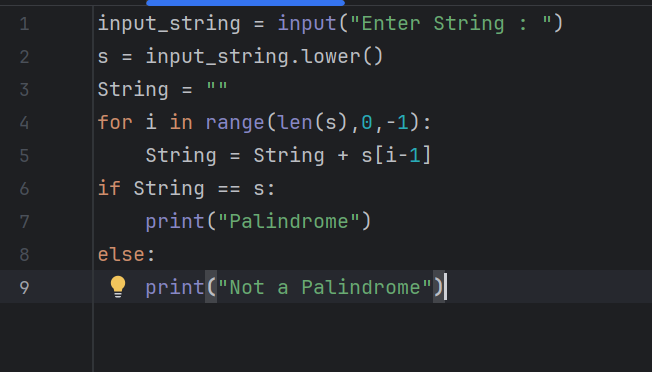
**IT11 & ITL11 - Python Programming**

**Lab Assignment 1**

================================================================

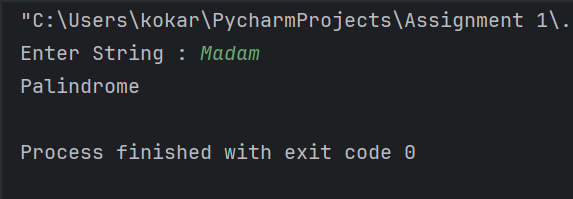
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 1 | Write a program to determine if a given string is a palindrome or not using combination of positive and negative indexing. Take the string as an input from the user. | “Madam”  “College” | Palindrome  Not a palindrome |

**Program :**

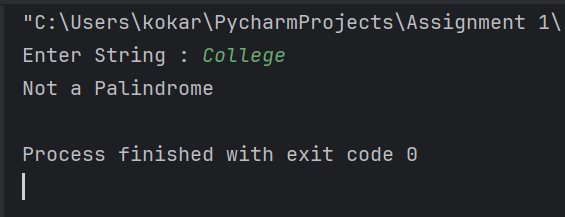


**Output :**

1. “Madam”

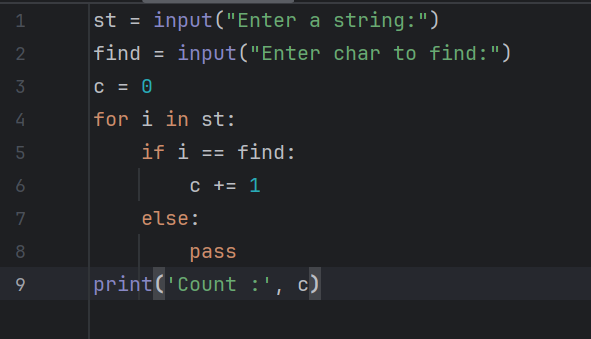


2. “College”

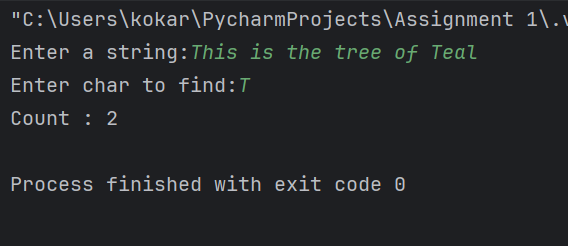


|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 2 | Without using count() demonstrate the use of for loop to determine the number of occurrences of a given character in a string. Take the string and character from the user. | String : This is the tree of Teal Character ‘T’ | 2 |

**Program :**

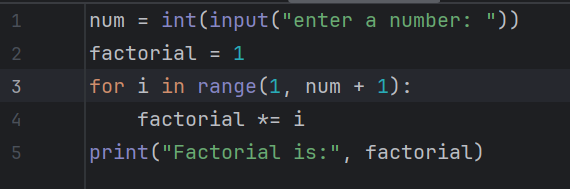


**Output :**

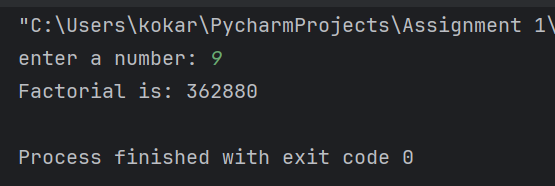


|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 3 | Without using readymade methods, write a program to find factorial of a given number. Take the number from the user. | 9 | 362880 |

**Program :**



**Output :**

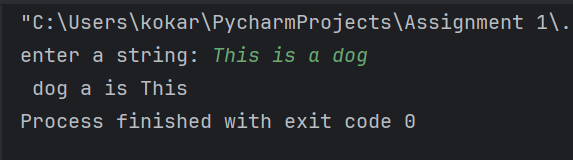


|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 4 | Without using any readymade methods, write a program in Python to reverse the words in a given string. Take the string from the user. | This is a dog | dog a is This |

**Program :**

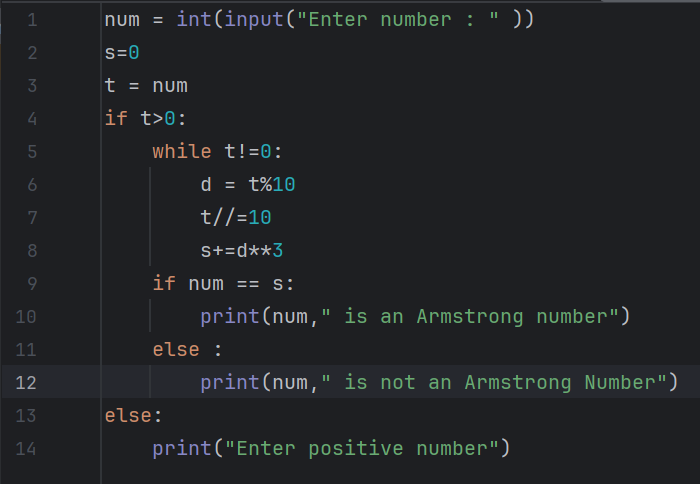


**Output :**



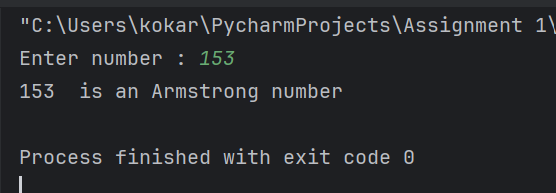
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 5 | Without using any readymade methods, write a program in Python to check if the given number is an Armstrong number or not. Take the number from the user.  Eg : If the number is 153 O/p - 153 is an Armstrong number. If the number is 25 O/p - 25 is not an Armstrong number | 153  25 | 153 is an Armstrong number  25 is not an Armstrong number |

**Program :**

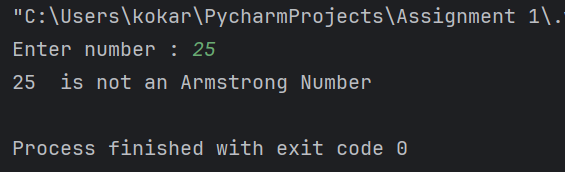


**Output :**

1. 153

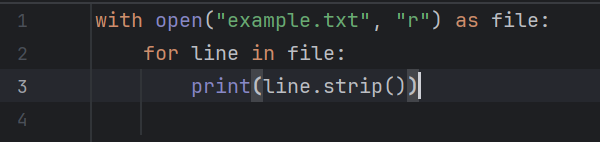


2. 25

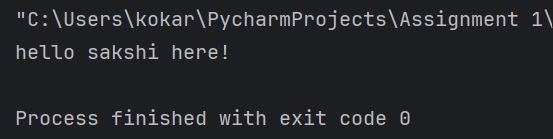


|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 6 | Without using readline() demonstrate a way in Python to read a multi line file line by line. |  |  |

**Program :**

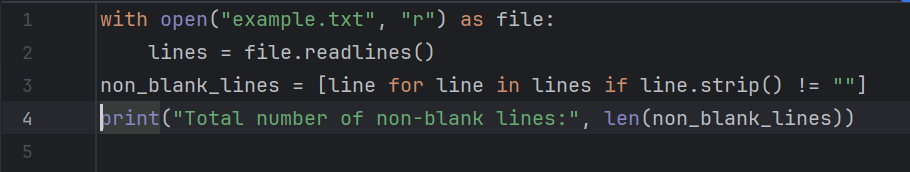


**Output :**

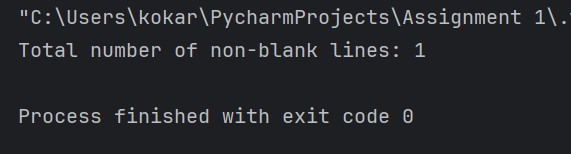


|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 7 | Using readlines() demonstrate a way to return the total number of NON BLANK lines in a file |  |  |

**Program :**

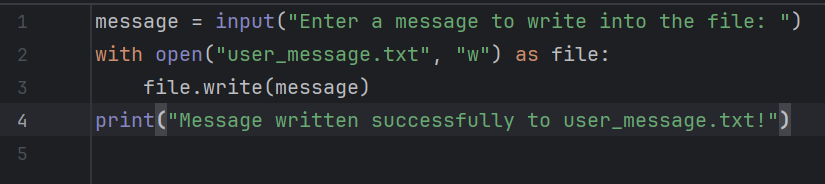


**Output :**



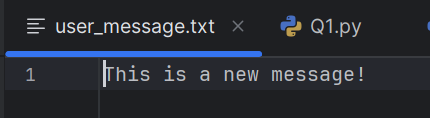
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 8 | Using file writing methods, to write a message from the user to a file. |  |  |

**Program :**



**Output :**



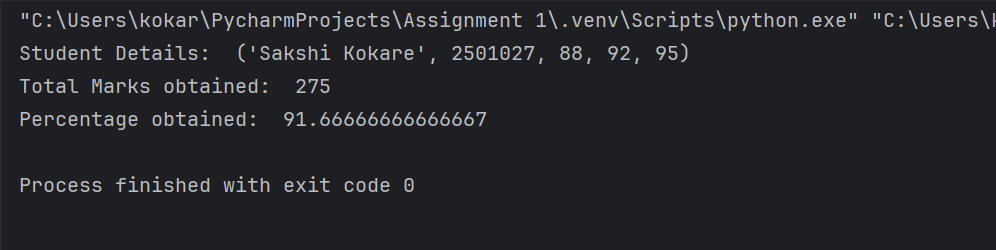


|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 9 | Write a class Student having attributes, name, rollNumber, mathsMks, scienceMks and engMks. Use getters and setters for these attributes.  Write another class Marksheet having the attributes totalMks and percentage. Define a method calculateMarks() and calculatePercentage().  Create a student class object in Marksheet class. Assign name, roll number, maths, science and english marks to the student class object. Invoke calculateMarks() and calculatePercentage() using the data of this Student object |  |  |

**Program :**

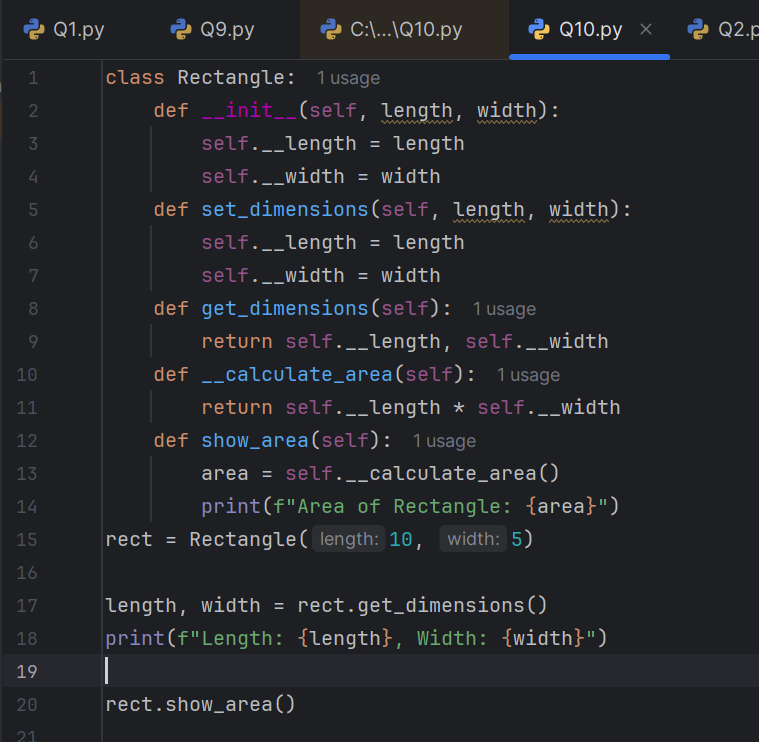


**Output :**



|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 10 | Using the concept of class, public and non-public attributes and methods write a program to calculate the area of a rectangle. |  |  |

**Program :**

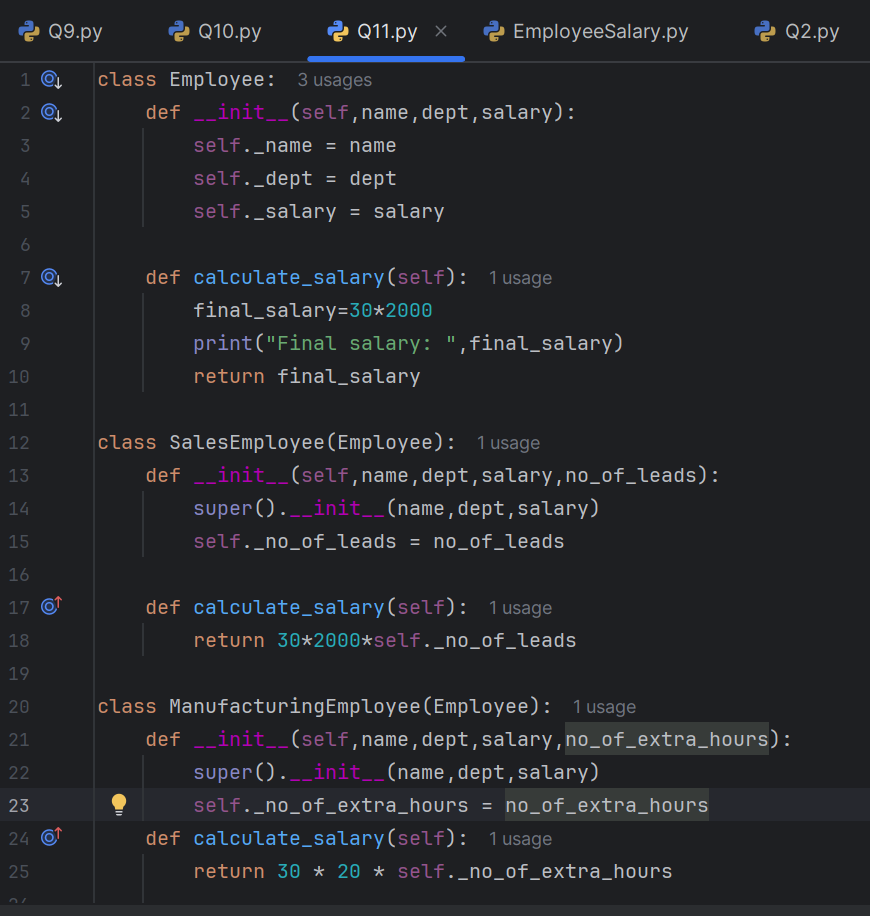


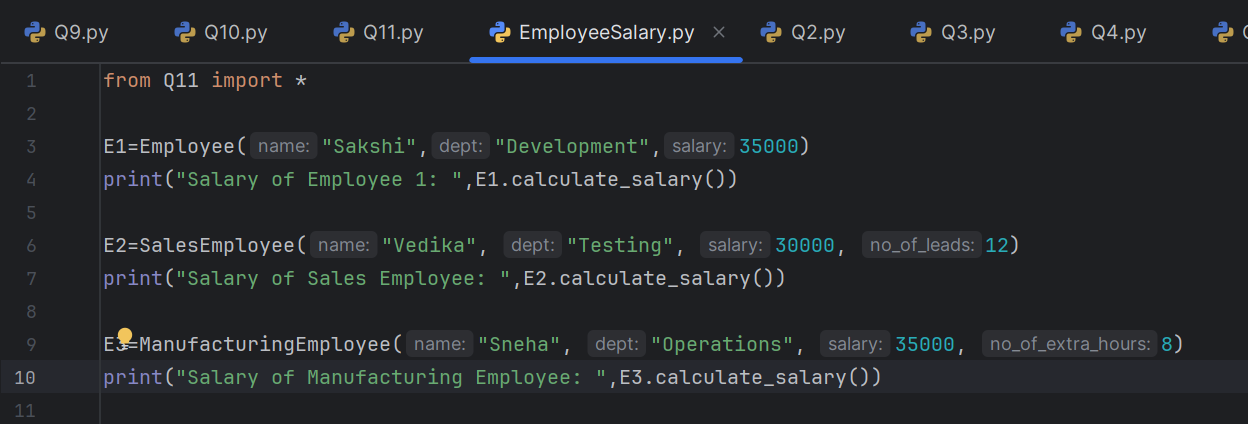
**Output :**



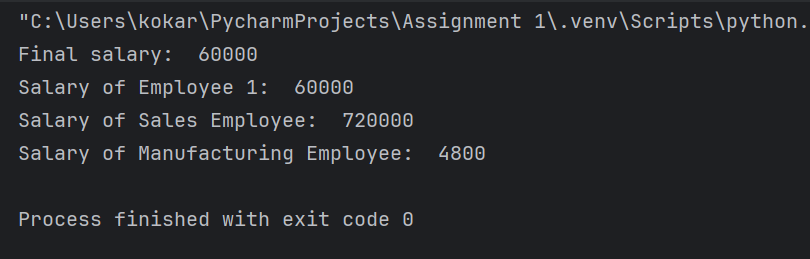
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 11 | Write a class Employee having attributes name, dept, sal. Add a method, calculate\_salary(). This method should calculate the salary using the logic, 30\*2000. Print the final salary calculated. Write a subclass, SalesEmployee having attribute no\_of\_leads. Override calculate\_salary() which uses the formula, salary = 30\*2000\*no\_of\_leads. Write another subclass, ManufacturingEmployee having attribute no\_of\_extra\_hours. Override calculate\_salary() which uses the formula, salary = 30\*20\*no\_of\_extra+hours. In a separate file, EmployeeSalary.py create objects of these classes and invoke their respective calculate\_salary(). |  |  |

**Program :**



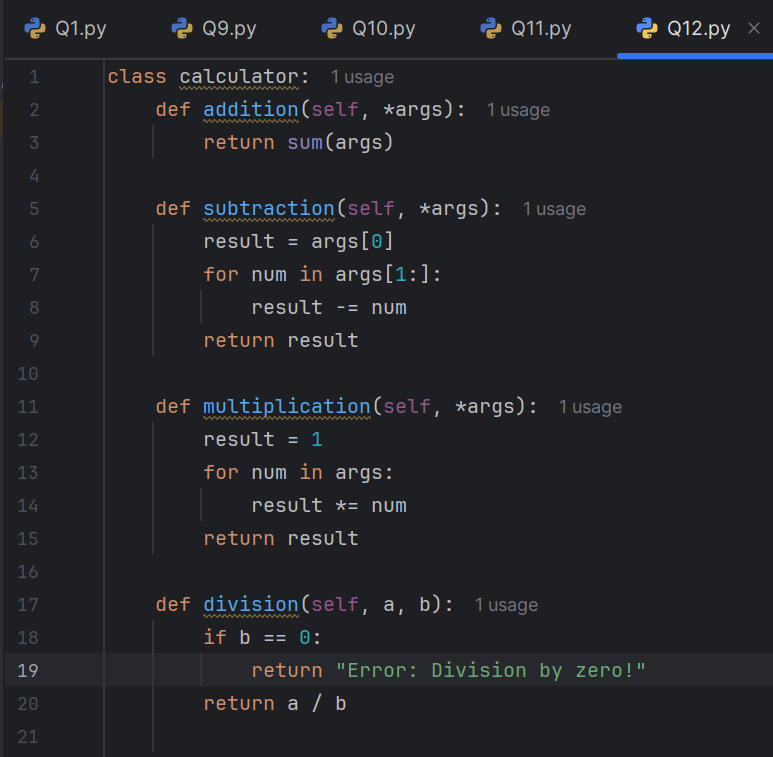


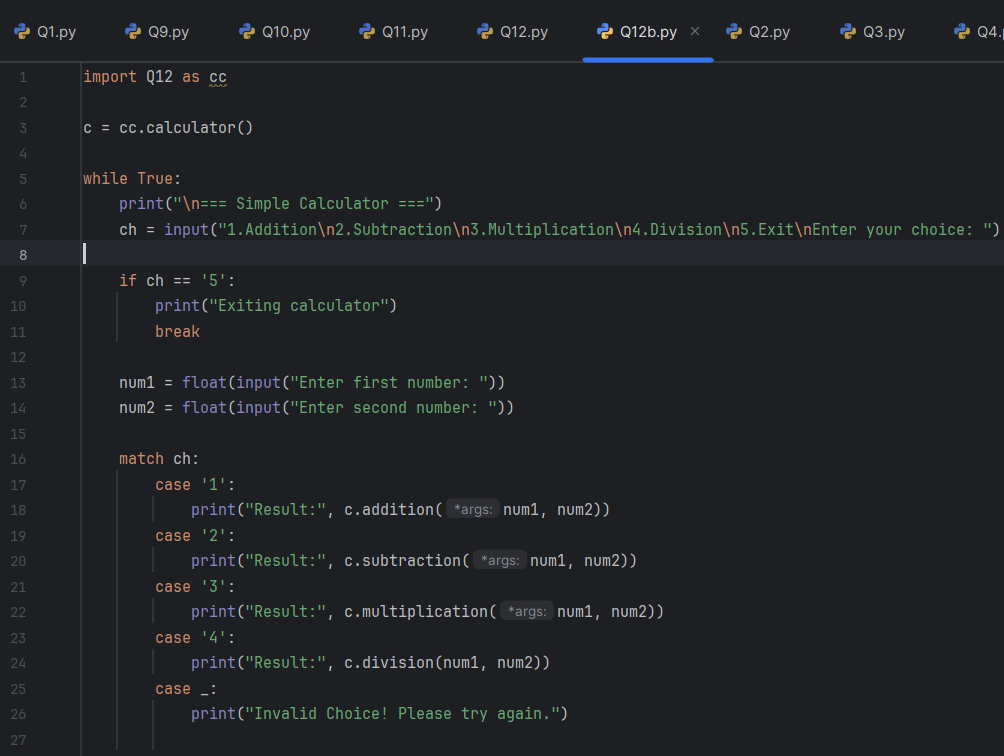
**Output :**



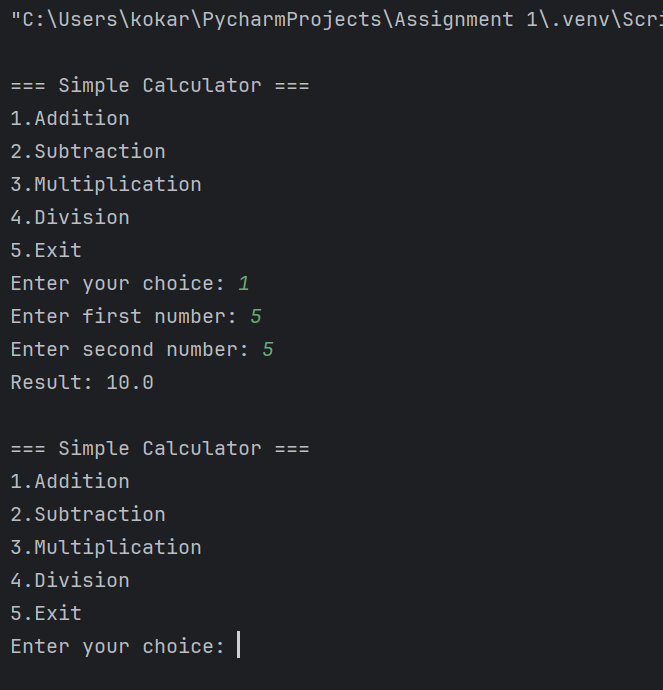
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Assignment** | **Test data Input** | **Test data output** |
| 12 | Design a calculator utility module having methods for addition, subtraction, division and multiplication. Use this module in a different file which takes the numbers from the user and the choice of operation. Exhibit support for arbitrary arguments in addition, subtraction and multiplication methods. |  |  |

**Program :**





**Output:**



**================================================================**

**Practical Assignment\_Numpy Case Studies**

**Case Study 1: Student Marks Analysis**

**Scenario:** A class of 5 students has marks in 3 subjects. You need to calculate the

average marks per student and per subject.

**Task:** Use NumPy arrays to store the data and apply mean() across rows and columns.

**Input:**

marks = np.array([[85, 90, 78],

[88, 92, 80],

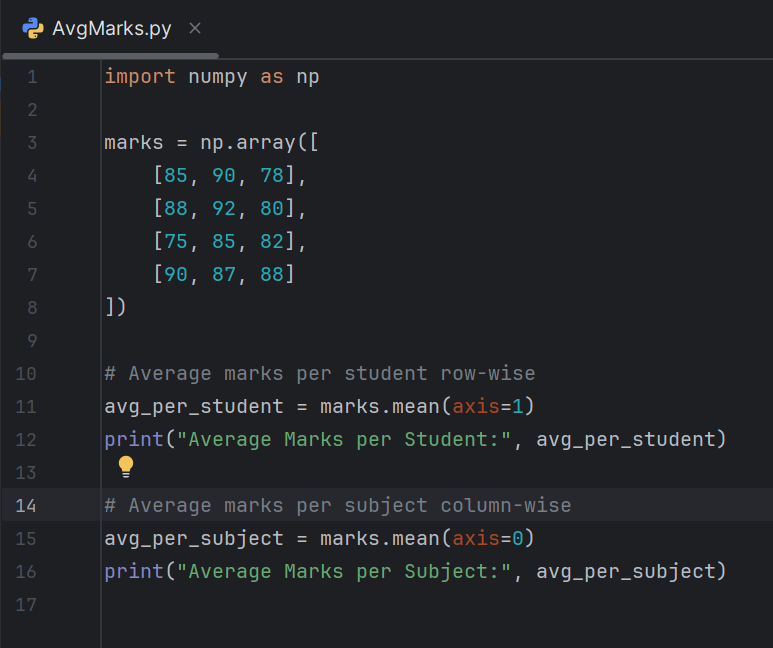
[75, 85, 82],

[90, 87, 88]])

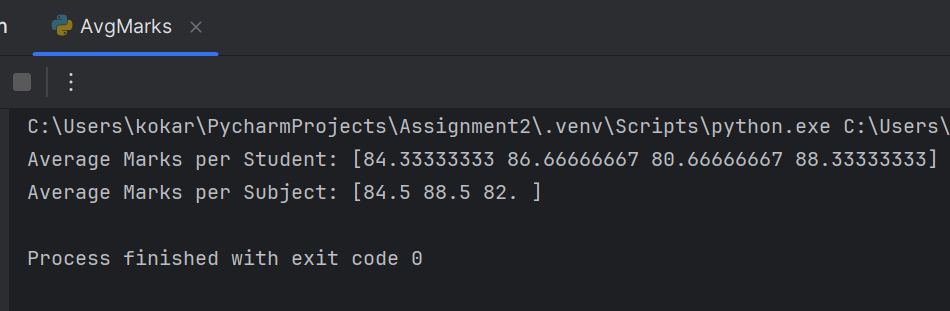
**Expected Output:**

[84.5 88.5 82.0]

**SOLUTION:**

****

**Output:**

****

**Case Study 2: Scalar Operations on Temperature Data**

**Scenario:** A weather station records daily temperatures in Celsius. You want to convert them to Fahrenheit.

**Task:** Use a NumPy array and apply scalar operations: F = C \* 9/5 + 32.

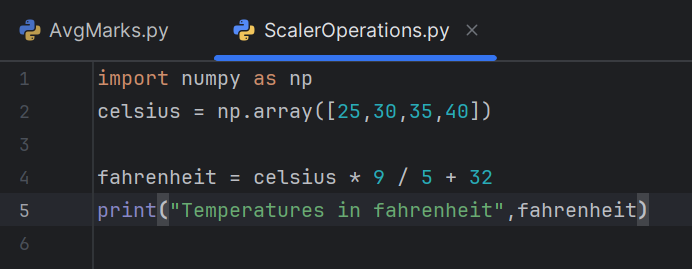
**Input:**

celsius = np.array([25, 30, 35, 40])

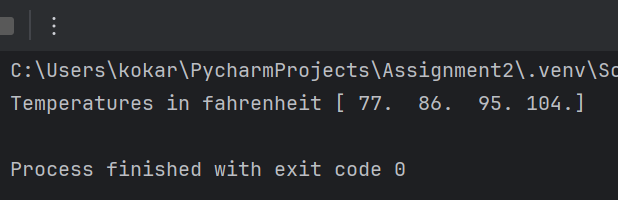
**Expected Output:**

[77. 86. 95. 104.]

**SOLUTION:**

****

**Output:**

****

**Case Study 3: Monthly Sales Growth**

**Scenario:** A company tracks monthly sales for two products. You need to compute the percentage growth month-over-month.

**Task:** Use NumPy arrays and apply element-wise operations to calculate growth using (current - previous) / previous \* 100.

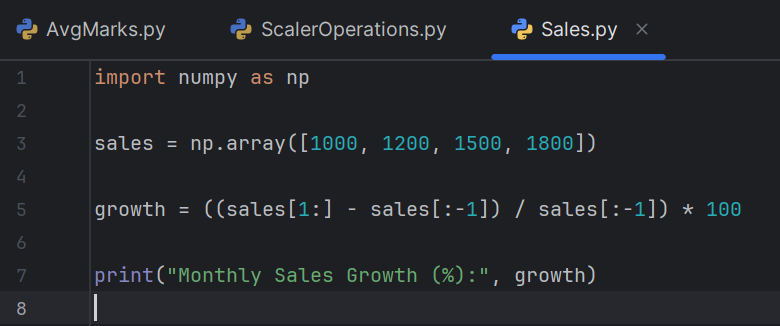
**Input:**

sales = np.array([1000, 1200, 1500, 1800])

**Expected Output:**

[20. 25. 20.]

**SOLUTION:**

****

**Output:**

****

**Case Study 4: Sensor Data Normalization**

**Scenario:** IoT sensors produce raw data between 0 and 1023. You need to normalize it to a 0–1 scale.

**Task:** Use universal functions to apply normalization: normalized = (data - min) / (max - min).

**Input:**

data = np.array([100, 300, 500, 700, 900])

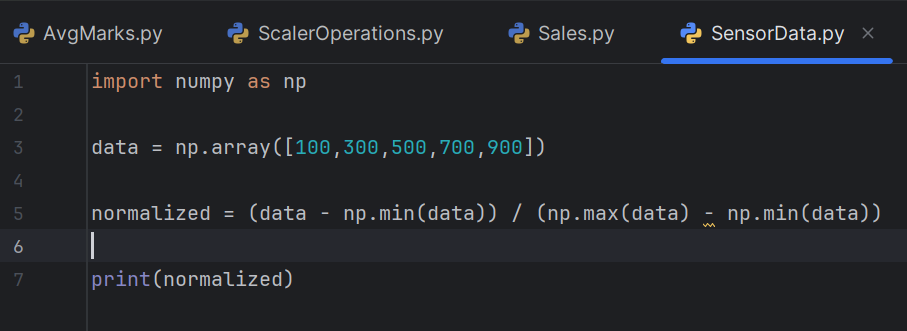
normalized = (data - np.min(data)) / (np.max(data) - np.min(data))

print(normalized)

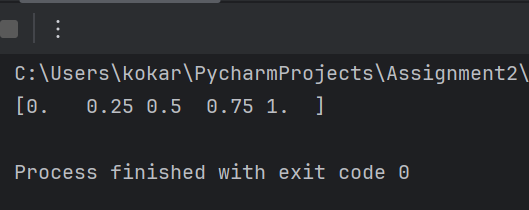
**Expected Output:**

[0. 0.25 0.5 0.75 1. ]

**SOLUTION:**



**Output:**

****

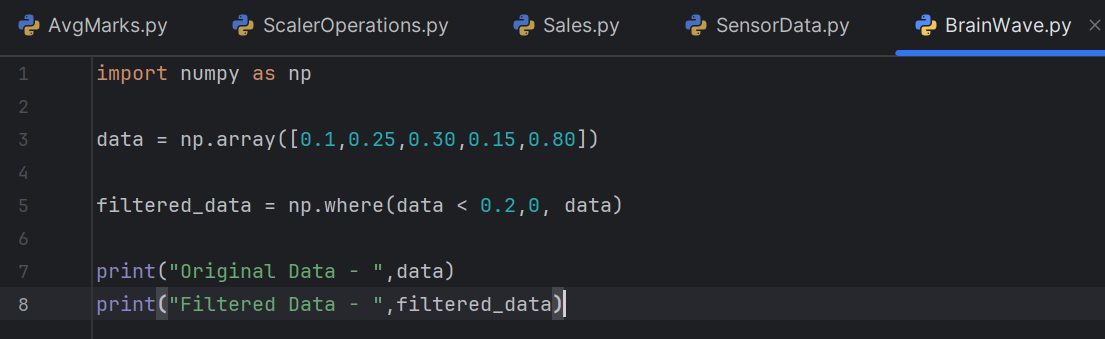
**Case Study 5: Brainwave Signal Filtering**

**Scenario:** EEG data contains noise. You want to apply a threshold filter to remove

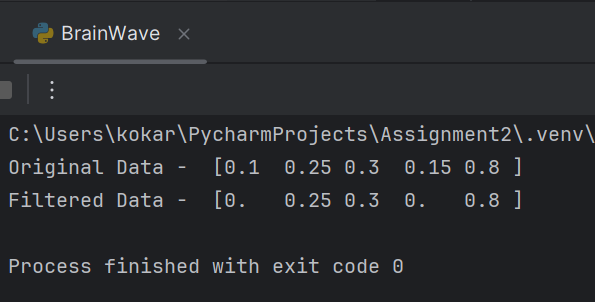
values below 0.2.

**Task:** Use NumPy’s boolean indexing and where() to filter and replace values.

**SOLUTION:**

****

**Output:**

****

**================================================================**

**Practical Assignment\_Pandas Case Studies**

**Case Study 1: Creating a Series of Student Scores**

**Scenario:** A teacher wants to store scores of 5 students in a subject.

**Input:**

import pandas as pd

scores = pd.Series([85, 90, 78, 92, 88], index=['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha'])

print(scores)

**Expected Output:**

Amit 85

Bhavna 90

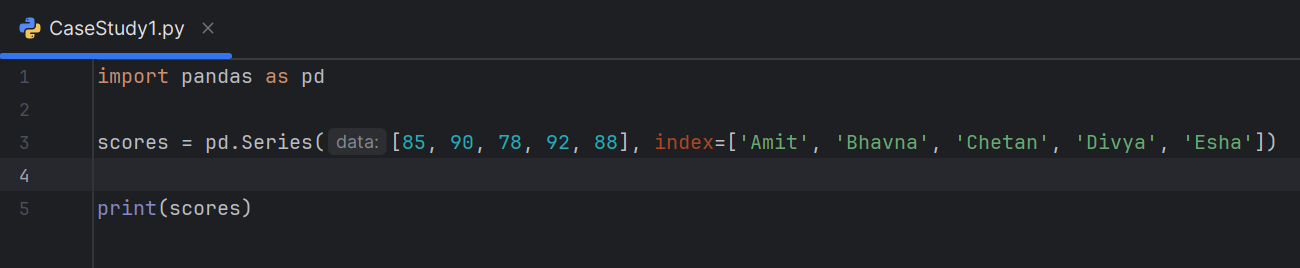
Chetan 78

Divya 92

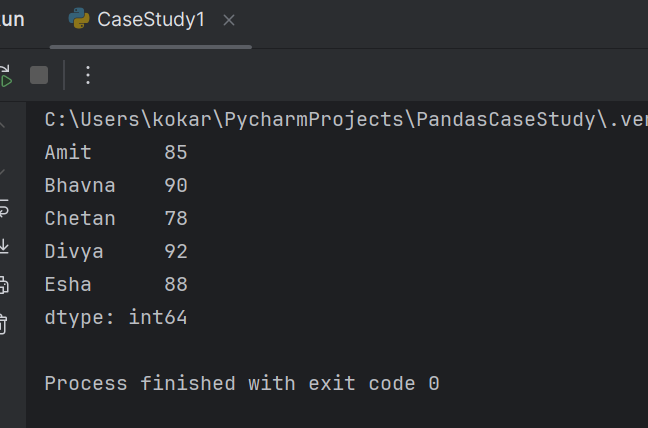
Esha 88

dtype: int64

**Program:**



**Output:**



**Case Study 2: Creating a DataFrame of Student Marks**

**Scenario:** Store marks of 3 subjects for 4 students.

**Input:**

data = {'Math': [85, 78, 92, 88],

'Science': [90, 82, 95, 89],

'English': [88, 76, 91, 84]}

df = pd.DataFrame(data, index=['Amit', 'Bhavna', 'Chetan', 'Divya'])

print(df)

**Expected Output:**

Math Science English

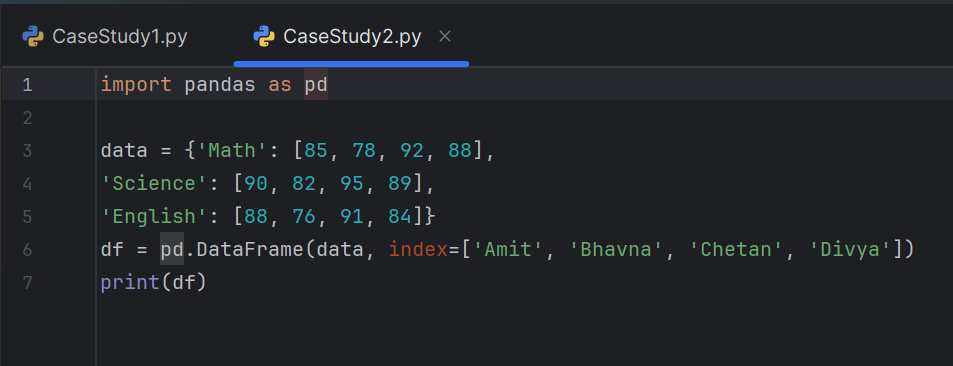
Amit 85 90 88

Bhavna 78 82 76

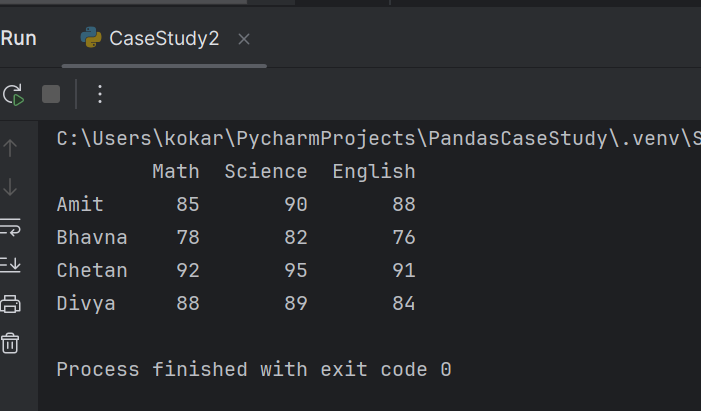
Chetan 92 95 91

Divya 88 89 84

**Program:**

****

**Output:**

****

**Case Study 3: Handling Missing Data**

**Scenario:** Some student scores are missing. Fill them with the subject average.

**Input:**

df.loc['Bhavna', 'Science'] = None

df.fillna(df.mean(), inplace=True)

print(df)

**Expected Output:**

Math Science English

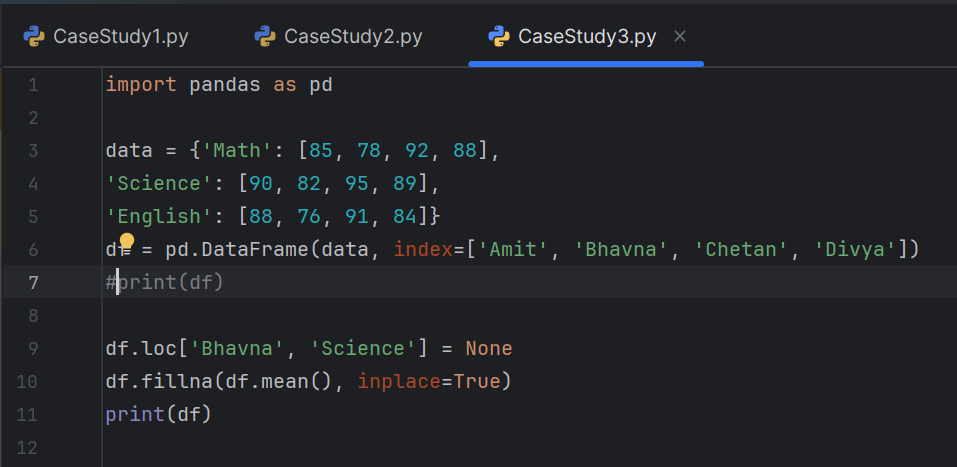
Amit 85 90.0 88

Bhavna 78 91.3 76

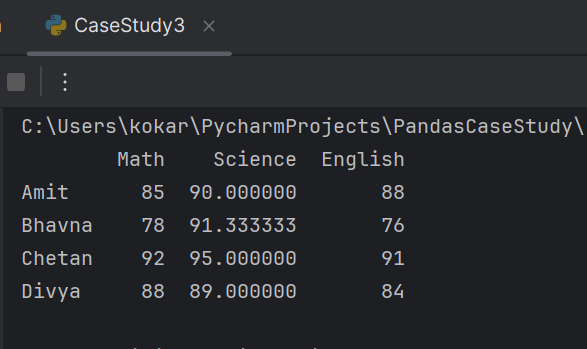
Chetan 92 95.0 91

Divya 88 89.0 84

**Program:**

****

**Output:**

****

**Case Study 4: Dropping an Entry**

**Scenario:** Drop a student who withdrew from the course.

**Input:**

df.drop('Chetan', inplace=True)

print(df)

**Expected Output:**

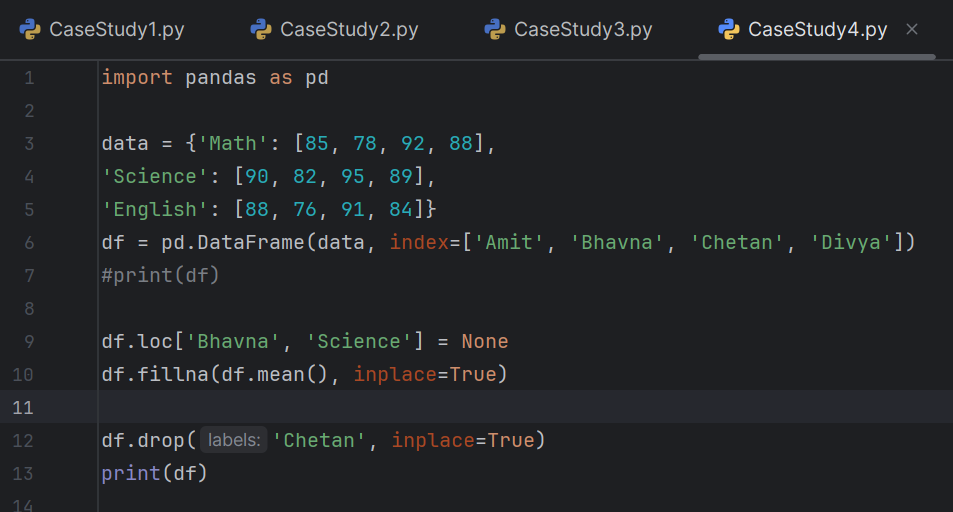
Math Science English

Amit 85 90.0 88

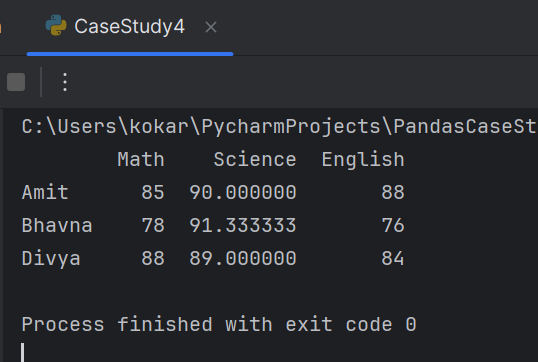
Bhavna 78 91.3 76

Divya 88 89.0 84

**Program:**

****

**Output:**

****

**Case Study 5: Selecting Entries by Condition**

**Scenario:** Select students who scored above 85 in Math.

**Input:**

high\_math = df[df['Math'] > 85]

print(high\_math)

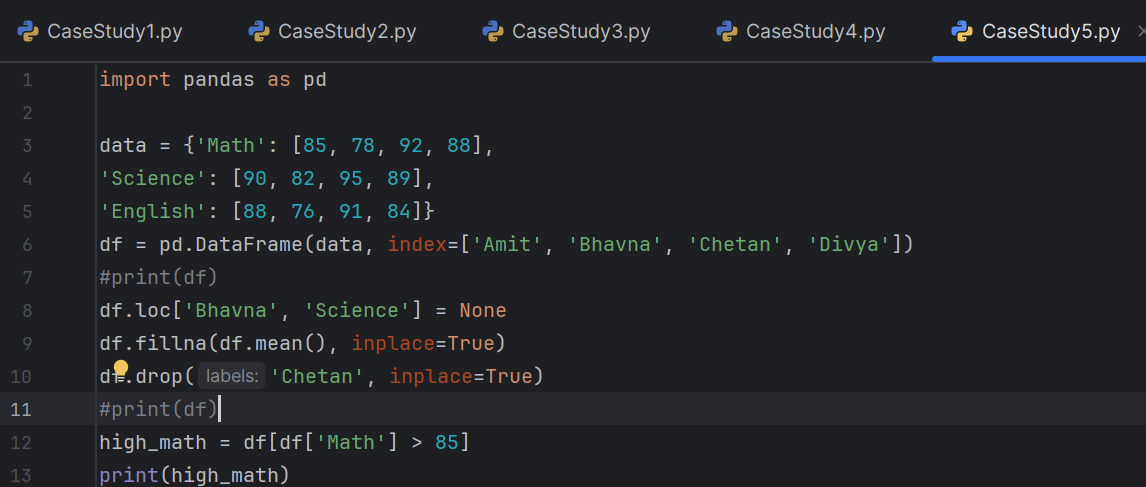
**Expected Output:**

Math Science English

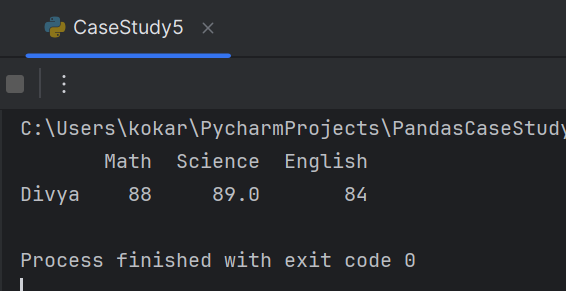
Amit 85 90.0 88

Divya 88 89.0 84

**Program:**

****

**Output:**

****

**Case Study 6: Data Alignment Between Two DataFrames**

**Scenario:** Align marks from two subjects stored in separate DataFrames.

**Input:**

math = pd.Series([85, 78, 92], index=['Amit', 'Bhavna', 'Chetan'])

science = pd.Series([90, 82, 95], index=['Bhavna', 'Chetan', 'Divya'])

aligned = pd.DataFrame({'Math': math, 'Science': science})

print(aligned)

**Expected Output:**

Math Science

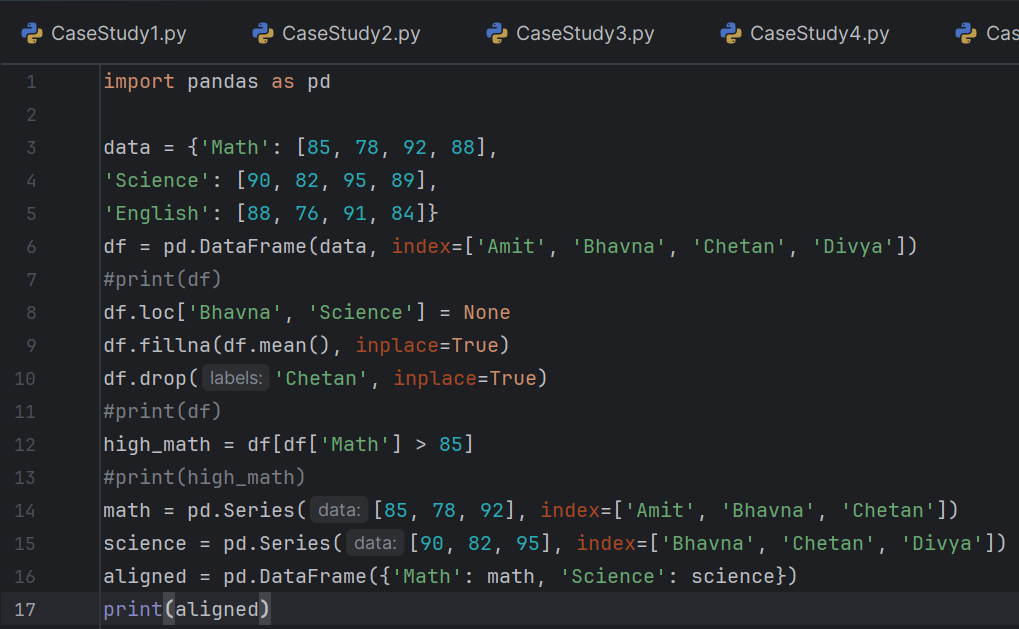
Amit 85.0 NaN

Bhavna 78.0 90.0

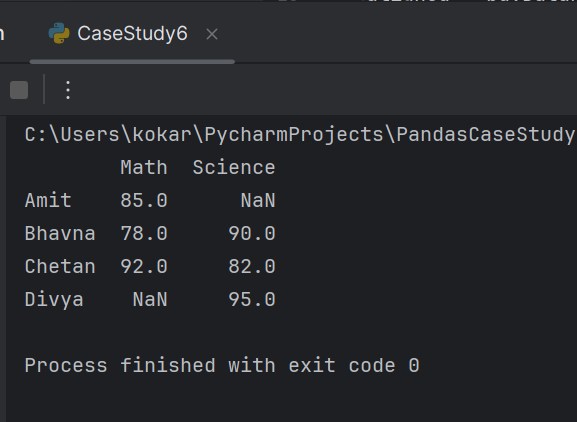
Chetan 92.0 82.0

Divya NaN 95.0

**Program:**

****

**Output:**

****

**Case Study 7: Ranking Students by Total Marks**

**Scenario:** Rank students based on total marks.

**Input:**

df['Total'] = df.sum(axis=1)

df['Rank'] = df['Total'].rank(ascending=False)

print(df[['Total', 'Rank']])

**Expected Output:**

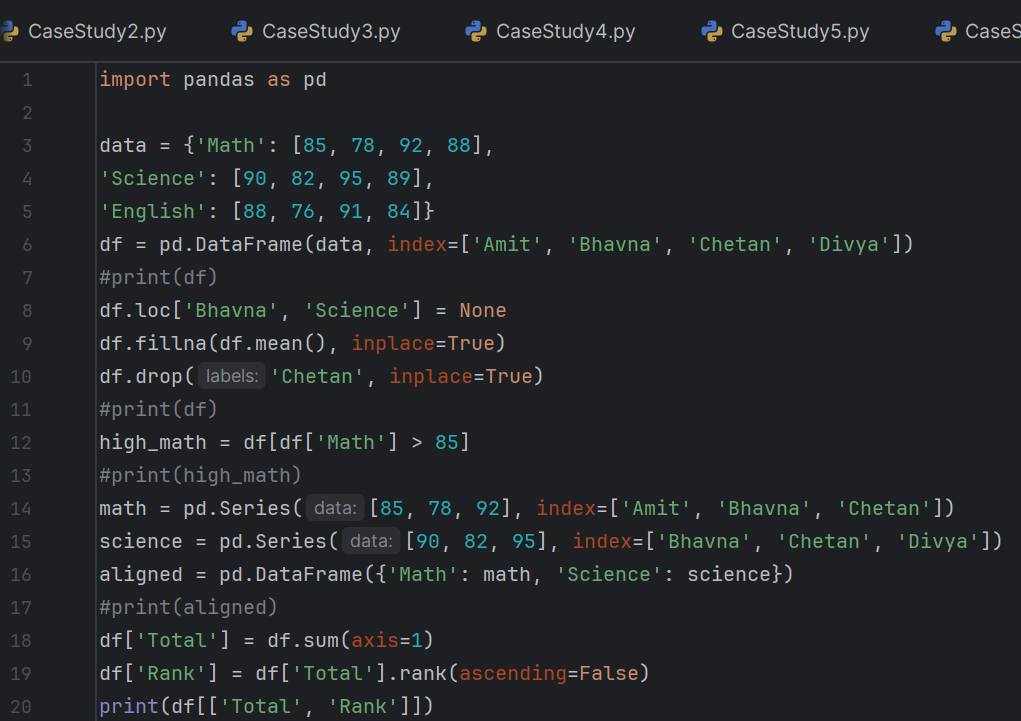
Total Rank

Amit 263.0 2.0

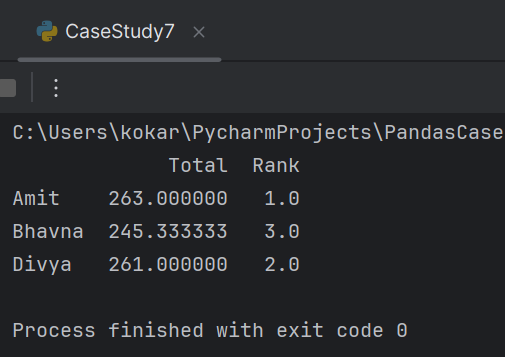
Bhavna 245.3 3.0

Divya 261.0 1.0

**Program:**

****

**Output:**

****

**Case Study 8: Sorting by English Marks**

**Scenario:** Sort students by their English scores.

**Input:**

sorted\_df = df.sort\_values(by='English', ascending=False)

print(sorted\_df)

**Expected Output:**

Math Science English Total Rank

Amit 85 90.0 88 263.0 2.0

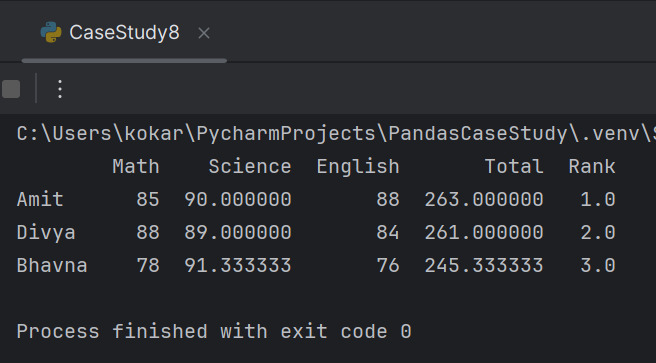
Divya 88 89.0 84 261.0 1.0

Bhavna 78 91.3 76 245.3 3.0

**Program:**

****

**Output:**

****

**Case Study 9: Adding a New Column – Attendance %**

**Scenario:** Add attendance percentage for each student.

**Input:**

df['Attendance'] = [95, 88, 92]

print(df[['Attendance']])

**Expected Output:**

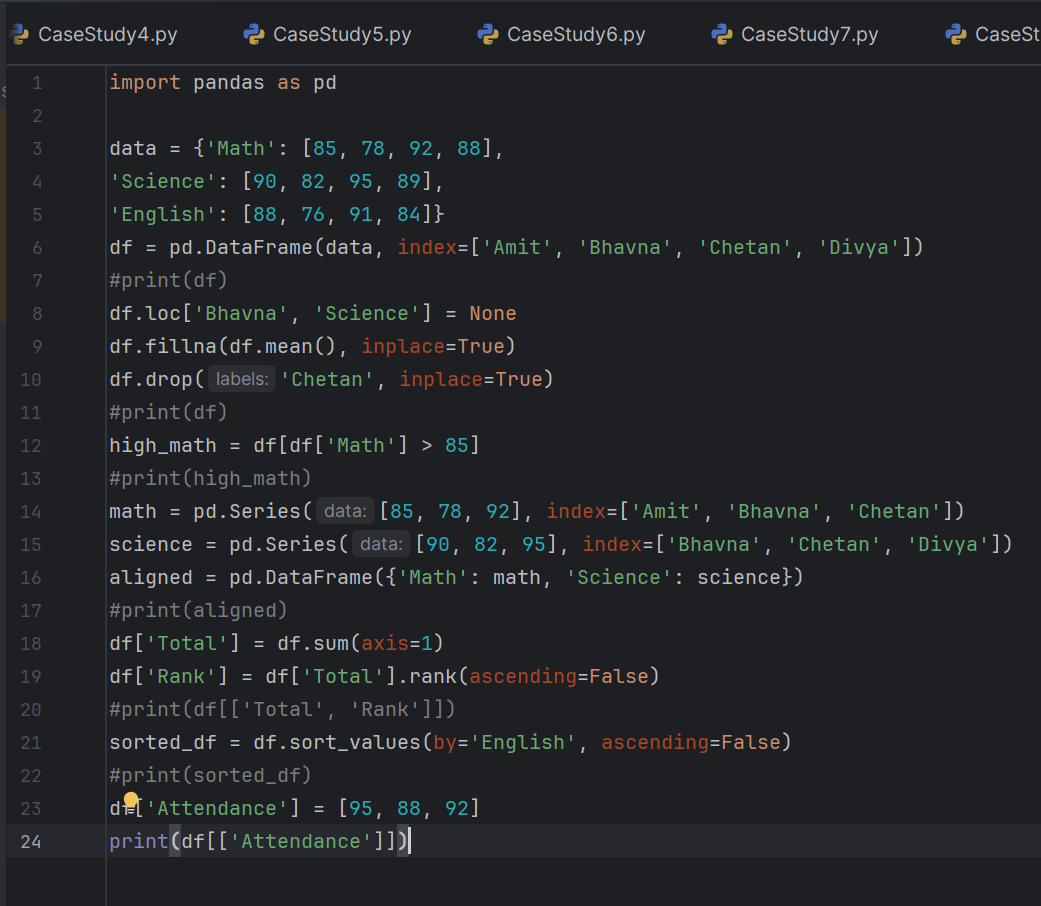
Attendance

Amit 95

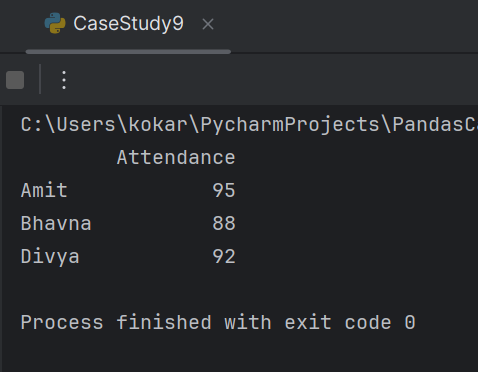
Bhavna 88

Divya 92

**Program:**

****

**Output:**

****

**Case Study 10: Selecting Specific Columns and Rows**

**Scenario:** Display only Math and English scores for Amit and Divya.

**Input:**

selected = df.loc[['Amit', 'Divya'], ['Math', 'English']]

print(selected)

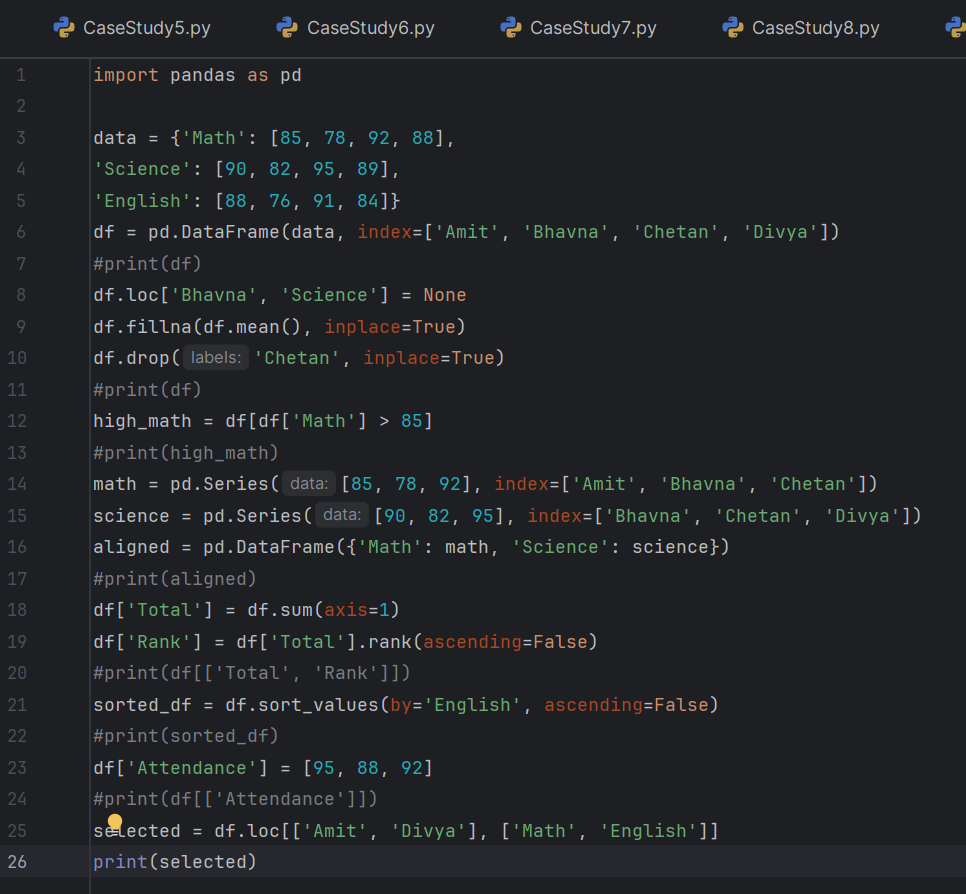
**Expected Output:**

Math English

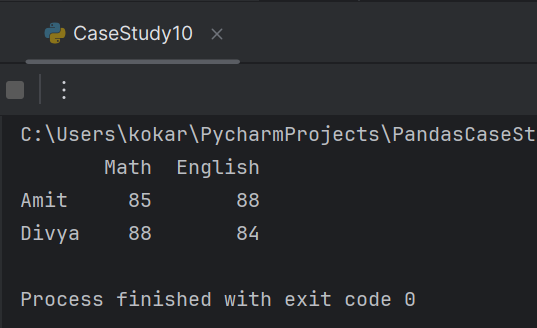
Amit 85 88

Divya 88 84

**Program:**

****

**Output:**

****

**================================================================**

**Car Crash Practice Assignment**

**Case Study Set (Car Crash Dataset)**

**Case Study 1: NumPy Arrays & Universal Functions**

**Problem: Using the car\_crashes dataset, extract the speeding column as a NumPy**

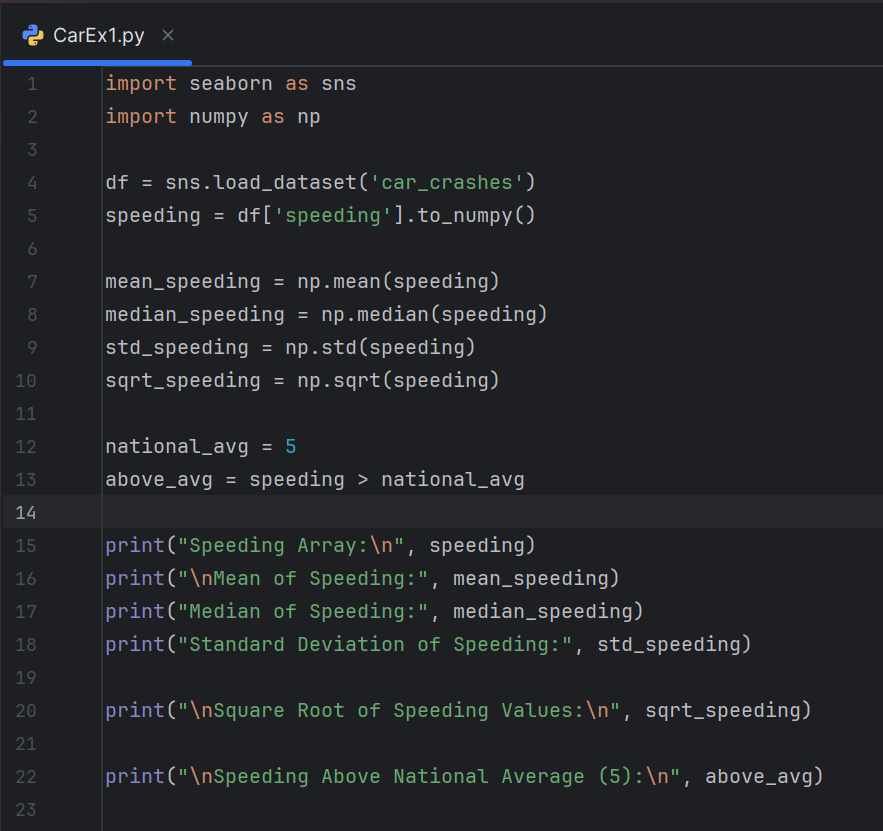
**array.**

**• Compute the mean, median, and standard deviation.**

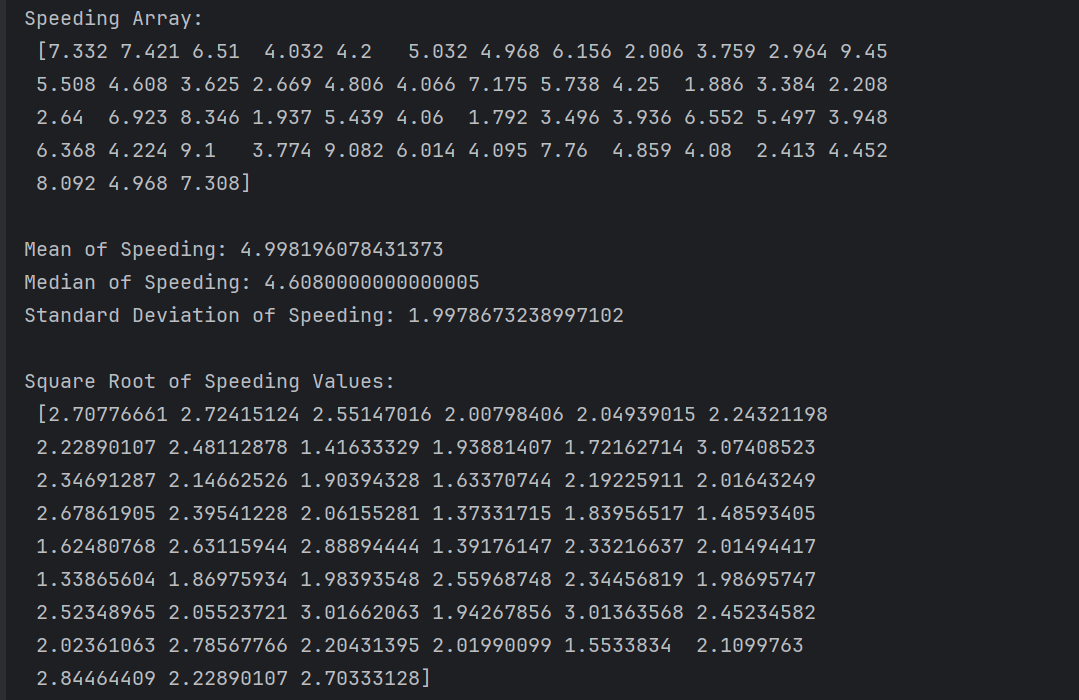
**• Apply a universal function (e.g., np.sqrt) to transform values.**

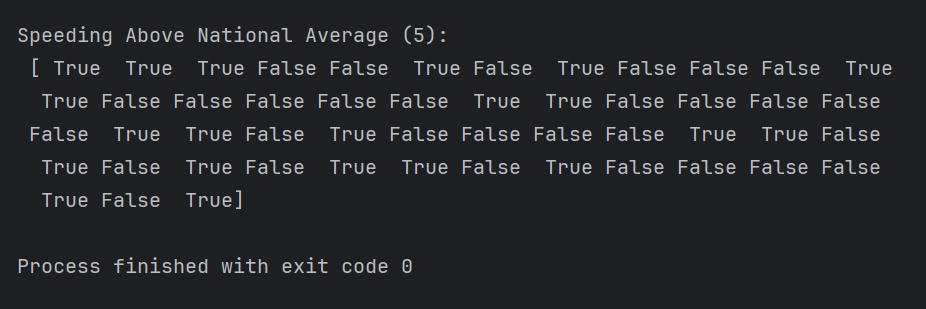
**• Compare speeding values against the national average (scalar).**

**Program:**



**Output:**





**Case Study 2: Pandas Series & DataFrames**

**Problem: Analyze the relationship between alcohol and total crashes.**

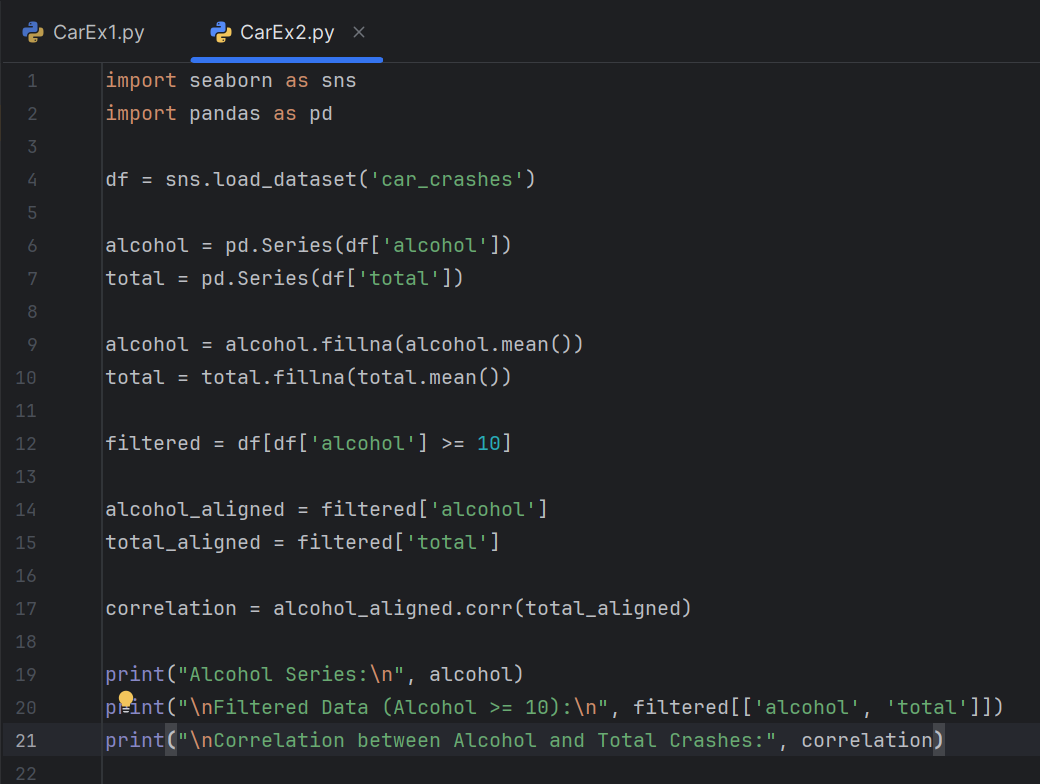
**• Create a Pandas Series for alcohol.**

**• Handle missing data (if any).**

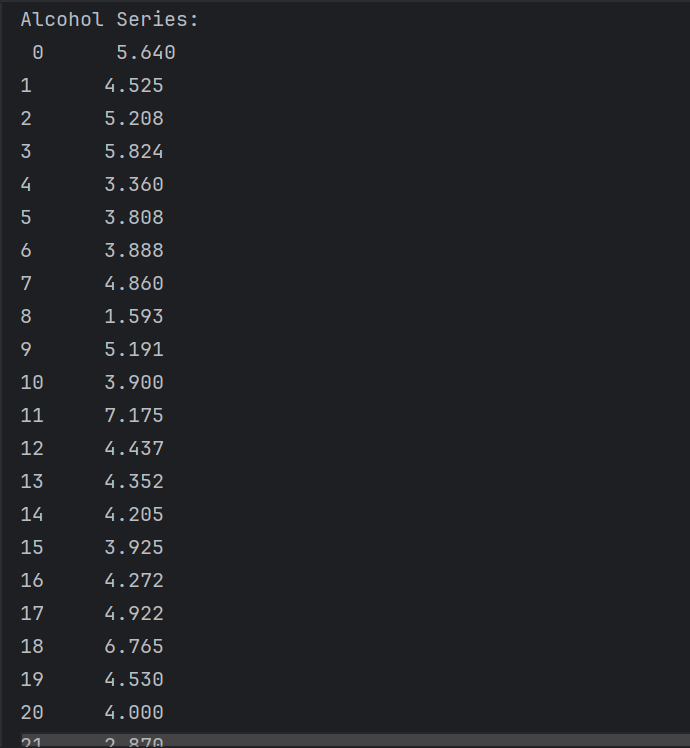
**• Drop entries where alcohol < 10.**

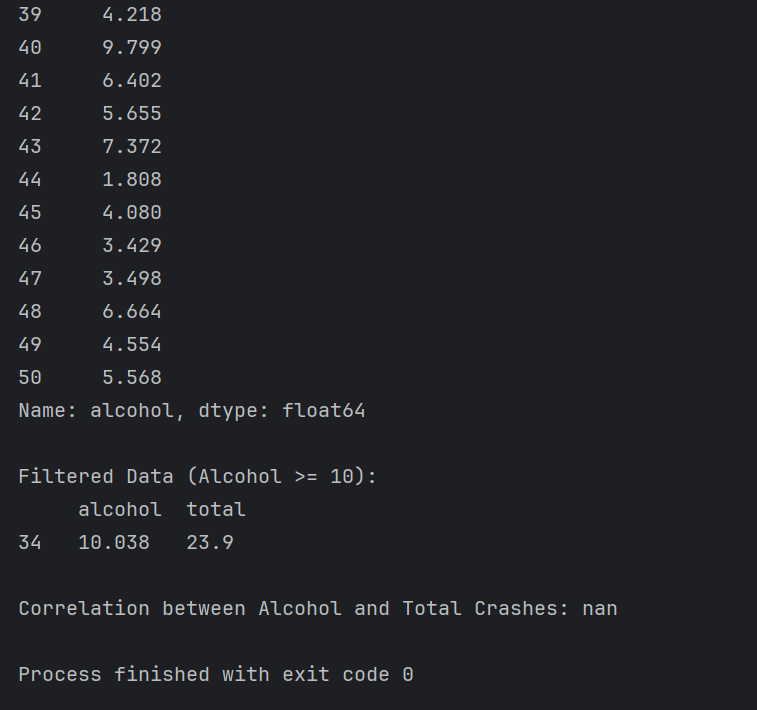
**• Align data with total crashes and compute correlation.**

**Program:**



**Output:**





**Case Study 3: Ranking & Sorting with Pandas**

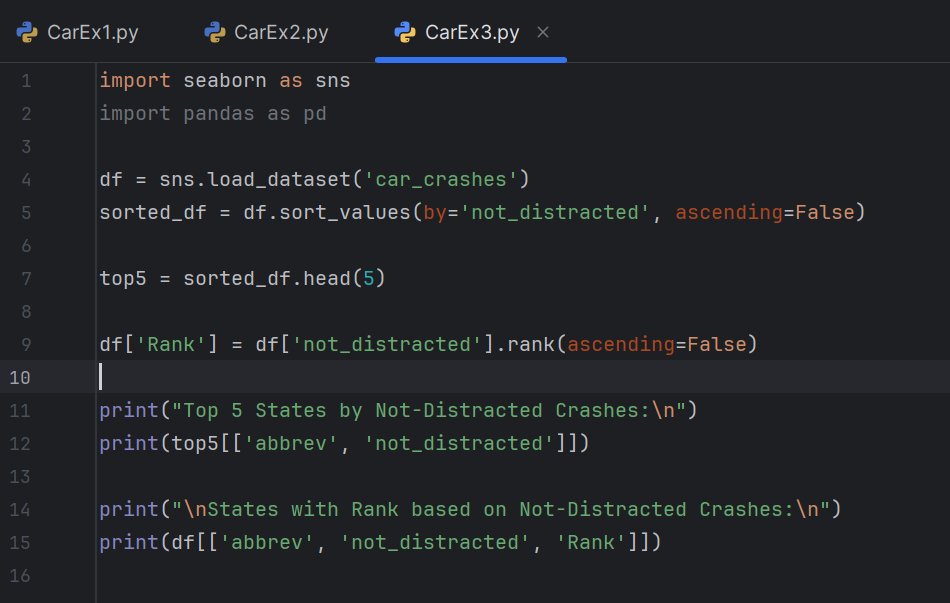
**Problem: Rank states by not\_distracted crashes.**

**• Sort values in descending order.**

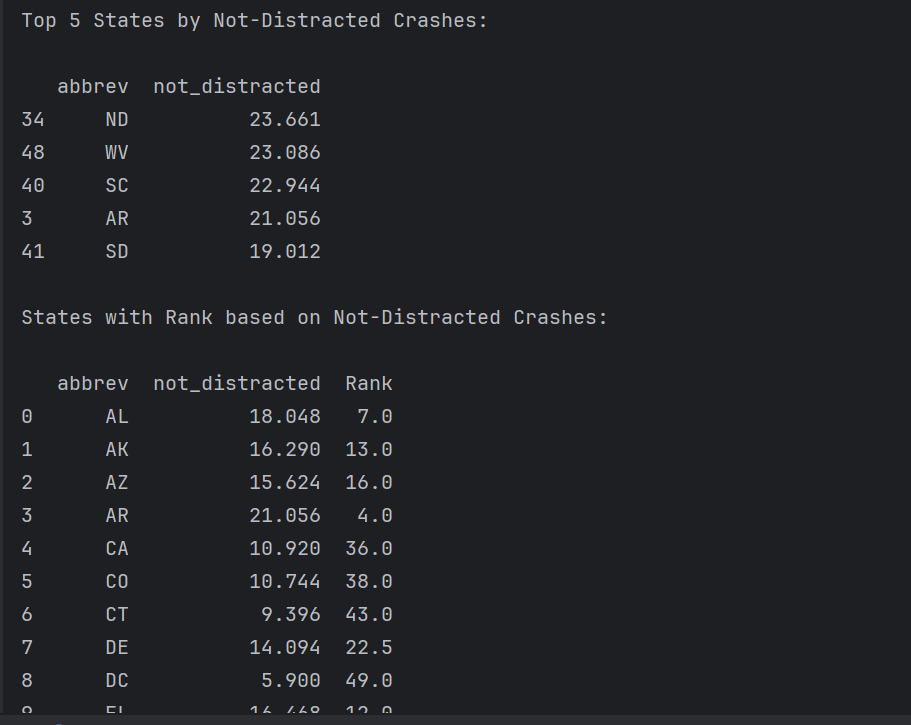
**• Display top 5 states.**

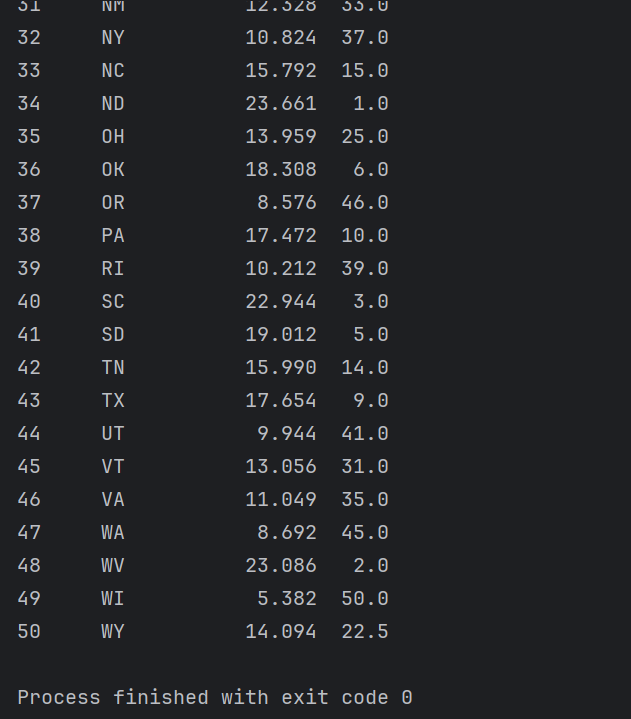
**• Assign ranks to each state.**

**Program:**



**Output:**





**Case Study 4: Matplotlib Visualization – Comparative Analysis**

**Problem: Visualize the distribution of speeding vs alcohol.**

**• Create a scatter plot with labels.**

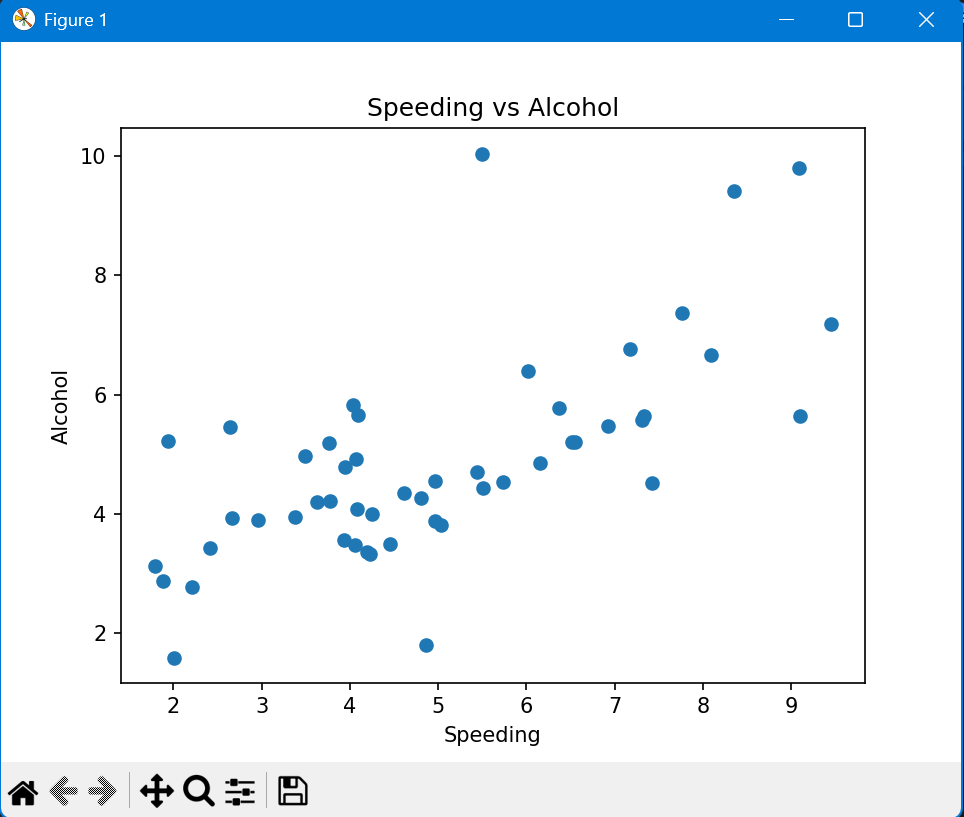
**• Add a histogram of total crashes.**

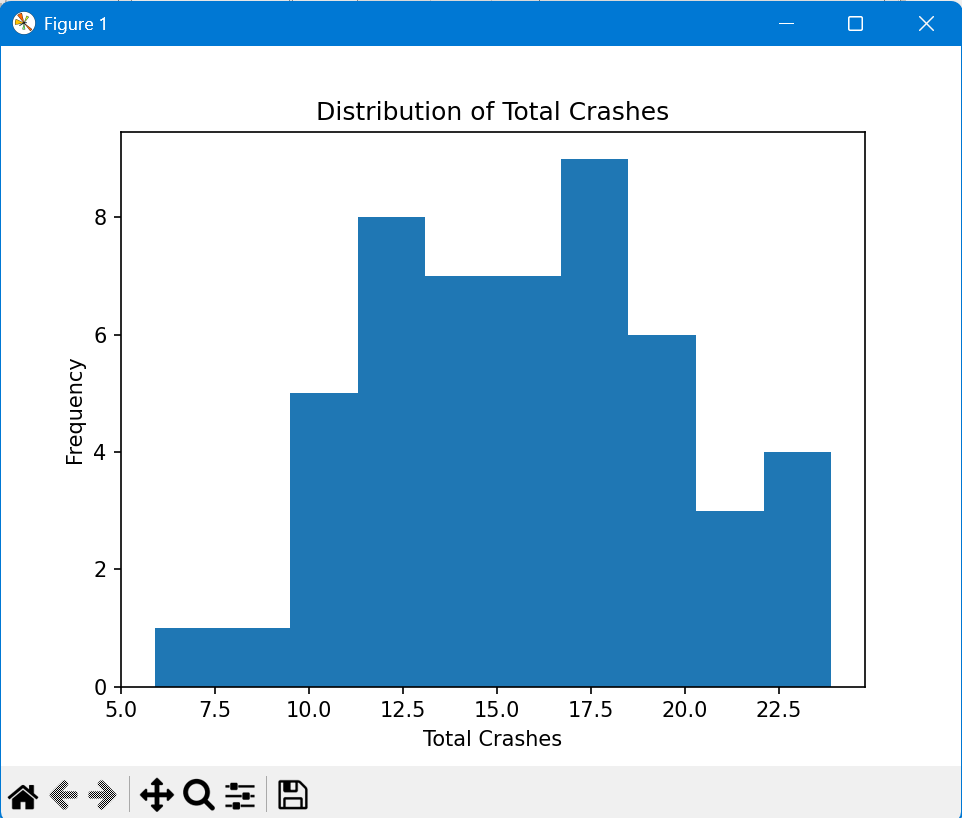
**• Compare trends visually.**

**Program:**



**Output:**





**================================================================**

**Pandas Case Study Assignment 2**

**Case Study 1: Data Alignment Between Two DataFrames**

**Scenario: Align marks from two subjects stored in separate DataFrames.**

**Input:**

python

math = pd.Series([85, 78, 92], index=['Amit', 'Bhavna', 'Chetan'])

science = pd.Series([90, 82, 95], index=['Bhavna', 'Chetan', 'Divya'])

aligned = pd.DataFrame({'Math': math, 'Science': science})

print(aligned)

**Expected Output:**

Code

Math Science

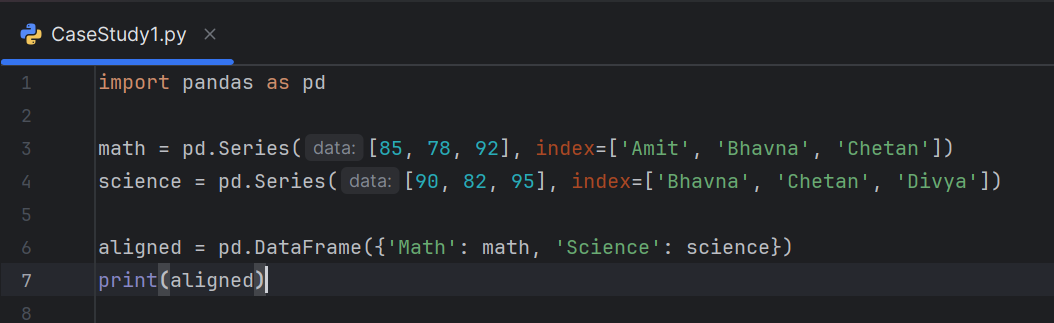
Amit 85.0 NaN

Bhavna 78.0 90.0

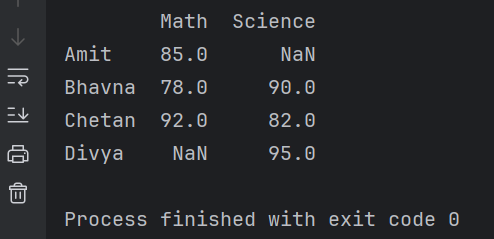
Chetan 92.0 82.0

Divya NaN 95.0

**Program:**



**Output:**



**Case Study 2: Ranking Students by Total Marks**

**Scenario: Rank students based on total marks.**

**Input:**

python

df['Total'] = df.sum(axis=1)

df['Rank'] = df['Total'].rank(ascending=False)

print(df[['Total', 'Rank']])

**Expected Output:**

Code

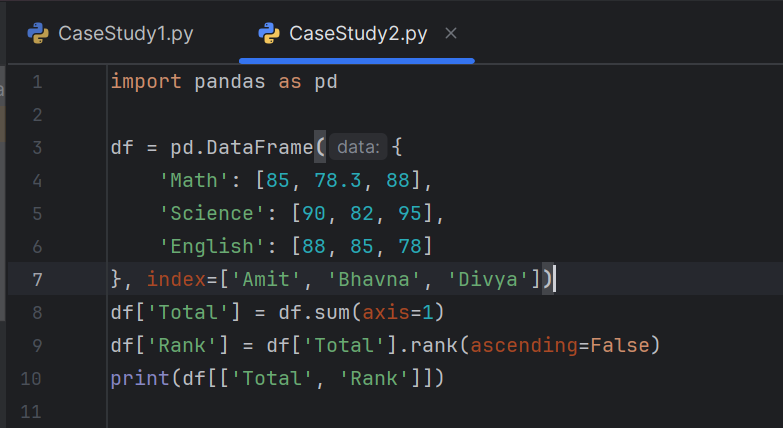
Total Rank

Amit 263.0 2.0

Bhavna 245.3 3.0

Divya 261.0 1.0

**Program:**



**Output:**



**Case Study 3: Sorting by English Marks**

**Scenario: Sort students by their English scores.**

**Input:**

python

sorted\_df = df.sort\_values(by='English', ascending=False)

print(sorted\_df)

**Expected Output:**

Code

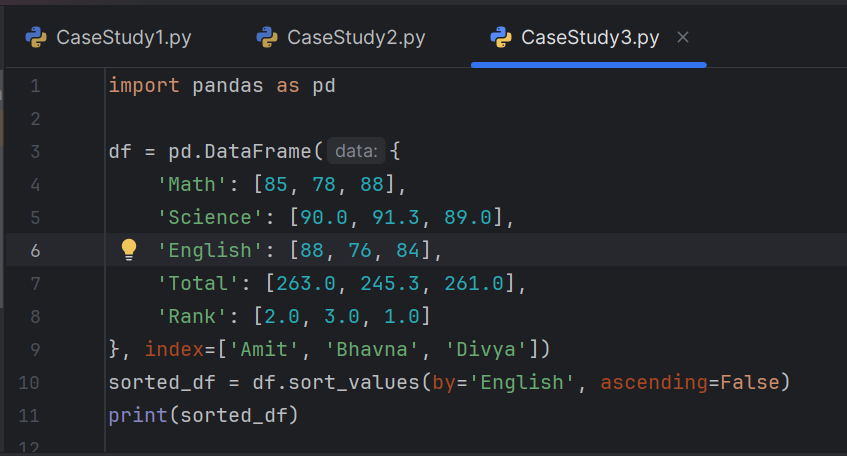
Math Science English Total Rank

Amit 85 90.0 88 263.0 2.0

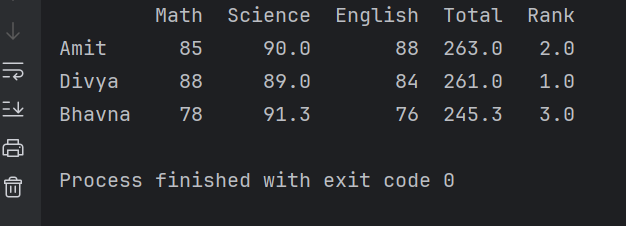
Divya 88 89.0 84 261.0 1.0

Bhavna 78 91.3 76 245.3 3.0

**Program:**



**Output:**



**Case Study 4: Adding a New Column – Attendance %**

**Scenario: Add attendance percentage for each student.**

**Input:**

python

df['Attendance'] = [95, 88, 92]

print(df[['Attendance']])

**Expected Output:**

Code

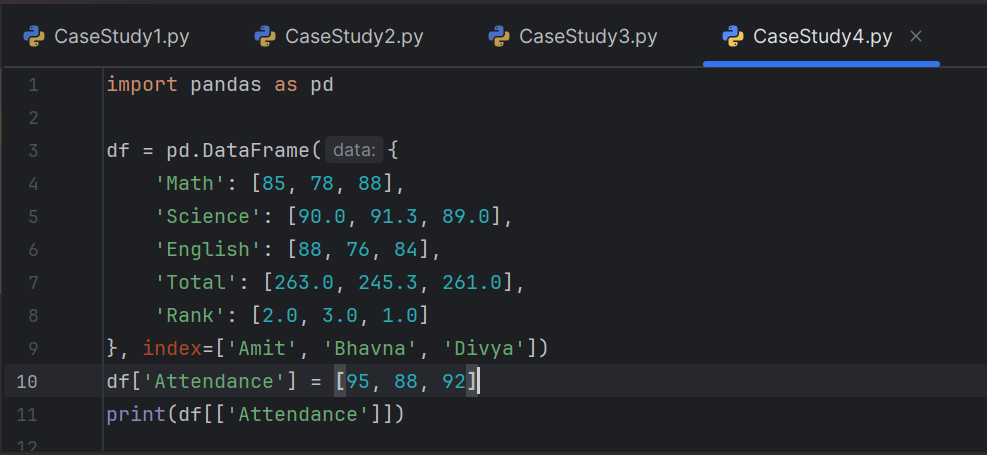
Attendance

Amit 95

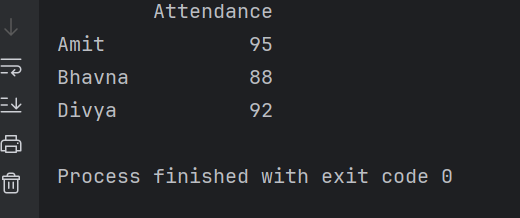
Bhavna 88

Divya 92

**Program:**



**Code:**



**Case Study 5: Selecting Specific Columns and Rows**

**Scenario: Display only Math and English scores for Amit and Divya.**

**Input:**

python

selected = df.loc[['Amit', 'Divya'], ['Math', 'English']]

print(selected)

**Expected Output:**

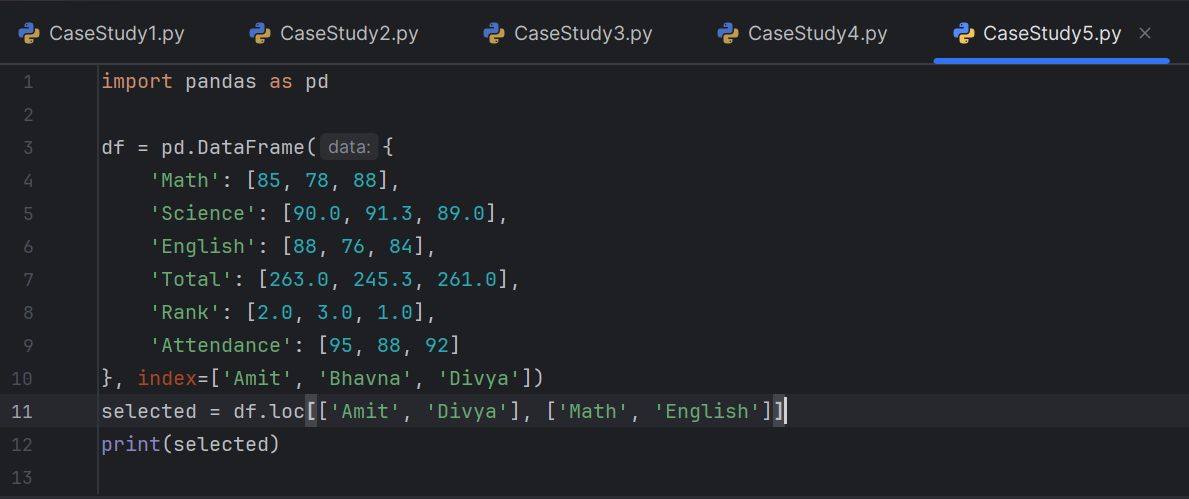
Code

Math English

Amit 85 88

Divya 88 84

**Program:**



**Output:**



**================================================================**

**Regular Expressions Assignments**

**Case Study 1: Regex in Hackathon Registration System**

**Background**

IMCC is organizing a state-level hackathon. As part of the registration process,

participants must submit details like:

• Full Name

• Email Address

• Password (for login)

• Project URL (GitHub or hosted demo link)

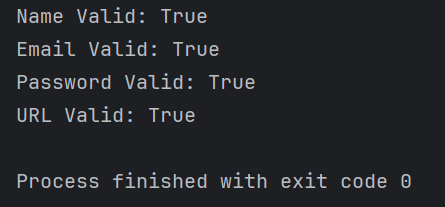
To ensure data integrity and security, the system uses Python Regular Expressions

(Regex) for validation.

**Program:**



**Output:**



**Case Study 2: Library Management System – Member Registration**

**Background**

A university library is digitizing its membership system. Students and faculty must

register online, and the system validates their details using regex.

**Problem Statement**

Design regex validations for:

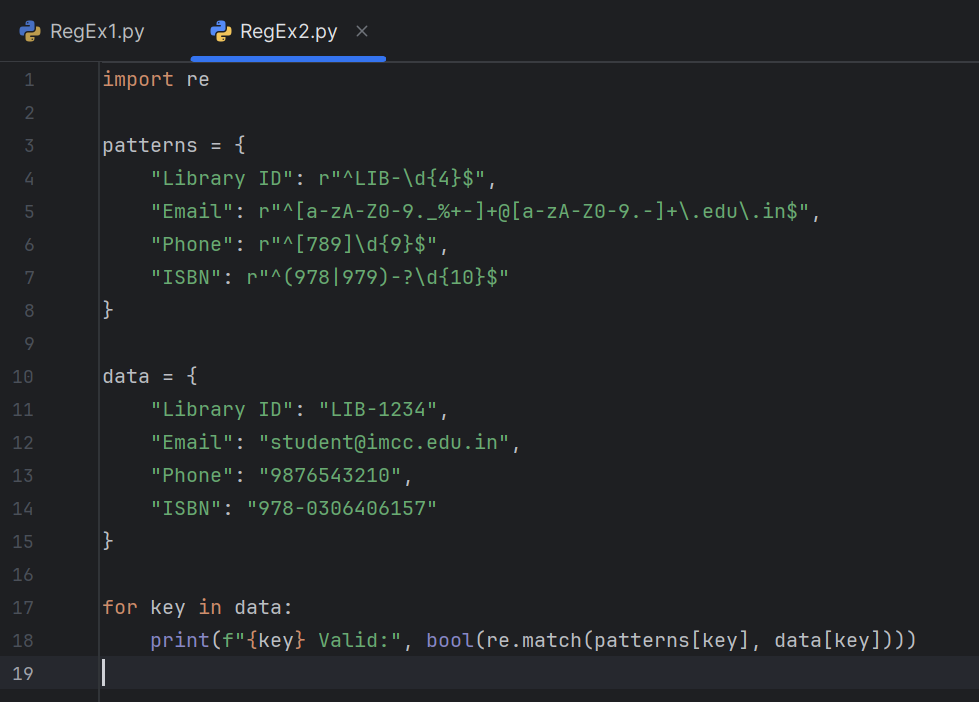
1. Library ID → Format: LIB-XXXX (4 digits).

2. Email → Must end with .edu.in.

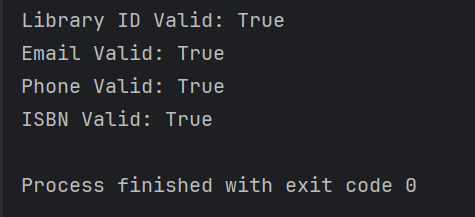
3. Phone Number → Exactly 10 digits, starting with 7, 8, or 9.

4. Book ISBN → Must match ISBN-13 format (978-... or 979-...).

**Program:**



**Output:**



**Case Study 3: E-Commerce Checkout Form**

**Background**

An e-commerce platform wants to ensure clean data entry during checkout.

**Problem Statement**

Validate the following using regex:

1. Customer Name → Alphabets only, min 2 characters.

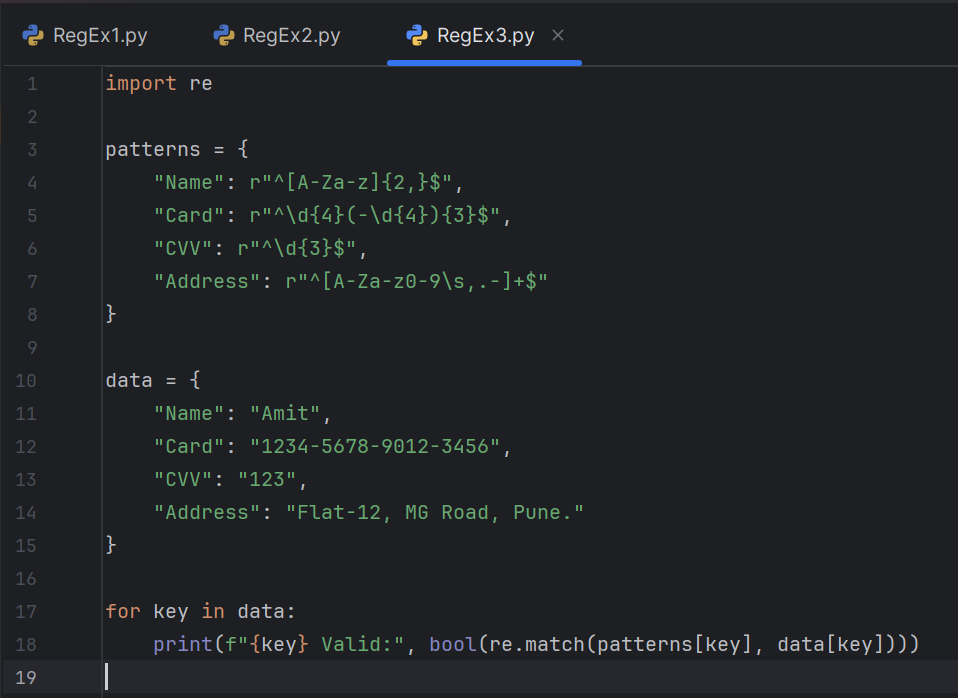
2. Credit Card Number → 16 digits, grouped as XXXX-XXXX-XXXX-XXXX.

3. CVV → Exactly 3 digits.

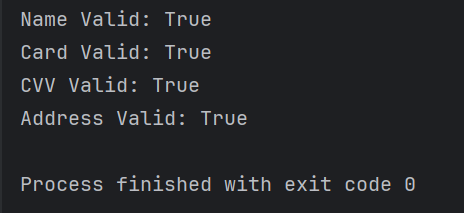
4. Shipping Address → Must contain alphanumeric characters and allow commas,

periods, and hyphens.

**Program:**



**Output:**



**Case Study 4: Healthcare Appointment System**

**Background**

A hospital introduces an online appointment booking system. Regex ensures patients

enter valid details.

**Problem Statement**

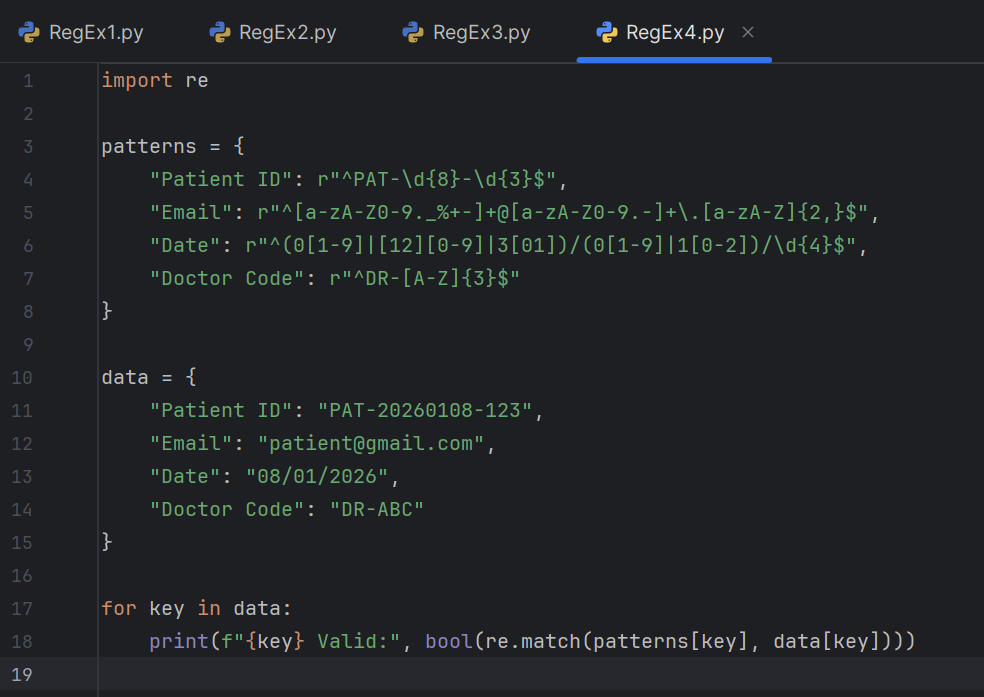
1. Patient ID → Format: PAT-YYYYMMDD-XXX (date + 3 digits).

2. Email → Standard email format.

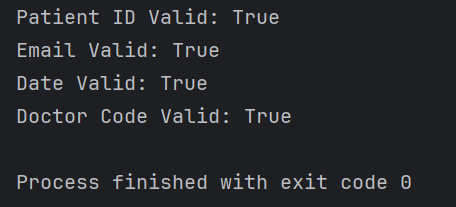
3. Appointment Date → Must follow DD/MM/YYYY.

4. Doctor Code → Format: DR-XXX (3 uppercase letters).

**Program:**



**Output:**



**Case Study 5: Online Examination Portal**

**Background**

Students log in to an exam portal where regex validates credentials and submissions.

**Problem Statement**

1. Username → Alphanumeric, 5–15 characters.

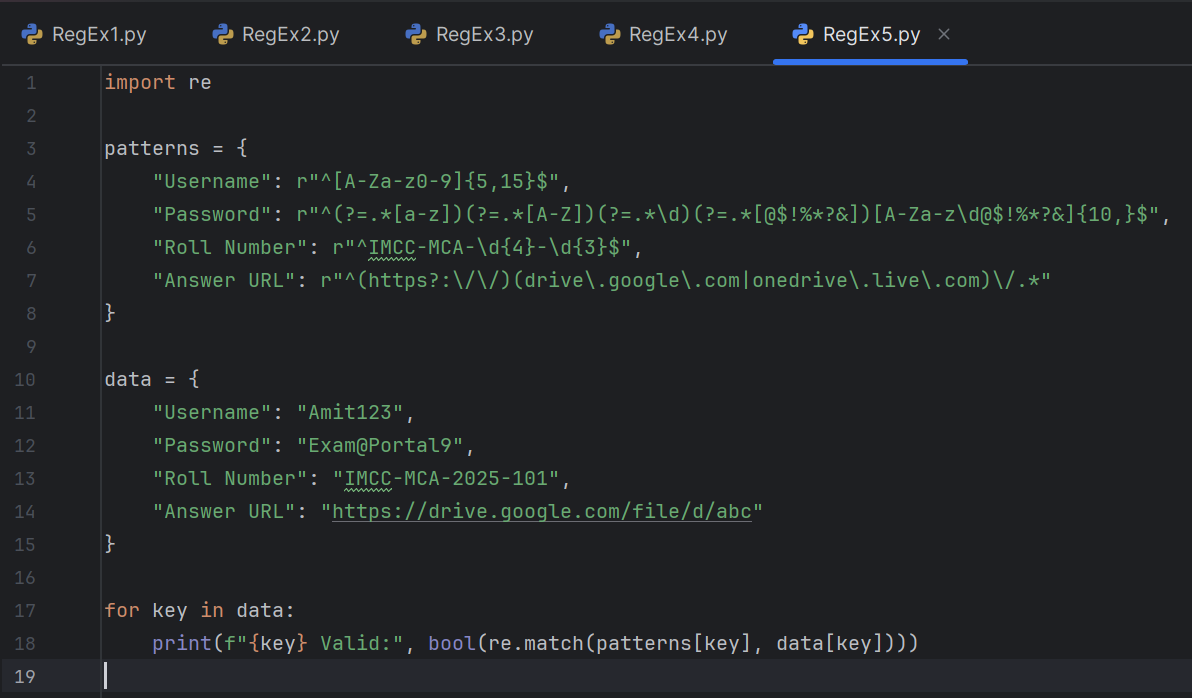
2. Password → At least 10 characters, must include uppercase, lowercase, digit,

and special character.

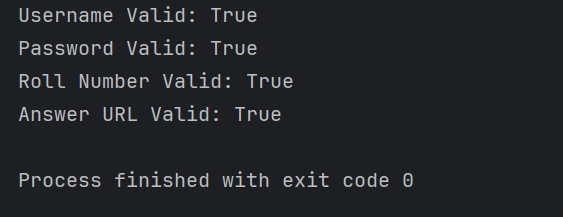
3. Roll Number → Format: IMCC-MCA-YYYY-XXX.

4. Answer File URL → Must be a valid Google Drive or OneDrive link.

**Program:**



**Output:**



**================================================================**