1.1.1. Calculate Momentum

03:05

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("%.2f"%p,end='')
5  print("kgm/s")
```

Debugger

Terminal Test cases

1.1.2. Conditional Calculation Based on the Number of Digits

10:56

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  n=int(input())
2  if n>=0 and n<=10:
3      p = n*n
4      print(p)
5  elif n>=10 and n<=99:
6      q=n**0.5
7      print("%0.2f"%q)
8  elif n>=100 and n<=999:
9      r=n**(1/3)
10     print("%0.2f"%r)
11 else:
12     print("Invalid")
13
14
```

Terminal Test cases

< Prev Reset Submit Next >

1.1.3. Age and Salary Calculation

30:10

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

```
1  from datetime import datetime
2
3  def calculate_age(birthdate):
4      date_object = datetime.strptime(birthdate, "%d-%m-%Y")
5      today = datetime.today()
6      if ((today.month,today.day) < (date_object.month, date_object.day)):
7          age = today.year-date_object.year-((today.month,today.day)
   <(date_object.month,date_object.day))
8          return age
9      elif ((today.month,today.day)>(date_object.month,date_object.day)):
10         age = today.year-date_object.year-((today.month,today.day)>
   (date_object.month,date_object.day))
11         return age
12
13
14
15 def convert_salary_to_dollars(salary_in_rupees):
16     salary = salary_in_rupees*0.012
17     return salary
18
19
20 birthdate = input()
21 salary_in_rupees = float(input())
22 age = calculate_age(birthdate)
23 salary_in_dollars = convert_salary_to_dollars(salary_in_rupees)
24 print(f"Age: {age}")
25 print(f"Salary in dollars: {salary_in_dollars:.2f}")
```

Terminal Test cases

< Prev Reset Submit Next >

1.1.4. Reverse a Number

06:51

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  n=int(input())
2  reverse = 0
3  while n!=0:
4      digit = n%10
5      reverse = reverse*10 + digit
6      n = n//10
7
8  print(reverse)
```

Debugger

Terminal Test cases

1.1.5. Multiplication Table

04:40

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  n=int(input())
2  i = 1
3  while i <=10:
4      print(n,"x",i,"=",n*i)
5      i = i+1
6
```

Debugger

Terminal Test cases

CODETANTRA Home 202401090091@mitaoe.ac.in Support Logout

1.2.1. Pass or Fail

14:34

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:

- "Fail".

Sample Test Cases +

Explorer

passorFa...

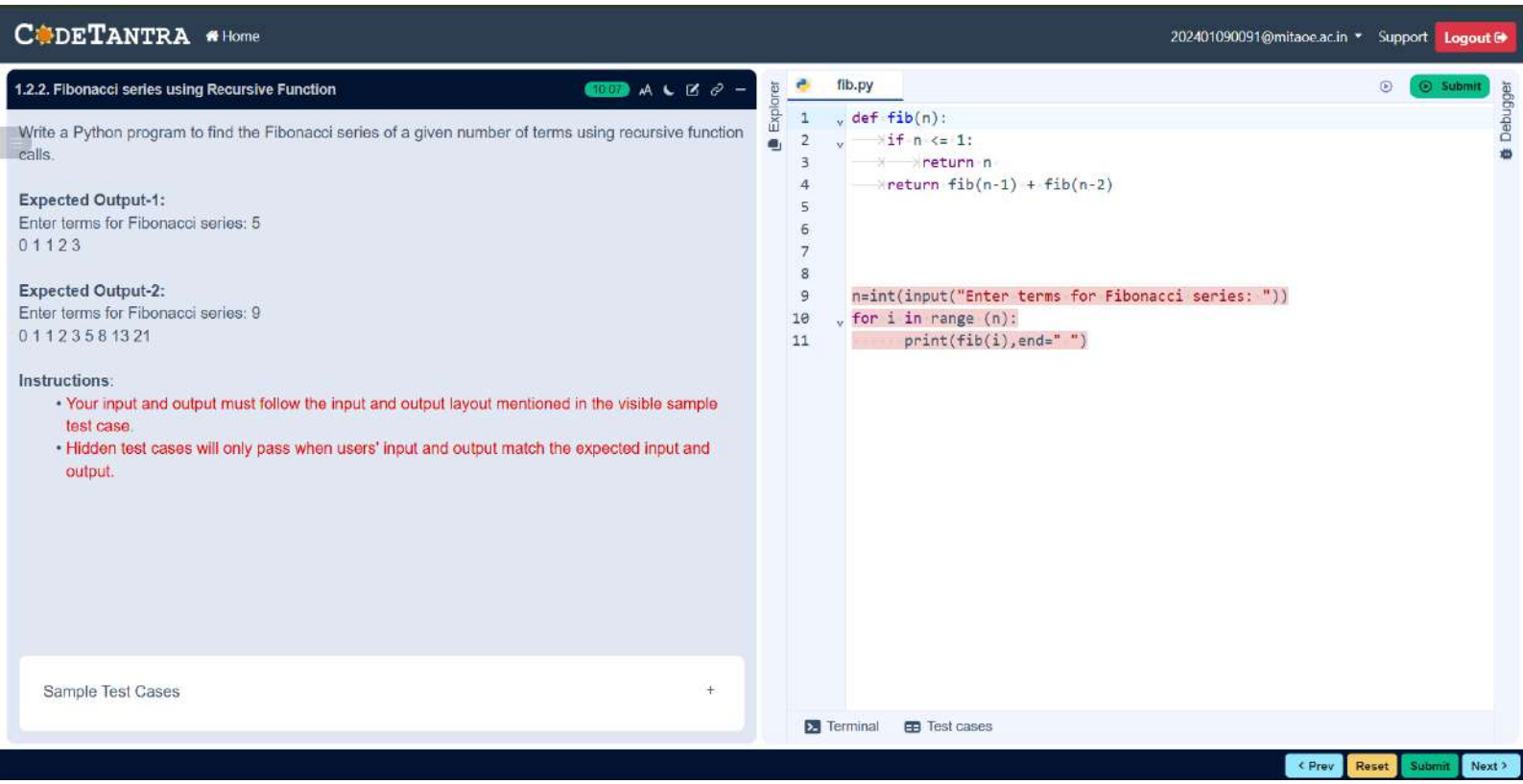
Submit

Debugger

```
1  def calculate():
2      courses = int(input())
3      marks = list(map(int,input().split()))
4
5      if min(marks)<40:
6          print ("Fail")
7          return
8      percentage = sum(marks)/courses
9      grade= "Distinction" if percentage>75 else\
10             "First Division" if percentage >60 else\
11             "Second Division" if percentage > 50 else "Third Division"
12     print(f"Aggregate Percentage: {percentage:.2f}\nGrade: {grade}")
13 calculate()
14
15
```

Terminal Test cases

< Prev Reset Submit Next >


CODETANTRA
Home
202401090091@mitaoe.ac.in
Support
Logout
1.2.2. Fibonacci series using Recursive Function
10:07
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
Debugger
1 def fib(n):
2 if n <= 1:
3 return n
4 return fib(n-1) + fib(n-2)
5
6
7
8
9 n=int(input("Enter terms for Fibonacci series: "))
10 for i in range (n):
11 print(fib(i),end=" ")
Terminal
Test cases
< Prev
Reset
Submit
Next >

1.2.4. Pattern - 2

07:57

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

```
1  n = int(input())
2  for i in range (1,n+1):
3      for j in range (1,i +1):
4          print(j,end=" ")
5      print()
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

2.1.1. List operations

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 +

listOps.py

```python
def menu_driven_program():
    lst = []  # Initialize an empty list
    while True:
        print("1. Add")
        print("2. Remove")
        print("3. Display")
        print("4. Quit")
        try:
            choice = int(input("Enter choice: "))
        except ValueError:
            print("Invalid choice")
            continue
        if choice == 1:
            try:
                num = int(input("Integer: "))
                lst.append(num)
                print(f"List after adding: {lst}")
            except ValueError:
                print("Invalid input")
        elif choice == 2:
            if lst:
                try:
                    num = int(input("Integer: "))
                    if num in lst:
                        lst.remove(num)
                        print(f"List after removing: {lst}")
                    else:
                        print("Element not found")
                except ValueError:
                    print("Invalid input")
            else:
                print("List is empty")
        elif choice == 3:
            if lst:
                print(lst)
            else:
                print("List is empty")
        elif choice == 4:
            break
        else:
            print("Invalid choice")

# Run the menu-driven program
menu_driven_program()
```

Terminal Test cases

2.1.2. Dictionary Operations

05:10

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

```python
print("Empty Dictionary:", dict_data)

# Ask the user how many items to add, then input key-value pairs
n = int(input("Number of items: "))
for _ in range(n):
    key = input("key: ")
    value = input("value: ")
    dict_data[key] = value
print("Dictionary:", dict_data)

# Ask the user to update a key's value
update_key = input("Enter the key to update: ")
if update_key in dict_data:
    new_value = input("Enter the new value: ")
    dict_data[update_key] = new_value
    print("Value updated")
else:
    print("Key not found")

# Retrieve and print a value using a key
retrieve_key = input("Enter the key to retrieve: ")
if retrieve_key in dict_data:
    print(f"Key: {retrieve_key}, Value: {dict_data[retrieve_key]}")
else:
    print("Key not found")

# Use get() to retrieve a value
get_key = input("Enter the key to get using the get() method: ")
value = dict_data.get(get_key, "Key not found")
if value == "Key not found":
    print(value)
else:
    print(f"Key: {get_key}, Value: {value}")

# Delete a key-value pair
deleted_key = input("Enter the key to delete: ")
if deleted_key in dict_data:
    del dict_data[deleted_key]
    print("Deleted")
else:
    print("Key not found")

# Display the updated dictionary
print("Updated Dictionary:", dict_data)
```

Terminal Test cases

< Prev Reset Submit Next >

2.2.1. Linear search Technique

07:15

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  arr = list(map(int,input().split()))
2  key = int(input())
3  for i in range (len(arr)):
4      if arr[i] == key:
5          print(i)
6          break
7
8  if arr[i] != key:
9      print("Not found")
10
```

Terminal Test cases

< Prev Reset Submit Next >

2.2.2. Captain of the Team

01:28

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

```
1  heights = list(map(int, input().split()))
2
3  captaain = max(heights)
4  print(captaain)
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

3.1.1. Numpy array operations

06:21

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

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

Terminal Test cases

< Prev Reset Submit Next >

3.2.1. Numpy: Matrix Operations

01:40

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.
3. The result of Element-wise Multiplication.
4. The result of Matrix Multiplication.

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
print(a)
```

Terminal Test cases

< Prev Reset Submit Next >

3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

03:10

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

```
1  import numpy as np
2
3  # Input matrices
4  print("Enter Array1:")
5  arr1 = np.array([list(map(int, input().split())) for i in range(3)])
6
7  print("Enter Array2:")
8  arr2 = np.array([list(map(int, input().split())) for i in range(3)])
9
10 # Perform horizontal stacking (hstack)
11 a = np.hstack((arr1,arr2))
12 print("Horizontal Stack:")
13 print(a)
14 b = np.vstack((arr1,arr2))
15 print("Vertical Stack:")
16 print(b)
17
18 # Perform vertical stacking (vstack)
19
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

3.2.3. Numpy: Custom Sequence Generation

01:31

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

customS...

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  a = np.arange(start,stop,step)
8  print(a)
9  # Generate the sequence using np.arange()
10
11 # Print the generated sequence
12
```

Terminal Test cases

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

05:52

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)))
    print("Element-wise Product:", ' '.join(map(str, prod_result)))

```

Terminal Test cases

3.2.5. Numpy: Copying and Viewing Arrays

01:00

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

02:30

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`.

Sample Test Cases +

Explorer

arrayOpe...

Submit

```
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)
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

3.2.7. Student Data Analysis and Operations

01:53

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.

Sample Test Cases +

Explorer

Operatio...

Submit

Debugger

```
1  import numpy as np
2
3  a = np.loadtxt("Sample.csv", delimiter=',', skiprows=1)
4
5  # 1. Print all student details
6  print("All student Details:\n",a    )
7
8  # 2. print total students
9
10 print("Total Students:", len(a))
11
12 # 3. Print all student Roll numbers
13 print("All Student Roll Nos", a[:,0]    )
14
15 # 4. Print subject 1 marks
16 print("Subject 1 Marks",a[:,1] )
17
18 # 5. print minimum marks of Subject 2
19 print("Min marks in Subject 2",np.min(a[:,2])      )
20
21 # 6. print maximum marks of Subject 3
22 print("Max marks in Subject 3",np.max(a[:,3])      )
23
24 # 7. Print All subject marks
25 print("All subject marks:",a[:,1:]       )
26
27 # 8. print Total marks of students
```

Terminal Test cases

< Prev Reset Submit Next >

4.1.1. Pandas - series creation and manipulation

02:50

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

```
1  import pandas as pd
2
3  # Take inputs from the user to create a list of numbers
4  numbers = list(map(int, input().split()))
5
6  # Create a Pandas series from the list of numbers
7  series = pd.Series(numbers)
8  # Grouping by even and odd numbers and calculating the mean
9  grouped = series.groupby(series%2==0).mean()
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
```

Terminal Test cases

4.1.2. Dictionary to dataframe

20:04

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
name=input("New name: ")
age = int(input("New age: "))
newrow = {'Name': name,'Age': age}
df = pd.concat([df,pd.DataFrame([newrow])],ignore_index = True)
# Display the DataFrame after adding a new row
print("After adding a row:\n",df)

# Modifying a row
index = int(input("Index of row to modify: "))
new_age_mod = int(input("New age: "))
df.loc[index,"Age"] = new_age_mod
# Display the DataFrame after modifying a row
print("After modifying a row:")
print(df)

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

Terminal Test cases

‹ Prev Reset Submit Next ›

4.1.3. Student Information

16:40

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

studentin... studentdat... Submit

```
1  import pandas as pd
2
3  # Read the text file into a DataFrame
4  file = input()
5  data = pd.read_csv(file, sep="\s+", header=None, names=["Name", "Age",
   "Grade"])
6  print("First five rows:")
7  print(data.head(5))
8  average = round(data['Age'].mean(),2)
9  print(f"Average age: {average}")
10 garde = 'B'
11 std = data[data['Grade']<=garde]
12 print("Students with a grade up to B")
13 print(std)
14 # write your code here..
15
16
```

Terminal Test cases

Debugger

4.2.1. Month with the Highest Total Sales

24:51

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:
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

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)
df['Date'] = pd.to_datetime(df['Date'])
df['Month'] = df['Date'].dt.to_period('M')
df['Total sales'] = df['Quantity'] * df['Price']
# Find the month with the highest total sales
monthly_sales = df.groupby('Month')['Total sales'].sum()
best_month = monthly_sales.idxmax()
highest_sales = monthly_sales.max()

print(f"Best month: {best_month}")
print(f"Total sales: ${highest_sales:.2f}")

```

Terminal Test cases

4.2.2. Best Selling Product

02:10

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

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)

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

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

```

Terminal Test cases

4.2.3. City that Sold the Most Products

01:25

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 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

```
1  import pandas as pd
2
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  sales = df.groupby('City')['Quantity'].sum()
9  # write the code..
10 best_city = sales.idxmax()
11 # Display the result
12 print(f"City sold the most products: {best_city}")
13
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

4.2.4. Most Frequently Sold Product Pairs

08:52

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

Sample Test Cases +

Explorer

frequentI... sales_dat... Submit

```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
date_products = {}
for date, group in df.groupby('Date'):
    products = group['Product'].unique()
    if len(products) > 1:
        date_products[date] = products

pair_counter = Counter()
for products in date_products.values():
    pairs = combinations(sorted(products),2)
    pair_counter.update(pairs)

if pair_counter:
    max_count = max(pair_counter.values())
    for pair,count in pair_counter.items():
        if count == max_count:
            print(f"{pair[0]} and {pair[1]}: {count} times")
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

4.2.5. Titanic Dataset Analysis and Data Cleaning

07:00

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 Test Cases +

Explorer

titanicDat...

Submit

Debugger

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

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

print(data.head())
print(data.tail())
print(data.shape)
print(data.info())
print(data.describe())
print(data.isnull().sum())
median_age = data['Age'].median()
data['Age'].fillna(median_age,inplace=True)
# 1. Display the first 5 rows of the dataset
mode_embarked = data['Age'].median()
data['Embarked'].fillna(mode_embarked,inplace = True)
data.drop('Cabin',axis = 1,inplace = True)
data['FamilySize'] = data['SibSp'] + data['Parch']
# 2. Display the last 5 rows of the dataset


# 3. Get the shape of the dataset


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

```

Terminal Test cases

< Prev Reset Submit Next >

5.2.3. Bar plot of survival rate of passengers

05:45

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,

Passengerid	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
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
```

Sample Test Cases +

Explorer

BarPlotOf... Submit

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  # Load the Titanic dataset
5  data = pd.read_csv('Titanic-Dataset.csv')
6
7  # Data Cleaning
8  data['Age'].fillna(data['Age'].median(), inplace=True)
9  data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10 data.drop('Cabin', axis=1, inplace=True)
11
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 # Write your code here for Bar Plot for Survival Rate
17 survival_counts = data['Survived'].value_counts()
18 survival_counts.plot(kind='bar')
19 plt.title('Survival Count')
20 plt.xlabel('Survived')
21 plt.ylabel('Count')
22 plt.show()
23
```

Debugger

Terminal Test cases

5.2.4. Bar Plot for Survival by Gender

04:39

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,

Passengerid	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

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)
plt.show()

```

Terminal Test cases

5.2.5. Bar Plot for Survival by Pclass

04:22

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:

```
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

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 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()


```

Terminal Test cases

< Prev Reset Submit Next >

5.2.6. Bar Plot for Survival by Embarked

04:48

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,

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

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 Embarked
grouped = data.groupby('Embarked_Q')
['Survived'].value_counts().unstack().fillna(0)
grouped.columns = ['Not Survived','Survived']
grouped.plot(kind = 'bar',stacked = True)
plt.title('Survival by Embarked')
plt.xlabel('Embarked')
plt.ylabel('Count')
plt.legend(title=None)
plt.show()
```

Terminal Test cases

5.2.7. Box plot for Age Distribution

02:26

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,

Passengerid	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
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
```

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()
```

Debugger

Terminal Test cases

< Prev Reset Submit Next >

5.2.8. Box Plot for Age by Survived

02:00

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,

Passengerid	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

BoxPlotF...

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 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

5.2.9. Box Plot for Fare by Pclass

02:09

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
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
```

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 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

Debugger

5.2.10. Scatter Plot for Age vs. Fare

12:27

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
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
```

Sample Test Cases +

AgeFareS... Submit

Explorer 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 Box Plot for Fare by Pclass
plt.figure(figsize=(6.4,4.8))
plt.scatter(data['Age'],data['Fare'])
plt.title('Age vs. Fare')
plt.xlabel('Age')
plt.ylabel('Fare')

plt.show()

```

Terminal Test cases

5.2.11. Scatter Plot for Age vs. Fare by Survived

40:27

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,

Passengerid	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

AgeFareS...

Submit

```python
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')
plt.ylabel('Fare')
plt.show()


```

Debugger

Terminal Test cases

Prev Reset Submit