1.1.1. Calculate Momentum

08:47

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

$$p = m \times v$$

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

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

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

Sample Test Cases

Explorer

calculate...

SUBMIT

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

Terminal Test cases

1.1.2. Conditional Calculation Based on the Number of Digits

38:59

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

Constraints:
$1 \leq n \leq 999$

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

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

Sample Test Cases

Explorer

condition...

SUBMIT

```
1   n=int(input())
2   if n>=0 and n<=10:
3       p=n*n
4       print('%0.0f'%p)
5   elif n>=10 and n<=99:
6       q=n**0.5
7       print('%0.2f'%q)
8   elif n>=100 and n<=999:
9       r=n**(1/3)
10      print('%.2f'%r)
11  else:
12      print("Invalid")
```

Terminal Test cases

1.1.3. Age and Salary Calculation

24:07

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

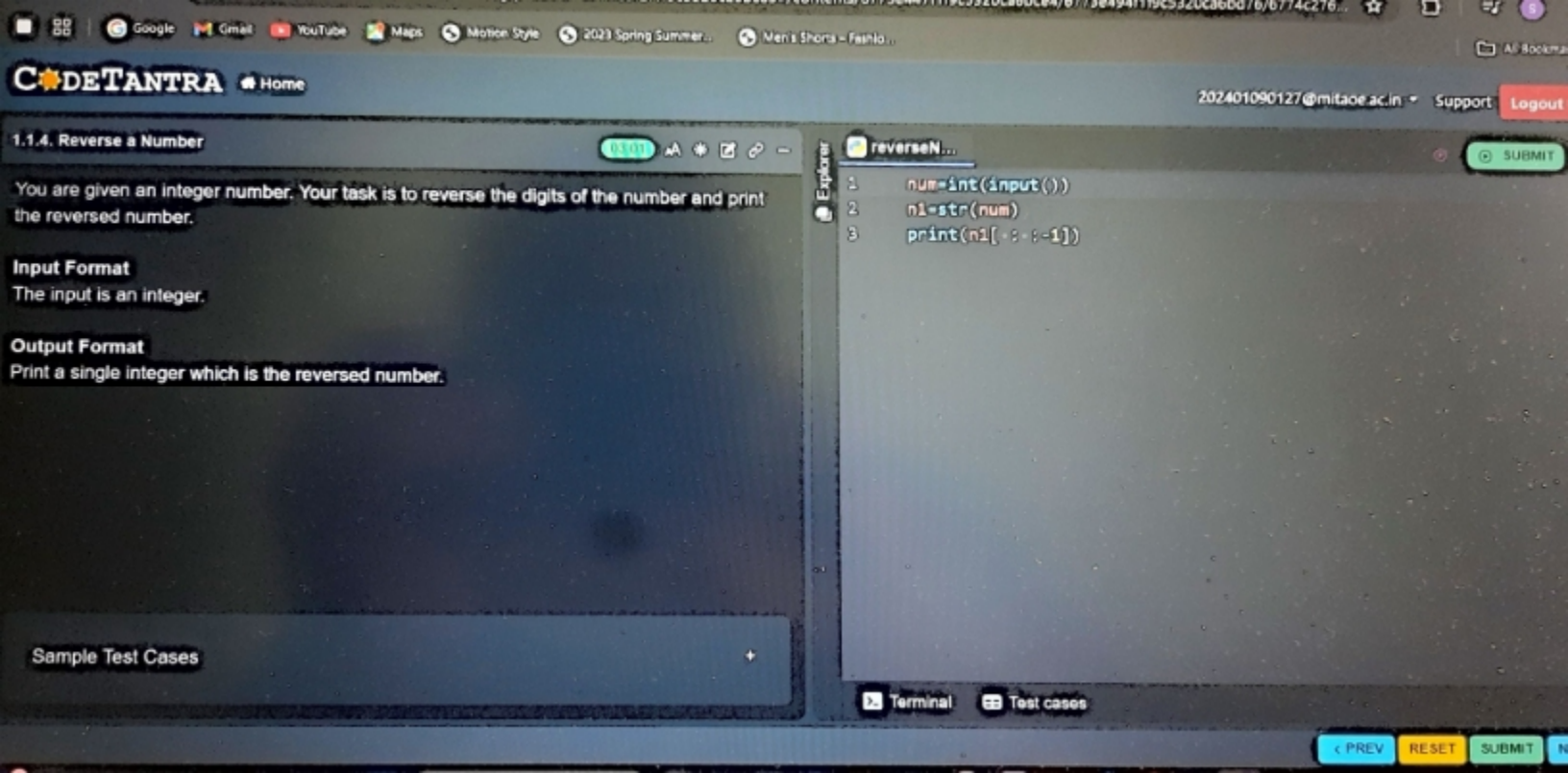
Sample Test Cases +

birthDate... SUBMIT

```python
from datetime import datetime

def calculate_age(birthdate):
    birth_year=int(birthdate.split('-')[-1])
    current_year=datetime.now().year
    return current_year-birth_year-1
def convert_salary_to_dollars(salary_in_rupees):
    inr_to_usd=0.012
    return salary_in_rupees*inr_to_usd
birthdate = input()
salary_in_rupees = float(input())
age = calculate_age(birthdate)
salary_in_dollars = convert_salary_to_dollars(salary_in_rupees)
print(f"Age: {age}")
print(f"Salary in dollars: {salary_in_dollars:.2f}")

```

Terminal Test cases

PREV RESET SUBMIT NEXT


Google
Gmail
YouTube
Maps
Motion Style
2023 Spring Summer...
Men's Shorts - Fashio...
CODETANTRA
Home
202401090127@mitaoe.ac.in
Support
Logout
1.1.4. Reverse a Number
You are given an integer number. Your task is to reverse the digits of the number and print the reversed number.
Input Format
The input is an integer.
Output Format
Print a single integer which is the reversed number.
Sample Test Cases
Explorer
reverseN...
SUBMIT
num=int(input())
n1=str(num)
print(n1[::-1])
Terminal
Test cases
PREV
RESET
SUBMIT

1.1.5. Multiplication Table

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

Terminal Test cases

1.2.1. Pass or Fail

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...

```
n=int(input())
marks=list(map(int,input().split( )))
if all(mark>40 for mark in marks):
    avg=sum(marks)/n
    print(f"Aggregate Percentage: {avg:.2f}")
    if (avg >=75):
        print("Grade: Distinction")
    elif (60<= avg <75):
        print("Grade: First Division")
    elif (avg>=50 and avg<60):
        print("Grade: Second Division")
    elif (avg>=40 and avg<50):
        print("Grade: Third Division")
else:
    print("Fail")
```

Terminal Test cases

CODETANTRA Home

2.2. Fibonacci series using Recursive Function

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

Expected Output-1:

Enter terms for Fibonacci series: 5
0 1 1 2 3

Expected Output-2:

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

Instructions:

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

Sample Test Cases

Explorer

fib.py

```
def fib(n):
    if n<=0:
        return 0
    elif n==1:
        return 1
    else:
        return fib(n-1)+fib(n-2)


n=int(input("Enter terms for Fibonacci series: "))
for i in range(n):
    print(fib(i),end=" ")
```

Terminal Test cases

2.3. Pattern - 1

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

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

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

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

Sample Test Cases

Explorer

rightangl...

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

Terminal Test cases

PREV RESET SUBMIT NEXT

1.2.4. Pattern : 2

03:10

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

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

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

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

Sample Test Cases

Explorer

numberP...

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

Terminal Test cases

< PREV RESET SUBMIT

2.1.1. List operations

43:15

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

Explorer

listOps.py

```
# write the code..
def main():
    integer_list = []
    
    while True:
        print("1. Add")
        print("2. Remove")
        print("3. Display")
        print("4. Quit")
        
        choice = input("Enter choice: ")
        
        if choice == "1":
            num = input("Integer: ")
            if num.isdigit(): # Check if input is a valid integer
                integer_list.append(int(num))
                print(f"List after adding: {integer_list}")
            else:
                print("Invalid input")
        
```

Terminal Test cases

< PREV RESET SUBMIT

25°C Partly cloudy

Search

ENG IN

2.1.2. Dictionary Operations

05:40

Write a Python program to perform the following dictionary operations:

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

Note: Refer to visible test cases.

Sample Test Cases

Explorer

dictOpera...

```
dict1={}
print("Empty Dictionary:",dict1)
n=int(input("Number of items: "))
for n in range(n):
    key=input("key : ")
    value=input("value: ")
    dict1[key]=value
print("Dictionary:",dict1)
ukey=input("Enter the key to update:")
if ukey in dict1:
    new=input("Enter the new value: ")
    dict1[ukey]=new
    print("value updated")
else:
    print("Key not found ")
rkey=input ("Enter the key to retrieve: ")
if rkey in dict1 :
    print (f"key: {rkey}, Value {dict1[rkey]}")
else:
    print("Key not found")
```

Terminal Test cases

‹ PREV RESET SUBMIT

2.2.1. Linear search Technique 14:48

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:

Sample Test Cases

Explorer CTP1709... SUBMIT

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

Terminal Test cases

< PREV RESET SUBMIT NEXT >

2.2.2. Captain of the Team

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

captainof...

```
a=list(map(int,input().split(" ")))
print(max(a))
```

Terminal Test cases

PREV RESET SUBMIT

3.1.1. Numpy array operations

05:35

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

Explorer

numpyarr...

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

Terminal Test cases

‹ PREV RESET SUBMIT

3.2.1. Numpy: Matrix Operations

03:30

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

Sample Test Cases

Explorer

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 i in
   range(3)])
6
7  print("Enter Matrix B:")
8  matrix_b = np.array([list(map(int, input().split())) for i in
   range(3)])
9
10
11 # Addition
12 print("Addition (A + B):")
13 print(matrix_a+matrix_b)
14 # Subtraction
15 print("Subtraction (A - B):")
16 print(matrix_a-matrix_b)
17 # Multiplication (element-wise)
18 print("Element-wise Multiplication (A * B):")
```

Terminal Test cases

3.2.2. Numpy: Horizontal and Vertical Stacking of Arrays

04:01

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

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

Input Format:

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

Output Format:

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

Sample Test Cases

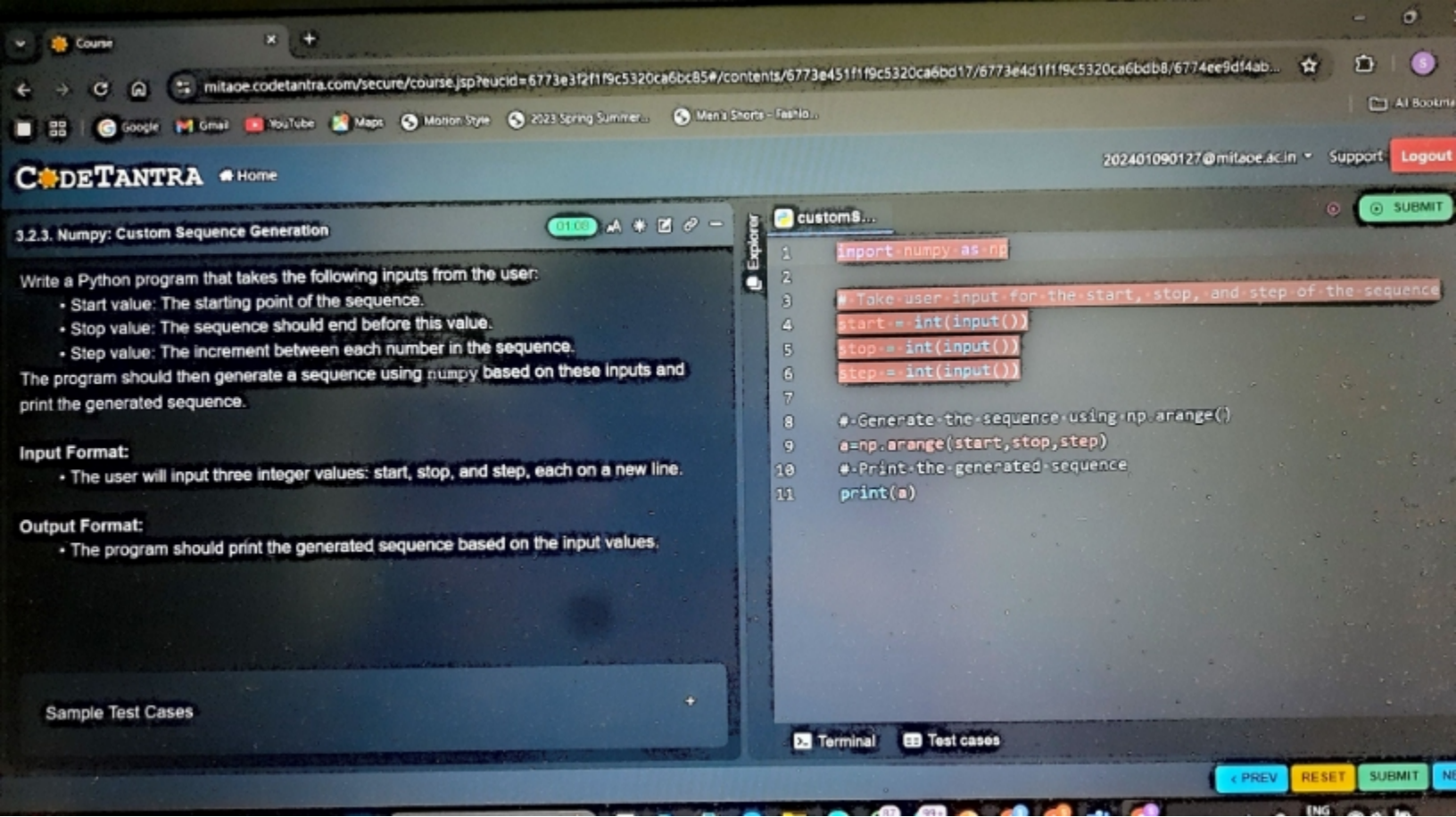
Explorer

stacking.py

```
import numpy as np

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

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

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


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

Terminal Test cases

‹ PREV RESET SUBMI


Course
mitaoe.codetantra.com/secure/course.jsp?eucid=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774ee9d14ab...
Google
Gmail
YouTube
Maps
Motion Style
2023 Spring Summer...
Men's Shorts - Fashio...
CODETANTRA
Home
202401090127@mitaoe.ac.in
Support
Logout
3.2.3. Numpy: Custom Sequence Generation
01:00
Write a Python program that takes the following inputs from the user:
• Start value: The starting point of the sequence.
• Stop value: The sequence should end before this value.
• Step value: The increment between each number in the sequence.
The program should then generate a sequence using numpy based on these inputs and print the generated sequence.
Input Format:
• The user will input three integer values: start, stop, and step, each on a new line.
Output Format:
• The program should print the generated sequence based on the input values.
Sample Test Cases
Explorer
customS...
SUBMIT
import numpy as np
# Take user input for the start, stop, and step of the sequence
start = int(input())
stop = int(input())
step = int(input())
# Generate the sequence using np.arange()
a=np.arange(start,stop,step)
# Print the generated sequence
print(a)
Terminal
Test cases
PREV
RESET
SUBMIT

Course

mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774ef2cf4ab7...

Google Gmail YouTube Maps Motion Style 2023 Spring Summer... Men's Shorts - Fashio...

CODETANTRA Home

202401090127@mitaoe.ac.in Support Logou

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Oper... 05:15

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

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

Input Format:

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

Sample Test Cases

Explorer

different...

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      # Arithmetic Operations
9      sum_result = A+B
10     diff_result = A-B
11     prod_result = 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 = A&B
20     or_result = A|B
```

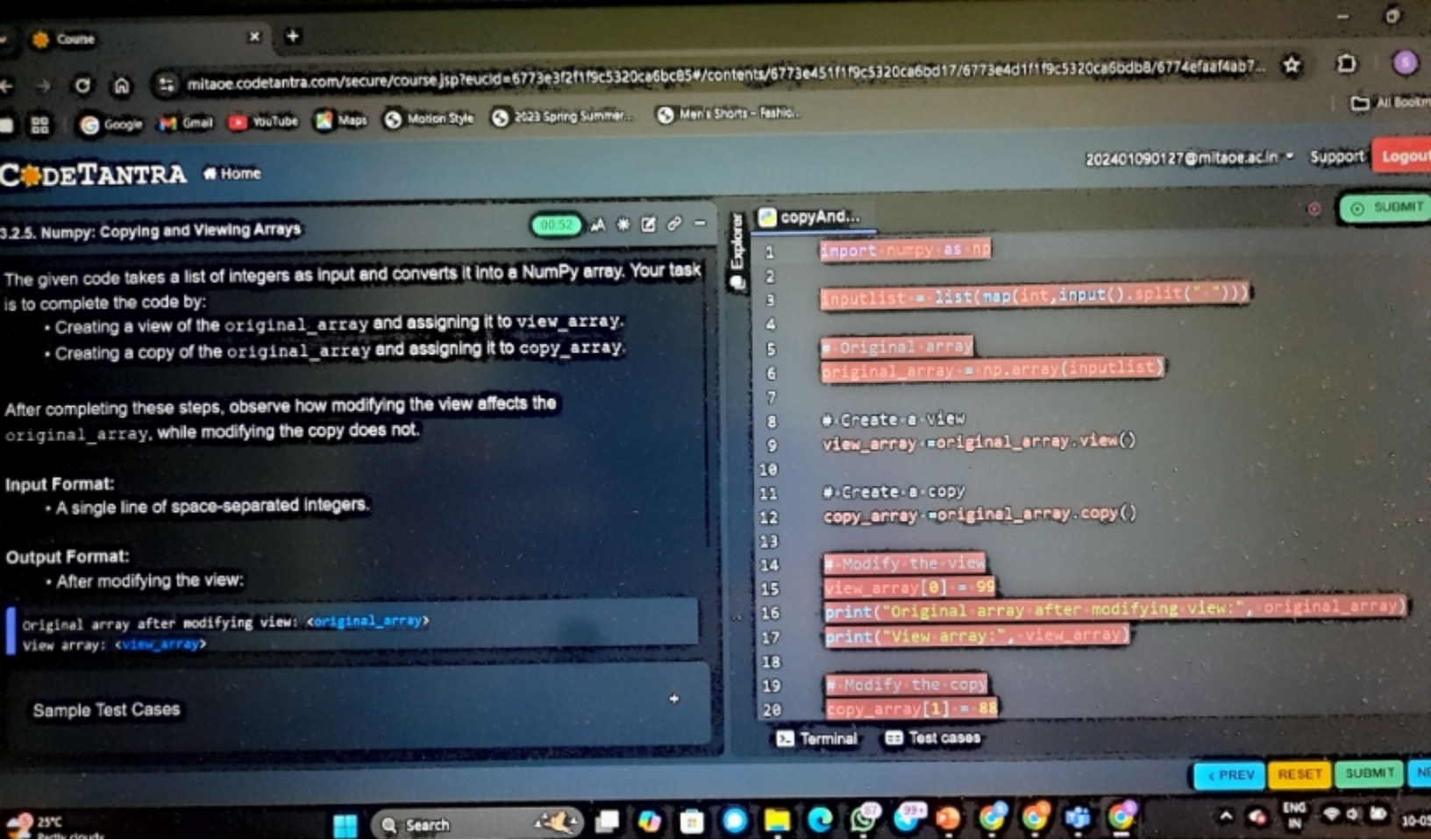
Terminal Test cases

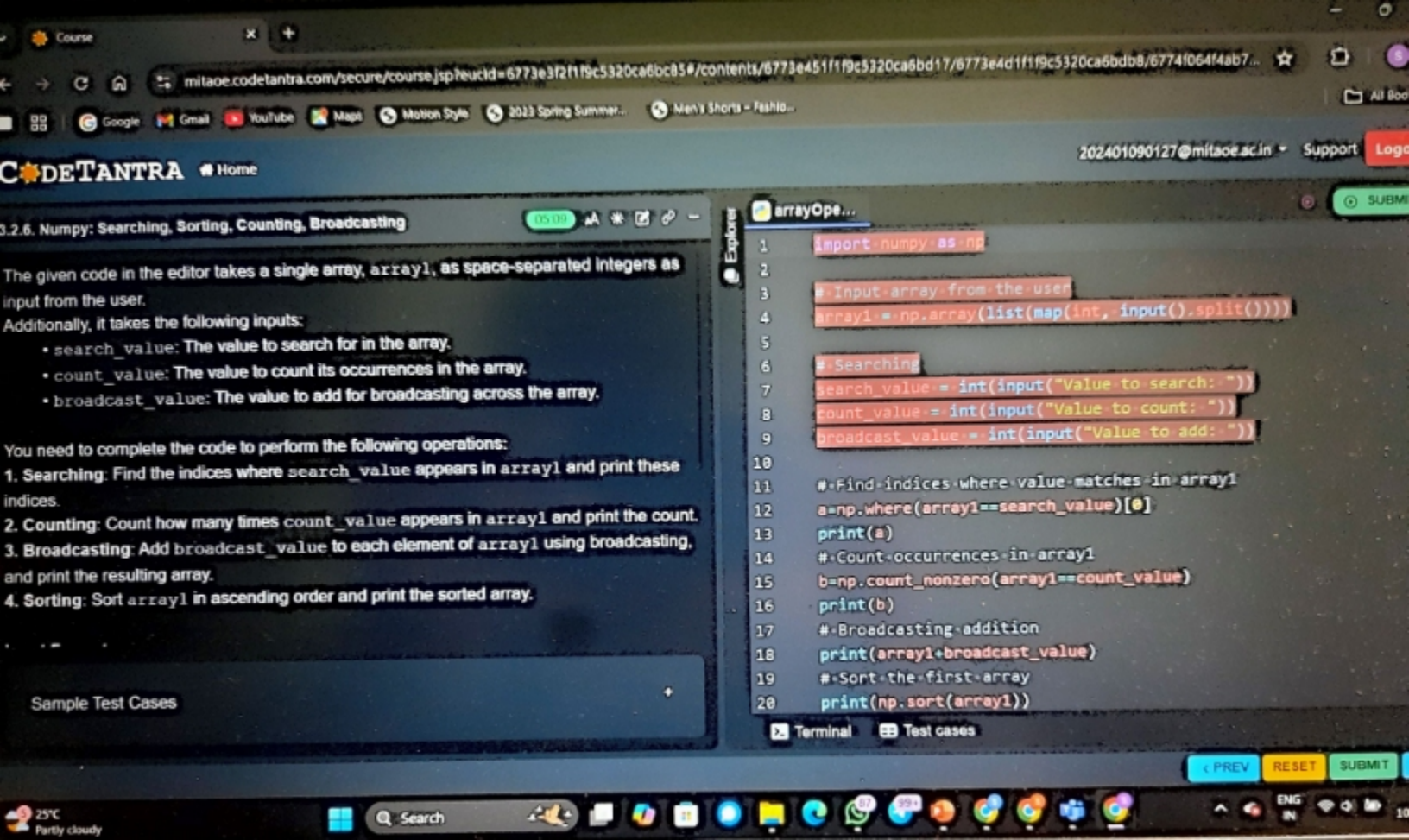
< PREV RESET SUBMIT

25°C Partly cloudy

Search

ENG IN


Course
mitaoe.codetantra.com/secure/course.jsp?eucid=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774efaaf4ab7...
Google
Gmail
YouTube
Maps
Motion Style
2023 Spring Summer...
Men's Shorts - Fashio...
All Bookm
CODETANTRA
Home
202401090127@mitaoe.ac.in
Support
Logout
3.2.5. Numpy: Copying and Viewing Arrays
00:52
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>
Sample Test Cases
Explorer
copyAnd...
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
Terminal
Test cases
‹ PREV
RESET
SUBMIT
25°C
Search
ENG
IN


Course
mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e451f1f9c5320ca6bd17/6773e4d1f1f9c5320ca6bdb8/6774f064f4ab7...
Google
Gmail
YouTube
Maps
Motion Style
2023 Spring Summer...
Men's Shorts - Fashio...
All Boo
CODETANTRA
Home
202401090127@mitaoe.ac.in
Support
Logo
3.2.6. Numpy: Searching, Sorting, Counting, Broadcasting
05:09
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.
Sample Test Cases
Explorer
arrayOpe...
SUBMI
1 import numpy as np
2
3 # Input array from the user
4 array1 = np.array(list(map(int, input().split())))
5
6 # Searching
7 search_value = int(input("Value to search: "))
8 count_value = int(input("Value to count: "))
9 broadcast_value = int(input("Value to add: "))
10
11 # Find indices where value matches in array1
12 a=np.where(array1==search_value)[0]
13 print(a)
14 # Count occurrences in array1
15 b=np.count_nonzero(array1==count_value)
16 print(b)
17 # Broadcasting addition
18 print(array1+broadcast_value)
19 # Sort the first array
20 print(np.sort(array1))
Terminal
Test cases
PREV
RESET
SUBMIT
25°C
Partly cloudy
Search
ENG
IN

3.2.7. Student Data Analysis and Operations

08:20

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

Sample Test Cases +

Explorer

Operatio...

SUBMIT

```
import numpy as np

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

import numpy as np

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


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

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

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

#4. Print subject 1 marks
```

Terminal Test cases

‹ PREV RESET SUBMIT N...

4.1.1. Pandas - series creation and manipulation

24:40

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

Input Format:

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

Output Format:

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

Sample Test Cases

Explorer

seriesMa...

SUBMIT

Debugger

```
import pandas as pd

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

#Create a Pandas series from the list of numbers
a=pd.Series(numbers)
#Grouping by even and odd numbers and calculating the mean
grouped=a.groupby(a%2==0).mean()
formatted_series=pd.Series({'Even':grouped.get(True,None),'Odd':
grouped.get(False,None)}).dropna()
#Display the mean of even and odd numbers with labels
grouped.index=['Even' if is_even else 'Odd' for is_even in
grouped.index]
print("Mean of even and odd numbers:")
print(grouped)

```

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

4.1.2. Dictionary to dataframe

30 50

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.

Sample Test Cases

dataframe...

SUBMIT

```
import pandas as pd

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

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

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

# Adding a new row
new_name = input("New name: ")
new_age = int(input("New age: "))
new_row = {
    'Name': new_name,
```

Terminal Test cases

PREV RESET SUBMIT NEXT

4.1.3. Student Information

24:41

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

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

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

Sample Test Cases

studentin... studentdat...

SUBMIT

```
import pandas as pd

#Read the text file into a DataFrame
file = input()
data = pd.read_csv(file, sep="\s+",header=None, names=["Name", "Age", "Grade"])
print("First five rows:")
print(data.head(5))
#write your code here..
age=round(data['Age'].mean(),2)
print("Average age:",age)
print("Students with a grade up to B")
df=pd.DataFrame(data)
a=df[df['Grade']<='B']
print(a)
```

Terminal Test cases

< PREV RESET SUBMIT NEXT >

4.2.1. Month with the Highest Total Sales

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
```

Sample Test Cases

monthFor... sales_dat...

SUBMIT

```python
import pandas as pd

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

# Load the data
df = pd.read_csv(file_name)
df['sales']=df['Quantity'].multiply(df['Price'])
df['month']=pd.to_datetime(df['Date']).dt.strftime("%Y-%m")

# Find the month with the highest total sales
best_month = df.groupby('month')['sales'].sum().idxmax()
highest_sales = df['sales'].sum()

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

Terminal Test cases

< PREV RESET SUBMIT NEXT >

4.2.2. Best Selling Product

02:53

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
```

Sample Test Cases

monthFor... sales_dat...

SUBMIT

```python
import pandas as pd

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

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


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

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

```

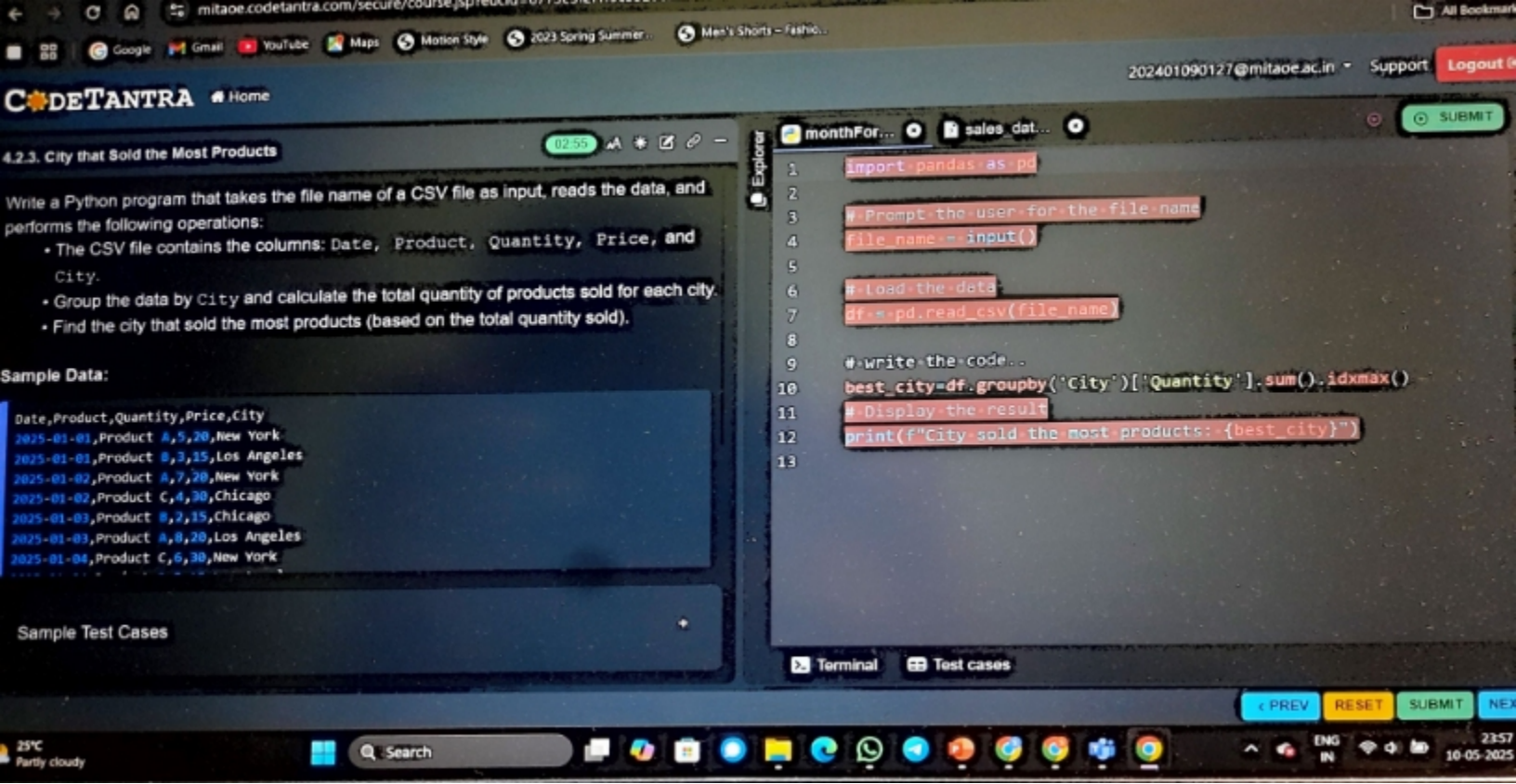
Terminal Test cases

PREV RESET SUBMIT NEXT

25°C Partly cloudy

Search

ENG IN 23:57 10-05-2025


mitaoe.codetantra.com/secure/course.jsp?eduId=
Google
Gmail
YouTube
Maps
Motion Style
2023 Spring Summer...
Men's Shorts – Fashio...
All Bookmarks
CODETANTRA
Home
202401090127@mitaoe.ac.in
Support
Logout
4.2.3. City that Sold the Most Products
02:55
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
Sample Test Cases
Explorer
monthFor...
sales_dat...
SUBMIT
1 import pandas as pd
2
3 # Prompt the user for the file name
4 file_name = input()
5
6 # Load the data
7 df = pd.read_csv(file_name)
8
9 # write the code..
10 best_city=df.groupby('City')['Quantity'].sum().idxmax()
11 # Display the result
12 print(f"City sold the most products: {best_city}")
13
Terminal
Test cases
‹ PREV
RESET
SUBMIT
NEX
25°C
Partly cloudy
Search
ENG
IN
23:57
10-05-2025

4.2.4. Most Frequently Sold Product Pairs

13:05

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
```

Sample Test Cases

Explorer frequenti... sales_dat... SUBMIT

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

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

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

# write the code
grouped = df.groupby("Date")["Product"].apply(list)

product_pair = []
for products in grouped:
    if len(products) > 1:
        product_pair.extend(combinations(sorted(products),2))

pair_counts = Counter(product_pair)

```

Terminal Test cases

PREV RESET SUBMIT NEXT

25°C Partly cloudy

Search

ENG IN 23:57 10-05-2025

4.2.5. Titanic Dataset Analysis and Cleaning

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.

Sample Test Cases

Explorer titanicDat... SUBMIT Debugger

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

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

# 1. Display the first 5 rows of the dataset
print(data.head())

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

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

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

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

Terminal Test cases

2.6. Titanic Dataset Analysis and Data Cleaning - 2

00:32

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

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.

e Titanic dataset contains columns as shown below,

Sample Test Cases

Explorer titanicDat... SUBMIT

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

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

# 1. Create a new column 'IsAlone' (1 if alone, 0 otherwise)
data['IsAlone']=(data['FamilySize']==0).astype(int)
# 2. Convert 'Sex' to numeric (male: 0, female: 1)
data['Sex']=data['Sex'].map({'male':0,'female':1})
# 3. One-hot encode the 'Embarked' column
embarked_dummies=pd.get_dummies(data['Embarked'],prefix='Embarked',drop_first=True)
data=pd.concat([data,embarked_dummies],axis=1)
# 4. Get the mean age of passengers
mean_age=data['Age'].mean()
print(mean_age)
# 5. Get the median fare of passengers
median_fare=data['Fare'].median()
```

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

3.2.7. Titanic Dataset Analysis and Data Cleaning - 3

00:28

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

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

The Titanic dataset contains columns as shown below,

Sample Test Cases

Explorer

titanicDat...

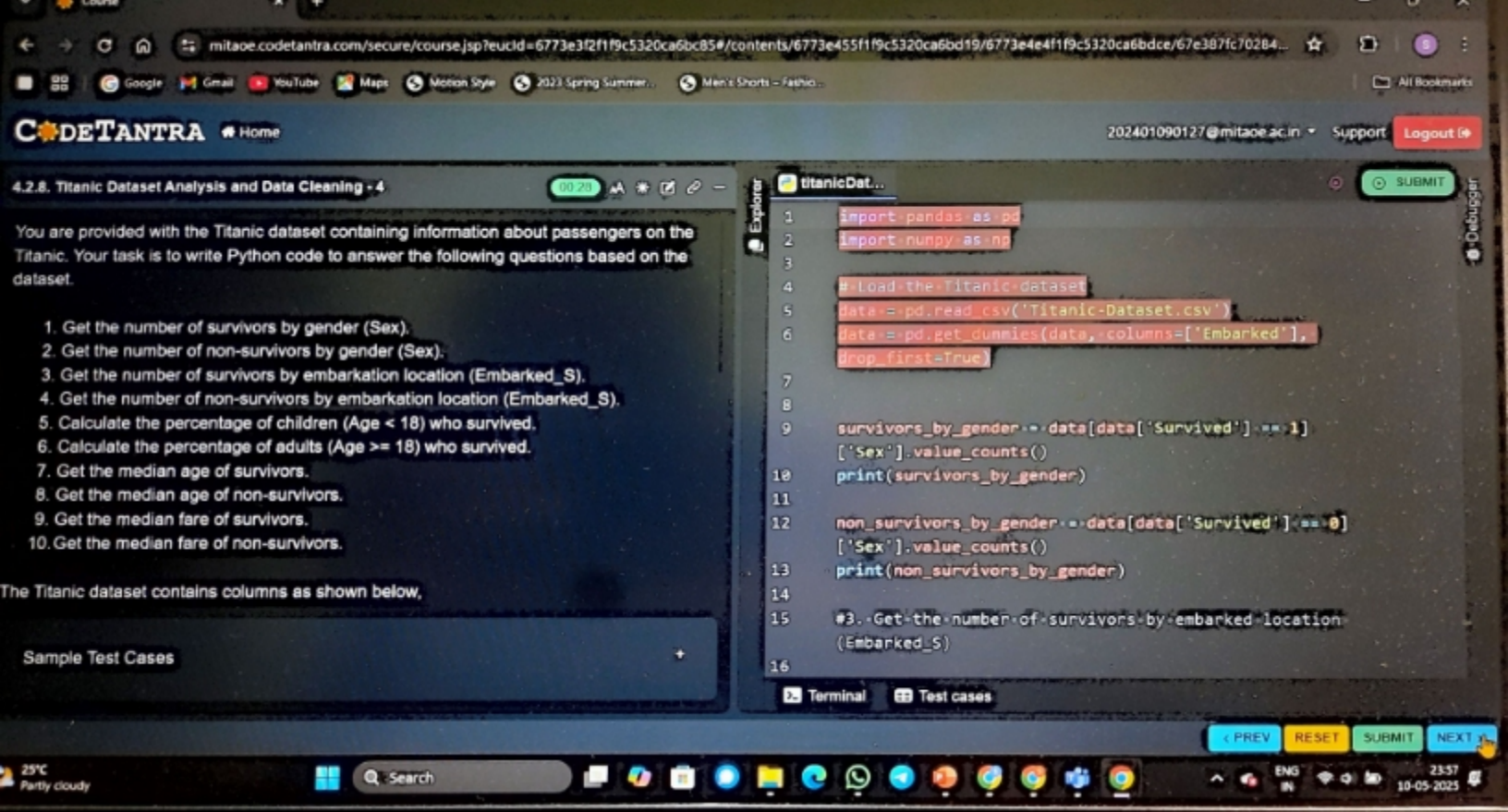
SUBMIT

Debugger

```
import pandas as pd
import numpy as np

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

#1. Calculate the survival rate by class
survival_by_class=data.groupby('Pclass')['Survived'].mean()
#2. Calculate the survival rate by embarked location
surivival_by_embarked=data.groupby('Embarked_S')
['Survived'].mean()
#3. Calculate the survival rate by family size
survival_by_family=data.groupby('FamilySize')
['Survived'].mean().sort_index()
#4. Calculate the survival rate by being alone
survival_by_alone=data.groupby('IsAlone')['Survived'].mean()
```

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›


mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e455f1f9c5320ca6bd19/6773e4e4f1f9c5320ca6bdce/67e387fc70284...
Google
Gmail
YouTube
Maps
Motion Style
2023 Spring Summer...
Men's Shorts – Fashio...
All Bookmarks
CODETANTRA
Home
202401090127@mitaoe.ac.in
Support
Logout
4.2.8. Titanic Dataset Analysis and Data Cleaning - 4
00:28
You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset.
1. 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,
Sample Test Cases
Explorer
titanicDat...
SUBMIT
Debugger
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 = data[data['Survived'] == 1]['Sex'].value_counts()
10 print(survivors_by_gender)
11
12 non_survivors_by_gender = data[data['Survived'] == 0]['Sex'].value_counts()
13 print(non_survivors_by_gender)
14
15 #3. Get the number of survivors by embarked location (Embarked_S)
16
Terminal
Test cases
‹ PREV
RESET
SUBMIT
NEXT ›
25°C
Partly cloudy
Search
ENG
IN
23:57
10-05-2025

mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e45af1f9c5320ca6bd1d/6773e4e9f1f9c5320ca6bdd2/675a8d2c09f9f...

Google Gmail YouTube Maps Motion Style 2023 Spring Summer... Men's Shorts - Fashio... All Bookmarks

CODETANTRA Home

202401090127@mitaoe.ac.in Support Logout

5.1.1. Stacked Plot

21:18

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

Your task is to:

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

Sample Test Cases

Explorer

stackedpl...

SUBMIT

Debugger

```
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', 'May',
    'June', 'July', 'August', 'September', 'October', 'November',
    'December'],
7       'City_A_Temperature': [5, 7, 10, 13, 17, 20, 22, 21, 18,
    12, 8, 6],
8       'City_B_Temperature': [2, 3, 5, 6, 10, 14, 16, 17, 12, 9,
    5, 3],
9       'City_C_Temperature': [3, 4, 6, 8, 9, 12, 15, 14, 10, 7, 4,
    2]
10  }
11
12  # Write your code...
13  plt.stackplot(data['Month'],data['City_A_Temperature'],data['Cit
    y_B_Temperature'],data['City_C_Temperature'])
14  plt.xlabel('Month')
```

Terminal Test cases

PREV RESET SUBMIT NEXT

25°C Partly cloudy

Search

ENG IN 23:57 10-05-2025

2.1. Titanic Dataset

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

Sample Test Cases

titanicDat...

SUBMIT

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

# Load the Titanic dataset from the CSV file
df = pd.read_csv('titanic.csv')

# Set up the figure for 5 subplots
fig, axes = plt.subplots(3, 2, figsize=(12, 12))

# write the code
# write the code..
import pandas as pd
import matplotlib.pyplot as plt

# Load the Titanic dataset from the CSV file
df = pd.read_csv('titanic.csv')

# Set up the figure for 5 subplots
fig, axes = plt.subplots(3, 2, figsize=(12, 12))

```

Terminal Test cases

< PREV RESET SUBMIT NEXT >

Course

mitaoe.codetantra.com/secure/course.jsp?eucId=6773e3f2f1f9c5320ca6bc85#/contents/6773e45af1f9c5320ca6bd1d/6773e4f1f1f9c5320ca6bdd5/67e28a397c0b9...

Google | Gmail | YouTube | Maps | Motion Style | 2023 Spring Summer... | Men's Shorts – Fashio... | All Bookmarks

CODETANTRA Home

202401090127@mitaoe.ac.in | Support | Logout

5.2.2. Histogram of passenger information of Titanic

00:29

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,

Passenger Id	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases

Explorer | Histogra... | SUBMIT | Debugger

```
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 Histogram
17 plt.hist(data['Age'], bins=30, edgecolor='k')
18 plt.xlabel('Age')
```

Terminal | Test cases

‹ PREV | RESET | SUBMIT | NEXT ›

ENG IN 23:58 10-05-2025

2.3. Bar plot of survival rate of passengers

00:31

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,

Passenger Id	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases

Explorer

BarPlotOf...

SUBMIT

Debugger

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  # Load the Titanic dataset
5  data = pd.read_csv('Titanic-Dataset.csv')
6
7  # Data Cleaning
8  data['Age'].fillna(data['Age'].median(), inplace=True)
9  data['Embarked'].fillna(data['Embarked'].mode()[0],
   inplace=True)
10 data.drop('Cabin', axis=1, inplace=True)
11
12 # Convert categorical features to numeric
13 data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14 data = pd.get_dummies(data, columns=['Embarked'],
   drop_first=True)
15
16 # Write your code here for Bar Plot for Survival Rate
17 survival_count = data['Survived'].value_counts()
18 survival_count.plot(kind='bar')
```

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

5.2.4. Bar Plot for Survival by Gender

01:03

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,

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

Sample Test Cases

Explorer

BarPlotOf...

SUBMIT

Debugger

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

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

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

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

# Write your code here for Bar Plot for Survival by Gender
survival_by_gender = data.groupby('Sex')
['Survived'].value_counts().unstack().fillna(0)
```

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

5.2.5. Bar Plot for Survival by Pclass

00:35

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,

Passenger	Survived	Pclass	Name	Sex	Age

Sample Test Cases

Explorer

BarPlotOf...

SUBMIT

Debugger

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

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

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

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

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

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

5.2.6. Bar Plot for Survival by Embarked

01:08

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,

Pa

Sample Test Cases

Explorer

BarPlotOf...

SUBMIT

Debugger

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

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

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

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

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

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

5.2.7. Box plot for Age Distribution

00:36

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,

Passenger Id	Survived	Pclass	Name	Sex	Age	Sib Sp	Pa rch	Ti cket	Fare	Cabin	Embarked

Sample Test Cases

BoxPlotF...

SUBMIT

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

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

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

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

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

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

CODETANTRA Home

202401090127@mitaoe.ac.in Support Logout

5.2.8. Box Plot for Age by Survived

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,

Passenger Id	Survived	Pclass	Name	Sex	Age	Sib Sp	Pa rch	Ti cket	Fa re	Ca bin	Em bar ked

Sample Test Cases

Explorer

BoxPlotF...

SUBMIT

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

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

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

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

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

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

25°C Partly cloudy

Search

ENG IN 23:58 10-05-2025

5.2.9. Box Plot for Fare by Pclass

12:04

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,

Passenger Id	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked

Sample Test Cases

Explorer

BoxPlotF...

SUBMIT

Debugger

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

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

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

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

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

Terminal Test cases

‹ PREV RESET SUBMIT NEXT ›

CODETANTRA Home

Scatter Plot for Age vs. Fare

Write the 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.

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

Sample Test Cases

AgeFareS...

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

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

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

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

# Write your code here for Box Plot for

plt.figure()
```

Terminal Test cases

PREV RESET

5.2.11. Scatter Plot for Age vs. Fare by Survived

12:55

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	Sib...	Pa...	Ti...	Fa...	Ca...	Em...

Sample Test Cases

Explorer

AgeFareS...

SUBMIT

Debugger

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

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

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

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

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

```

Terminal Test cases

PREV RESET SUBM