1.1.1. Calculate Momentum

`20:59`

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   +

---

calculate...

```python
mass=float(input(""))
velocity=float(input(""))
p=mass*velocity
print("%0.2f"%p,end="")
print("kgm/s")
```

Submit

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

1.1.2. Conditional Calculation Based on the Number of Digits

`15:45` A ☾ ✎ 🔗 —

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...   ⊙   ✅ Submit

```python
n=int(input(""))
if(n>0 and n<=9):
    print(n*n)
elif(n>10 and n<=99):
    r=n**0.5
    print("%0.2f"%r)
elif(n>100 and n<=999):
    p=n**(1/3)
    print("%0.2f"%p)
else:
    print("Invalid")
```

▶ Terminal   ⊞ Test cases

1.1.3. Age and Salary Calculation

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):
    date_object = datetime.strptime(birthdate, "%d-%m-%Y")
    today = datetime.today()
    if ((today.month, today.day ) < (date_object.month, date_object.day)):
        age = today.year-date_object.year-((today.month, today.day) < (date_object.month, date_object.day))
        return age
    elif((today.month,today.day ) > (date_object.month,date_object.day)):
        age = today.year-date_object.year-((today.month,today.day) > (date_object.month, date_object.day))
        return age


def convert_salary_to_dollars(salary_in_rupees):
    salary=salary_in_rupees*0.012
    return salary
```

Terminal   Test cases

1.1.3. Age and Salary Calculation

`25:42` ⒜ 🌙 ✎ ⊘ —

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):
    date_object = datetime.strptime(birthdate, "%d-%m-%Y")
    today = datetime.today()
    if ((today.month, today.day ) < (date_object.month, date_object.day)):
        age = today.year-date_object.year-((today.month, today.day) < (date_object.month, date_object.day))
        return age
    elif((today.month,today.day ) > (date_object.month,date_object.day)):
        age = today.year-date_object.year-((today.month,today.day) > (date_object.month, date_object.day))
        return age


def convert_salary_to_dollars(salary_in_rupees):
    salary=salary_in_rupees*0.012
    return salary
```

Terminal   ⊞ Test cases

1.1.4. Reverse a Number

`19:44`

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...   Submit

```python
n=int(input(""))
reverse=0
while n != 0:
    digit = n % 10
    reverse = reverse*10 + digit
    n = n // 10
print(reverse)
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

1.1.5. Multiplication Table

06:49

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... Submit

```python
n=int(input(""))
i=1
while i<=10:
    print(n,"x",i,"=",i*n)
    i=i+1
```

Terminal Test cases

‹ Prev Reset Submit Next ›

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.

Output Format:

If the student passes all courses:

Sample Test Cases   +

---

passorFa...   Submit

```python
n=int(input(""))
marks = list(map(int, input().split()))
if any(mark<40 for mark in marks):
    print("Fail")
else:
    totalmarks = sum(marks)
    aggregate = (totalmarks/(n*100))*100
    print(f"Aggregate Percentage: {aggregate:.2f}")
    if aggregate>75:
        print("Grade: Distinction")
    elif aggregate>60 and aggregate<75:
        print("Grade: First Division")
    elif aggregate>50 and aggregate<60:
        print("Grade: Second Division")
    elif aggregate>40 and aggregate<50:
        print("Grade: Third Division")
    else:
        print("Fail")
```

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

1.2.2. Fibonacci series using Recursive Function   `02:10` 𝐀 ☾ ☑ ⊘ −

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

Expected Output-1:

Enter terms for Fibonacci series: 5

0 1 1 2 3

Expected Output-2:

Enter terms for Fibonacci series: 9

0 1 1 2 3 5 8 13 21

Instructions:

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

Sample Test Cases   +

🐍 **fib.py**   ⊙ ▶ Submit

```python
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

< Prev   Reset   Submit   Next >

1.2.3. 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...   Submit

```python
n=int(input(""))
for i in range(0, n):
    for j in range(0, i+1):
        print("*", end=" ")
    print()
```

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

1.2.4. Pattern - 2

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...   Submit

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

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

2.1.1. List operations   `08:29` ᴀA ☾ ☑ ⧉ —

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

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

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

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

Sample Test Cases   +

🐍 listOps.py   ⊙ ▶ **Submit**

```python
1   list1=[]
2   while True:
3       print("1. Add")
4       print("2. Remove")
5       print("3. Display")
6       print("4. Quit")
7       n=int(input("Enter choice: "))
8       if n == 1:
9           add=int(input("Integer: "))
10          list1.append(add)
11          print(f"List after adding: {list1}")
12      elif n==2:
13          if len(list1)==0:
14              print("List is empty")
15          elif len(list1)!=0:
16              remove=int(input("Integer: "))
17              if remove not in list1 :
18                  print("Element not found")
19              else:
20                  list1.remove(remove)
21                  print(f"List after removing: {list1}")
22      elif n==3:
```

>_ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

2.1.1. List operations   08:29   A̶A ☾ ☑ 𝒫 −

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

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

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

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

Sample Test Cases   +

listOps.py   ⊙   ⊙ Submit

```python
      add=int(input("Integer: "))
10      list1.append(add)
11      print(f"List after adding: {list1}")
12  elif n==2:
13      if len(list1)==0:
14          print("List is empty")
15      elif len(list1)!=0:
16          remove=int(input("Integer: "))
17          if remove not in list1 :
18              print("Element not found")
19          else:
20              list1.remove(remove)
21              print(f"List after removing: {list1}")
22  elif n==3:
23      if len(list1)==0:
24          print("List is empty")
25      else:
26          print(list1)
27  elif n==4:
28      break
29  else:
30      print("Invalid choice")
```

▶ Terminal   ⊞ Test cases

2.1.2. Dictionary Operations   `21.13` 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...   ⊙   **⊙ Submit**

```python
1   dict = {}
2   print("Empty Dictionary:",dict)
3   n = int(input("Number of items: "))
4   for _ in range (n):
5       key = input("key: ")
6       value = input("value: ")
7       dict[key] = value
8   print("Dictionary:",dict)
9   update_key = input("Enter the key to update: ")
10  if update_key in dict:
11      new_value = input("Enter the new value: ")
12      dict[update_key] = new_value
13      print("Value updated")
14  else:
15      print("Key not found")
16  retrieve_key = input("Enter the key to retrieve: ")
17  if retrieve_key in dict:
18      print(f"Key: {retrieve_key}, Value: {dict[retrieve_key]}")
19  else:
20      print("Key not found")
21  get_key = input("Enter the key to get using the get() method: ")
22  value = dict.get(get_key, "Key not found")
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

2.1.2. Dictionary Operations

21:13 ᴀ̌ ☾ ☑ ⊘ —

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...   ⊙   ▶ Submit

```
13        print("value updated")
14    else:
15        print("Key not found")
16    retrieve_key = input("Enter the key to retrieve: ")
17    if retrieve_key in dict:
18        print(f"Key: {retrieve_key}, Value: {dict[retrieve_key]}")
19    else:
20        print("Key not found")
21    get_key = input("Enter the key to get using the get() method: ")
22    value = dict.get(get_key,"Key not found")
23    if value != "Key not found":
24        print(f"Key: {get_key}, Value: {value}")
25    else:
26        print("value")
27    deleted_key = input("Enter the key to delete: ")
28    if deleted_key in dict:
29        del dict[deleted_key]
30        print("Deleted")
31    else:
32        print("Key not found")
33    print("Updated Dictionary:",dict)
34
```

▶ Terminal   ⊞ Test cases

2.2.1. Linear search Technique

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

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

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

Sample Test Case:
Input:
1 2 3 4 3 5 6
3
Output:
2

Sample Test Cases   +

```python
arr = list(map(int,input().split(" ")))
key = int(input())
for i in range(len(arr)):
    if arr[i] == key:
        print(i)
        break
if arr[i] != key:
    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...

```python
heights = list(map(int,input().split(" ")))

captain = max(heights)

print(captain)
```

Terminal   Test cases

‹ Prev Reset Submit Next ›

3.1.1. Numpy array operations

Write a python program to demonstrate the usage of ndim, shape and size for a Numpy Array. The program should create a NumPy array using the entered elements and display it. Assume all input elements are valid numeric values.

Input Format:
- User inputs the number of rows and columns with space separated values.
- User inputs elements of the array row-wise followed line by line, separated by spaces.

Output Format:
- The created NumPy array based on the input dimensions and elements.
- Dimensions (ndim): Number of dimensions of the array.
- Shape: Tuple representing the shape of the array (number of rows, number of columns).
- Size: Total number of elements in the array.

Note: Use reshape() function to reshape the input array with the specified number of rows and columns.

Sample Test Cases   +

---

numpyarr...   ⊙   ⊙ Submit

```python
import numpy as np
rows,cols= list(map(int,input().split()))
matrix=[]
for i in range(rows):
    row= list(map(int,input().split()))
    matrix.append(row)
matrix=np.array(matrix).reshape(rows,cols)
print(matrix)
print(matrix.ndim)
print(matrix.shape)
print(matrix.size)
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

3.2.1. Numpy: Matrix Operations

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** $(\text{matrix\_a} + \text{matrix\_b})$
2. **Subtraction** $(\text{matrix\_a} - \text{matrix\_b})$
3. **Element-wise Multiplication** $(\text{matrix\_a} * \text{matrix\_b})$
4. **Matrix Multiplication** $(\text{matrix\_a} \cdot \text{matrix\_b})$
5. **Transpose of Matrix A**

Input Format:

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

Sample Test Cases   +

---

matrixOp...   ⊙ ⊙ Submit

```python
import numpy as np

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

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


# Addition
print("Addition (A + B):")
print(matrix_a + matrix_b)
# Subtraction
print("Subtraction (A - B):")
print(matrix_a - matrix_b)
# Multiplication (element-wise)
print("Element-wise Multiplication (A * B):")
print(matrix_a * matrix_b)
# Matrix multiplication (dot product)
```

> Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

3.2.1. Numpy: Matrix Operations

`03:38` ᴀA ☾ ☑ ⧉ —

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** $(\text{matrix\_a} + \text{matrix\_b})$
2. **Subtraction** $(\text{matrix\_a} - \text{matrix\_b})$
3. **Element-wise Multiplication** $(\text{matrix\_a} * \text{matrix\_b})$
4. **Matrix Multiplication** $(\text{matrix\_a} \cdot \text{matrix\_b})$
5. **Transpose of Matrix A**

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

Sample Test Cases +

matrixOp... ⊙ Submit

```python
 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):")
19    print(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    a= matrix_a.T
26    print(a)
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

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   +

---

stacking.py   ⊙   Submit

```python
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)
horizontal_stack = np.hstack((arr1,arr2))

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

Terminal   Test cases

< Prev   Reset   Submit   Next >

3.2.3. Numpy: Custom Sequence Generation

02:53

Write a Python program that takes the following inputs from the user:
- Start value: The starting point of the sequence.
- Stop value: The sequence should end before this value.
- Step value: The increment between each number in the sequence.

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

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

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

Sample Test Cases   +

---

customS...   ⊙ ● Submit

```python
import numpy as np


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


# Generate the sequence using np.arange()
sequence = np.arange(start, stop, step)


# Print the generated sequence
print(sequence)
```

▸ Terminal   ⊞ Test cases

< Prev   Reset   Submit   Next >

3.2.4. Numpy: Arithmetic and Statistical Operations, Mathematical Operati... `08:57` ᴀA ☾ ☑ 🔗 —

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

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

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

Output Format:
 • For each operation (arithmetic, statistical, and bitwise), print the results in the specified

Sample Test Cases   +

🐍 different...   ⊙   ▶ Submit

```python
1    import numpy as np
2
3    def array_operations(A, B):
4
5        # Convert A and B to NumPy arrays
6        A = np.array(A)
7        B = np.array(B)
8
9        # Arithmetic Operations
10       sum_result = A+B
11       diff_result =A-B
12       prod_result =A*B
13
14       # Statistical Operations
15       mean_A = np.mean(A)
16       median_A = np.median(A)
17       std_dev_A = np.std(A)
18
19       # Bitwise Operations
20       and_result = A & B
21       or_result = A | B
22       xor result = A ^ B
```

▶ Terminal   ▦ Test cases

‹ Prev   Reset   Submit   Next ›

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

1. Arithmetic Operations:
- Compute the element-wise sum, difference, and product of the two arrays.

2. Statistical Operations:
- Calculate the mean, median, and standard deviation of array `A`.

3. Bitwise Operations:
- Perform bitwise `AND`, bitwise `OR`, and bitwise `XOR` on the arrays (ex: A_i `OR` B_i).

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

Output Format:
- For each operation (arithmetic, statistical, and bitwise), print the results in the specified

Sample Test Cases   +

---

different...   ⊙ Submit

```
21        or_result = A | B
22        xor_result = A ^ B
23
24        # Output results with one space between each element
25        print("Element-wise Sum:", ' '.join(map(str, sum_result)))
26        print("Element-wise Difference:", ' '.join(map(str, diff_result)))
27        print("Element-wise Product:", ' '.join(map(str, prod_result)))
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        print("Bitwise AND:", ' '.join(map(str, and_result)))
34        print("Bitwise OR:", ' '.join(map(str, or_result)))
35        print("Bitwise XOR:", ' '.join(map(str, xor_result)))
36
37    A = list(map(int, input().split()))  # Elements of array A
38    B = list(map(int, input().split()))  # Elements of array B
39    array_operations(A, B)
40
```

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

3.2.5. Numpy: Copying and Viewing Arrays

The given code takes a list of integers as input and converts it into a NumPy array. Your task is to complete the code by:

- Creating a view of the `original_array` and assigning it to `view_array`.
- Creating a copy of the `original_array` and assigning it to `copy_array`.

After completing these steps, observe how modifying the view affects the `original_array`, while modifying the copy does not.

Input Format:
- A single line of space-separated integers.

Output Format:
- After modifying the view:

```
Original array after modifying view: <original_array>
View array: <view_array>
```

- After modifying the copy:

Sample Test Cases   +

📄 copyAnd...   ⊙   ⊙ Submit

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

⌨ Terminal   ⊞ Test cases

3.2.5. Numpy: Copying and Viewing Arrays

`03:07`

The given code takes a list of integers as input and converts it into a NumPy array. Your task is to complete the code by:

- Creating a view of the `original_array` and assigning it to `view_array`.
- Creating a copy of the `original_array` and assigning it to `copy_array`.

After completing these steps, observe how modifying the view affects the `original_array`, while modifying the copy does not.

Input Format:
- A single line of space-separated integers.

Output Format:
- After modifying the view:

```
Original array after modifying view: <original_array>
View array: <view_array>
```

- After modifying the copy:

Sample Test Cases   +

📄 copyAnd...   ⊙ Submit

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

▶ Terminal   ⊞ Test cases

The given code in the editor takes a single array, `array1`, as space-separated integers as input from the user.
Additionally, it takes the following inputs:
- `search_value`: The value to search for in the array.
- `count_value`: The value to count its occurrences in the array.
- `broadcast_value`: The value to add for broadcasting across the array.

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

Input Format:
   1. A single line containing space-separated integers representing `array1`.

Sample Test Cases   +

---

arrayOpe...   ⊙   Submit

```python
1    import numpy as np
2
3    # Input array from the user
4    array1 = np.array(list(map(int, input().split())))
5
6    # Searching
7    search_value = int(input("Value to search: "))
8    count_value = int(input("Value to count: "))
9    broadcast_value = int(input("Value to add: "))
10
11   # Find indices where value matches in array1
12   a=np.where(array1==search_value)[0]
13   print(a)
14   # Count occurrences in array1
15   b=np.count_nonzero(array1==count_value)
16   print(b)
17   # Broadcasting addition
18   c= array1+broadcast_value
19   print(c)
20   # Sort the first array
21   d=np.sort(array1)
22   print(d)
```

>_ Terminal   ⊞ Test cases

3.2.7. Student Data Analysis and Operations

`40:52` AA ☾ ☑ 🔗 —

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

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Sample Test Cases   +

Operatio...

🐍 Operatio...   ⊙ ● Submit

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

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

3.2.7. Student Data Analysis and Operations

40:52

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

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Find average marks of Subject 1 and Subject 3: Compute the average marks for

Sample Test Cases +

🐍 Operatio...   ⊙ ⊙ Submit

```
27    # 8. print Total marks of students
28    print("Total Marks",np.sum(a[:,1:],axis=1))
29
30    # 9. print average marks of each student
31    avg=np.mean(a[:,1:],axis=1)
32    print(np.round(avg,1))
33    # 10. print average marks of each subject
34    print("Average Marks of each subject",np.mean(a[:,1:],axis=0))
35
36    # 11. print average marks of S1 and S2
37    print("Average Marks of S1 and S2",np.mean(a[:,1:3],axis=0))
38    # 12. print average marks of S1 and S3
39    print("Average Marks of S1 and S3",np.mean(a[:,[1,3]],axis=0))
40
41    # 13. print Roll number who got maximum marks in Subject 3
42    i=np.argmax(a[:,3])
43    print("Roll no who got maximum marks in Subject 3",a[i,0])
44
45    # 14. print Roll number who got minimum marks in Subject 2
46    mn=np.argmin(a[:,2])
47    print("Roll no who got minimum marks in Subject 2",a[mn,0])
48
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

3.2.7. Student Data Analysis and Operations   40:52   A ☾ ☑ ∂ —

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

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.
- Find overage marks of Subject 1 and Subject 2: Compute the overage marks for

Sample Test Cases   +

Operatio...   ⊙   Submit

```python
49    # 15. print Roll number who got 24 marks in Subject 2
50    whr=np.where(a[:,2]==24)
51    print("Roll no who got 24 marks in Subject 2",a[whr,0])
52
53    # 16. print count of students who got marks in Subject 1 < 40
54    ct=np.count_nonzero(a[:,1]<40)
55    print("Count of students who got marks in Subject 1 < 40",ct)
56
57    # 17. print count of students who got marks in Subject 2 > 90
58    ct1=np.count_nonzero(a[:,2]>90)
59    print("Count of students who got marks in Subject 2 > 90:",ct1)
60
61    # 18. print count of students in each subject who got marks >= 90
62    print("Count of students in each subject who got marks >=
      90:",np.count_nonzero(a[:,1:]>90,axis=0))
63
64    # 19. print count of subjects in which each student got marks >=
      90
65    print("Roll no:",a[:,0])
66    print("Count of subjects in which student got marks >=
      90:",np.count_nonzero(a[:,1:]>90,axis=1))
67
```

⯈ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

3.2.7. Student Data Analysis and Operations

40:52

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

- **Print all student details**: Display the complete details of all students, including roll numbers and marks for all subjects.
- **Find total students**: Determine the total number of students in the dataset.
- **Print all student roll numbers**: Extract and print the roll numbers of all students.
- **Print Subject 1 marks**: Extract and print the marks of all students in Subject 1.
- **Find minimum marks in Subject 2**: Identify the lowest marks in Subject 2.
- **Find maximum marks in Subject 3**: Identify the highest marks in Subject 3.
- **Print all subject marks**: Display the marks of all students for each subject.
- **Find total marks of students**: Compute the total marks for each student across all subjects.
- **Find the average marks of each student**: Compute the average marks for each student.
- **Find average marks of each subject**: Compute the average marks for all students in each subject.

Find average marks of Subject 4 and Subject 3: Compute the average marks for

Sample Test Cases   +

Operatio...   ⊙ ⊙ Submit

```python
65   print("Roll no:",a[:,0])
66   print("Count of subjects in which student got marks >=
     90:",np.count_nonzero(a[:,1:]>90,axis=1))
67
68
69   # 20. Print S1 marks in ascending order
70   srt=np.sort(a[:,1])
71   print(srt)
72
73
74   # 21. Print S1 marks >= 50 and <= 90
75   print(a[(a[:,1]>=50)&(a[:,1]<90)])
76
77
78   # 22. Print the index position of marks 79
79   print(a)
80   ip=np.where(a[:,1]==79)
81   print(ip)
82
83
84
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.1.1. Pandas - series creation and manipulation

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   +

---

seriesMa...   Submit

```python
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

series= pd.Series(numbers)
# Grouping by even and odd numbers and calculating the mean
grouped = series.groupby(series%2==0).mean()

# 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

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 +

datafram...

```python
import pandas as pd

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

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

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

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

Terminal   Test cases

4.1.2. Dictionary to dataframe   28:04 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.
  • Display the DataFrame after modifying the row.

~~Delete a row:~~

Sample Test Cases	+

```python
23
24   # Modifying a row
25   modify_index=int(input("Index of row to modify: "))
26   new_age_mod=int(input("New age: "))
27   df.loc[modify_index,"Age"]=new_age_mod
28   # Display the DataFrame after modifying a row
29   print("After modifying a row:")
30   print(df)
31
32   # Deleting a row
33   delete_index=int(input("Index of row to delete: "))
34   df=df.drop(delete_index).reset_index(drop=True)
35
36   # Display the DataFrame after deleting a row
37   print("After deleting a row:")
38   print(df)
39
40   # Adding a new column
41   gender_input=input("Enter genders separated by space: ")
42   genders=gender_input.split()
43   df["Gender"]=genders
44
```

>_ Terminal   ⊞ Test cases

4.1.2. Dictionary to dataframe   28:04 A 🌙 ☑ 𝒫 —

A dictionary of lists has been provided to you in the editor. Create a DataFrame from the dictionary of lists and perform the listed operations, then display the DataFrame before and after each manipulation.

Create the DataFrame:
 • Convert the dictionary to a Pandas DataFrame.

Add a new row:
 • Take inputs from the user for the new row data (name, age).
 • Add the new row to the DataFrame.
 • Display the DataFrame after adding the new row.

Modify a row:
 • Modify a specific row by changing the age. Take the row index and new age value from the user.
 • Display the DataFrame after modifying the row.

~~Delete a row~~

Sample Test Cases   +

---

datafram...   ⊙ ▶ Submit

```
41    gender_input=input("Enter genders separated by space.")
42    genders=gender_input.split()
43    df["Gender"]=genders
44
45
46    # Display the DataFrame after adding a new column
47    print("After adding a new column:")
48    print(df)
49
50    # Modifying a column
51    df["Name"]=df["Name"].str.upper()
52    # Display the DataFrame after modifying a column
53    print("After modifying a column:")
54    print(df)
55
56    # Deleting a column
57    df=df.drop(columns=["Age"])
58
59    # Display the DataFrame after deleting a column
60    print("After deleting a column:")
61    print(df)
62
```

▶ Terminal   ⊞ Test cases

4.1.2. Dictionary to dataframe   `28:04` ᴀ̇ ☾ ☑ ∂ —

A dictionary of lists has been provided to you in the editor. Create a DataFrame from the dictionary of lists and perform the listed operations, then display the DataFrame before and after each manipulation.

Create the DataFrame:
- Convert the dictionary to a Pandas DataFrame.

Add a new row:
- Take inputs from the user for the new row data (name, age).
- Add the new row to the DataFrame.
- Display the DataFrame after adding the new row.

Modify a row:
- Modify a specific row by changing the age. Take the row index and new age value from the user.
- Display the DataFrame after modifying the row.

~~Delete a row~~

Sample Test Cases   +

🐍 datafram...   ⊙   ⊙ Submit

```python
41   gender_input=input("Enter genders separated by space:")
42   genders=gender_input.split()
43   df["Gender"]=genders
44
45
46   # Display the DataFrame after adding a new column
47   print("After adding a new column:")
48   print(df)
49
50   # Modifying a column
51   df["Name"]=df["Name"].str.upper()
52   # Display the DataFrame after modifying a column
53   print("After modifying a column:")
54   print(df)
55
56   # Deleting a column
57   df=df.drop(columns=["Age"])
58
59   # Display the DataFrame after deleting a column
60   print("After deleting a column:")
61   print(df)
62
```

>_ Terminal   ⊞ Test cases

4.1.3. Student Information

05:38 A̲A̲ ☾ ☑ ⌲ —

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

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

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

Sample Test Cases   +

🐍 studentin... ⊗   📄 studentdat... ⊗   ⊙   ⊙ Submit

```python
import pandas as pd

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

4.2.1. Month with the Highest Total Sales

`04:57` ⋀ ☾ ☑ ⬀ —

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

- The CSV file contains the columns: Date, Product, Quantity, Price, and City.
- Group the data by Month and calculate the total sales for each month.
- Find the month with the highest total sales and display it.
- Also, display the total sales for the best month.

Sample Data:

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

Sample Test Cases   +

---

monthFor... ●   sales_dat... ●   ⊙   Submit

```python
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["sales"]=df["Quantity"].multiply(df["Price"])
9   df["month"]=pd.to_datetime(df["Date"]).dt.strftime("%Y-%m")
10
11  # Find the month with the highest total sales
12  best_month =df.groupby("month")["sales"].sum().idxmax()
13  highest_sales =df["sales"].sum()
14
15  print(f"Best month: {best_month}")
16  print(f"Total sales: ${highest_sales:.2f}")
17
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.2. Best Selling Product 22.01 AA ☾ ☑ 𝒫 —

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

- The CSV file contains the columns: Date, Product, Quantity, Price, and City.
- Find the product that sold the most in terms of quantity sold.
- Display the product that sold the most and the total quantity sold for that product.

Sample Data:

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

Sample Test Cases   +

📄 monthFor... ⊗ 📄 sales_dat... ⊗   ⊙ ⊙ Submit

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

⊵ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.3. City that Sold the Most Products

Write a Python program that takes the file name of a CSV file as input, reads the data, and performs the following operations:
- The CSV file contains the columns: Date, Product, Quantity, Price, and City.
- Group the data by City and calculate the total quantity of products sold for each city.
- Find the city that sold the most products (based on the total quantity sold).

Sample Data:

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

Sample Test Cases   +

---

monthFor... ⊗   sales_dat... ⊗   ⊙ Submit

```python
1   import pandas as pd
2
3   # Prompt the user for the file name
4   file_name = input()
5
6   # Load the data
7   df = pd.read_csv(file_name)
8
9   # write the code..
10  city_sales=df.groupby("City")["Quantity"].sum()
11  best_city = city_sales.idxmax()
12
13  # Display the result
14  print(f"City sold the most products: {best_city}")
15
```

Terminal   Test cases

4.2.4. Most Frequently Sold Product Pairs   10:02 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
2025-01-03,Product A,8,20,Los Angeles
2025-01-04,Product C,6,30,New York
2025-01-04,Product B,5,15,Los Angeles
2025-01-05,Product A,3,20,Chicago
2025-01-05,Product C,10,20,Los Angeles
```

Sample Test Cases   +

---

 frequentl... ⊗   sales_dat... ⊗   ⊙   ⊙ Submit

```python
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   date_products = {}
13   for date,group in df.groupby("Date"):
14       products=group["Product"].unique()
15       if len(products) > 1:
16           date_products[date]=products
17   pair_count = Counter()
18   for products in date_products.values():
19       pairs = combinations(sorted(products), 2)
20       pair_count.update(pairs)
21
22   if pair count:
```

 Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

4.2.4. Most Frequently Sold Product Pairs

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

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

Sample Data:

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

Sample Test Cases   +

---

frequenti... ⊗   **sales_dat...** ⊗   ⊙ ☉ Submit

```python
 8    # Read data from the specified CSV file
 9    df = pd.read_csv(file_name)
10
11    # write the code
12    date_products = {}
13    for date,group in df.groupby("Date"):
14        products=group["Product"].unique()
15        if len(products) > 1:
16            date_products[date]=products
17    pair_count = Counter()
18    for products in date_products.values():
19        pairs = combinations(sorted(products), 2)
20        pair_count.update(pairs)
21
22    if pair_count:
23        max_count=max(pair_count.values())
24        for pair,count in pair_count.items():
25            if count == max_count:
26                print(f"{pair[0]} and {pair[1]}: {count} times")
27    else:
28        print("No product pairs found.")
29    # Output the most frequent product pairs
```

▶ Terminal   ⊞ Test cases

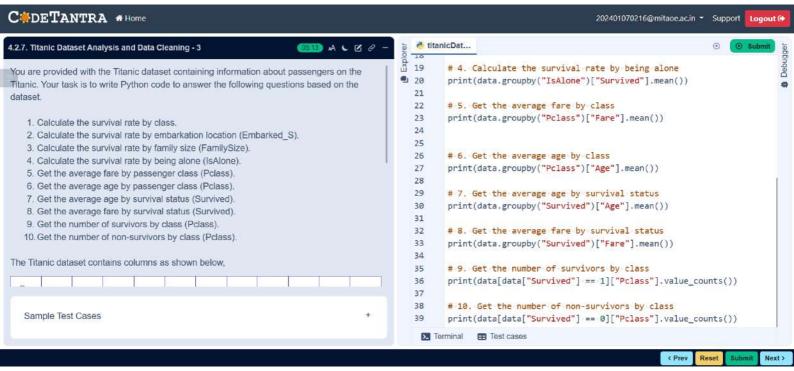## 4.2.5. Titanic Dataset Analysis and Data Cleaning   `05:53` A ☾ ☑ 🔗 —

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. For each question, perform necessary data cleaning, transformations, and calculations as required.

1. Display the first 5 rows of the dataset.
2. Display the last 5 rows of the dataset.
3. Get the shape of the dataset (number of rows and columns).
4. Get a summary of the dataset (using .info()).
5. Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
6. Check for missing values and display the count of missing values for each column.
7. Fill missing values in the 'Age' column with the median age.
8. Fill missing values in the 'Embarked' column with the most frequent value (mode).
9. Drop the 'Cabin' column due to many missing values.
10. Create a new column, 'FamilySize' by adding the 'SibSp' and 'Parch' columns.

The Titanic dataset contains columns as shown below,

Sample Test Cases   +

🐍 titanicDat...   ⊙   **Submit**

```python
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
20   print(data.describe())
21
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   median_age=data["Age"].median()
27   data["Age"].fillna(median_age, inplace=True)
```
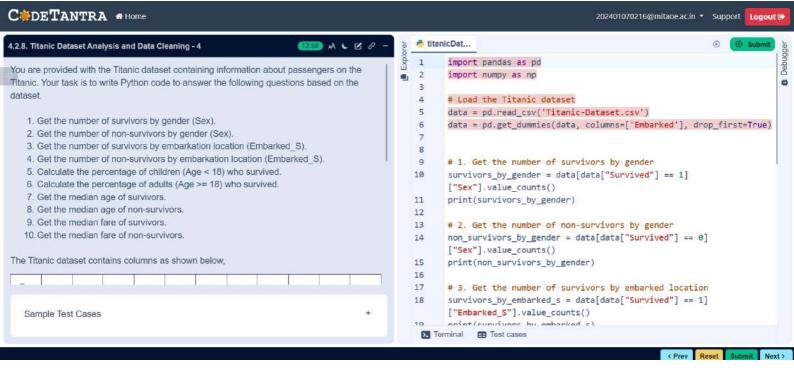
⏵ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.5. Titanic Dataset Analysis and Data Cleaning   05:53 ᴀA ☾ ☑ 𝒫 —

You are provided with the Titanic dataset containing information about passengers on the Titanic. Your task is to write Python code to answer the following questions based on the dataset. For each question, perform necessary data cleaning, transformations, and calculations as required.

1. Display the first 5 rows of the dataset.
2. Display the last 5 rows of the dataset.
3. Get the shape of the dataset (number of rows and columns).
4. Get a summary of the dataset (using .info()).
5. Get basic statistics (mean, standard deviation, etc.) of the dataset using .describe().
6. Check for missing values and display the count of missing values for each column.
7. Fill missing values in the 'Age' column with the median age.
8. Fill missing values in the 'Embarked' column with the most frequent value (mode).
9. Drop the 'Cabin' column due to many missing values.
10. Create a new column, 'FamilySize' by adding the 'SibSp' and 'Parch' columns.

The Titanic dataset contains columns as shown below,

Sample Test Cases   +

🐍 titanicDat...   ⊙ ⊙ Submit

```python
18
19    # 5. Get basic statistics of the dataset
20    print(data.describe())
21
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    median_age=data["Age"].median()
27    data["Age"].fillna(median_age, inplace=True)
28
29    # 8. Fill missing values in the 'Embarked' column with the mode
30    mode_embarked=data["Embarked"].mode()[0]
31    data["Embarked"].fillna(mode_embarked, inplace=True)
32
33    # 9. Drop the 'Cabin' column due to many missing values
34    data.drop("Cabin", axis=1, inplace=True)
35
36    # 10. Create a new column 'FamilySize' by adding 'SibSp' and
      'Parch'
37    data["FamilySize"] = data["SibSp"] + data["Parch"]
38
```
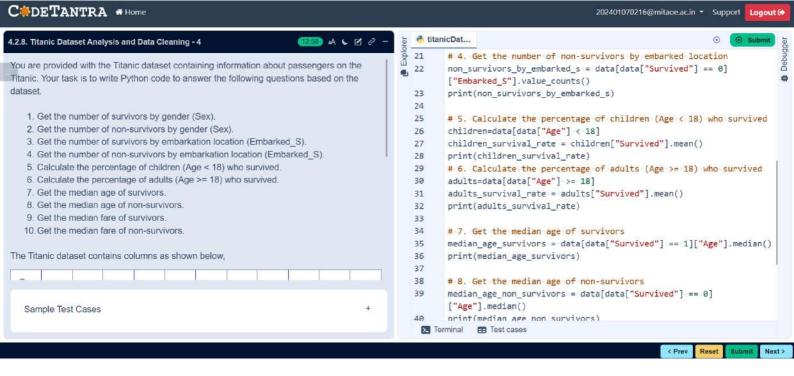
▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2

08:44 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. Create a new column 'IsAlone' which is 1 if the passenger is alone (FamilySize = 0), otherwise 0.
2. Convert the 'Sex' column to numeric values (male: 0, female: 1).
3. One-hot encode the 'Embarked' column, dropping the first category.
4. Get the mean age of passengers.
5. Get the median fare of passengers.
6. Get the number of passengers by class.
7. Get the number of passengers by gender.
8. Get the number of passengers by survival status.
9. Calculate the survival rate of passengers.
10. Calculate the survival rate by gender.

The Titanic dataset contains columns as shown below,

Sample Test Cases   +

🐍 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"] = np.where(data["FamilySize"] == 0, 1, 0)

# 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
data = pd.get_dummies(data, columns=["Embarked"], drop_first=True)

# 4. Get the mean age of passengers
mean_age = data["Age"].mean()
print(mean_age)

# 5. Get the median fare of passengers
```

▶ Terminal   ▦ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2   08:44 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.

The Titanic dataset contains columns as shown below.

```python
22   # 5. Get the median fare of passengers
23   median_fare = data["Fare"].median()
24   print(median_fare)
25
26
27   # 6. Get the number of passengers by
28   passengers_by_class = data["Pclass"].v
29   print(passengers_by_class)
30
31
32   # 7. Get the number of passengers by g
33   passengers_by_gender = data["Sex"].val
34   print(passengers_by_gender)
35
36
37   # 8. Get the number of passengers by s
38   passengers by survival =
```

4.2.6. Titanic Dataset Analysis and Data Cleaning - 2   `08:44` ⅍ ☾ ☑ ⅋ —

titanicDat...   ⊙ ⊙ Submit

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

1. Create a new column 'IsAlone' which is 1 if the passenger is alone (FamilySize = 0), otherwise 0.
2. Convert the 'Sex' column to numeric values (male: 0, female: 1).
3. One-hot encode the 'Embarked' column, dropping the first category.
4. Get the mean age of passengers.
5. Get the median fare of passengers.
6. Get the number of passengers by class.
7. Get the number of passengers by gender.
8. Get the number of passengers by survival status.
9. Calculate the survival rate of passengers.
10. Calculate the survival rate by gender.

The Titanic dataset contains columns as shown below,

Sample Test Cases   +

```python
22   # 5. Get the median fare of passengers
23   median_fare = data["Fare"].median()
24   print(median_fare)
25
26
27   # 6. Get the number of passengers by class
28   passengers_by_class = data["Pclass"].value_counts()
29   print(passengers_by_class)
30
31
32   # 7. Get the number of passengers by gender
33   passengers_by_gender = data["Sex"].value_counts().sort_index()
34   print(passengers_by_gender)
35
36
37   # 8. Get the number of passengers by survival status
38   passengers_by_survival = data["Survived"].value_counts().sort_index()
39   print(passengers_by_survival)
40
41
42   # 9. Calculate the survival rate
```

⟩ Terminal   ⊟ Test cases

‹ Prev   Reset   Submit   Next ›

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

1. Create a new column 'IsAlone' which is 1 if the passenger is alone (FamilySize = 0), otherwise 0.
2. Convert the 'Sex' column to numeric values (male: 0, female: 1).
3. One-hot encode the 'Embarked' column, dropping the first category.
4. Get the mean age of passengers.
5. Get the median fare of passengers.
6. Get the number of passengers by class.
7. Get the number of passengers by gender.
8. Get the number of passengers by survival status.
9. Calculate the survival rate of passengers.
10. Calculate the survival rate by gender.

The Titanic dataset contains columns as shown below,

Sample Test Cases   +

🐍 titanicDat...   ⊙   ⊙ Submit

```python
31
32    # 7. Get the number of passengers by gender
33    passengers_by_gender = data["Sex"].value_counts().sort_index()
34    print(passengers_by_gender)
35
36
37    # 8. Get the number of passengers by survival status
38    passengers_by_survival =
      data["Survived"].value_counts().sort_index()
39    print(passengers_by_survival)
40
41
42    # 9. Calculate the survival rate
43    survival_rate=data["Survived"].mean()
44    print(survival_rate)
45
46
47    # 10. Calculate the survival rate by gender
48    survival_rate_by_gender=data.groupby("Sex")["Survived"].mean()
49    print(survival_rate_by_gender)
50
51
```

▶ Terminal   ⊞ Test cases

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3   `05:13` ᴬA ☾ ☑ 𝒫 —

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   +

---

titanicDat...   ⊙   ⊙ Submit

```python
1   import pandas as pd
2   import numpy as np
3
4   # Load the Titanic dataset
5   data = pd.read_csv('Titanic-Dataset.csv')
6   data['FamilySize'] = data['SibSp'] + data['Parch']
7   data['IsAlone'] = np.where(data['FamilySize'] > 0, 0, 1)
8   data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
9
10  # 1. Calculate the survival rate by class
11  print(data.groupby("Pclass")["Survived"].mean())
12
13  # 2. Calculate the survival rate by embarked location
14  print(data.groupby("Embarked_S")["Survived"].mean())
15
16  # 3. Calculate the survival rate by family size
17  print(data.groupby("FamilySize")["Survived"].mean())
18
19  # 4. Calculate the survival rate by being alone
20  print(data.groupby("IsAlone")["Survived"].mean())
21
22  # 5. Get the average fare by class
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.7. Titanic Dataset Analysis and Data Cleaning - 3   05:13 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. 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   +

📄 titanicDat...   ⊙   ⊙ Submit

```python
19    # 4. Calculate the survival rate by being alone
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
26    # 6. Get the average age by class
27    print(data.groupby("Pclass")["Age"].mean())
28
29    # 7. Get the average age by survival status
30    print(data.groupby("Survived")["Age"].mean())
31
32    # 8. Get the average fare by survival status
33    print(data.groupby("Survived")["Fare"].mean())
34
35    # 9. Get the number of survivors by class
36    print(data[data["Survived"] == 1]["Pclass"].value_counts())
37
38    # 10. Get the number of non-survivors by class
39    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   12:58 ⁝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.

The Titanic dataset contains columns as shown below,

Sample Test Cases   +

titanicDat...   ⊙ **Submit**

```python
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  # 1. Get the number of survivors by gender
10 survivors_by_gender = data[data["Survived"] == 1]
   ["Sex"].value_counts()
11 print(survivors_by_gender)
12
13 # 2. Get the number of non-survivors by gender
14 non_survivors_by_gender = data[data["Survived"] == 0]
   ["Sex"].value_counts()
15 print(non_survivors_by_gender)
16
17 # 3. Get the number of survivors by embarked location
18 survivors_by_embarked_s = data[data["Survived"] == 1]
   ["Embarked_S"].value_counts()
19 print(survivors_by_embarked_s)
```

⌕ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

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   +

---

titanicDat...   ⊙ Submit

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

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

4.2.8. Titanic Dataset Analysis and Data Cleaning - 4

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   +

titanicDat...   Submit

```
31    adults_survival_rate = adults["Survived"].mean()
32    print(adults_survival_rate)
33
34    # 7. Get the median age of survivors
35    median_age_survivors = data[data["Survived"] == 1]["Age"].median()
36    print(median_age_survivors)
37
38    # 8. Get the median age of non-survivors
39    median_age_non_survivors = data[data["Survived"] == 0]
      ["Age"].median()
40    print(median_age_non_survivors)
41
42    # 9. Get the median fare of survivors
43    median_fare_survivors = data[data["Survived"] == 1]
      ["Fare"].median()
44    print(median_fare_survivors)
45
46    # 10. Get the median fare of non-survivors
47    median_fare_non_survivors = data[data["Survived"] == 0]
      ["Fare"].median()
48    print(median_fare_non_survivors)
49
```

Terminal   Test cases

5.1.1. Stacked Plot

`06:07` Aᴀ ☾ ☑ 🔗 —

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   +

🐍 stackedpl...   ⊙   Submit

```python
import pandas as pd

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

df = pd.DataFrame(data)
plt.stackplot(df["Month"],df["City_A_Temperature"],df["City_B_Temp
erature"],df["City_C_Temperature"])
plt.title("Temperature Variation")
plt.xlabel("Month")
plt.ylabel("Temperature")
plt.show()
```

⊵ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

5.2.1. Titanic Dataset   00:25

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

Dataset Information:

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

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

Visualization:

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

Sample Test Cases	+

titanicDat...   ⊙ **Submit**

```python
1   import pandas as pd
2   import matplotlib.pyplot as plt
3
4   # Load the Titanic dataset from the CSV file
5   df = pd.read_csv('titanic.csv')
6
7   # Set up the figure for 5 subplots
8   fig, axes = plt.subplots(3, 2, figsize=(12, 12))
9
10
11  axes[0,
    0].bar(df['Pclass'].value_counts().index,df['Pclass'].value_counts
    (), color='skyblue')
12  axes[0, 0].set_title("Passenger Class Distribution")
13  axes[0, 0].set_xlabel("Pclass")
14  axes[0, 0].set_ylabel("Count")
15
16
17  axes[0, 1].pie(df['Gender'].value_counts(),
    labels=df['Gender'].value_counts().index, autopct='%1.1f%%',
    colors=['lightblue', 'lightcoral'])
18  axes[0, 1].set_title("Gender Distribution")
```

▶ Terminal   ⊞ Test cases

5.2.1. Titanic Dataset

00:25 ᴀA ☾ ☑ 𝒫 −

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

Dataset Information:

The dataset is stored in a CSV file named `titanic.csv` and has been loaded using the `pandas` library. It contains the following columns:
- `Pclass`: Passenger class (1 = First, 2 = Second, 3 = Third).
- `Gender`: Gender of the passenger (male/female).
- `Age`: Age of the passenger.
- `Survived`: Survival status (0 = Did not survive, 1 = Survived).
- `Fare`: Ticket fare paid by the passenger.

Visualization:

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

Sample Test Cases   +

```
19
20    axes[1, 0].hist(df['Age'].dropna(), bins=8, color='lightgreen',
      edgecolor='black')
21    axes[1, 0].set_title("Age Distribution")
22    axes[1, 0].set_xlabel("Age")
23    axes[1, 0].set_ylabel("Frequency")
24
25    axes[1,
      1].bar(df['Survived'].value_counts().index,df['Survived'].value_co
      unts(), color=['lightblue', 'lightcoral'])
26    axes[1, 1].set_title("Survival Count")
27    axes[1, 1].set_xlabel("Survived (0 = No, 1 = Yes)")
28    axes[1, 1].set_ylabel("Count")
29
30    axes[2, 0].scatter(df['Age'], df['Fare'],
      color='orange',edgecolors='black')
31    axes[2, 0].set_title("Fare vs Age")
32    axes[2, 0].set_xlabel("Age")
33    axes[2, 0].set_ylabel("Fare")
34
35    plt.tight_layout()
36    plt.show()
```

▶ Terminal   ⊞ Test cases

5.2.2. Histogram of passenger information of Titanic   01:33

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases   +

Histogra...   ⊙   ⊙ Submit

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

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

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

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

# Write your code here for Histogram
plt.hist(data["Age"],bins=30,edgecolor="k")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title('Age Distribution')
plt.show()
```

Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

5.2.3. Bar plot of survival rate of passengers   08:34 ᴬA ☾ ☑ 𝒫 —

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases   +

BarPlotOf...   ⊙ ⊙ Submit

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

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

...hon code to plot a stacked bar chart that shows the count of passengers who
...d did not survive, grouped by gender, in the Titanic dataset. The chart should
...following specifications:
...up the data by the **'Sex'** column, then use the **value_counts()** function to count the
...urrences of survivors (0 = Did not survive, 1 = Survived) for each gender.
... a **stacked bar chart** to display the survival counts.
... the title **"Survival by Gender"** to the chart.
...el the x-axis as **'Gender'** and the y-axis as **'Count'**.
...legend should indicate **'Not Survived'** and **'Survived'**.

... dataset contains columns as shown below,

...ur iv d	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Test Cases   +

BarPlotOf...

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

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

# Write your code here for Bar Plot for Survival by Gender

survival_by_gender=data.groupby("Sex")
["Survived"].value_counts().unstack().fillna(0)
survival_by_gender.columns=["Not Survived", "Survived"]
survival_by_gender.index=["0","1"]
survival_by_gender.plot(kind="bar",stacked=True)
plt.title("Survival by Gender")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.legend(title=None)
plt.show()
```

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

5.2.5. Bar Plot for Survival by Pclass `01:20` ᴬA ☾ ✎ 𝒫 —

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases   +

🐍 BarPlotOf...   ⊙   ⊙ Submit   ⚙ Debugger

```
 7    # Data Cleaning
 8    data['Age'].fillna(data['Age'].median(), inplace=True)
 9    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10    data.drop('Cabin', axis=1, inplace=True)
11
12    # Convert categorical features to numeric
13    data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14    data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16    # Write your code here for Bar Plot for Survival by Pclass
17    survival_by_class=data.groupby("Pclass")
      ["Survived"].value_counts().unstack().fillna(0)
18    survival_by_class.columns=["Not Survived", "Survived"]
19    survival_by_class.plot(kind="bar",stacked=True)
20    plt.title("Survival by Pclass")
21    plt.xlabel("Pclass")
22    plt.ylabel("Count")
23    plt.legend(title=None)
24    plt.show()
25
26
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

5.2.6. Bar Plot for Survival by Embarked

02:25

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 sse ng	Sur viv	Pcl	Na	Se	Ag	Sib	Par	Tic	Far	Ca	Em bar

Sample Test Cases   +

BarPlotOf...   ⊙ Submit

```python
6
7    # Data Cleaning
8    data['Age'].fillna(data['Age'].median(), inplace=True)
9    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10   data.drop('Cabin', axis=1, inplace=True)
11
12   # Convert categorical features to numeric
13   data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14   data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16   # Write your code here for Bar Plot for Survival by Embarked
17   grouped=data.groupby("Embarked_Q")
     ["Survived"].value_counts().unstack().fillna(0)
18   grouped.columns=["Not Survived", "Survived"]
19   grouped.plot(kind="bar",stacked=True)
20   plt.title("Survival by Embarked")
21   plt.xlabel("Embarked")
22   plt.ylabel("Count")
23   plt.legend(title=None)
24   plt.show()
25
26
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

5.2.7. Box plot for Age Distribution   `02:10` ⩗ ☾ ☑ ⊘ —

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases   +

BoxPlotF...   ⊙   ▶ Submit

```python
 4    # Load the Titanic dataset
 5    data = pd.read_csv('Titanic-Dataset.csv')
 6
 7    # Data Cleaning
 8    data['Age'].fillna(data['Age'].median(), inplace=True)
 9    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10    data.drop('Cabin', axis=1, inplace=True)
11
12    # Convert categorical features to numeric
13    data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14    data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16    # Write your code here for Box Plot for Age by Pclass
17    plt.figure(figsize=(8,6))
18    data.boxplot(column="Age", by="Pclass")
19    plt.suptitle("")
20    plt.title("Age by Pclass")
21    plt.xlabel("Pclass")
22    plt.ylabel("Age")
23    plt.show()
24
25
```

▶ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

5.2.8. Box Plot for Age by Survived

`02:47` ‹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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases   +

🐍 BoxPlotF...   ⊙   ⊙ Submit

```python
6
7   # Data Cleaning
8   data['Age'].fillna(data['Age'].median(), inplace=True)
9   data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10  data.drop('Cabin', axis=1, inplace=True)
11
12  # Convert categorical features to numeric
13  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14  data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16  # Write your code here for Box Plot for Age by Survived
17  plt.figure(figsize=(8,6))
18  data.boxplot(column="Age", by="Survived")
19  plt.suptitle("")
20  plt.title("Age by Survival")
21  plt.xlabel("Survived")
22  plt.ylabel("Age")
23  plt.show()
24
25
26
27
```

Terminal   Test cases

‹ Prev   Reset   Submit   Next ›

5.2.9. Box Plot for Fare by Pclass　　　`01:03` ᴬᴬ ☽ ☑ ∥ —

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases　　　　　　　　　　　　　　　　　　　+

🐍 BoxPlotF...　　　　　　　　　　　　　　　　　　ⓘ　**⊙ Submit**

```python
 7   # Data Cleaning
 8   data['Age'].fillna(data['Age'].median(), inplace=True)
 9   data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10   data.drop('Cabin', axis=1, inplace=True)
11
12   # Convert categorical features to numeric
13   data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14   data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16   # Write your code here for Box Plot for Fare by Pclass
17   plt.figure(figsize=(8,6))
18   data.boxplot(column="Fare", by="Pclass")
19   plt.suptitle("")
20   plt.title("Fare by Pclass")
21   plt.xlabel("Pclass")
22   plt.ylabel("Fare")
23   plt.show()
24
25
26
27
```

▶ Terminal　　⊞ Test cases

‹ Prev　Reset　Submit　Next ›

5.2.10. Scatter Plot for Age vs. Fare
`02:09` ⅍A ☾ ☑ 𝒫 —

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Data:

Sample Test Cases   +

AgeFareS...   ⊙ ⊙ Submit

```python
 5    data = pd.read_csv("Titanic-Dataset.csv")
 6
 7    # Data Cleaning
 8    data['Age'].fillna(data['Age'].median(), inplace=True)
 9    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
10    data.drop('Cabin', axis=1, inplace=True)
11
12    # Convert categorical features to numeric
13    data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
14    data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
15
16    # Write your code here for Box Plot for Fare by Pclass
17    plt.figure(figsize=(6.4,4.8))
18    data.scatter(data["Age"],data["Fare"])
19    plt.title("Age vs. Fare")
20    plt.xlabel("Age")
21    plt.ylabel("Fare")
22    plt.show()
23
24
25
26
```

▷ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit   Next ›

5.2.10. Scatter Plot for Age vs. Fare

`09:47` 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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Data:

Sample Test Cases   +

🐍 AgeFareS...   ⊙ ⊙ Submit

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

▶ Terminal   ⊞ Test cases

‹ Prev Reset Submit Next ›

5.2.11. Scatter Plot for Age vs. Fare by Survived   `03:17` ᴬA ☾ ☑ 𝒫 —

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,

Pa sse ng erl d	Sur viv ed	Pcl ass	Na me	Se x	Ag e	Sib Sp	Par ch	Tic ket	Far e	Ca bin	Em bar ked

Sample Test Cases   +

AgeFareS...   ⊙   ✅ Submit

```python
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  colors = data['Survived'].map({0: 'red',1: 'blue'})
17  plt.scatter(data['Age'],data['Fare'],c=colors)
18  plt.title('Age vs. Fare by Survival')
19  plt.xlabel('Age')
20  plt.ylabel('Fare')
21  plt.show()
22
```

⬛ Terminal   ⊞ Test cases

‹ Prev   Reset   Submit

Essentials of Data Science Laboratory - 2304102L - 2304102L

Dashboard   Contents   Calendar   Assessments   Syllabus

100%

Completed

Course progress

0%   100%

⟳ 0 units left   **Completed**

Resume >

January

		1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

February

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

March

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

April

	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

May

			1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

← Prev   Next →

Less ▮▮▮▮ More

2000   4000   6000

Description

Essentials of Data Science Laboratory - 2304102L

Upcoming tests

No upcoming exams in the next 7 days

ılıılı. Networking
CISCO. Academy

This certificate is awarded to

AASHMI PATRO

for successfully completing

Python Essentials 1

offered by MIT Academy of Engineering, alandi Pune
through the Cisco Networking Academy program.

Anuradha Pawar
Instructor
MIT Academy of Engineering, alandi Pune

08 Apr 2025
Completion Date

illlilli Networking
CISCO. Academy

This certificate is awarded to

AASHMI PATRO

for successfully completing

Python Essentials 2

offered by Networking Academy
through the Cisco Networking Academy program.

Lynn Bloomer

Lynn Bloomer
Director
Cisco Networking Academy

10 Apr 2025
Completion Date