1.1.1. Calculate Momentum

08:05

Write a program that accepts the mass of an object (in kilograms) and its velocity (in meters per second), then calculates and displays the momentum of the object. The momentum p is calculated using the formula:

$$p = m \times v$$

where:
m is the mass of the object (in kilograms).
v is the velocity of the object (in meters per second).

Input Format:
A single floating-point number representing the mass of the object in kilograms.
A single floating-point number representing the velocity of the object in meters per second.

Output Format:
The output will display calculated momentum with appropriate units (kgm/s) (rounded up to 2 decimal places).

Sample Test Cases +

Explorer

calculate...

Submit

```
1  m=float(input())
2  v=float(input())
3  p=m*v
4  print('%0.02f'%p,end='')
5  print ("kgm/s",)
```

Debugger

Terminal Test cases

Prev Reset Submit Next

1.1.2. Conditional Calculation Based on the Number of Digits

11:40

Write a Python program that accepts an integer n as input. Depending on the number of digits in n.

Constraints:
$1 \leq n \leq 999$

Input Format:
The input consists of a single integer n.

Output Format:
If n is a single-digit number, print its square.
If n is a two-digit number, print its square root (rounded to two decimal places).
If n is a three-digit number, print its cube root (rounded to two decimal places).
Else print "Invalid".

Sample Test Cases +

Explorer

condition...

Submit

Debugger

```
1
2  n=int(input())
3  if 0<=n<=9:
4      a=n*n
5      print ('%0.0f'%a)
6  elif 10<=n<=99:
7      b=n**(1/2)
8      print('%0.02f'%b)
9  elif 100<=n<999:
10     c=n**(1/3)
11     print('%0.02f'%c)
12 else:
13     print("Invalid")
```

Terminal Test cases

Prev Reset Submit Next

1.1.3. Age and Salary Calculation

28:38

Write a Python program that reads the birth date and salary of employees.

Input Format:
The input consists of:
A string representing the birth date of the employee in the format $DD-MM-YYYY$.
A floating-point number representing the salary of the employee in rupees.

Output Format:
The output should include:
The age of the employee.
The salary of the employee in dollars.

Note:
1INR=0.012USD

Sample Test Cases +

Explorer

birthDate...

Submit

Debugger

```python
from datetime import datetime

def calculate_age(birthdate):
    date_object=datetime.strptime(birthdate,"%d-%m-%Y")
    today = datetime.today()
    if((today.month, today.day) < (date_object.month, date_object.day)):
        age = today.year-date_object.year-((today.month, today.day) < (date_object.month, date_object.day))
        return age
    elif ((today.month, today.day) > (date_object.month, date_object.day)):
        age = today.year-date_object.year-((today.month, today.day) > (date_object.month, date_object.day))
        return age
def convert_salary_to_dollars(salary_in_rupees):
    salary=salary_in_rupees*0.012
    return salary
birthdate = input()
salary_in_rupees = float(input())
age = calculate_age(birthdate)
salary_in_dollars = convert_salary_to_dollars(salary_in_rupees)
print(f"Age: {age}")
print(f"Salary in dollars: {salary_in_dollars:.2f}")

```

Terminal Test cases

Prev Reset Submit Next

1.1.4. Reverse a Number

11:02

You are given an integer number. Your task is to reverse the digits of the number and print the reversed number.

Input Format
The input is an integer.

Output Format
Print a single integer which is the reversed number.

Sample Test Cases +

Explorer

reverseN...

Submit

```
1  number=int(input())
2  n1=str(number)
3  print(n1[::-1])
4
```

Debugger

Terminal Test cases

Prev Reset Submit Next

1.1.5. Multiplication Table

07:54

Write a Python program that takes an integer as input and prints the multiplication table for that integer from 1 to 10.

Input Format:
The first line of input contains an integer that represents the number for which the multiplication table is to be printed.

Output Format:
Print the multiplication table for the given number .

Sample Test Cases +

Explorer

multiplica...

Submit

```
1  i=int(input())
2  a=1
3  while a<=10:
4      print(i,"x",a,"=",i*a)
5      a=a+1
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

1.2.1. Pass or Fail

01:48

Write a Python program that accepts the number of courses and the marks of a student in those courses.

The grade is determined based on the aggregate percentage:

- If the aggregate percentage is greater than 75, the grade is Distinction.
- If the aggregate percentage is greater than or equal to 60 but less than 75, the grade is First Division.
- If the aggregate percentage is greater than or equal to 50 but less than 60, the grade is Second Division.
- If the aggregate percentage is greater than or equal to 40 but less than 50, the grade is Third Division.

Input Format:

The first input will be an integer n, the number of courses.

The second input will be n integers representing the marks of the student in each of the n courses, separated by a space.

Output Format:

If the student passes all courses:

- Print the aggregate percentage (rounded to two decimal places).
- Print the grade based on the aggregate percentage.

If the student fails any course (marks < 40 in any course), print:

Sample Test Cases +

Explorer

passorFa...

Submit

Debugger

```python
# Read the number of courses
num_courses = int(input())

# Read the marks as a list of integers
marks = list(map(int, input().split()))

# Check if any mark is below 40 (fail condition)
if any(mark < 40 for mark in marks):
    print("Fail")
else:
    # Calculate aggregate percentage
    total_marks = sum(marks)
    aggregate_percentage = (total_marks / (num_courses * 100)) * 100

    # Round to two decimal places
    rounded_percentage = round(aggregate_percentage, 2)

    # Determine grade
    if aggregate_percentage > 75:
        grade = "Distinction"
    elif 60 <= aggregate_percentage < 75:
        grade = "First Division"
    elif 50 <= aggregate_percentage < 60:
        grade = "Second Division"
    elif 40 <= aggregate_percentage < 50:
```

Terminal | Test cases

< Prev | Reset | Submit | Next >

1.2.2. Fibonacci series using Recursive Function

02:00

Write a Python program to find the Fibonacci series of a given number of terms using recursive function calls.

Expected Output-1:
Enter terms for Fibonacci series: 5
0 1 1 2 3

Expected Output-2:
Enter terms for Fibonacci series: 9
0 1 1 2 3 5 8 13 21

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when users' input and output match the expected input and output.

Sample Test Cases +

Explorer

fib.py

Submit

```python
# n=int(input("Enter terms for Fibonacci series: "))
# for i in range (n):
#     print(fib(i),end=" ")
def fibonacci(n, memo={0: 0, 1: 1}):
    if n not in memo:
        memo[n] = fibonacci(n-1, memo) + fibonacci(n-2, memo)
    return memo[n]

terms = int(input("Enter terms for Fibonacci series: "))

if terms <= 0:
    print("Please enter a positive integer")
else:
    for i in range(terms):
        print(fibonacci(i), end=" ")
    print()  # This ensures the output ends with a newline
```

Debugger

Terminal Test cases

Prev Reset Submit Next

1.2.3. Pattern - 1

01:07

Write a Python program to print a pattern of asterisks in the form of a right-angled triangle.

Input Format:
The input is an integer, representing the number of rows in the pattern.

Output Format
The output should display the pattern of asterisks (*), with each row containing an increasing number of asterisks.

Note:
Refer to the displayed test cases for the sample pattern.

Sample Test Cases +

Explorer

rightangl...

Submit

```
1  n = int(input())
2  for i in range(1, n + 1):
3      print('* ' * i)
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

1.2.4. Pattern - 2

00:51

Write a Python program to print a right-angled triangle pattern of numbers.

Input Format:
The input is an integer, representing the number of rows in the pattern.

Output Format:
The output should display the pattern of numbers, with each row containing increasing numbers starting from 1 up to the row number.

Note:
Refer to the displayed test cases for the sample pattern.

Sample Test Cases +

Explorer

numberP...

Submit

```
n = int(input())
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(j, end=' ')
    print()
```

Debugger

Terminal Test cases

Prev Reset Submit Next

2.1.1. List operations

04:51

Write a Python program that implements a menu-driven interface for managing a list of integers. The program should have the following menu options:

1. Add
2. Remove
3. Display
4. Quit

The program should repeatedly prompt the user to enter a choice from the menu. Depending on the choice selected, the program should perform the following actions:

- **Add**: Prompts the user to enter an integer and add it to the integer list. If the input is not a valid integer, display "Invalid input".
- **Remove:** Prompts the user to enter an integer to remove from the list. If the integer is found in the list, remove it; otherwise, display "Element not found". If the list is empty, display "List is empty".
- **Display**: Displays the current list of integers. If the list is empty, display "List is empty".
- **Quit**: Exits the program.
- The program should handle invalid menu choices by displaying "Invalid choice".Ensure that the program continues to prompt the user until they choose to quit (option 4).

Sample Test Cases +

Explorer

listOps.py

Submit

```python
def menu():
    print("1. Add")
    print("2. Remove")
    print("3. Display")
    print("4. Quit")

def main():
    int_list = []

    while True:
        menu()
        choice = input("Enter choice: ")

        if choice == '1':
            num = input("Integer: ")
            if num.lstrip('-').isdigit():
                int_list.append(int(num))
                print(f"List after adding: {int_list}")
            else:
                print("Invalid input")

        elif choice == '2':
            if not int_list:
                print("List is empty")
            else:
```

Debugger

Terminal Test cases

Prev Reset Submit Next

2.1.2. Dictionary Operations

04:45

Write a Python program to perform the following dictionary operations:

- Create an empty dictionary and display it.
- Ask the user how many items to add, then input key-value pairs.
- Show the dictionary after adding items.
- Ask the user to update a key's value. Print "Value updated" if the key exists, otherwise print "Key not found".
- Retrieve and print a value using a key. If not found, print "Key not found".
- Use get() to retrieve a value. If the key doesn't exist, print "Key not found".
- Delete a key-value pair. If the key exists, delete and print "Deleted". If not, print "Key not found".
- Display the updated dictionary.

Note: Refer to visible test cases.

Sample Test Cases +

Explorer

dictOpera...

Submit

Debugger

```
37  def main():
38      # Step 1: Create an empty dictionary and display it
39      my_dict = {}
40      print("Empty Dictionary:", my_dict)
41
42      # Step 2: Ask the user how many items to add
43      try:
44          n = int(input("Number of items: "))
45      except ValueError:
46          print("Invalid input. Please enter an integer.")
47          return
48
49      # Step 3: Input key-value pairs
50      for _ in range(n):
51          key = input("key: ")
52          value = input("value: ")
53          my_dict[key] = value
54
55      # Step 4: Show dictionary after adding items
56      print("Dictionary:", my_dict)
57
58      # Step 5: Update a key's value
59      update_key = input("Enter the key to update: ")
60      if update_key in my_dict:
61          new_value = input("Enter the new value: ")
```

Terminal Test cases

Prev Reset Submit Next

2.2.1. Linear search Technique

01:18

Write a program to check whether the given element is present or not in the array of elements using linear search.

Input format:

- The first line of input contains the array of integers which are separated by space
- The last line of input contains the key element to be searched

Output format:

- If the element is found, print the index.
- If the element is not found, print **Not found**.

Sample Test Case:

Input:

1 2 3 4 3 5 6

3

Output:

2

Sample Test Cases +

Explorer

CTP1709...

Submit

Debugger

```
1  # Read the array input
2  arr = list(map(int, input().split()))
3
4  # Read the key to search
5  key = int(input())
6
7  # Perform linear search
8  found = False
9  for i in range(len(arr)):
10     if arr[i] == key:
11         print(i)
12         found = True
13         break
14
15 if not found:
16     print("Not found")
17
18
```

Terminal Test cases

Prev Reset Submit Next

2.2.2. Captain of the Team

00:42

You are provided with the heights of 11 cricket players (in centimeters). Your task is to identify the tallest player, who will be selected as the captain of the team.

Input Format:
The first line of input will contain 11 integers, each representing the height of a player (in centimeters), each separated by a space.

Output Format
The output should be the height (in centimeters) of the tallest player.

Sample Test Cases +

Explorer

captainof...

Submit

```python
# Read the heights of 11 players
heights = list(map(int, input().split()))

# Find the tallest height
tallest = max(heights)

# Print the tallest height
print(tallest)
```

Terminal Test cases

Debugger

Prev Reset Submit Next

3.1.1. Numpy array operations

02:58

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...

Submit

Debugger

```python
# import numpy as np
import numpy as np

# Read number of rows and columns
rows, cols = map(int, input().split())

# Read elements row-wise
elements = []
for _ in range(rows):
    elements.extend(map(int, input().split()))  # use int, not float

# Create numpy array and reshape
arr = np.array(elements, dtype=int).reshape(rows, cols)

# Display the array
print(arr)

# Display properties
print(arr.ndim)
print(arr.shape)
print(arr.size)
```

Terminal Test cases

Prev Reset Submit Next

3.2.1. Numpy: Matrix Operations

17:34

The given code takes two 3×3 matrices, $\mathrm{matrix\_a}$, and $\mathrm{matrix\_b}$, 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** ($\mathrm{matrix\_a} + \mathrm{matrix\_b}$)
2. **Subtraction** ($\mathrm{matrix\_a} - \mathrm{matrix\_b}$)
3. **Element-wise Multiplication** ($\mathrm{matrix\_a} * \mathrm{matrix\_b}$)
4. **Matrix Multiplication** ($\mathrm{matrix\_a} \cdot \mathrm{matrix\_b}$)
5. **Transpose of Matrix A**

Input Format:

- The user will input 3 rows for $\mathrm{matrix\_a}$, each containing 3 integers separated by spaces.
- Similarly, the user will input 3 rows for $\mathrm{matrix\_b}$, 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.

Sample Test Cases +

Explorer | matrixOp... | 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
print("Addition (A + B):")
print(matrix_a + matrix_b)
# Subtraction
print("Subtraction (A - B):")
print(matrix_a - matrix_b)
# Multiplication (element-wise)
print("Element-wise Multiplication (A * B):")
print(matrix_a * matrix_b)
# Matrix multiplication (dot product)
print("A dot B:")
print(np.dot(matrix_a,matrix_b))
# Transpose
print("Transpose of A:")
a = matrix_a.T
```

Terminal | Test cases

3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

15:14

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

```
# # Perform horizontal stacking (hstack)



# # Perform vertical stacking (vstack)
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)
b = np.vstack((arr1,arr2))
print("Vertical Stack:")
print(b)

# Perform vertical stacking (vstack)
```

Debugger

Terminal Test cases

Prev Reset Submit Next

3.2.3. Numpy: Custom Sequence Generation

09:03

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 +

customS...

Submit

```python
# import numpy as np

# # Take user input for the start, stop, and step of the sequence
# start = int(input())
# stop = int(input())
# step = int(input())

# # Generate the sequence using np.arange()

# # Print the generated sequence
import numpy as np

# Take user input for the start, stop, and step of the sequence
start = int(input())
stop = int(input())
step = int(input())
a = np.arange(start,stop,step)
print(a)
# Generate the sequence using np.arange()

# Print the generated sequence


```

Terminal Test cases

Prev Reset Submit Next

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Operations, Bitw... 01:41

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

different...

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)))
```

Terminal Test cases

Prev Reset Submit Next

3.2.5. Numpy: Copying and Viewing Arrays

03:37

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

copyAnd...

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.view()


# 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)


```

Terminal Test cases

Prev Reset Submit Next

3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting

04:00

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.

Sample Test Cases +

Explorer | arrayOpe... | Submit | Debugger

```
15  # # Broadcasting addition
16
17  # # Sort the first array
18  import numpy as np
19
20  # Input array from the user
21  array1 = np.array(list(map(int, input().split())))
22
23  # Searching
24  search_value = int(input("Value to search: "))
25  count_value = int(input("Value to count: "))
26  broadcast_value = int(input("Value to add: "))
27
28  # Find indices where value matches in array1
29  a=np.where(array1==search_value)[0]
30  print(a)
31  # Count occurrences in array1
32  b=np.count_nonzero(array1==count_value)
33  print(b)
34  # Broadcasting addition
35  c = array1 + broadcast_value
36  print(c)
37  # Sort the first array
38  d=np.sort(array1)
39  print(d)
```

Terminal | Test cases

Prev | Reset | Submit | Next

3.2.7. Student Data Analysis and Operations

05:08

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.

Sample Test Cases +

Explorer

Operatio...

Submit

Debugger

```
74
75  # # 22. Print the index position of marks
76  import numpy as np
77
78  a = np.loadtxt("Sample.csv", delimiter=',', skiprows=1)
79
80  # 1. Print all student details
81  print("All student Details:\n",a)
82
83  # 2. print total students
84
85  print("Total Students:", len(a))
86
87  # 3. Print all student Roll numbers
88  print("All Student Roll Nos", a[:,0])
89
90  # 4. Print subject 1 marks
91  print("Subject 1 Marks",a[:,1])
92
93  # 5. print minimum marks of Subject 2
94  print("Min marks in Subject 2",np.min(a[:,2]))
95
96  # 6. print maximum marks of Subject 3
97  print("Max marks in Subject 3",np.max(a[:,3]))
98
```

Terminal Test cases

Prev Reset Submit Next

4.1.1. Pandas - series creation and manipulation

01:24

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

seriesMa...

Submit

Debugger

```
8  # # Grouping by even and odd numbers and calculating the mean
9  # grouped =
10
11 # # Display the mean of even and odd numbers with labels
12 # grouped.index = ['Even' if is_even else 'Odd' for is_even in
   grouped.index]
13 # print("Mean of even and odd numbers:")
14 # print(grouped)
15 import pandas as pd
16
17 # Take inputs from the user to create a list of numbers
18 numbers = list(map(int, input().split()))
19
20 # Create a Pandas series from the list of numbers
21 series= pd.Series(numbers)
22 # Grouping by even and odd numbers and calculating the mean
23 grouped = series.groupby(series%2==0).mean()
24
25 # Display the mean of even and odd numbers with labels
26 grouped.index = ['Even' if is_even else 'Odd' for is_even in
   grouped.index]
27 print("Mean of even and odd numbers:")
28 print(grouped)
29
30
```

Terminal Test cases

Prev Reset Submit Next

4.1.2. Dictionary to dataframe

46:02

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:

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: "))
new_row={'Name': new_name,'Age':new_age}
df=pd.concat([df,pd.DataFrame([new_row])],ignore_index=True)
# Display the DataFrame after adding a new row
print("After adding a row:\n",df)

# Modifying a row
modify_index=int(input("Index of row to modify: "))
```

Terminal | Test cases

Prev | Reset | Submit | Next

4.1.3. Student Information

27:58

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 +

studentin... studentdat... Submit

```python
import pandas as pd

# Read the text file into a DataFrame
file = input()
data = pd.read_csv(file, sep="\s+", header=None, names=["Name","Age","Grade"])

# write your code here..

print("First five rows:")
print(data.head(5))
age =round(data['Age'].mean(),2)
print("Average age:",age)
print("Students with a grade up to B")
df=pd.DataFrame(data)
a=df[df['Grade']<='B']
print(a)
```

Terminal Test cases

Prev Reset Submit Next

4.2.1. Month with the Highest Total Sales

00:49

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 Month and calculate the total sales for each month.
- Find the month with the highest total sales and display it.
- Also, display the total sales for the best month.

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:

Sample Test Cases +

Explorer | monthFor... | sales_dat... | Submit | Debugger

```
13
14  # print(f"Best month: {best_month}")
15  # print(f"Total sales: ${highest_sales:.2f}")
16  import pandas as pd
17
18  # Prompt the user for the file name
19  file_name = input()
20
21  # Load the data
22  df = pd.read_csv(file_name)
23  df['Date'] = pd.to_datetime(df['Date'])
24  #Extract the month from the Date column
25  df ['Month'] = df['Date'].dt.to_period('M')
26  #Calculate the total sales for each row
27  df['Total_Sales'] = df['Quantity'] * df['Price']
28  #Group the data by Month and calculate the total sales for each month
29  monthly_sales = df.groupby('Month') ['Total_Sales'].sum()
30  # Find the month with the highest total sales
31  best_month = monthly_sales.idxmax()
32  highest_sales = monthly_sales.max()
33
34  print(f"Best month: {best_month}")
35  print(f"Total sales: ${highest_sales:.2f}")
36
37
```

Terminal | Test cases

Prev | Reset | Submit | Next

4.2.2. Best Selling Product

00:37

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

monthFor... sales_dat...

Submit

```python
# # Find the product with the highest total quantity sold
# best_product =
# highest_quantity =

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

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

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


product_sales = df.groupby("Product")["Quantity"].sum()
best_product = product_sales.idxmax()
highest_quantity = product_sales.max()

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


```

Debugger

Terminal Test cases

Prev Reset Submit Next

4.2.3. City that Sold the Most Products

00:31

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

monthFor... sales_dat...

Submit

```
3  # # Prompt the user for the file name
4  # file_name = input()
5
6  # # Load the data
7  # df = pd.read_csv(file_name)
8
9  # # write the code..
10
11 # # Display the result
12 # print(f"City sold the most products: {best_city}")
13 import pandas as pd
14
15 # Prompt the user for the file name
16 file_name = input()
17
18 # Load the data
19 df = pd.read_csv(file_name)
20
21 # write the code..
22 city_sales = df.groupby("City")["Quantity"].sum()
23 best_city = city_sales.idxmax()
24 # Display the result
25 print(f"City sold the most products: {best_city}")
26
27
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

4.2.4. Most Frequently Sold Product Pairs

00:35

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:

Sample Test Cases +

Explorer | frequentI... | sales_dat... | Submit | Debugger

```python
# # Output the most frequent product pairs
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)

date_products = {}
#-Group-products-by-date
for date, group in df.groupby('Date'):
    products = group['Product'].unique()
    if len(products) > 1:
        date_products[date] = products
#Count-product pairs
pair_counter = Counter()
for products in date_products.values():
    pairs = combinations(sorted(products), 2)
    pair_counter.update(pairs)
#Find the maximum-frequency
if pair_counter:
    max_count=max(pair_counter.values())
```

Terminal | Test cases

< Prev | Reset | Submit | Next >

4.2.5. Titanic Dataset Analysis and Data Cleaning

00:43

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,

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

Sample Test Cases +

Explorer

titanicDat...

Submit

```
21
22  #
23
24
25
26  # #
27
28  import pandas as pd
29  import numpy as np
30
31  # Load the Titanic dataset
32  data = pd.read_csv('Titanic-Dataset.csv')
33
34  print(data.head())
35  print(data.tail())
36  print(data.shape)
37  print(data.info())
38  print(data.describe())
39  print(data.isnull().sum())
40  median_age = data['Age'].median()
41  data['Age'].fillna(median_age, inplace=True)
42  mode_embarked = data['Embarked'].mode()[0]
43  data['Embarked'].fillna(mode_embarked, inplace=True)
44  data.drop('Cabin', axis=1, inplace=True)
45  data['FamilySize'] = data['SibSp'] + data['Parch']
```

Debugger

Terminal Test cases

Prev Reset Submit Next

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

00:35

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,

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

Sample Test Cases +

Explorer

titanicDat...

Submit

```
35
36  import pandas as pd
37  import numpy as np
38
39  # Load the Titanic dataset
40  data = pd.read_csv('Titanic-Dataset.csv')
41  data['FamilySize'] = data['SibSp'] + data['Parch']
42
43  data['IsAlone'] = np.where(data['FamilySize'] == 0, 1, 0)
44  #2-2. Convert-Sex-to-numeric (male:, forale: 1)
45  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
46  #a-3.-One-hot-encode the Embarked column, dropping the first-category
47  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
48  #4-4. Get the mean-age-of-passengers
49  mean_age = data['Age'].mean()
50  print(mean_age)
51  #-5. Get the median fare of passengers
52  median_fare = data['Fare'].median()
53  print(median_fare)
54  #-6. Get the number of passengers-by-class
55  passengers_by_class = data['Pclass'].value_counts()
56  print(passengers_by_class)
57  #7. Get the number-of-passengers-by-gender
58  passengers_by_gender = data['Sex'].value_counts().sort_index()
59  print(passengers_by_gender)
```

Terminal Test cases

Debugger

Prev Reset Submit Next

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3

00:40

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,

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

Sample Test Cases +

Explorer titanicDat... Submit Debugger

```
38 import pandas as pd
39 import numpy as np
40
41 # Load the Titanic dataset
42 data = pd.read_csv('Titanic-Dataset.csv')
43 data['FamilySize'] = data['SibSp'] + data['Parch']
44 data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)
45 data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
46
47 # 1. Calculate the survival rate by class
48 print(data.groupby('Pclass') ['Survived'].mean())
49 #2. Calculate the survival rate by embarked location (Embarked_S)
50 print(data.groupby('Embarked_S') ['Survived'].mean())
51 #3. Calculate the survival rate by family size
52 print(data.groupby('FamilySize') ['Survived'].mean())
53 #4. Calculate the survival rate by being alone
54 print(data.groupby('IsAlone') ['Survived'].mean())
55 #5. Get the average fare by class
56 print(data.groupby('Pclass') ['Fare'].mean())
57 #6. Get the average age by class
58 print(data.groupby('Pclass') ['Age'].mean())
59 #7. Get the average age by survival status
60 print(data.groupby('Survived')['Age'].mean())
61 #8. Get the average fare by survival status
62 print(data.groupby('Survived') ['Fare'].mean())
```

Terminal Test cases

Prev Reset Submit Next

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

00:39

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,

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

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```
37  import pandas as pd
38  import numpy as np
39
40  # Load the Titanic dataset
41  data = pd.read_csv('Titanic-Dataset.csv')
42  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
43
44
45  survivors_by_gender = data[data['Survived'] == 1]['Sex'].value_counts()
46  print(survivors_by_gender)
47
48  non_survivors_by_gender = data[data['Survived'] == 0]
    ['Sex'].value_counts()
49  print(non_survivors_by_gender)
50
51  #3. Get the number of survivors by embarked location (Embarked_S)
52
53  survivors_by_embarked_s= data[data['Survived'] == 1]
    ['Embarked_S'].value_counts()
54
55  print(survivors_by_embarked_s)
56
57  #4. Get the number of non-survivors by embarked location (Embarked_S)
58
59  non_survivors_by_embarked_s= data[data['Survived'] == 0]
```

Terminal Test cases

Prev Reset Submit Next

5.1.1. Stacked Plot

01:34

Create a stacked area plot to visualize the temperature variations for three different cities (City A, City B, and City C) across the months of the year. The temperature data is provided for each city in the editor.

Your task is to:

- Create a stacked area plot using the data.
- Label the x-axis as "Month", the y-axis as "Temperature", and provide the title "Temperature Variation" for the plot.
- Display the plot showing the temperature variation for each city throughout the months of the year.

Sample Test Cases +

Explorer

stackedpl...

Submit

```
#     'City_B_Temperature': [2, 3, 5, 6, 10, 14, 16, 17, 12, 9, 5, 3],
#     'City_C_Temperature': [3, 4, 6, 8, 9, 12, 15, 14, 10, 7, 4, 2]
# }
import matplotlib.pyplot as plt
import pandas as pd

# Data for Months and Temperature for three cities
data = {
    'Month': ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'],
    'City_A_Temperature': [5, 7, 10, 13, 17, 20, 22, 21, 18, 12, 8, 6],
    'City_B_Temperature': [2, 3, 5, 6, 10, 14, 16, 17, 12, 9, 5, 3],
    'City_C_Temperature': [3, 4, 6, 8, 9, 12, 15, 14, 10, 7, 4, 2]
}

# Write your code...
df = pd.DataFrame(data)
plt.stackplot(df['Month'],df['City_A_Temperature'],df['City_B_Temperature'],df['City_C_Temperature'])
plt.xlabel('Month')
plt.ylabel('Temperature')
plt.title('Temperature Variation')
plt.legend(loc = 'upper left')
plt.show()
```

Terminal Test cases

Debugger

< Prev Reset Submit Next >

5.2.1. Titanic Dataset

08:23

Write a Python program to analyze and visualize data from the Titanic dataset based on the following instructions:

Dataset Information:

The dataset is stored in a CSV file named `titanic.csv` and has been loaded using the `pandas` library. It contains the following columns:

- `Pclass`: Passenger class (1 = First, 2 = Second, 3 = Third).
- `Gender`: Gender of the passenger (male/female).
- `Age`: Age of the passenger.
- `Survived`: Survival status (0 = Did not survive, 1 = Survived).
- `Fare`: Ticket fare paid by the passenger.

Visualization:

To represent these trends, you will create 5 visualizations using Matplotlib. The visualizations should be arranged in a 3x2 grid (3 rows and 2 columns).

Visualization Details:

Write the code to create a series of visualizations as follows:

Sample Test Cases +

Explorer

titanicDat...

Submit

Debugger

```
10  # # write the code..
11  import pandas as pd
12  import matplotlib.pyplot as plt
13
14  # Load the Titanic dataset from the CSV file
15  df = pd.read_csv('titanic.csv')
16
17  # Set up the figure for 5 subplots
18  fig, axes = plt.subplots(3, 2, figsize=(12, 12))
19  # write the code..
20  import pandas as pd
21  import matplotlib.pyplot as plt
22
23  # Load the Titanic dataset from the CSV file
24  df = pd.read_csv('titanic.csv')
25
26  # Set up the figure for 5 subplots
27  fig, axes = plt.subplots(3, 2, figsize=(12, 12))
28
29  # Plot 1: Count of passengers by class
30  axes[0, 0].bar(df['Pclass'].value_counts().index,
    df['Pclass'].value_counts(), color='skyblue')
31  axes[0, 0].set_title("Passenger Class Distribution")
32  axes[0, 0].set_xlabel("Pclass")
33  axes[0, 0].set_ylabel("Count")
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.2. Histogram of passenger information of Titanic

04:11

Write a Python code to plot a histogram for the distribution of the 'Age' column from the Titanic dataset. The histogram should display the frequency of different age ranges with the following specifications:

1. Use **30 bins** for the histogram.
2. Set the **edge color** of the bars to **black** (k).
3. Label the x-axis as **'Age'** and the y-axis as **'Frequency'**.
4. Add the title **"Age Distribution"** to the histogram.

The Titanic dataset contains columns as shown below,

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

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
```

Sample Test Cases +

Explorer

Histogra...

Submit

Debugger

```
12  # # Convert categorical features to numeric
13  # data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14  # data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16  import pandas as pd
17  import matplotlib.pyplot as plt
18
19  # Load the Titanic dataset
20  data = pd.read_csv('Titanic-Dataset.csv')
21
22  # Data Cleaning
23  data['Age'].fillna(data['Age'].median(), inplace=True)
24  data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
25  data.drop('Cabin', axis=1, inplace=True)
26
27  # Convert categorical features to numeric
28  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
29  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
30
31  # Write your code here for Histogram
32  plt.hist(data['Age'],bins=30,edgecolor = 'k')
33  plt.title('Age Distribution')
34  plt.xlabel('Age')
35  plt.ylabel('Frequency')
36  plt.show()
```

Terminal Test cases

Prev Reset Submit Next

5.2.3. Bar plot of survival rate of passengers

01:02

Write a Python code to plot a bar chart that shows the count of passengers who survived and did not survive in the Titanic dataset. The chart should display the following specifications:

1. Use the '**Survived**' column to show the count of survivors (0 = Did not survive, 1 = Survived).
2. Set the chart type to '**bar**'.
3. Add the title "**Survival Count**" to the chart.
4. Label the x-axis as '**Survived**' and the y-axis as '**Count**'.

The Titanic dataset contains columns as shown below,

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

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
```

Sample Test Cases +

Explorer | BarPlotOf... | Submit | Debugger

```
15
16  import pandas as pd
17  import matplotlib.pyplot as plt
18
19  # Load the Titanic dataset
20  data = pd.read_csv('Titanic-Dataset.csv')
21
22  # Data Cleaning
23  data['Age'].fillna(data['Age'].median(), inplace=True)
24  data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
25  data.drop('Cabin', axis=1, inplace=True)
26
27  # Convert categorical features to numeric
28  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
29  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
30
31  # Write your code here for Bar Plot for Survival Rate
32  survival_counts = data['Survived'].value_counts()
33  survival_counts.plot(kind='bar')
34  plt.title('Survival Count')
35  plt.xlabel('Survived')
36  plt.ylabel('Count')
37  plt.show()
38
39
```

Terminal | Test cases

5.2.4. Bar Plot for Survival by Gender

00:45

Write a Python code to plot a stacked bar chart that shows the count of passengers who survived and did not survive, grouped by gender, in the Titanic dataset. The chart should display the following specifications:

1. Group the data by the **'Sex'** column, then use the **value_counts()** function to count the occurrences of survivors (0 = Did not survive, 1 = Survived) for each gender.
2. Use a **stacked bar chart** to display the survival counts.
3. Add the title **"Survival by Gender"** to the chart.
4. Label the x-axis as **'Gender'** and the y-axis as **'Count'**.
5. The legend should indicate **'Not Survived'** and **'Survived'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

Sample Test Cases +

Explorer | BarPlotOf... | Submit | Debugger

```python
import pandas as pd
import matplotlib.pyplot as plt

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

# Data Cleaning
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop('Cabin', axis=1, inplace=True)

# Convert categorical features to numeric
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# Write your code here for Bar Plot for Survival by Gender
survival_by_gender = data.groupby('Sex')
['Survived'].value_counts().unstack().fillna(0)
survival_by_gender.columns = ['Not Survived','Survived']
survival_by_gender.index = ['0','1']
survival_by_gender.plot(kind = 'bar',stacked = True)
plt.title('Survival by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title=None)
```

Terminal | Test cases

< Prev | Reset | Submit | Next >

5.2.5. Bar Plot for Survival by Pclass

00:47

Write a Python code to plot a stacked bar chart that shows the count of passengers who survived and did not survive, grouped by passenger class (**Pclass**), in the Titanic dataset. The chart should display the following specifications:

1. Group the data by the **Pclass** column and count the number of survivors (0 = Did not survive, 1 = Survived) for each class using **value_counts()**.
2. Use a **stacked bar chart** to display the survival counts.
3. Add the title **"Survival by Pclass"** to the chart.
4. Label the x-axis as **'Pclass'** and the y-axis as **'Count'**.
5. The legend should indicate **'Not Survived'** and **'Survived'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

Sample Test Cases +

Explorer

BarPlotOf...

Submit

```
# # Write your code here for Bar Plot for Survival by Pclass
import pandas as pd
import matplotlib.pyplot as plt

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

# Data Cleaning
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop('Cabin', axis=1, inplace=True)

# Convert categorical features to numeric
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# Write your code here for Bar Plot for Survival by Pclass
survival_by_class = data.groupby('Pclass')
['Survived'].value_counts().unstack().fillna(0)
survival_by_class.columns = ['Not Survived','Survived']
survival_by_class.plot(kind = 'bar',stacked = True)
plt.title('Survival by Pclass')
plt.xlabel('Pclass')
plt.ylabel('Count')
plt.legend(title = None)
plt.show()
```

Debugger

Terminal Test cases

Prev Reset Submit Next

5.2.6. Bar Plot for Survival by Embarked

00:48

Write a Python code to plot a stacked bar chart showing the survival count for passengers based on their embarkation location in the Titanic dataset.
The chart should display the following specifications:

1. Use the **Embarked** column to determine the embarkation location. After converting this column into dummy variables (using **pd.get_dummies()**), plot the survival count based on the **Embarked_Q** column (representing passengers who embarked from Queenstown) in relation to survival.
2. Set the chart type to 'bar' and make it stacked.
3. Add the title **"Survival by Embarked "** to the chart.
4. Label the x-axis as **'Embarked'** and the y-axis as **'Count'**.
5. Include a legend to distinguish between survivors and non-survivors (label the legend as **'Survived'** and **'Not Survived'**).

The Titanic dataset contains columns as shown below,

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

Sample Test Cases +

BarPlotOf... Submit

```
14  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16  import pandas as pd
17  import matplotlib.pyplot as plt
18
19  # Load the Titanic dataset
20  data = pd.read_csv('Titanic-Dataset.csv')
21
22  # Data Cleaning
23  data['Age'].fillna(data['Age'].median(), inplace=True)
24  data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
25  data.drop('Cabin', axis=1, inplace=True)
26
27  # Convert categorical features to numeric
28  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
29  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
30
31  # Write your code here for Bar Plot for Survival by Embarked
32  grouped = data.groupby('Embarked_Q')
    ['Survived'].value_counts().unstack().fillna(0)
33  grouped.columns = ['Not Survived','Survived']
34  grouped.plot(kind = 'bar',stacked = True)
35  plt.title('Survival by Embarked')
36  plt.xlabel('Embarked')
37  plt.ylabel('Count')
```

Explorer Debugger Terminal Test cases

Prev Reset Submit Next

5.2.7. Box plot for Age Distribution

00:46

Write a Python code to plot a boxplot that shows the distribution of the 'Age' column from the Titanic dataset across different passenger classes. The boxplot should display the following specifications:

1. Use the **Pclass** column to group the data for the boxplot.
2. Set the title of the plot to **"Age by Pclass"**.
3. Remove the default subtitle with **plt.suptitle('')**.
4. Label the x-axis as **'Pclass'** and the y-axis as **'Age'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
```

Sample Test Cases +

Explorer

BoxPlotF...

Submit

```python
import pandas as pd
import matplotlib.pyplot as plt

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

# Data Cleaning
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop('Cabin', axis=1, inplace=True)

# Convert categorical features to numeric
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# Write your code here for Box Plot for Age by Pclass
plt.figure(figsize = (8,6))
data.boxplot(column = 'Age',by = 'Pclass')
plt.suptitle('')
plt.title('Age by Pclass')
plt.xlabel('Pclass')
plt.ylabel('Age')
plt.show()


```

Terminal Test cases

Debugger

Prev Reset Submit Next

5.2.8. Box Plot for Age by Survived

00:49

Write a Python code to plot a boxplot that shows the distribution of the 'Age' column from the Titanic dataset based on whether passengers survived or not. The boxplot should display the following specifications:

1. Use the **Survived** column to group the data for the boxplot (0 = Did not survive, 1 = Survived).
2. Set the title of the plot to **"Age by Survival"**.
3. Remove the default subtitle with **plt.suptitle('')**.
4. Label the x-axis as **'Survived'** and the y-axis as **'Age'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
```

Sample Test Cases +

Explorer

BoxPlotF...

Submit

```python
# # Write your code here for Box Plot for Age by Survived
import pandas as pd
import matplotlib.pyplot as plt

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

# Data Cleaning
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop('Cabin', axis=1, inplace=True)

# Convert categorical features to numeric
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# Write your code here for Box Plot for Age by Survived
plt.figure(figsize=(8,6))
data.boxplot(column = 'Age',by = 'Survived')
plt.suptitle('')
plt.title('Age by Survival')
plt.xlabel('Survived')
plt.ylabel('Age')
plt.show()
```

Terminal Test cases

Debugger

Prev Reset Submit Next

5.2.9. Box Plot for Fare by Pclass

00:41

Write a Python code to plot a boxplot that shows the distribution of the 'Fare' column from the Titanic dataset based on the passenger class (Pclass). The boxplot should display the following specifications:

1. Use the **Pclass** column to group the data for the boxplot.
2. Set the title of the plot to **"Fare by Pclass"**.
3. Remove the default subtitle with **plt.suptitle('')**.
4. Label the x-axis as **'Pclass'** and the y-axis as **'Fare'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
```

Sample Test Cases +

Explorer | BoxPlotF... | Submit | Debugger

```python
# # Write your code here for Box Plot for Fare by Pclass
import pandas as pd
import matplotlib.pyplot as plt

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

# Data Cleaning
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop('Cabin', axis=1, inplace=True)

# Convert categorical features to numeric
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# Write your code here for Box Plot for Fare by Pclass
plt.figure(figsize=(8,6))
data.boxplot(column='Fare',by='Pclass')
plt.suptitle('')
plt.title('Fare by Pclass')
plt.xlabel('Pclass')
plt.ylabel('Fare')
plt.show()
```

Terminal | Test cases

< Prev | Reset | Submit | Next >

5.2.10. Scatter Plot for Age vs. Fare

00:43

Write a Python code to plot a scatter plot showing the relationship between the 'Age' and 'Fare' columns in the Titanic dataset. The scatter plot should display the following specifications:

1. Use the **Age** column for the x-axis and the **Fare** column for the y-axis.
2. Set the title of the plot to **"Age vs. Fare"**.
3. Label the x-axis as **'Age'** and the y-axis as **'Fare'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
```

Sample Test Cases +

Explorer | AgeFareS... | Submit | Debugger

```
14  # data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16  # # Write your code here for Box Plot for Fare by Pclass
17  import pandas as pd
18  import matplotlib.pyplot as plt
19
20  # Load the Titanic dataset
21  data = pd.read_csv('Titanic-Dataset.csv')
22
23  # Data Cleaning
24  data['Age'].fillna(data['Age'].median(), inplace=True)
25  data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
26  data.drop('Cabin', axis=1, inplace=True)
27
28  # Convert categorical features to numeric
29  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
30  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
31
32  # Write your code here for Box Plot for Fare by Pclass
33  plt.figure(figsize=(6.4,4.8))
34  plt.scatter(data['Age'],data['Fare'])
35  plt.title('Age vs. Fare')
36  plt.xlabel('Age')
37  plt.ylabel('Fare')
38
```

Terminal | Test cases

< Prev | Reset | Submit | Next >

5.2.11. Scatter Plot for Age vs. Fare by Survived

01:26

Write a Python code to plot a scatter plot showing the relationship between the 'Age' and 'Fare' columns in the Titanic dataset, with points color-coded by survival status. The scatter plot should display the following specifications:

1. Use the **Age** column for the x-axis and the **Fare** column for the y-axis.
2. Color the points based on the **Survived** column: **Red** for passengers who did not survive (**Survived = 0**). **Blue** for passengers who survived (**Survived = 1**).
3. Set the title of the plot to **"Age vs. Fare by Survival"**.
4. Label the x-axis as **'Age'** and the y-axis as **'Fare'**.

The Titanic dataset contains columns as shown below,

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

Sample Data:

Sample Test Cases +

Explorer | AgeFareS... | Submit | Debugger

```python
# # Write your code here for Scatter Plot for Age vs. Fare by Survived
import pandas as pd
import matplotlib.pyplot as plt

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

# Data Cleaning
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop('Cabin', axis=1, inplace=True)

# Convert categorical features to numeric
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)

# Write your code here for Scatter Plot for Age vs. Fare by Survived


plt.figure()
colors = {0: 'red', 1: 'blue'}
plt.scatter(data['Age'], data['Fare'], c=data['Survived'].apply(lambda x: colors[x]))
plt.title('Age vs. Fare by Survival')
plt.xlabel('Age')
```

Terminal | Test cases

Prev | Reset | Submit