containing space-separated integers representing `array1`.
2. An integer `search_value` represents the value to search for in the array.
3. An integer `count_value` represents the value to count in the array.
4. An integer `broadcast_value` represents the value to add to each element of the array.

Output Format:

1. The indices where `search_value` occurs in `array1`.
2. The count of occurrences of `count_value` in `array1`.
3. The array after adding the `broadcast_value` to each element.
4. The sorted array.

Sample Test Cases +

Explorer

arrayOpera...

Submit

Debugger

```
1  import numpy as np
2
3  # Input array from the user
4  array1 = np.array(list(map(int, input().split())))
5
6  # Searching
7  search_value = int(input("Value to search: "))
8  count_value = int(input("Value to count: "))
9  broadcast_value = int(input("Value to add: "))
10
11 # Find indices where value matches in array1
12 a=np.where(array1==search_value)[0]
13 print(a)
14 # Count occurrences in array1
15 b=np.count_nonzero(array1==count_value)
16 print(b)
17 # Broadcasting addition
18 c=array1+broadcast_value
19 print(c)
20 # Sort the first array
21 d= np.sort(array1)
22 print(d)
```

Average time 0.041 s 40.75 ms

Maximum time 0.051 s 51.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 51 ms Debug

Expected output	Actual output
1 1 1 2 2 2	1 1 1 2 2 2
Value to search: 1	Value to search: 1
Value to count: 2	Value to count: 2

‹ Prev Reset Submit Next ›

3.2.7. Student Data Analysis and Operations

53:50

Write a Python program that takes the file name of a CSV file containing student details, including roll numbers and their marks in three subjects as input, reads the data, and performs the following operations:

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.
- **Find average marks of Subject 1 and Subject 2**: Compute the average marks for Subject 1 and Subject 2.
- **Find average marks of Subject 1 and Subject 3**: Compute the average marks for Subject 1 and Subject 3.
- **Find the roll number of the student with maximum marks in Subject 3**: Identify the student with the highest marks in Subject 3 and print their roll number.
- **Find the roll number of the student with minimum marks in Subject 2**: Identify the student with the lowest marks in Subject 2 and print their roll number.
- **Find the roll number of students who scored 24 marks in Subject 2**: Identify students who obtained exactly 24 marks in Subject 2 and print their roll numbers.
- **Find the count of students who got less than 40 marks in Subject 1**: Count the number of students who scored less than 40 marks in Subject 1.
- **Find the count of students who got more than 90 marks in Subject 2**: Count the number of students who scored more than 90 marks in Subject 2.
- **Find the count of students who scored >=90 in each subject**: Count the number of students who scored 90 or more marks in each subject.
- **Find the count of subjects in which each student scored >=90**: Determine how many subjects each student scored 90 or more marks in.
- **Print Subject 1 marks in ascending order**: Sort and print the marks of students in Subject 1 in ascending order.
- **Print students who scored between 50 and 90 in Subject 1**: Display students who scored marks between 50 and 90 in Subject 1.
- **Find index positions of students who scored 79 in Subject 1**: Identify the index positions of students who scored exactly 79 marks in Subject 1.

Note: Fill in the missing code to perform the above-mentioned operations.

Sample Test Cases +

Explorer

Operations... Submit

```python
import numpy as np

a = np.loadtxt("Sample.csv", delimiter=',', skiprows=1)


# 1. Print all student details
print("All student Details:\n", a )

# 2. print total students
r, c=a.shape
print("Total Students:",r )

# 3. Print all student Roll numbers
print("All Student Roll Nos", a[:,0] )

# 4. Print subject 1 marks
print("Subject 1 Marks", a[:,1])

# 5. print minimum marks of Subject 2
print("Min marks in Subject 2", np.min(a[:,2]) )

# 6. print maximum marks of Subject 3
print("Max marks in Subject 3", np.max(a[:,3]) )

# 7. Print All subject marks
print("All subject marks:", a[:,1:] )

# 8. print Total marks of students
print("Total Marks", np.sum(a[:,1:],axis=1) )

# 9. print average marks of each student
avg = np.mean(a[:,1:],axis=1)
print(np.round(avg,1))


# 10. print average marks of each subject
print("Average Marks of each subject", np.mean(a[:,1:],axis=0) )

# 11. print average marks of S1 and S2
print("Average Marks of S1 and S2", np.mean(a[:,1:3],axis=0) )

# 12. print average marks of S1 and S3
print("Average Marks of S1 and S3", np.mean(a[:,[1,3]],axis=0) )
```

Debugger

Average time **0.090 s** 90.00 ms

Maximum time **0.090 s** 90.00 ms

1 out of 1 shown test case(s) passed

Test case 1 90 ms Debug

Expected output	Actual output
All student Details:	All student Details:
[[301. 67. 77. 88.]	[[301. 67. 77. 88.]
[302. 78. 88. 77.]	[302. 78. 88. 77.]
[303. 45. 56. 89.]	[303. 45. 56. 89.]

CodeTantra

4.1.1. Pandas - series creation and manipula... 03:57

Write a Python program that takes a list of numbers from the user, creates a Pandas series from it, and then calculates the mean of even and odd numbers separately using the **groupby** and **mean()** operations.

Input Format:

- The user should enter a list of numbers separated by space when prompted.

Output Format:

- The program should display the mean of even and odd numbers separately.
- Each mean value should be displayed with a label indicating whether it corresponds to even or odd numbers.

Sample Test Cases +

Explorer

seriesMani...

Submit

```python
import pandas as pd

# Take inputs from the user to create a list of numbers
numbers = list(map(int, input().split()))

# Create a Pandas series from the list of numbers
series = pd.Series(numbers)
# Grouping by even and odd numbers and calculating the mean
grouped = series.groupby(series % 2 == 0).mean()

# Display the mean of even and odd numbers with labels
grouped.index = ['Even' if is_even else 'Odd' for is_even in grouped.index]
print("Mean of even and odd numbers:")
print(grouped)

```

Debugger

Average time
0.034 s
34.00 ms

Maximum time
0.073 s
73.00 ms

3 out of 3 shown test case(s) passed

3 out of 3 hidden test case(s) passed

Test case 1 73 ms Debug

Expected output	Actual output
1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
Mean of even and odd numbers:	Mean of even and odd numbers:
Odd 5.0	Odd 5.0

‹ Prev Reset Submit Next ›

4.1.2. Dictionary to dataframe

17:27

A dictionary of lists has been provided to you in the editor. Create a DataFrame from the dictionary of lists and perform the listed operations, then display the DataFrame before and after each manipulation.

Create the DataFrame:
- Convert the dictionary to a Pandas DataFrame.

Add a new row:
- Take inputs from the user for the new row data (name, age).
- Add the new row to the DataFrame.
- Display the DataFrame after adding the new row.

Modify a row:
- Modify a specific row by changing the age. Take the row index and new age value from the user.
- Display the DataFrame after modifying the row.

Delete a row:
- Take the row index to be deleted from the user.
- Remove the specified row.
- Display the DataFrame after deleting the row.

Add a new column:
- Add a column **Gender** with values taken from the user.
- Display the DataFrame after adding the new column.

Modify a column:
- Convert names to uppercase.
- Display the DataFrame after modifying the column.

Delete a column:
- Remove the **Age** column.
- Display the DataFrame after deleting the column.

Sample Test Cases +

Explorer

dataframe...

Submit

Debugger

```python
import pandas as pd

# Provided dictionary of lists
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
}

# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Adding a new row
new_name = input("New name: ")
new_age = int(input("New age: "))
df.loc[len(df)] = [new_name, new_age]


# Display the DataFrame after adding a new row
print("After adding a row:\n",df)

# Modifying a row
row_to_modify = int(input("Index of row to modify: "))
new_age_value = int(input("New age: "))
df.at[row_to_modify, "Age"] = new_age_value

# Display the DataFrame after modifying a row
print("After modifying a row:")
print(df)

# Deleting a row
row_to_delete = int(input("Index of row to delete: "))
df = df.drop(index=row_to_delete).reset_index(drop=True)
# Display the DataFrame after deleting a row
print("After deleting a row:")
print(df)

# Adding a new column
genders = input("Enter genders separated by space: ").split()
df["Gender"] = genders

# Display the DataFrame after adding a new column
```

Average time **0.372 s** 372.00 ms

Maximum time **0.436 s** 436.00 ms

1 out of 1 shown test case(s) passed

1 out of 1 hidden test case(s) passed

Test case 1 436 ms Debug

Expected output	Actual output
Original DataFrame:	Original DataFrame:
Name Age	Name Age
0 Alice 25	0 Alice 25

CodeTantra

4.1.3. Student Information

01:16

Write a program to read a text file containing student information (name, age, and grade) using Pandas. Perform the following tasks:

- Display the first five rows of the data frame.
- Calculate the average age of the students(limit the average age up to 2 decimal places).
- Filter out the students who have a grade above a certain threshold(consider the threshold grade is 'B').

Note:
Refer to the displayed test cases for better understanding.

Sample Test Cases +

Explorer

studentinf... studentdat... Submit

Debugger

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data into a Pandas DataFrame
df = pd.read_csv(file_name, sep="\s+", names=["Name", "Age", "Grade"])

# Display the first five rows
print("First five rows:")
print(df.head())

# Calculate the average age of students (limited to 2 decimal places)
average_age = round(df["Age"].mean(), 2)
print(f"Average age: {average_age}")

# Filter students with a grade up to 'B'
filtered_students = df[df["Grade"] <= "B"]
print("Students with a grade up to B")
print(filtered_students)
```

Average time
0.166 s
165.50 ms

Maximum time
0.218 s
218.00 ms

1 out of 1 shown test case(s) passed

1 out of 1 hidden test case(s) passed

Test case 1 218 ms Debug

Expected output	Actual output
studentdata.txt	studentdata.txt
First five rows:	First five rows:
Name Age Grade	Name Age Grade

‹ Prev Reset Submit Next ›

4.2.2. Best Selling Product

01:36

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: `Date, Product, Quantity, Price,` and `City`.
- Find the product that sold the most in terms of quantity sold.
- Display the product that sold the most and the total quantity sold for that product.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases +

Explorer

monthForS... | sales_data.... | Submit

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)


# Find the product with the highest total quantity sold
best_product =df.groupby("Product")["Quantity"].sum().idxmax()
highest_quantity = df.groupby("Product")["Quantity"].sum().max()

# Display the result
print(f"Best selling product: {best_product}")
print(f"Total quantity sold: {highest_quantity}")

```

Debugger

Average time **0.078 s** 78.33 ms

Maximum time **0.146 s** 146.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 146 ms | Debug

Expected output	Actual output
sales_data.csv	sales_data.csv
Best selling product: Product A	Best selling product: Product A
Total quantity sold: 23	Total quantity sold: 23

CODETANTRA

4.2.3. City that Sold the Most Products

01:18

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the columns: `Date, Product, Quantity, Price,` and `City`.
- Group the data by `City` and calculate the total quantity of products sold for each city.
- Find the city that sold the most products (based on the total quantity sold).

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases +

Explorer

monthForS... sales_data.... Submit

Debugger

```python
import pandas as pd

# Prompt the user for the file name
file_name = input()

# Load the data
df = pd.read_csv(file_name)

# write the code..
city_sales = df.groupby("City")["Quantity"].sum()

# Find the city that sold the most products
best_city = city_sales.idxmax()
highest_quantity = city_sales.max()
# Display the result
print(f"City sold the most products: {best_city}")

```

Average time
0.062 s
61.67 ms

Maximum time
0.113 s
113.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 113 ms Debug

Expected output	Actual output
sales_data.csv	sales_data.csv
City sold the most products: Los Angeles	City sold the most products: Los Angeles

< Prev Reset Submit Next >

4.2.4. Most Frequently Sold Product Pairs

04:25

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:

- The CSV file contains the following columns: Date, Product, Quantity, Price, and City.
- For each date, find all pairs of products that were sold together (i.e., two products sold on the same date).
- Output the product pair/s that was sold most frequently.

Sample Data:

```
Date,Product,Quantity,Price,City
2025-01-01,Product A,5,20,New York
2025-01-01,Product B,3,15,Los Angeles
2025-01-02,Product A,7,20,New York
2025-01-02,Product C,4,30,Chicago
2025-01-03,Product B,2,15,Chicago
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,30,Los Angeles
```

Explanation:
Transactions:

- **2025-01-01:** Product A, Product B
- **2025-01-02:** Product A, Product C
- **2025-01-03:** Product B, Product A
- **2025-01-04:** Product C, Product B
- **2025-01-05:** Product A, Product C

Now, let's count how often the pairs of products appear together:

- **Product A and Product B**: Appear in transactions on 2025-01-01 and 2025-01-03.
- **Product A and Product C**: Appear in transactions on 2025-01-02 and 2025-01-05.
- **Product B and Product C**: Appears in transactions on 2025-01-04.

Most Frequent Product Combinations:

- **Product A and Product B** (2 times)
- **Product A and Product C** (2 times)

Note:
The data cannot be displayed in the file. You can refer to the sample data provided for insights.

Sample Test Cases +

Explorer

frequently... sales_data.... Submit

Debugger

```python
import pandas as pd
from itertools import combinations
from collections import Counter

# Prompt user to input the file name
file_name = input()

# Read data from the specified CSV file
df = pd.read_csv(file_name)

# write the code
product_pairs = []

for _, group in df.groupby("Date"):
    products = list(group["Product"].unique())
    product_pairs.extend(combinations(sorted(products), 2))  # Generate unique product pairs

# Count occurrences of each product pair
pair_counts = Counter(product_pairs)

# Find the maximum frequency
max_count = max(pair_counts.values())

# Find the most frequent product pairs
most_frequent_pairs = [pair for pair, count in pair_counts.items() if count == max_count]

# Output the most frequent product pairs

for pair in most_frequent_pairs:
    print(f"{pair[0]} and {pair[1]}: {max_count} times")

# Output the most frequent product pairs
```

Average time
0.077 s
77.33 ms

Maximum time
0.145 s
145.00 ms

1 out of 1 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 145 ms Debug

Expected output	Actual output
sales_data.csv	sales_data.csv
Product A and Product B: 2 times	Product A and Product B: 2 times
Product A and Product C:	Product A and Product C:

4.2.5. Titanic Dataset Analysis and Data Cle... 03:38

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset. For each question, perform necessary data cleaning, transformations, and calculations as required.

1. Display the first 5 rows of the dataset.
2. Display the last 5 rows of the dataset.
3. Get the shape of the dataset (number of rows and columns).
4. Get a summary of the dataset (using .info()).
5. Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
6. Check for missing values and display the count of missing values for each column.
7. Fill missing values in the 'Age' column with the median age.
8. Fill missing values in the 'Embarked' column with the most frequent value (mode).
9. Drop the 'Cabin' column due to many missing values.
10. Create a new column, 'FamilySize' by adding the 'SibSp' and 'Parch' columns.

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

titanicData... Submit

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')


print(data.head())

# 2. Display the last 5 rows of the dataset
print(data.tail())

# 3. Get the shape of the dataset
print(data.shape)

# 4. Get a summary of the dataset (info)
print(data.info())

# 5. Get basic statistics of the dataset
print(data.describe())

# 6. Check for missing values
print(data.isnull().sum())

# 7. Fill missing values in the 'Age' column with the median age
median_age = data['Age'].median()
data['Age'].fillna(median_age, inplace=True)

# 8. Fill missing values in the 'Embarked' column with the most frequent value
mode_embarked = data['Embarked'].mode()[0]
data['Embarked'].fillna(mode_embarked, inplace=True)

# 9. Drop the 'Cabin' column due to many missing values
data.drop('Cabin', axis=1, inplace=True)

# 10. Create a new column 'FamilySize' by adding 'SibSp' and 'Parch'
data['FamilySize'] = data['SibSp'] + data['Parch']
```

Average time 0.833 s 833.00 ms | Maximum time 0.833 s 833.00 ms

1 out of 1 shown test case(s) passed

Test case 1 833 ms Debug

Expected output

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
```

Actual output

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
```

4.2.6. Titanic Dataset Analysis and Data Cle... 04:24

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Create a new column 'IsAlone' which is 1 if the passenger is alone (FamilySize = 0), otherwise 0.
2. Convert the 'Sex' column to numeric values (male: 0, female: 1).
3. One-hot encode the 'Embarked' column, dropping the first category.
4. Get the mean age of passengers.
5. Get the median fare of passengers.
6. Get the number of passengers by class.
7. Get the number of passengers by gender.
8. Get the number of passengers by survival status.
9. Calculate the survival rate of passengers.
10. Calculate the survival rate by gender.

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

Explorer

titanicData...

Submit

Debugger

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data['FamilySize'] = data['SibSp'] + data['Parch']
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')

data['FamilySize'] = data['SibSp'] + data['Parch']
data['Alone'] = np.where(data['FamilySize'] == 0, 1, 0)

# 2. Convert 'Sex' to numeric (male: 0, female: 1)
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# 3. One-hot encode the 'Embarked' column, dropping the first category
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# 4. Get the mean age of passengers
mean_age = data['Age'].mean()
print(mean_age)

# 5. Get the median fare of passengers
median_fare = data['Fare'].median()
print(median_fare)

# 6. Get the number of passengers by class
passengers_by_class = data['Pclass'].value_counts()
print(passengers_by_class)

# 7. Get the number of passengers by gender
passengers_by_gender = data['Sex'].value_counts().sort_index()
print(passengers_by_gender)

# 8. Get the number of passengers by survival status
passengers_by_survival = data['Survived'].value_counts().sort
```

Average time **0.334 s** 334.00 ms

Maximum time **0.334 s** 334.00 ms

1 out of 1 shown test case(s) passed

Test case 1 334 ms Debug

Expected output	Actual output
29.69911764705882	29.69911764705882
14.4542	14.4542
3 491	3 491
1 216	1 216
2 184	2 184

4.2.7. Titanic Dataset Analysis and Data Cle... 02:28

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Calculate the survival rate by class.
2. Calculate the survival rate by embarkation location (Embarked_S).
3. Calculate the survival rate by family size (FamilySize).
4. Calculate the survival rate by being alone (IsAlone).
5. Get the average fare by passenger class (Pclass).
6. Get the average age by passenger class (Pclass).
7. Get the average age by survival status (Survived).
8. Get the average fare by survival status (Survived).
9. Get the number of survivors by class (Pclass).
10. Get the number of non-survivors by class (Pclass).

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

Explorer

titanicData...

Submit

Debugger

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data['FamilySize'] = data['SibSp'] + data['Parch']
data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
data = pd.read_csv('Titanic-Dataset.csv')
data['FamilySize'] = data['SibSp'] + data['Parch']
data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

print(data.groupby('Pclass')['Survived'].mean())

# 2. Calculate the survival rate by embarked location (Embarked_S)
print(data.groupby('Embarked_S')['Survived'].mean())

# 3. Calculate the survival rate by family size
print(data.groupby('FamilySize')['Survived'].mean())

# 4. Calculate the survival rate by being alone
print(data.groupby('IsAlone')['Survived'].mean())

# 5. Get the average fare by class
print(data.groupby('Pclass')['Fare'].mean())

# 6. Get the average age by class
print(data.groupby('Pclass')['Age'].mean())

# 7. Get the average age by survival status
print(data.groupby('Survived')['Age'].mean())

# 8. Get the average fare by survival status
print(data.groupby('Survived')['Fare'].mean())
```

Average time **0.482 s** 482.00 ms

Maximum time **0.482 s** 482.00 ms

1 out of 1 shown test case(s) passed

Test case 1 482 ms Debug

Expected output	Actual output
Pclass	Pclass
1 0.629630	1 0.629630
2 0.472826	2 0.472826
3 0.242363	3 0.242363
Name: Survived, dtype: fl	Name: Survived, dtype: f

‹ Prev Reset Submit Next ›

4.2.8. Titanic Dataset Analysis and Data Cle... 02:11

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.

1. Get the number of survivors by gender (Sex).
2. Get the number of non-survivors by gender (Sex).
3. Get the number of survivors by embarkation location (Embarked_S).
4. Get the number of non-survivors by embarkation location (Embarked_S).
5. Calculate the percentage of children (Age < 18) who survived.
6. Calculate the percentage of adults (Age >= 18) who survived.
7. Get the median age of survivors.
8. Get the median age of non-survivors.
9. Get the median fare of survivors.
10. Get the median fare of non-survivors.

The Titanic dataset contains columns as shown below,

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thay
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",fe
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

Note: Refer to the visible test case for better reference.

Sample Test Cases +

Explorer

titanicData... Submit

```python
import pandas as pd
import numpy as np

# Load the Titanic dataset
data = pd.read_csv('Titanic-Dataset.csv')
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)


# 1. Get the number of survivors by gender
survivors_by_gender = data[data['Survived'] == 1]['Sex'].value_counts()
print(survivors_by_gender)

# 2. Get the number of non-survivors by gender
non_survivors_by_gender = data[data['Survived'] == 0]['Sex'].value_counts()
print(non_survivors_by_gender)

# 3. Get the number of survivors by embarked location (Embarked_S)
survivors_by_embarked_s = data[data['Survived'] == 1]['Embarked_S'].value_counts()
print(survivors_by_embarked_s)

# 4. Get the number of non-survivors by embarked location (Embarked_S)
non_survivors_by_embarked_s = data[data['Survived'] == 0]['Embarked_S'].value_counts()
print(non_survivors_by_embarked_s)

# 5. Percentage of children (Age < 18) who survived
children = data[data['Age'] < 18]
children_survival_rate = children['Survived'].mean()
print(children_survival_rate)

# 6. Percentage of adults (Age >= 18) who survived
adults = data[data['Age'] >= 18]
adults_survival_rate = adults['Survived'].mean()
print(adults_survival_rate)

# 7. Median age of survivors
median_age_survivors = data[data['Survived'] == 1]['Age'].median()
print(median_age_survivors)

```

Debugger

Average time **0.352 s** 352.00 ms

Maximum time **0.352 s** 352.00 ms

1 out of 1 shown test case(s) passed

Test case 1 352 ms Debug

Expected output	Actual output
female 233	female 233
male 109	male 109
Name: Sex, dtype: int64	Name: Sex, dtype: int64
male 468	male 468
female 81	female 81

< Prev Reset Submit Next >