

EDS Laboratory Assignments

Name : Shreya Abhay Bhagat

Division : CS3

Roll No. : 83

PRN : 202401040020

Practical 1 Lab Assignment 1:

CODETANTRA Home 202401040020@mitaoe.ac.in Support Logout

1.1.1. Calculate Momentum

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)

Sample Test Cases +

calculate... Explorer

```
1 m = float(input(""))
2 v = float(input(""))
3 p = m * v
4 print("%.2fkgm/s"%p)
5
```

Submit Debugger screencast

< Prev Reset Next >

CODETANTRA Home 202401040020@mitaoe.ac.in Support Logout

1.1.2. Conditional Calculation Based on the Number of Digits

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 +

condition... Explorer

```
1 n = int(input(""))
2 if 0<=n<10:
3     print(n**2)
4 elif 10<=n<100:
5     print("%.2f"%n**(1/2))
6 elif 100<=n<1000 :
7     print("%.2f"%n**(1/3))
8 else :
9     print("Invalid")
```

Terminal Test cases

< Prev Reset Submit Next >

1.1.3. Age and Salary Calculation

29:43 A ☾ ☽ -

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:

INR → USD

Sample Test Cases

```
birthDate...
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     elif((today.month, today.day) > (date_object.month, date_object.day)):
9         age = today.year - date_object.year - ((today.month, today.day) > (date_object.month, date_object.day))
10    return age
11
12
13
14
15 def convert_salary_to_dollars(salary_in_rupees):
```

Terminal Test cases

< Prev Reset Submit Next >

1.1.3. Age and Salary Calculation

29:43 A ☾ ☽ -

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:

INR → USD

Sample Test Cases

```
birthDate...
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_in_dollars = salary_in_rupees * 0.012
17     return salary_in_dollars
18
19
20 birthdate = input()
21 salary_in_rupees = float(input())
22 age = calculate_age(birthdate)
23 salary_in_dollars =
24 convert_salary_to_dollars(salary_in_rupees)
25 print(f"Age: {age}")
26 print(f"Salary in dollars: {salary_in_dollars:.2f}")
```

Terminal Test cases

< Prev Reset Submit Next >

1.1.4. Reverse a Number

07:46 A ☾ ☽ -

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

```
reverseN...
1 n = int(input(""))
2 sum = 0
3 while(n>0):
4     rem = n%10
5     sum = sum*10 + rem
6     n = n//10
7 print(sum)
```

Terminal Test cases

< Prev Reset Submit Next >

1.1.5. Multiplication Table

16:05 AA ☾ ⚡ -

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

+

multiplica...

```
1 num = int(input(""))
2 for i in range(1,11):
3     print(f"{num} x {i} = {i*num}")
4
5
```



Submit



Debugger

screenrec

Terminal Test cases

< Prev

Reset

Submit

Next >

Practical 1 Lab Assignment 2 :

1.2.1. Pass or Fail

27:52 AA ☾ ⚡ -

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.

Sample Test Cases

+

passorFa...

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



Submit



Debugger

screenrec

Terminal Test cases

< Prev

Reset

Submit

Next >

1.2.2. Fibonacci series using Recursive Function

04:14 AA ☾ ⚡ -

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 output.

Sample Test Cases



fib.py

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

Terminal Test cases

< Prev Reset Submit Next >

1.2.3. Pattern - 1

25:50 AA ☾ ⚡ -

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



rightangl...

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

Terminal Test cases

< Prev Reset Submit Next >

1.2.4. Pattern - 2

01:30 AA ☾ ⚡ -

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



numberP...

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

Terminal Test cases

< Prev Reset Submit Next >

Practical 2 Lab Assignment 1:

CODETANTRA [Home](#)

2.1.1. List operations 51:43 A ☾ ↻

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".

Sample Test Cases +

```
listOps.py
1  a = []
2  v while 1:
3      v     print("1. Add")
4      v     print("2. Remove")
5      v     print("3. Display")
6      v     print("4. Quit")
7      v     c = int(input("Enter choice: "))
8      v     if(c==1):
9          v         i = int(input("Integer: "))
10         v         a.append(i)
11         v         print(f"List after adding: {a}")
12         v         elif(c==2):
13             v             if a:
14                 v                     i = int(input("Integer: "))
15                 v                     if (i in a):
16                     v                         a.remove(i)
17                     v                     print(f"List after removing: {a}")
18                 v                     else:
19                     v                         print("Element not found")
```

Terminal Test cases

Submit Reset Next > < Prev

CODETANTRA [Home](#)

2.1.1. List operations 51:43 A ☾ ↻

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".

Sample Test Cases +

```
listOps.py
15  v             if (i in a):
16      v                 a.remove(i)
17      v                 print(f"List after removing: {a}")
18  v             else:
19      v                 print("Element not found")
20  v             else:
21      v                 print("List is empty")
22  v             elif(c==3):
23      v                 if a:
24                      v                     print(a)
25                  v                 else:
26                      v                         print("List is empty")
27  v             elif(c == 4):
28                  v                     break
29  v             else:
30                  v                     print("Invalid choice")
```

Terminal Test cases

Submit Reset Next > < Prev

2.1.2. Dictionary Operations

21:19 A ☾ ☿ -

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



```
dictOpera...
1 # 1. Create an empty dictionary and display it
2 my_dict = {}
3 print("Empty Dictionary:", my_dict)
4
5 # 2. Ask the user how many items to add, then input key-
6 # value pairs
7 size = int(input("Number of items: "))
8 for _ in range(size):
9     key = input("key: ")
10    value = input("value: ")
11    my_dict[key] = value
12
13 # 3. Show the dictionary after adding items
14 print("Dictionary:", my_dict)
15 # 4. Update a key's value
16 key_to_update = input("Enter the key to update: ")
17 if key_to_update in my_dict:
18     new_value = input("Enter the new value: ")
19     my_dict[key_to_update] = new_value
20     print("Value updated")
21 else:
22     print("Key not found")
23
24 # 5. Retrieve and print a value using a key
25 key_to_access = input("Enter the key to retrieve: ")
26 if key_to_access in my_dict:
27     print(f"Key: {key_to_access}, Value: {my_dict[key_to_access]}")
28 else:
29     print("Key not found")
30
31 # 6. Use `get()` to retrieve a value
32 key_to_get = input("Enter the key to get using the get() method: ")
33 value = my_dict.get(key_to_get)
34 if value is not None:
35     print(f"Key: {key_to_get}, Value: {value}")
36 else:
37     print("Key not found")
38
39 # 7. Delete a key-value pair
40 key_to_delete = input("Enter the key to delete: ")
41 if key_to_delete in my_dict:
42     del my_dict[key_to_delete]
43     print("Deleted")
44 else:
45     print("Key not found")
46
47 # 8. Display the updated dictionary
48 print("Updated Dictionary:", my_dict)
```

Terminal Test cases

< Prev Reset Submit Next >

2.1.2. Dictionary Operations

21:19 A ☾ ☿ -

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



```
dictOpera...
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
```

Terminal Test cases

< Prev Reset Submit Next >

2.1.2. Dictionary Operations

21:19 A ☾ ☿ -

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



```
dictOpera...
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
```

Terminal Test cases

< Prev Reset Submit Next >

Practical 2 Assignment 2 :

CODETANTRA [Home](#)

2.2.1. Linear search Technique 19:35 A ☾ ⌂ ⌂ -

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 5 6

Sample Test Cases +

Code Editor (CTP1709...):

```
1 n = list(map(int,input().split()))
2 z = int(input())
3 v = False
4 for i in range(len(n)):
5     if n[i] == z:
6         print(i)
7         v = True
8         break
9 if not v:
10    print("Not found")
```

Terminal **Test cases** ◀ Prev Reset Submit Next ▶

CODETANTRA [Home](#)

2.2.2. Captain of the Team 03:25 A ☾ ⌂ ⌂ -

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 +

Code Editor (captainof...):

```
1 a = list(map(int,input().split()))
2 z = max(a)
3 print(z)
```

Terminal **Test cases** ◀ Prev Reset Submit Next ▶

Practical 3 Assignment 1 :

3.1.1. Numpy array operations

05:51 A ☾ ✎

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.

Sample Test Cases

+

numpyarr...

```

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

```

Terminal Test cases

< Prev Reset Submit Next >

Practical 3 Assignment 2 :

3.2.1. Numpy: Matrix Operations

24:46 A ☾ ✎

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:

- Addition** (matrix_a + matrix_b)
- Subtraction** (matrix_a - matrix_b)
- Element-wise Multiplication** (matrix_a * matrix_b)
- Matrix Multiplication** (matrix_a . matrix_b)
- Transpose of Matrix A**

Input Format:

- The user will input 3 rows for matrix_a, each containing 3 integers separated

Sample Test Cases

+

matrixOp...

```

1 import numpy as np
2
3 # Input matrices
4 print("Enter Matrix A:")
5 matrix_a = np.array([list(map(int, input().split())) for
6 i in range(3)])
7
8 print("Enter Matrix B:")
9 matrix_b = np.array([list(map(int, input().split())) for
10 i in range(3)])
11
12 # Addition
13 print("Addition (A + B):")
14 print(np.add(matrix_a,matrix_b))
15 # Subtraction
16 print("Subtraction (A - B):")
17 print(np.subtract(matrix_a,matrix_b))
18 # Multiplication (element-wise)

```

Terminal Test cases

< Prev Reset Submit Next >

3.2.1. Numpy: Matrix Operations

24:46 AA ☾ ⌂ ⌂ -

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

Sample Test Cases



matrixOp...

```

8  matrix_d = np.array([list(map(int, input().split())) for i in range(3)])
9
10
11 # Addition
12 print("Addition (A + B):")
13 print(np.add(matrix_a,matrix_b))
14 # Subtraction
15 print("Subtraction (A - B):")
16 print(np.subtract(matrix_a,matrix_b))
17 # Multiplication (element-wise)
18 print("Element-wise Multiplication (A * B):")
19 print(np.multiply(matrix_a,matrix_b))
20 # Matrix multiplication (dot product)
21 print("A dot B:")
22 print(np.dot(matrix_a,matrix_b))
23 # Transpose
24 print("Transpose of A:")
25 print(matrix_a.T)

```

Terminal Test cases

< Prev Reset Submit Next >

3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

11:42 AA ☾ ⌂ ⌂ -

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 3×3 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).

Sample Test Cases



stacking.py

```

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 print("Horizontal Stack:")
12 x = np.hstack((arr1,arr2))
13 print(x)
14 # Perform vertical stacking (vstack)
15 print("Vertical Stack:")
16 y=np.vstack((arr1, arr2))
17 print(y)

```

Terminal Test cases

< Prev Reset Submit Next >

3.2.3. Numpy: Custom Sequence Generation

04:26 AA ☾ ⌂ ⌂ -

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

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

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

Input Format:

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

Output Format:

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

Sample Test Cases



customS...

```

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

```

Terminal Test cases

< Prev Reset Submit Next >

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematic...

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: $A_i \text{ OR } B_i$).

Input Format:

- The first line contains space-separated integers representing the elements of

Sample Test Cases



```
different...
1 import numpy as np
2
3 def array_operations(A, B):
4     # Convert A and B to NumPy arrays
5     A = np.array(A)
6     B = np.array(B)
7
8     # Arithmetic Operations
9     sum_result = np.add(A,B)
10    diff_result = np.subtract(A,B)
11    prod_result = np.multiply(A,B)
12
13    # Statistical Operations
14    mean_A = np.mean(A)
15    median_A = np.median(A)
16    std_dev_A = np.std(A)
17
18    # Bitwise Operations
19    and_result = np.bitwise_and(A,B)
20    or_result = np.bitwise_or(A,B)
21    xor_result = np.bitwise_xor(A,B)
```

Terminal Test cases

< Prev Reset Submit Next >

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematic...

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: $A_i \text{ OR } B_i$).

Input Format:

- The first line contains space-separated integers representing the elements of

Sample Test Cases



```
different...
20 # Output results with one space between each element
21 print("Element-wise Sum:", ' '.join(map(str,
22 sum_result)))
23 print("Element-wise Difference:", ' '.join(map(str,
24 diff_result)))
25 print("Element-wise Product:", ' '.join(map(str,
26 prod_result)))
27
28
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
34 A = list(map(int, input().split())) # Elements of array A
35 B = list(map(int, input().split())) # Elements of array B
36 array_operations(A, B)
```

Terminal Test cases

< Prev Reset Submit Next >

3.2.5. Numpy: Copying and Viewing Arrays

14:58 AA ☾ ↻

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:

Sample Test Cases



```
copyAnd...
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)
```

Terminal Test cases

< Prev Reset Submit Next >

3.2.5. Numpy: Copying and Viewing Arrays

14:58 A ☾ ☽ -

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:

Sample Test Cases

```
copyAnd...
1 # Create a view
2 view_array = original_array.view()
3
4 # Create a copy
5 copy_array = original_array.copy()
6
7 # Modify the view
8 view_array[0] = 99
9 print("Original array after modifying view:", original_array)
10 print("View array:", view_array)
11
12 # Modify the copy
13 copy_array[1] = 88
14 print("Original array after modifying copy:", original_array)
15 print("Copy array:", copy_array)
```

Terminal Test cases

< Prev Reset Submit Next >

3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting

12:33 A ☾ ☽ -

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:

- Searching:** Find the indices where `search_value` appears in `array1` and print these indices.
- Counting:** Count how many times `count_value` appears in `array1` and print the count.
- Broadcasting:** Add `broadcast_value` to each element of `array1` using broadcasting, and print the resulting array.

Sample Test Cases

```
arrayOpe...
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 z = np.where(array1==search_value)
13 print(z[0])
14 # Count occurrences in array1
15 c = np.count_nonzero(array1 == count_value)
16 print(c)
17 # Broadcasting addition
18 broadcasted_array = array1 + broadcast_value
19 print(broadcasted_array)
```

Terminal Test cases

< Prev Reset Submit Next >

3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting

12:33 A ☾ ☽ -

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:

- Searching:** Find the indices where `search_value` appears in `array1` and print these indices.
- Counting:** Count how many times `count_value` appears in `array1` and print the count.
- Broadcasting:** Add `broadcast_value` to each element of `array1` using broadcasting, and print the resulting array.

Sample Test Cases

```
arrayOpe...
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 z = np.where(array1==search_value)
13 print(z[0])
14 # Count occurrences in array1
15 c = np.count_nonzero(array1 == count_value)
16 print(c)
17 # Broadcasting addition
18 broadcasted_array = array1 + broadcast_value
19 print(broadcasted_array)
20 # Sort the first array
21 sorted_arr = np.sort(array1)
22 print(sorted_arr)
```

Terminal Test cases

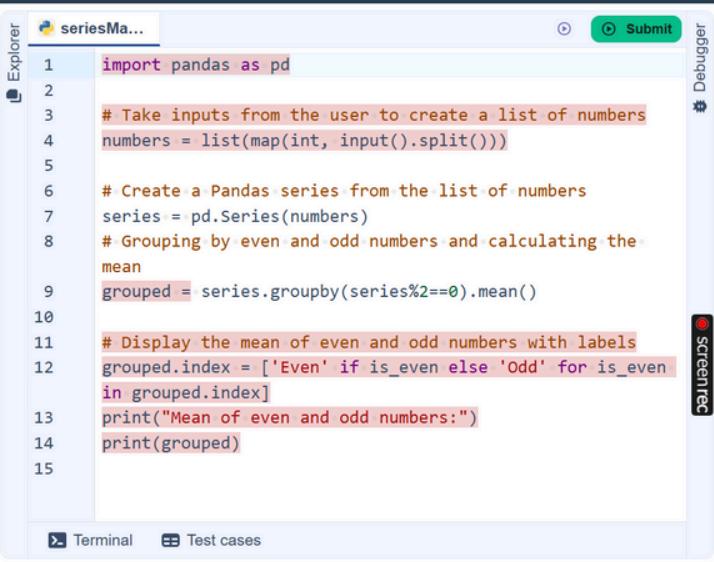
< Prev Reset Submit Next >

Practical 4 Assignment 1 :

CODETANTRA [Home](#)

202401040020@mitaoe.ac.in [Support](#) [Logout](#)

4.1.1. Pandas - series creation and manipulation

01:13 

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.

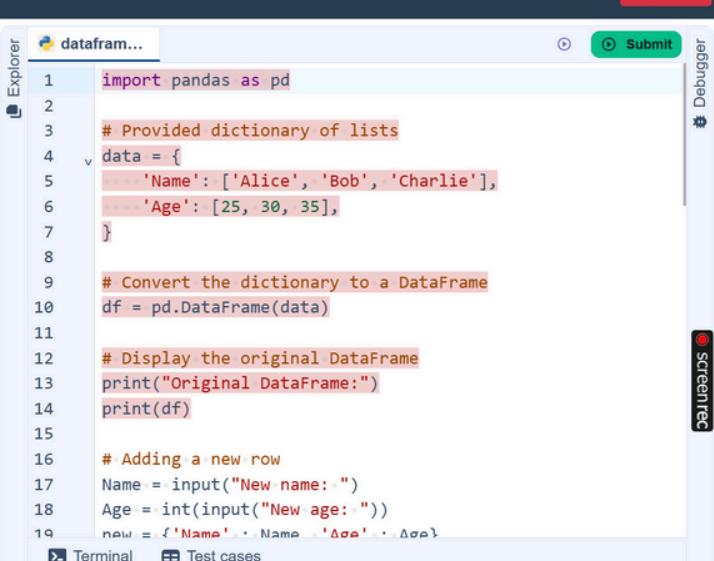
Sample Test Cases [+](#)

[Submit](#) [Reset](#) [Next >](#)

CODETANTRA [Home](#)

202401040020@mitaoe.ac.in [Support](#) [Logout](#)

4.1.2. Dictionary to dataframe

16:40 

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.

Sample Test Cases [+](#)

[Submit](#) [Reset](#) [Next >](#)

4.1.2. Dictionary to dataframe

16:40 AA ☾ ↻ -

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.

Sample Test Cases



datafram...

```

19 new = {'Name' : Name, 'Age' : Age}
20 df = df.append(new, ignore_index=True)
# Display the DataFrame after adding a new row
21 print("After adding a row:\n", df)
22 index=int(input("Index of row to modify: "))
23 age=int(input("New age: "))
24 df.at[index, 'Age']=age
# Modifying a row
25
26
27
28
29 # Display the DataFrame after modifying a row
30 print("After modifying a row:")
31 print(df)
32
33 # Deleting a row
34 delt=int(input("Index of row to delete: "))
35 df=df.drop(delt).reset_index(drop=True)
36
37 # Display the DataFrame after deleting a row

```

Terminal Test cases

< Prev Reset Submit Next >

4.1.2. Dictionary to dataframe

16:40 AA ☾ ↻ -

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.

Sample Test Cases



datafram...

```

36 # Display the DataFrame after deleting a row
37 print("After deleting a row:")
38 print(df)
39
40
41 # Adding a new column
42 gen=input("Enter genders separated by space: ").split()
43 df['Gender']=gen
44
45
46
47 # Display the DataFrame after adding a new column
48 print("After adding a new column:")
49 print(df)
50
51 # Modifying a column
52 df['Name']=df['Name'].str.upper()
53 # Display the DataFrame after modifying a column
54 print("After modifying a column:")

```

Terminal Test cases

< Prev Reset Submit Next >

4.1.2. Dictionary to dataframe

16:40 AA ☾ ↻ -

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.

Sample Test Cases



datafram...

```

44
45
46
47 # Display the DataFrame after adding a new column
48 print("After adding a new column:")
49 print(df)
50
51 # Modifying a column
52 df['Name']=df['Name'].str.upper()
53 # Display the DataFrame after modifying a column
54 print("After modifying a column:")
55 print(df)
56
57 # Deleting a column
58 df=df.drop(columns=['Age'])
59 # Display the DataFrame after deleting a column
60 print("After deleting a column:")
61 print(df)

```

Terminal Test cases

< Prev Reset Submit Next >

4.1.3. Student Information

05:50 AA ☾ ⌂ ⌃ -

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

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

Note:

Refer to the displayed test cases for better understanding.

Sample Test Cases

+

```
studentin... studentdat...
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=
6 ["Name", "Age", "Grade"])
7 print("First five rows:")
8 print(data.head())
9 avg = round(data["Age"].mean(),2)
10 print("Average age:",avg)
11 # write your code here..
12 filter = data[data['Grade'] <= 'B']
13 print("Students with a grade up to B")
14 print(filter)
```

Terminal Test cases

< Prev Reset Submit Next >

Practical 4 Assignment 2 :

4.2.1. Month with the Highest Total Sales

07:53 AA ☾ ⌂ ⌃ -

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
2025-01-02,Product B,2,15,Chicago
```

Sample Test Cases

```
monthFor... sales_dat...
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 df['Total_Sales']=df['Quantity']*df['Price']
9 df['Date']=pd.to_datetime(df['Date'])
10 df['Month']=df['Date'].dt.to_period('M')
11 monthly_sales=df.groupby('Month')['Total_Sales'].sum()
12 # Find the month with the highest total sales
13 best_month = monthly_sales.idxmax()
14 highest_sales = monthly_sales.max()
15 print(f"Best month: {best_month}")
16 print(f"Total sales: ${highest_sales:.2f}")
```

Terminal Test cases

< Prev Reset Submit Next >

4.2.2. Best Selling Product

23:52 A ☾ ☽ -

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
2025-01-02,Product B,2,15,Chicago
```

Sample Test Cases

+

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

Terminal Test cases

< Prev Reset Submit Next >

4.2.3. City that Sold the Most Products

05:58 A ☾ ☽ -

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
2025-01-02,Product B,2,15,Chicago
```

Sample Test Cases

+

```
monthFor... sales_dat...
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 city_wise_sell=df.groupby('City')
9 ['Quantity'].sum().reset_index()
10
11 # write the code..
12 best_city=city_wise_sell.loc[city_wise_sell['Quantity'].id
13 xmax()]
14 """
15 # Display the result
16 print(f"City sold the most products: {best_city}")"
17
18 print(f"City sold the most products: {best_city['City']}")"
```

Terminal Test cases

< Prev Reset Submit Next >

4.2.4. Most Frequently Sold Product Pairs

10:12 A ☾ ☽ -

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
```

Sample Test Cases

+

```
frequentl... sales_dat...
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 file
9 df = pd.read_csv(file_name)
10
11 # write the code
12 grouped=df.groupby('Date')['Product'].apply(list)
13 # create list
14 product_combinations=[]
15 v for products in grouped:
16     -->
17         product_combinations.extend(combinations(sorted(set(products)),2))
18
19 combinations_count=Counter(product_combinations)
```

Terminal Test cases

< Prev Reset Submit Next >

4.2.4. Most Frequently Sold Product Pairs

10:12 AA ☾ ⚡ -

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
```

Sample Test Cases

```
frequentl... sales_dat...
8 # Read data from the specified CSV file
9 df = pd.read_csv(file_name)
10
11 # write the code
12 grouped=df.groupby('Date')['Product'].apply(list)
13 # create list
14 product_combinations=[]
15 for products in grouped:
16     product_combinations.extend(combinations(sorted(set(products)),2))
17 combinations_count=Counter(product_combinations)
18 max_count=combinations_count.most_common(1)[0][1]
19 # Output the most frequent product pairs
20 for combo, count in combinations_count.items():
21     if count==max_count:
22         print(f'{combo[0]} and {combo[1]}: {count} times')
23
```

Terminal Test cases

< Prev Reset Submit Next >

4.2.5. Titanic Dataset Analysis and Data Cleaning

23:58 AA ☾ ⚡ -

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.

- Display the first 5 rows of the dataset.
- Display the last 5 rows of the dataset.
- Get the shape of the dataset (number of rows and columns).
- Get a summary of the dataset (using .info()).
- Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
- Check for missing values and display the count of missing values for each column.
- Fill missing values in the 'Age' column with the median age.
- Fill missing values in the 'Embarked' column with the most frequent value.

Sample Test Cases

```
titanicDat...
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
7 # 1. Display the first 5 rows of the dataset
8 print(data.head())
9
10 # 2. Display the last 5 rows of the dataset
11 print(data.tail())
12
13 # 3. Get the shape of the dataset
14 print(data.shape)
15
16 # 4. Get a summary of the dataset (info)
17 print(data.info())
18
19 # 5. Get basic statistics of the dataset
```

Terminal Test cases

< Prev Reset Submit Next >

4.2.5. Titanic Dataset Analysis and Data Cleaning

23:58 AA ☾ ⚡ -

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.

- Display the first 5 rows of the dataset.
- Display the last 5 rows of the dataset.
- Get the shape of the dataset (number of rows and columns).
- Get a summary of the dataset (using .info()).
- Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
- Check for missing values and display the count of missing values for each column.
- Fill missing values in the 'Age' column with the median age.
- Fill missing values in the 'Embarked' column with the most frequent value.

Sample Test Cases

```
titanicDat...
22 # 6. Check for missing values
23 print(data.isnull().sum())
24
25 # 7. Fill missing values in the 'Age' column with the median age
26 data['Age'].fillna(data['Age'].median(), inplace=True)
27
28 # 8. Fill missing values in the 'Embarked' column with the mode
29 data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
30
31 # 9. Drop the 'Cabin' column due to many missing values
32 data.drop('Cabin', axis=1, inplace=True)
33
34 # 10. Create a new column 'FamilySize' by adding 'SibSp' and 'Parch'
35 data['FamilySize']=data['SibSp']+ data['Parch']
```

Terminal Test cases

< Prev Reset Submit Next >

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

23:16 AA ☾ ⌂ ⌂ -

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.

Sample Test Cases



titanicDat...

```

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 # 1. Create a new column 'IsAlone' (1 if alone, 0 otherwise)
9 data['IsAlone']=np.where(data['FamilySize']>0,0,1)
10
11 # 2. Convert 'Sex' to numeric (male: 0, female: 1)
12 data['Sex']=data['Sex'].map({'male':0,'female':1})
13
14 # 3. One-hot encode the 'Embarked' column
15 data=pd.get_dummies(data,columns=['Embarked'],drop_first=True)
16
17 # 4. Get the mean age of passengers

```

Terminal Test cases

< Prev Reset Submit Next >

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

23:16 AA ☾ ⌂ ⌂ -

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.

Sample Test Cases



titanicDat...

```

17 # 4. Get the mean age of passengers
18 print(data['Age'].mean())
19
20
21 # 5. Get the median fare of passengers
22 print(data['Fare'].median())
23
24
25 # 6. Get the number of passengers by class
26 print(data['Pclass'].value_counts())
27
28
29 # 7. Get the number of passengers by gender
30 print(data['Sex'].value_counts())
31
32
33 # 8. Get the number of passengers by survival status
34 print(data['Survived'].value_counts())
35
36
37
38
39
40
41
42
43
44

```

Terminal Test cases

< Prev Reset Submit Next >

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

23:16 AA ☾ ⌂ ⌂ -

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.

Sample Test Cases



titanicDat...

```

26 print(data['Pclass'].value_counts())
27
28
29 # 7. Get the number of passengers by gender
30 print(data['Sex'].value_counts())
31
32
33 # 8. Get the number of passengers by survival status
34 print(data['Survived'].value_counts())
35
36
37 # 9. Calculate the survival rate
38 print(data['Survived'].mean())
39
40
41 # 10. Calculate the survival rate by gender
42 print(data.groupby('Sex')['Survived'].mean())
43
44

```

Terminal Test cases

< Prev Reset Submit Next >

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3

21:27 AA ☾

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).

Sample Test Cases



titanicDat...

```

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'],
9 drop_first=True)
10
11 # 1. Calculate the survival rate by class
12 print(data.groupby('Pclass')['Survived'].mean())
13
14 # 2. Calculate the survival rate by embarked location
15 print(data.groupby('Embarked_S')['Survived'].mean())
16
17 # 3. Calculate the survival rate by family size
18 print(data.groupby('FamilySize')['Survived'].mean())
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

```

Terminal

Test cases

< Prev

Reset

Submit

Next >

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3

21:27 AA ☾

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).

Sample Test Cases



titanicDat...

```

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
37
38

```

Terminal

Test cases

< Prev

Reset

Submit

Next >

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3

21:27 AA ☾

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).

Sample Test Cases



titanicDat...

```

20 print(data.groupby('IsAlone')['Survived'].mean())
21
22 # 5. Get the average fare by class
23 print(data.groupby('Pclass')['Fare'].mean())
24
25 # 6. Get the average age by class
26 print(data.groupby('Pclass')['Age'].mean())
27
28 # 7. Get the average age by survival status
29 print(data.groupby('Survived')['Age'].mean())
30
31 # 8. Get the average fare by survival status
32 print(data.groupby('Survived')['Fare'].mean())
33
34 # 9. Get the number of survivors by class
35 print(data[data['Survived']==1]['Pclass'].value_counts())
36
37 # 10. Get the number of non-survivors by class
38 print(data[data['Survived']==0]['Pclass'].value_counts())

```

Terminal

Test cases

< Prev

Reset

Submit

Next >

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

13:07 A ☾ ☽ -

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.

Sample Test Cases

+

titanicDat...

```

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'],
7 drop_first=True)
8
9 # 1. Get the number of survivors by gender
10 print(data[data['Survived']==1]['Sex'].value_counts())
11
12 # 2. Get the number of non-survivors by gender
13 print(data[data['Survived']==0]['Sex'].value_counts())
14
15 # 3. Get the number of survivors by embarked location
16 print(data[data['Survived']==1]
17 ['Embarked_S'].value_counts())

```

Terminal Test cases

< Prev Reset Submit Next >

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

13:07 A ☾ ☽ -

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.

Sample Test Cases

+

titanicDat...

```

18 # 4. Get the number of non-survivors by embarked location
19 print(data[data['Survived']==0]
20 ['Embarked_S'].value_counts())
21
22 # 5. Calculate the percentage of children (Age < 18) who
23 survived
24 children=data[data['Age']<18]
25 print(children['Survived'].mean())
26 # 6. Calculate the percentage of adults (Age >= 18) who
27 survived
28 adults=data[data['Age']>=18]
29 print(adults['Survived'].mean())
30 # 7. Get the median age of survivors
31 print(data[data['Survived']==1]['Age'].median())
32
33 # 8. Get the median age of non-survivors
34 print(data[data['Survived']==0]['Age'].median())
35
36 # 9. Get the median fare of survivors
37 print(data[data['Survived']==1]['Fare'].median())
38
39 # 10. Get the median fare of non-survivors
40 print(data[data['Survived']==0]['Fare'].median())

```

Terminal Test cases

< Prev Reset Submit Next >

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

13:07 A ☾ ☽ -

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.

Sample Test Cases

+

titanicDat...

```

22 survived
23 children=data[data['Age']<18]
24 print(children['Survived'].mean())
25 # 6. Calculate the percentage of adults (Age >= 18) who
26 survived
27 adults=data['Age']>=18
28 print(adults['Survived'].mean())
29 # 7. Get the median age of survivors
30 print(data[data['Survived']==1]['Age'].median())
31
32 # 8. Get the median age of non-survivors
33 print(data[data['Survived']==0]['Age'].median())
34
35 # 9. Get the median fare of survivors
36 print(data[data['Survived']==1]['Fare'].median())
37
38 # 10. Get the median fare of non-survivors
39 print(data[data['Survived']==0]['Fare'].median())

```

Terminal Test cases

< Prev Reset Submit Next >

Practical 5 Assignment 1:

CODETANTRA [Home](#)

202401040020@mitaoe.ac.in [Support](#) [Logout](#)

5.1.1. Stacked Plot

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

Your task is to:

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

Sample Test Cases +

Explorer **stackedpl...**

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

Terminal Test cases

< Prev Reset Submit Next >

CODETANTRA [Home](#)

202401040020@mitaoe.ac.in [Support](#) [Logout](#)

5.1.1. Stacked Plot

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

Your task is to:

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

Sample Test Cases +

Explorer **stackedpl...**

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

Terminal Test cases

< Prev Reset Submit Next >

Practical 5 Assignment 2:

5.2.2. Histogram of passenger information of Titanic

03:01 A ☽ ☁ ☀ -

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	S	Pc	N	S	A	Si	P	Ti	Fa	C	E
as	ur	la	a	m	g	b	ar	ck	re	ab	m
se	vi	ss	n	e	e	s	ch	et		in	ba
ng	ve		m	x	p	p				ba	rk
erl	d		e	e						rk	ed

Sample Test Cases



Histogram...

```
# 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 Histogram
plt.figure()
plt.hist(data['Age'], bins=30, edgecolor='k')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.3. Bar plot of survival rate of passengers

02:30 A ☽ ☁ ☀ -

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	S	Pc	N	S	A	Si	P	Ti	Fa	C	E
as	ur	la	a	m	g	b	ar	ck	re	ab	m
se	vi	ss	n	e	e	s	ch	et		in	ba
ng	ve		m	x	p	p				ba	rk
erl	d		e	e						rk	ed

Sample Test Cases



BarPlotOf...

```
# 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 Rate
plt.figure()
data['Survived'].value_counts().plot(kind='bar')
plt.title('Survival Count')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.show()
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.4. Bar Plot for Survival by Gender

03:33 A ☽ ☁ ☀ -

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	N		Si						E
as	ur	a		m						m
se	vi	ss		e						ba
ng	ve			x						rk
erl	d			e						ed

Sample Test Cases



BarPlotOf...

```
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
plt.figure()
data.groupby('Sex')[['Survived']].value_counts().unstack().plot(kind='bar', stacked=True)
plt.title('Survival by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(['Not Survived', 'Survived'])
plt.show()
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.5. Bar Plot for Survival by Pclass

05:30 AA ☾ ☽ -

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	S	Pc	N	S	A	Si	P	Ti	Fa	C	E	m
as	ur	ce	a	s	ge	b	ar	ck	re	ab	mb	rk
se	vi	la	n	ex	sp	si	ch	et	re	in	ba	rd

Sample Test Cases

+

BarPlotOf...

```

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'],
15 drop_first=True)
16
17 # Write your code here for Bar Plot for Survival by Pclass
18 plt.figure()
19 data.groupby('Pclass')[['Survived']].value_counts().unstack().plot(kind='bar',
20 stacked=True)
21 plt.title('Survival by Pclass')
22 plt.xlabel('Pclass')
23 plt.ylabel('Count')
24 plt.legend(['Not Survived', 'Survived'])
25 plt.show()

```

Terminal Test cases

< Prev Reset Submit Next >

5.2.6. Bar Plot for Survival by Embarked

03:20 AA ☾ ☽ -

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,

Sample Test Cases

+

BarPlotOf...

```

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'],
15 drop_first=True)
16
17 # Write your code here for Bar Plot for Survival by Embarked
18 plt.figure()
19 data.groupby('Embarked_Q')[['Survived']].value_counts().unstack().plot(kind='bar',
20 stacked=True)
21 plt.title('Survival by Embarked')
22 plt.xlabel('Embarked')
23 plt.ylabel('Count')
24 plt.legend(['Not Survived', 'Survived'])
25 plt.show()

```

Terminal Test cases

< Prev Reset Submit Next >

5.2.7. Box plot for Age Distribution

02:09 AA ☾ ☽ -

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	S	Pc	N	S	A	Si	P	Ti	Fa	C	E	m
as	ur	ce	a	s	ge	b	ar	ck	re	ab	mb	rk
se	vi	la	n	ex	sp	si	ch	et	re	in	ba	rd

Sample Test Cases

+

BoxPlotF...

```

9 data['EMBARKED'].fillna(data['EMBARKED'].mode(),
10 inplace=True)
11 data.drop('Cabin', axis=1, inplace=True)
12
13 # Convert categorical features to numeric
14 data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
15 data = pd.get_dummies(data, columns=['Embarked'],
16 drop_first=True)
17
18 # Write your code here for Box Plot for Age by Pclass
19 plt.figure()
20 data.boxplot(column='Age', by='Pclass')
21 plt.title('Age by Pclass')
22 plt.suptitle('')
23 plt.xlabel('Pclass')
24 plt.ylabel('Age')
25 plt.show()

```

Terminal Test cases

< Prev Reset Submit Next >

5.2.8. Box Plot for Age by Survived

02:16 A ☾ ☽ -

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,

P	S	Pc	N	S	A	Si	P	Ti	Fa	C	E
as	ur	la	a	ex	ge	b	ar	ck	re	ab	m
se	vi	l	m			s					ba
ng

Sample Test Cases

+

BoxPlotF...

```
9 data[embarked].fillna(data[embarked].mode()[0], inplace=True)
10 data.drop('Cabin', axis=1, inplace=True)

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

# Write your code here for Box Plot for Age by Survived
16 plt.figure()
17 data.boxplot(column='Age', by='Survived')
18 plt.title('Age by Survival')
19 plt.suptitle('')
20 plt.xlabel('Survived')
21 plt.ylabel('Age')
22 plt.show()
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.9. Box Plot for Fare by Pclass

02:04 A ☾ ☽ -

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	S	Pc	N	S	A	Si	P	Ti	Fa	C	E
as	ur	la	a	ex	ge	b	ar	ck	re	ab	m
se	vi	l	m	e	p	s	p	ch	et	in	ba
ng

Sample Test Cases

+

BoxPlotF...

```
9 data[embarked].fillna(data[embarked].mode()[0], inplace=True)
10 data.drop('Cabin', axis=1, inplace=True)

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

# Write your code here for Box Plot for Fare by Pclass
16 plt.figure()
17 data.boxplot(column='Fare', by='Pclass')
18 plt.title('Fare by Pclass')
19 plt.suptitle('')
20 plt.xlabel('Pclass')
21 plt.ylabel('Fare')
22 plt.show()
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.10. Scatter Plot for Age vs. Fare

02:52 A ☾ ☽ -

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	Pc	N	S	A	Si	P	Ti	Fa	C	E
as	ur	la	a	ex	ge	b	ar	ck	re	ab	m
se	vi	l	m	e	p	s	p	ch	et	in	ba
ng

Sample Test Cases

+

AgeFareS...

```
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)

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

# Write your code here for Box Plot for Fare by Pclass
16 plt.figure()
17 plt.scatter(data['Age'], data['Fare'])
18 plt.title('Age vs. Fare')
19 plt.xlabel('Age')
20 plt.ylabel('Fare')
21 plt.show()
```

Terminal Test cases

< Prev Reset Submit Next >

5.2.11. Scatter Plot for Age vs. Fare by Survived

03:22 AA ☺ ☰ -

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,

P	S	Pc	N	S	A	Si	P	Ti	Fa	C	E
as	ur	la	a	ex	ge	b	ar	ck	re	ab	m
se	vi	...	m	...	s	s	ba
ng

Sample Test Cases

+

Explorer  AgeFareS...

```
11 # Convert categorical features to numeric
12 data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
13 data = pd.get_dummies(data, columns=['Embarked'],
14 drop_first=True)
15
16 # Write your code here for Scatter Plot for Age vs. Fare
17 by Survived
18 plt.figure()
19 colors = {0: 'red', 1: 'blue'}
20 plt.scatter(data['Age'], data['Fare'],
21 c=data['Survived'].apply(lambda x:colors[x]))
22 plt.title('Age vs. Fare by Survival')
23 plt.xlabel('Age')
24 plt.ylabel('Fare')
25 plt.show()
```

Terminal  Test cases 

< Prev Reset Submit

