



CODETANTRA

1.1.1. Calculate Momentum

12:26



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



calculateM...

Submit

```
1
2 m=float(input())
3 v=float(input())
4 p=m*v
5 print("%0.2f"%p,end='')
6 print("kgm/s")
7
```

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



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

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

+

birthDate...

Submit

```
1 from datetime import datetime
2
3 def calculate_age(birthdate):
4     date_object =
5         datetime.strptime(birthdate, "%d-%m-%Y")
6     today = datetime.today()
7     age = today.year -
8         date_object.year
9     if (today.month, today.day) <
10         (date_object.month, date_object.day):
11         age -= 1
12     return age
13
14 def
15     convert_salary_to_dollars(salary_in_r
16         upees):
17     return salary_in_rupees * 0.012
18
19 birthdate = input()
20 salary_in_rupees = float(input())
21 age = calculate_age(birthdate)
22 salary_in_dollars =
23     convert_salary_to_dollars(salary_in_r
24         upees)
25 print(f"Age: {age}")
26 print(f"Salary in dollars:
27     {salary_in_dollars:.2f}")
28
```

1.1.4. Reverse a Number

04/05



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



reverseNu...

Submit

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

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

+

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

Lab Assignment

About this unit

Lab Assignment

 **Pass or Fail**

Question



 **Fibonacci series using Recursive Function**

Question



 **Pattern - 1**

Question



 **Pattern - 2**

Question



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



```

1 n = int(input())
2 marks =
  list(map(int, input().split()))
3 if any(mark < 40 for mark in marks):
4     print("Fail")
5 else:
6     totalmarks = sum(marks)
7     aggregate =
      (totalmarks/(n*100))*100
8     print(f"Aggregate Percentage:
      {aggregate:.2f}")
9     if aggregate > 75:
10        print("Grade: Distinction")
11    elif aggregate > 60 and
      aggregate < 75:
12        print("Grade: First
      Division")
13    elif aggregate > 50 and
      aggregate < 60:
14        print("Grade: Second
      Division")
15    elif aggregate > 40 and
      aggregate < 50:
16        print("Grade: Third
      Division")
17    else:
18        print("Fail")

```

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



fib.py

```
1  def fib(n):
2      if n <= 0:
3          return 0
4      elif n == 1:
5          return 1
6      else:
7          return fib(n-1) + fib(n-2)
8
9
10
11
12
13
14
15
16
17
18  n=int(input("Enter terms for
19  Fibonacci series: "))
20  for i in range (n):
    print(fib(i),end=" ")
```


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



```
1 def print_triangle_pattern(rows):
2     i = 1
3     while i <= rows :
4         print("* " * i)
5         i += 1
6     rows = int(input(""))
7     print_triangle_pattern(rows)
8
```

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



```
1 def print_number_pattern(rows):
2     for i in range(1, rows + 1):
3         print(" ".join(map(str,
4                             range(1, i + 1))) + "\n")
5
6 rows = int(input().strip())
print_number_pattern(rows)
```

Practical 2

About this unit

Practical 2

Practice Lab Assignment

Unit • 100% completed



Lab Assignment

Unit • 100% completed



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



```

1  l = []
2  while True :
3      print("1. Add\n2. Remove\n3. Display\n4. Quit")
4      n = int(input("Enter choice: "))
5      if n == 1:
6          a = int(input("Integer: "))
7          l.append(a)
8          print(f"List after adding: {l}")
9      elif n == 2:
10         if len(l) != 0:
11             b = int(input("Integer: "))
12             if b in l:
13                 l.remove(b)
14                 print(f"List after removing: {l}")
15             else:
16                 print("Element not found")
17         else:
18             print("List is empty")
19         elif n == 3:
20             if len(l) == 0:
21                 print("List is empty")
22             else:
23                 print(l)
24         elif n==4:
25             break
26         else:
27             print("Invalid choice")

```

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



```
dictOperati... Submit
1 dict = {}
2 print("Empty Dictionary:", dict)
3
4 n = int(input("Number of items: "))
5 for _ in range(n):
6     key = input("key: ")
7     value = input("value: ")
8     dict[key] = value
9     print("Dictionary:", dict)
10
11 update_key = input("Enter the key to
update: ")
12 if update_key in dict:
13     new_value = input("Enter the new
value: ")
14     dict[update_key] = new_value
15     print("Value updated")
16 else:
17     print("Key not found")
18
19 retrieve_key = input("Enter the key
to retrieve: ")
20 if retrieve_key in dict:
21     print(f"Key: {retrieve_key},
Value: {dict[retrieve_key]}")
22 else:
23     print("Key not found")
24
25 get_key = input("Enter the key to
get using the get() method: ")
26 value = dict.get(get_key, "Key not
found")
27 if value != "Key not found":
28     print(f"Key: {get_key}, Value:
{value}")
29 else:
30     print(value)
31
32 deleted_key = input("Enter the key
to delete: ")
33 if deleted_key in dict:
34     del dict[deleted_key]
35     print("Deleted")
36 else:
37     print("Key not found")
38
39 print("Updated Dictionary:", dict)
```

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



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

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



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

Practical 3

About this unit

Practical 3

Practice Lab Assignment

Unit • 100% completed



Lab Assignment

Unit • 100% completed



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



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

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

Task:

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

1. **Addition** (`matrix_a + matrix_b`)
2. **Subtraction** (`matrix_a - matrix_b`)
3. **Element-wise Multiplication** (`matrix_a * matrix_b`)
4. **Matrix Multiplication** (`matrix_a · matrix_b`)
5. **Transpose of Matrix A**

Input Format:

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

Output Format:

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

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

Sample Test Cases



```

1 import numpy as np
2
3 # Input matrices
4 print("Enter Matrix A:")
5 matrix_a = np.array([list(map(int,
6 input().split())) for i in range(3)])
7
8 print("Enter Matrix B:")
9 matrix_b = np.array([list(map(int,
10 input().split())) for i in range(3)])
11
12 # Addition
13 print("Addition (A + B):")
14 print(matrix_a + matrix_b)
15 # Subtraction
16 print("Subtraction (A - B):")
17 print(matrix_a - matrix_b)
18 # Multiplication (element-wise)
19 print("Element-wise Multiplication
20 (A * B):")
21 print(matrix_a * matrix_b)
22 # Matrix multiplication (dot product)
23 print("A dot B:")
24 print(np.dot(matrix_a, matrix_b))
25 # Transpose
26 print("Transpose of A:")
27 print(matrix_a.T)

```

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



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

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



customSe...

Submit

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

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_1 OR B_1).

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



differentO...

Submit

```

1  import numpy as np
2
3  def array_operations(A, B):
4
5      # Convert A and B to NumPy arrays
6      A = np.array(A)
7      B = np.array(B)
8
9      # Arithmetic Operations
10     sum_result = A + B
11     diff_result = A - B
12     prod_result = A * B
13
14     # Statistical Operations
15     mean_A = np.mean(A)
16     median_A = np.median(A)
17     std_dev_A = np.std(A)
18
19     # Bitwise Operations
20     and_result = A & B
21     or_result = A | B
22     xor_result = A ^ B
23
24     # Output results with one space
25     # between each element
26     print("Element-wise Sum:", ' '.join(map(str, sum_result)))
27     print("Element-wise Difference:", ' '.join(map(str, diff_result)))
28     print("Element-wise Product:", ' '.join(map(str, prod_result)))
29     print(f"Mean of A: {mean_A}")
30     print(f"Median of A: {median_A}")
31     print(f"Standard Deviation of A: {std_dev_A}")
32
33     print("Bitwise AND:", ' '.join(map(str, and_result)))
34     print("Bitwise OR:", ' '.join(map(str, or_result)))
35     print("Bitwise XOR:", ' '.join(map(str, xor_result)))
36
37     A = list(map(int, input().split()))
38     # Elements of array A
39     B = list(map(int, input().split()))
40     # Elements of array B
41     array_operations(A, B)

```



CODETANTRA

3.2.5. Numpy: Copying and Viewing Arrays

01:56



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



copyAndvi...

Submit

```
1 import numpy as np  
2  
3 inputlist =  
   list(map(int, input().split(" ")))  
4  
5 # Original array  
6 original_array = np.array(inputlist)  
7  
8 # Create a view  
9 view_array = original_array.view()  
10  
11 # Create a copy  
12 copy_array = original_array.copy()  
13  
14 # Modify the view  
15 view_array[0] = 99  
16 print("Original array after  
   modifying view:", original_array)  
17 print("View array:", view_array)  
18  
19 # Modify the copy  
20 copy_array[1] = 88  
21 print("Original array after  
   modifying copy:", original_array)  
22 print("Copy array:", copy_array)  
23
```




CODETANTRA

3.2.6. Numpy: Searching, Sorting, Counting, ... 05:04

The given code in the editor takes a single array, array1, as space-separated integers as input from the user.

Additionally, it takes the following inputs:

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

You need to complete the code to perform the following operations:

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

Input Format:

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

Output Format:

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

Sample Test Cases



arrayOpera...

Submit

```

1  import numpy as np
2
3  # Input array from the user
4  array1 = np.array(list(map(int,
5  input().split()))))
6
7  # Searching
8  search_value = int(input("Value to
9  search: "))
10 count_value = int(input("Value to
11 count: "))
12 broadcast_value = int(input("Value
13 to add: "))
14
15 # Find indices where value matches
16 in array1
17 search_indices = np.where(array1 ==
18 search_value)[0]
19 print( search_indices)
20
21 # Count occurrences in array1
22 count_occurrence =
23 np.count_nonzero(array1 ==
24 count_value)
25 print( count_occurrence)
26
27 # Broadcasting addition
28 broadcast_result = array1 +
29 broadcast_value
30 print( broadcast_result)
31
32 # Sort the first array
33 sorted_array = np.sort(array1)
34 print( sorted_array)
35

```

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

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

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

Sample Test Cases

+

Operations...

Submit

```

1 import numpy as np
2
3 a = np.loadtxt("Sample.csv",
4               delimiter=',', skiprows=1)
5
6 # 1. Print all student details
7 print("All student Details:\n",a)
8
9 # 2. print total students
10 r,c=a.shape
11 print("Total Students:",r)
12
13 # 3. Print all student Roll numbers
14 print("All Student Roll Nos",a[:,0])
15
16 # 4. Print subject 1 marks
17 print("Subject 1 Marks",a[:,1])
18
19 # 5. print minimum marks of Subject 2
20 print("Min marks in Subject 2",
21       np.min(a[:,2]))
22
23 # 6. print maximum marks of Subject 3
24 print("Max marks in Subject 3",
25       np.max(a[:,3]))
26
27 # 7. Print All subject marks
28 print("All subject marks:",a[:,1:])
29
30 # 8. print Total marks of students
31 total_marks = np.sum(a[:, 1:], axis=1)
32 print("Total Marks", total_marks)
33
34 # 9. print average marks of each
35 student
36 avg=np.mean(a[:,1:],axis=1)
37 print(np.round(avg,1))
38
39 # 10. print average marks of each
40 subject
41 print("Average Marks of each
42 subject",np.mean(a[:, 1:],axis=0))
43
44 # 11. print average marks of S1 and
45 S2
46 print("Average Marks of S1 and
47 S2",np.mean(a[:, 1:3],axis=0))
48
49 # 12. print average marks of S1 and
50 S3
51 print("Average Marks of S1 and
52 S3",np.mean(a[:, [1, 3]],axis=0))
53
54 # 13. print Roll number who got
55 maximum marks in Subject 3
56 i=np.argmax(a[:,3])
57 print("Roll no who got maximum marks
58 in Subject 3",a[i,0])
59
60 # 14. print Roll number who got
61 minimum marks in Subject 2
62 mn=np.argmin(a[:,2])
63 print("Roll no who got minimum marks
64 in Subject 2",a[mn,0])
65
66 # 15. print Roll number who got 24
67 marks in Subject 2
68 whr=np.where(a[:, 2] == 24)
69 print("Roll no who got 24 marks in
70 Subject 2",a[whr,0])
71
72 # 16. print count of students who
73 got marks in Subject 1 < 40
74 ct=np.count_nonzero(a[:,1]<40)
75 print("Count of students who got
76 marks in Subject 1 < 40", ct)
77
78
79
80

```


Practical 4

About this unit

Practical 4

Practice Lab Assignment

Unit • 100% completed



Lab Assignment

Unit • 100% completed



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



```
1 import pandas as pd
2
3 # Take inputs from the user to
4 # create a list of numbers
5 numbers = list(map(int,
6 input().split()))
7
8 # Create a Pandas series from the
9 # list of numbers
10 series = pd.Series(numbers)
11 # Grouping by even and odd numbers
12 # and calculating the mean
13 grouped = series.groupby(series % 2
14 == 0).mean()
15
16 # Display the mean of even and odd
17 # numbers with labels
18 grouped.index = ['Even' if is_even
19 else 'Odd' for is_even in
20 grouped.index]
21 print("Mean of even and odd
22 numbers:")
23 print(grouped)
```

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



dataframe...

Submit

```

1 import pandas as pd
2
3 # Provided dictionary of lists
4 data = {
5     'Name': ['Alice', 'Bob',
6             'Charlie'],
7     'Age': [25, 30, 35],
8 }
9
10 # Convert the dictionary to a
11 # DataFrame
12 df = pd.DataFrame(data)
13
14 # Display the original DataFrame
15 print("Original DataFrame:")
16 print(df)
17
18 # Adding a new row
19 new_name = input("New name: ")
20 new_age = int(input("New age: "))
21 df.loc[len(df)] = [new_name, new_age]
22
23 # Display the DataFrame after adding
24 # a new row
25 print("After adding a row:\n", df)
26
27 # Modifying a row
28 mod_index = int(input("Index of row
29 to modify: "))
30 new_age = int(input("New age: "))
31 if 0 <= mod_index < len(df):
32     df.at[mod_index, 'Age'] = new_age
33 else:
34     print("Invalid index!")
35
36 # Display the DataFrame after
37 # modifying a row
38 print("After modifying a row:")
39 print(df)
40
41 # Deleting a row
42 del_index = int(input("Index of row
43 to delete: "))
44 if 0 <= del_index < len(df):
45     df = df.drop(del_index).reset_index(drop=True)
46 else:
47     print("Invalid index!")
48
49 # Display the DataFrame after
50 # deleting a row
51 print("After deleting a row:")
52 print(df)
53
54 # Adding a new column
55 genders = input("Enter genders
56 separated by space: ").split()
57 df['Gender'] = genders
58
59 # Display the DataFrame after adding
60 # a new column
61 print("After adding a new column:")
62 print(df)
63
64 # Modifying a column
65 df['Name'] = df['Name'].str.upper()
66
67 # Display the DataFrame after
68 # modifying a column
69 print("After modifying a column:")
70 print(df)
71
72 # Deleting a column
73 df.drop('Age', axis=1, inplace=True)
74
75 # Display the DataFrame after
76 # deleting a column

```

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



```
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+",
6 header=None, names=["Name", "Age",
7 "Grade"])
8 print("First five rows:")
9 print(data.head())
10 average_age =
11 round(data["Age"].mean(), 2)
12 print("Average age:", average_age)
13
14 print("Students with a grade up to
15 B")
16 filtered_students =
17 data[data["Grade"].isin(['A+', 'A',
18 'B'])]
19 print(filtered_students)
```

Lab Assignment

About this unit

Lab Assignment

</> Month with the Highest Total Sales

Question



</> Best Selling Product

Question



</> City that Sold the Most Products

Question



</> Most Frequently Sold Product Pairs

Question



</> Titanic Dataset Analysis and Data Cleaning

Question



</> Titanic Dataset Analysis and Data Cleaning - 2

Question



</> Titanic Dataset Analysis and Data Cleaning - 3

Question



</> Titanic Dataset Analysis and Data Cleaning - 4

Question



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



```
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
9 df['Date'] =
10 pd.to_datetime(df['Date'])
11 df['Month'] =
12 df['Date'].dt.to_period('M').astype(str)
13
14 df['Total'] = df['Quantity'] *
15 df['Price']
16
17 monthly_sales = df.groupby('Month')
18 ['Total'].sum()
19
20 # Find the month with the highest
21 total sales
22 best_month = monthly_sales.idxmax()
23 highest_sales = monthly_sales.max()
24
25 print(f"Best month: {best_month}")
26 print(f"Total sales:
27 ${highest_sales:.2f}")
```

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



```
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 product_sales = df.groupby('Product')
9 ['Quantity'].sum()
10 # Find the product with the highest
11 total quantity sold
12 best_product = product_sales.idxmax()
13 highest_quantity =
14 product_sales.max()
15
16 # Display the result
17 print(f"Best selling product:
18 {best_product}")
19 print(f"Total quantity sold:
20 {highest_quantity}")
```


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



```
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
9 # write the code..
10 city_sales = df.groupby('City')
11    ['Quantity'].sum()
12 best_city = city_sales.idxmax()
13
14 # Display the result
15 print(f"City sold the most products:
    {best_city}")
```


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

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

Sample Data:

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

Explanation:

Transactions:

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

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

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

Most Frequent Product Combinations:

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

Note:

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

Sample Test Cases



```
1 import pandas as pd
2 from itertools import combinations
3 from collections import Counter
4
5 # Prompt user to input the file name
6 file_name = input()
7
8 # Read data from the specified CSV
9 # file
10 df = pd.read_csv(file_name)
11
12 # write the code
13 date_products = {}
14
15 for date, group in
16 df.groupby('Date'):
17     products =
18     group['Product'].unique()
19     if len(products) > 1:
20         date_products[date] =
21         products
22
23 pair_counter = Counter()
24
25 for products in
26 date_products.values():
27     pairs =
28     combinations(sorted(products), 2)
29     pair_counter.update(pairs)
30
31 if pair_counter:
32     max_count =
33     max(pair_counter.values())
34
35 for pair, count in
36 pair_counter.items():
37     if count == max_count:
38         print(f"{pair[0]} and
39 {pair[1]}: {count} times")
40
41 else:
42     print("No product pairs found.")
43
44 # Output the most frequent product
45 pairs
```

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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,35,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,0,3,373450,8.0
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,41,0,0,330914,53.1
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,
```

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

Sample Test Cases



```
1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-
6 Dataset.csv')
7
8 # 1. Display the first 5 rows of the
9 dataset
10 print(data.head())
11
12 # 2. Display the last 5 rows of the
13 dataset
14 print(data.tail())
15
16 # 3. Get the shape of the dataset
17 print(data.shape)
18
19 # 4. Get a summary of the dataset
20 (info)
21 print(data.info())
22
23 # 5. Get basic statistics of the
24 dataset
25 print(data.describe())
26
27 # 6. Check for missing values
28 print(data.isnull().sum())
29
30
31 # 7. Fill missing values in the
32 'Age' column with the median age
33 median_age = data['Age'].median()
34 data['Age'].fillna(median_age,
35 inplace=True)
36
37 # 8. Fill missing values in the
38 'Embarked' column with the mode
39 mode_embarked =
40 data['Embarked'].mode()[0]
41 data['Embarked'].fillna(mode_embarked,
42 inplace=True)
43
44 # 9. Drop the 'Cabin' column due to
45 many missing values
46 data.drop('Cabin', axis=1,
47 inplace=True)
48
49 # 10. Create a new column
50 'FamilySize' by adding 'SibSp' and
51 'Parch'
52 data['FamilySize'] = data['SibSp'] +
53 data['Parch']
54
```

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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

```

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

```

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

Sample Test Cases

+

titanicData...

Submit

```

1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-
Dataset.csv')
6 data['FamilySize'] = data['SibSp'] +
data['Parch']
7
8 import pandas as pd
9 import numpy as np
10
11 # Load the Titanic dataset
12 data = pd.read_csv('Titanic-
Dataset.csv')
13 data['FamilySize'] = data['SibSp'] +
data['Parch']
14
15 data['Alone'] =
np.where(data['FamilySize'] == 0, 1,
0)
16
17 # 3. Convert 'Sex' to numeric (male:
0, female: 1)
18 data['Sex'] =
data['Sex'].map({'male': 0,
'female': 1})
19
20 # 4. One-hot encode the 'Embarked'
column, dropping the first category
21 data = pd.get_dummies(data, columns=
['Embarked'], drop_first=True)
22
23 # 6. Get the mean age of passengers
24 mean_age = data['Age'].mean()
25 print(mean_age)
26
27 # 7. Get the median fare of
passengers
28 median_fare = data['Fare'].median()
29 print(median_fare)
30
31 # 8. Get the number of passengers by
class
32 passengers_by_class =
data['Pclass'].value_counts()
33 print(passengers_by_class)
34
35 # 9. Get the number of passengers by
gender
36 passengers_by_gender =
data['Sex'].value_counts().sort_index
()
37 print(passengers_by_gender)
38
39 # 10. Get the number of passengers
by survival status
40 passengers_by_survival =
data['Survived'].value_counts().sort_
index()
41 print(passengers_by_survival)
42
43 # 11. Calculate the survival rate
44 survival_rate =
data['Survived'].mean()
45 print(survival_rate)
46
47 # 12. Calculate the survival rate by
gender
48 survival_rate_by_gender =
data.groupby('Sex')
['Survived'].mean()
49 print(survival_rate_by_gender)

```


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,

P	S	P	N	S	A	S	P	T	F	C	E
a	u	c	a	e	g	i	a	i	a	a	m
s	r	l	m	x		b	r	c	r	b	b
e	v	a	e			S	c	k	e	i	r
n	i	s				p	h	e		n	k
g	v										e
e	e										d
d	d										

Sample Data:

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

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

Sample Test Cases



titanicData...

```
1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-
Dataset.csv')
6 data['FamilySize'] = data['SibSp'] +
data['Parch']
7 data['IsAlone'] =
np.where(data['FamilySize'] > 0, 0,
1)
8 data = pd.get_dummies(data, columns=
['Embarked'], drop_first=True)
9
10 print(data.groupby('Pclass')
['Survived'].mean())
11
12 # 2. Calculate the survival rate by
embarked location
13 print(data.groupby('Embarked_S')
['Survived'].mean())
14
15 # 3. Calculate the survival rate by
family size
16 print(data.groupby('FamilySize')
['Survived'].mean())
17
18 # 4. Calculate the survival rate by
being alone
19 print(data.groupby('IsAlone')
['Survived'].mean())
20
21 # 5. Get the average fare by class
22 print(data.groupby('Pclass')
['Fare'].mean())
23
24 # 6. Get the average age by class
25 print(data.groupby('Pclass')
['Age'].mean())
26
27 # 7. Get the average age by survival
status
28 print(data.groupby('Survived')
['Age'].mean())
29
30 # 8. Get the average fare by
survival status
31 print(data.groupby('Survived')
['Fare'].mean())
32
33 # 9. Get the number of survivors by
class
34 print(data[data['Survived'] == 1]
['Pclass'].value_counts())
35
36 # 10. Get the number of non-
survivors by class
37 print(data[data['Survived'] == 0]
['Pclass'].value_counts())
```



CODETANTRA

4.2.8. Titanic Dataset Analysis and Data Cle...

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,

P	S	P	N	S	A	S	P	T	F	C	E
a	u	c	a	e	g	i	a	i	a	a	
s	r	l	m	x	e	b	r	c	r	b	
s	v	a	e			s	c	k	e		
e	i	s				p	h	e			
n	v										
g	e										
e	r										
r	i										
i	d										

Sample Data:

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

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

Sample Test Cases



```
1 import pandas as pd
2 import numpy as np
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-
Dataset.csv')
6 data = pd.get_dummies(data, columns=
['Embarked'], drop_first=True)
7
8
9 survivors_by_gender =
10 data[data['Survived'] == 1]
11 ['Sex'].value_counts()
12 print(survivors_by_gender)
13
14 # 2. Get the number of non-survivors
15 by gender
16 non_survivors_by_gender =
17 data[data['Survived'] == 0]
18 ['Sex'].value_counts()
19 print(non_survivors_by_gender)
20
21 # 3. Get the number of survivors by
22 embarked location
23 survivors_by_embarked_s =
24 data[data['Survived'] == 1]
25 ['Embarked_S'].value_counts()
26 print(survivors_by_embarked_s)
27
28 # 4. Get the number of non-survivors
29 by embarked location
30 non_survivors_by_embarked_s =
31 data[data['Survived'] == 0]
32 ['Embarked_S'].value_counts()
33 print(non_survivors_by_embarked_s)
34
35 # 5. Calculate the percentage of
36 children (Age < 18) who survived
37 children=data[data['Age'] < 18]
38 children_survival_rate=
39 children['Survived'].mean()
40 print(children_survival_rate)
41
42 # 6. Calculate the percentage of
43 adults (Age >= 18) who survived
44 adults= data[data['Age'] >=18]
45 adults_survival_rate=
46 adults['Survived'].mean()
47 print(adults_survival_rate)
48
49 # 7. Get the median age of survivors
50 median_age_survivors =
51 data[data['Survived'] == 1]
52 ['Age'].median()
53 print(median_age_survivors)
54
55 # 8. Get the median age of non-
56 survivors
57 median_age_non_survivors =
58 data[data['Survived'] == 0]
59 ['Age'].median()
60 print(median_age_non_survivors)
61
62 # 9. Get the median fare of survivors
63 median_fare_survivors =
64 data[data['Survived'] == 1]
65 ['Fare'].median()
66 print(median_fare_survivors)
67
68 # 10. Get the median fare of non-
69 survivors
70 median_fare_non_survivors =
71 data[data['Survived'] == 0]
72 ['Fare'].median()
73 print(median_fare_non_survivors)
```

Practical 5

About this unit

Practical 5

Practice Lab Assignment

Unit • 100% completed



Lab Assignment

Unit • 100% completed



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



```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3
4 # Data for Months and Temperature
5 # for three cities
6 data = {
7     'Month': ['January', 'February',
8               'March', 'April', 'May', 'June',
9               'July', 'August', 'September',
10              'October', 'November', 'December'],
11     'City_A_Temperature': [5, 7, 10,
12                            13, 17, 20, 22, 21, 18, 12, 8, 6],
13     'City_B_Temperature': [2, 3, 5,
14                            6, 10, 14, 16, 17, 12, 9, 5, 3],
15     'City_C_Temperature': [3, 4, 6,
16                            8, 9, 12, 15, 14, 10, 7, 4, 2]
17 }
18
19 # Write your code...
20 df = pd.DataFrame(data)
21 plt.stackplot(df['Month'], df['City_A_
22 Temperature'], df['City_B_Temperature'
23 ], df['City_C_Temperature'])
24 plt.title('Temperature Variation')
25 plt.xlabel('Month')
26 plt.ylabel('Temperature')
27 plt.show()
```


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:

Bar Plot (Pclass Distribution):

- Create a bar plot to show the distribution of passengers across the different passenger classes (`Pclass`).
- Use the color skyblue for the bars.
- Title the plot as "Passenger Class Distribution".
- Label the x-axis as "Pclass" and the y-axis as "Count".

Pie Chart (Gender Distribution):

- Create a pie chart to display the distribution of male and female passengers.
- Use lightblue for males and lightcoral for females.
- Include percentages on the slices (use `autopct='%1.1f%%'`).
- Title the plot as "Gender Distribution".

Histogram (Age Distribution):

- Create a histogram to visualize the distribution of passengers' ages.
- Use lightgreen for the bars with black edges (`edgecolor = 'black'`).
- Set the number of bins to 8 for the histogram.
- Title the plot as "Age Distribution".
- Label the x-axis as "Age" and the y-axis as "Frequency".

Bar Plot (Survival Count):

- Create a bar plot to show the count of passengers who survived and those who did not, based on the `Survived` column.
- Use the colors lightblue for survivors (1) and lightcoral for non-survivors (0).
- Title the plot as "Survival Count".
- Label the x-axis as "Survived (0 = No, 1 = Yes)" and the y-axis as "Count".

Scatter Plot (Fare vs Age):

- Create a scatter plot to visualize the relationship between the `Fare` and `Age` of passengers.
- Use orange for the data points.
- Title the plot as "Fare vs Age".
- Label the x-axis as "Age" and the y-axis as "Fare".

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

Sample Test Cases



```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Load the Titanic dataset from the CSV file
5 df = pd.read_csv('titanic.csv')
6
7 # Set up the figure for 5 subplots
8 fig, axes = plt.subplots(3, 2,
9                           figsize=(12, 12))
10
11 # write the code..
12 # Plot 1: Count of passengers by class
13 axes[0, 0].bar(df['Pclass'].value_counts().index, df['Pclass'].value_counts(),
14               color='skyblue')
15 axes[0, 0].set_title("Passenger Class Distribution")
16 axes[0, 0].set_xlabel("Pclass")
17 axes[0, 0].set_ylabel("Count")
18
19 # plot 2: Gender distribution
20 axes[0, 1].pie(df['Gender'].value_counts(),
21               labels=df['Gender'].value_counts().index, autopct='%1.1f%%', colors=
22               ['lightblue', 'lightcoral'])
23 axes[0, 1].set_title("Gender Distribution")
24
25 # plot 3: Age distribution
26 axes[1, 0].hist(df['Age'].dropna(),
27               bins=8, color='lightgreen',
28               edgecolor='black')
29 axes[1, 0].set_title("Age Distribution")
30 axes[1, 0].set_xlabel("Age")
31 axes[1, 0].set_ylabel("Frequency")
32
33 # plot 4: Survival count
34 axes[1, 1].bar(df['Survived'].value_counts().index,
35               df['Survived'].value_counts(), color=
36               ['lightblue', 'lightcoral'])
37 axes[1, 1].set_title("Survival Count")
38 axes[1, 1].set_xlabel("Survived (0 = No, 1 = Yes)")
39 axes[1, 1].set_ylabel("Count")
40
41 # plot 5: Fare vs Age
42 axes[2, 0].scatter(df['Age'],
43                   df['Fare'], color='orange',
44                   edgecolors='black')
45 axes[2, 0].set_title("Fare vs Age")
46 axes[2, 0].set_xlabel("Age")
47 axes[2, 0].set_ylabel("Fare")
48
49 plt.tight_layout()
50 plt.show()
51
52
53
54

```


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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

```

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

```

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

Sample Test Cases



```

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['Embarke
   d'].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 Histogram
17 plt.hist(data['Age'], bins=30,
   edgecolor='k')
18 plt.xlabel('Age')
19 plt.ylabel('Frequency')
20 plt.title('Age Distribution')
21 plt.show()

```

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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

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

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

Sample Test Cases

+

```
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['Embarke
d'].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
18 survival_counts =
data['Survived'].value_counts()
19 survival_counts.plot(kind='bar')
20 plt.title('Survival Count')
21 plt.xlabel('Survived')
22 plt.ylabel('Count')
23 plt.show()
```

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,

P	S	P	N	S	A	S	P	T	F	C	E
a	u	c	a	e	g	i	a	i	a	a	m
s	r	l	m	e	e	b	r	c	r	b	b
s	v	a	e			s	c	k	e	r	k
e	i	s	s			p	h	e		i	e
r	v										
i	e										
d	d										

Sample Data:

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

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

Sample Test Cases

+

```
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['Embarke
d'].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 by Gender
17
18
19 survival_by_gender =
data.groupby('Sex')
['Survived'].value_counts().unstack()
.fillna(0)
20 survival_by_gender.columns = ['Not
Survived', 'Survived']
21 survival_by_gender.index = ['0', '1']
22 survival_by_gender.plot(kind='bar',
stacked=True)
23 plt.title('Survival by Gender')
24 plt.xlabel('Gender')
25 plt.ylabel('Count')
26 plt.legend(title=None)
27 plt.show()
```

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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

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

Note:

- Refer to the visible test case for better reference.
- Ensure you use the **groupby()** function with **value_counts()** to count the survivors and non-survivors for each **Pclass**.
- Do **not** manually use **size()** or **unstack()** without **value_counts()**. Use the **value_counts()** method for counting survival status directly.

Sample Test Cases

+

```
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['Embarke
d'].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 by Pclass
17
18
19 survival_by_class =
data.groupby('Pclass')
['Survived'].value_counts().unstack()
.fillna(0)
20 survival_by_class.columns = ['Not
Survived', 'Survived']
21 survival_by_class.plot(kind='bar',
stacked=True)
22 plt.title('Survival by Pclass')
23 plt.xlabel('Pclass')
24 plt.ylabel('Count')
25 plt.legend(title=None)
26 plt.show()
```


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
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	9140	53.1		C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	9.58		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1		C
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male	0	0	0	330877	8.45		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.86		S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	34990	16.99		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	41	0	0	237733	16.99		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	5153	30.0		C

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, "9140", 53.1, , C
3, 1, 3, "Heikkinen, Miss. Laina", female, 26, 0, 0, "STON/O2. 3101282", 9.58, , S
4, 1, 1, "Futrelle, Mrs. Jacques Heath (Lily May Peel)", female, 35, 1, 0, "113803", 53.1, , C
5, 0, 3, "Allen, Mr. William Henry", male, 35, 0, 0, "373450", 8.05, , S
6, 0, 3, "Moran, Mr. James", male, 0, 0, 0, "330877", 8.45, , Q
7, 0, 1, "McCarthy, Mr. Timothy J", male, 54, 0, 0, "17463", 51.86, , S
8, 0, 3, "Palsson, Master. Gosta Leonard", male, 2, 3, 1, "34990", 16.99, , S
9, 1, 3, "Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)", female, 41, 0, 0, "237733", 16.99, , S
10, 1, 2, "Nasser, Mrs. Nicholas (Adele Achem)", female, 14, 1, 0, "5153", 30.0, , C
```

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

Sample Test Cases



```
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 by Embarked
17
18
19 grouped = data.groupby('Embarked_Q')
20 ['Survived'].value_counts().unstack().fillna(0)
21 grouped.columns = ['Not Survived', 'Survived']
22 grouped.plot(kind='bar', stacked=True)
23 plt.title('Survival by Embarked')
24 plt.xlabel('Embarked')
25 plt.ylabel('Count')
26 plt.legend(title=None)
27 plt.show()
```

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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

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

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

Sample Test Cases

+

```
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['Embarke
   d'].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 Box Plot
   for Age by Pclass
17
18
19 plt.figure(figsize=(8, 6))
20 data.boxplot(column='Age',
   by='Pclass')
21 plt.suptitle('')
22 plt.title('Age by Pclass')
23 plt.xlabel('Pclass')
24 plt.ylabel('Age')
25 plt.show()
```

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
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,34,1,0,STON/O2. 3101282,53.0
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,53.0
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,0,0,373450,8.0
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,0,0,330877,8.4583, Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990,79.2
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,41,0,0,34990,79.2
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,0,0,2101282,53.0
```

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

Sample Test Cases



```
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 Box Plot for Age by Survived
17
18
19 plt.figure(figsize=(8, 6))
20 data.boxplot(column='Age', by='Survived')
21 plt.suptitle('')
22 plt.title('Age by Survival')
23 plt.xlabel('Survived')
24 plt.ylabel('Age')
25 plt.show()
```


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,

P a s s e n g e r I d	S u r v i v e d	P c l a s s	N a m e	S e x	A g e	S i b S p	P a r c h	T i c k e t	F a r e	C a b i n	E m b a r k e d

Sample Data:

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

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

Sample Test Cases

+

```
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['Embarke
   d'].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 Box Plot
   for Fare by Pclass
17
18
19 plt.figure(figsize=(8, 6))
20 data.boxplot(column='Fare',
   by='Pclass')
21 plt.suptitle('')
22 plt.title('Fare by Pclass')
23 plt.xlabel('Pclass')
24 plt.ylabel('Fare')
25 plt.show()
```

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,

P	S	P	N	S	A	S	P	T	F	C	E
a	u	c	a	e	g	i	a	i	a	a	m
s	r	l	m	x	e	b	r	c	r	b	b
s	v	a	e			s	c	k	e	i	r
e	i	s				p	h	e		n	k
r	v										e
i	e										d
d	d										

Sample Data:

```

PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ti
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,34,1,0,STON/O2. 3
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,0,3,373450,8.0
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.0
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.86
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,34990
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,4,1,0,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,

```

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

Sample Test Cases



```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Load the Titanic dataset
5 data = pd.read_csv('Titanic-
6 Dataset.csv')
7
8 # Data Cleaning
9 data['Age'].fillna(data['Age'].median
10 (), inplace=True)
11 data['Embarked'].fillna(data['Embarke
12 d'].mode()[0], inplace=True)
13 data.drop('Cabin', axis=1,
14 inplace=True)
15
16 # Convert categorical features to
17 numeric
18 data['Sex'] =
19 data['Sex'].map({'male': 0,
20 'female': 1})
21 data = pd.get_dummies(data, columns=
22 ['Embarked'], drop_first=True)
23
24 # Write your code here for Box Plot
25 for Fare by Pclass
26
27 plt.figure(figsize=(6.4,4.8))
28 plt.scatter(data['Age'],data['Fare'])
29 plt.title('Age vs. Fare')
30 plt.xlabel('Age')
31 plt.ylabel('Fare')
32
33 plt.show()

```

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, 35, 1, 3, 33091, 53.0, , S
3, 1, 3, "Heikkinen, Miss. Laina", female, 26, 0, 0, STON/O2. 3101282, 51.0, , S
4, 1, 1, "Futrelle, Mrs. Jacques Heath (Lily May Peel)", female, 35, 0, 0, 3101538, 53.0, , S
5, 0, 3, "Allen, Mr. William Henry", male, 35, 0, 0, 373450, 8.0, , S
6, 0, 3, "Moran, Mr. James", male, 0, 0, 330877, 8.4583, , Q
7, 0, 1, "McCarthy, Mr. Timothy J", male, 54, 0, 0, 17463, 51.86, , S
8, 0, 3, "Palsson, Master. Gosta Leonard", male, 2, 3, 1, 34990, 53.0, , S
9, 1, 3, "Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)", female, 41, 0, 0, 3101282, 51.0, , S
10, 1, 2, "Nasser, Mrs. Nicholas (Adele Achem)", female, 14, 1, 1, 5149, 51.0, , S
```

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

Sample Test Cases

+

```
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 Scatter Plot for Age vs. Fare by Survived
17 colors = data['Survived'].map({0: 'red', 1: 'blue'})
18 plt.scatter(data['Age'], data['Fare'], c=colors)
19
20 # Set labels and title
21 plt.xlabel("Age")
22 plt.ylabel("Fare")
23 plt.title("Age vs. Fare by Survival")
24
25 # Show the plot
26 plt.show()
27
28
```