



Alliance School of Advanced Computing

Department of Computer Science and Engineering

Class Assignment-1

Course Code: 5CS1025

Course Title: Artificial Intelligence

Semester: 04

Class : AIML

Name :- Vidya Khairnar

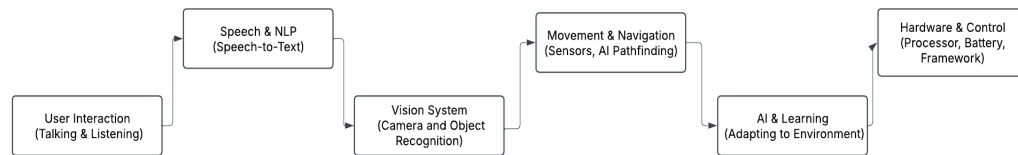
Class :- AIML-E

Reg.no :- 2023BCSE07AED421

https://github.com/GITVIDHUB-1010/AI_codes

2024-25

1. Imagine you are tasked with designing a humanoid robot to assist in a home or office environment. The robot must be capable of interacting with people by **talking** and **listening**, **walking** to different locations, **seeing** and recognizing objects, and **learning** from its surroundings to adapt its behavior. **What technologies, tools, and frameworks would you need to build such a robot?** Give as flow chart



2. Calculate and interpret mean, median, mode, variance and standard deviation for a given dataset. Data = [15,21,29,21,15,24,32,21,15,30]

```
import numpy as np
import pandas as pd

data = [15, 21, 29, 21, 15, 24, 32, 21, 15, 30]
data_series = pd.Series(data)

mean = np.mean(data)
median = np.median(data)
mode = data_series.mode().values[0]
variance = np.var(data)
std_dev = np.std(data)

print(f"Mean {mean}")
print(f"Median {median}")
print(f"Mode {mode}")
print(f"Variance {variance}")
print(f"Standard Deviation {std_dev}")

Mean 22.3
Median 21.0
Mode 15
Variance 36.61
Standard Deviation 6.050619802962338
```

3. You are analyzing a dataset that captures the daily performance and activity of a humanoid robot in a simulated environment. The dataset link [robot_dataset\(robot_dataset\)_1.csv](#) includes the following attributes

Interaction_Count: Number of conversations the robot had daily.
Steps_Walked: Total steps taken each day.
Objects_Recognized: Number of objects successfully identified by the robot.
Learning_Sessions: Number of learning tasks completed.
Energy_Consumption (kWh): Daily energy usage of robots.

Perform Basic Statistical Operations:

- 1) What is the **average (mean)** number of conversations the robot has daily?
- 2) Find the **total steps walked** by the robot over a given period.
- 3) Determine the **maximum and minimum energy consumption** in the dataset.
- 4) Calculate the **correlation** between the number of steps walked and energy consumption.

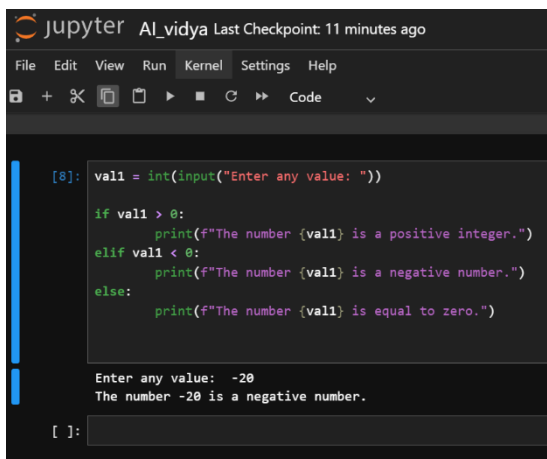
- 5) Analyze the **distribution** of objects recognized daily (e.g., histogram or box plot).
 - 6) What is the **variance** in the number of learning sessions completed?
-
4. Write a Python program that declares variables of different data types (e.g., string, integer, float, and boolean). Output the variables in a sentence format using print() and f-strings.

```
name = "Vidya"
ID = 456
marks = 85.7
is_student = True

print(f"My name is {name}. My ID number is {ID} and I scored {marks} in 12th, and it is {is_student}, that I am a student.")

My name is Vidya. My ID number is 456 and I scored 85.7 in 12th, and it is True, that I am a student.
```

5. Write a Python program that takes an integer input and checks whether the number is positive, negative, or zero using conditional statements (if-else)



A screenshot of a Jupyter Notebook interface. The title bar says "Jupyter AI_vidya Last Checkpoint: 11 minutes ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. Below the menu is a toolbar with icons for file operations and execution. The code cell [8] contains the following Python code:

```
val1 = int(input("Enter any value: "))

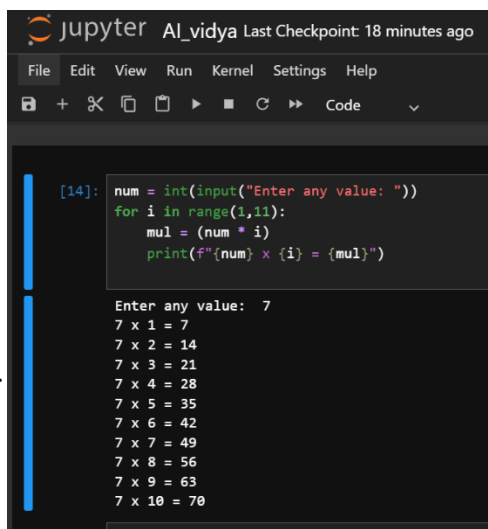
if val1 > 0:
    print(f"The number {val1} is a positive integer.")
elif val1 < 0:
    print(f"The number {val1} is a negative number.")
else:
    print(f"The number {val1} is equal to zero.")
```

The output of the code is shown below the code cell:

```
Enter any value: -20
The number -20 is a negative number.
```

The prompt []: is visible at the bottom of the code cell.

6. Write a Python program that takes a number as input and prints the multiplication table for that number (from 1 to 10)



A screenshot of a Jupyter Notebook interface. The title bar says "Jupyter AI_vidya Last Checkpoint: 18 minutes ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. Below the menu is a toolbar with icons for file operations and execution. The code cell [14] contains the following Python code:

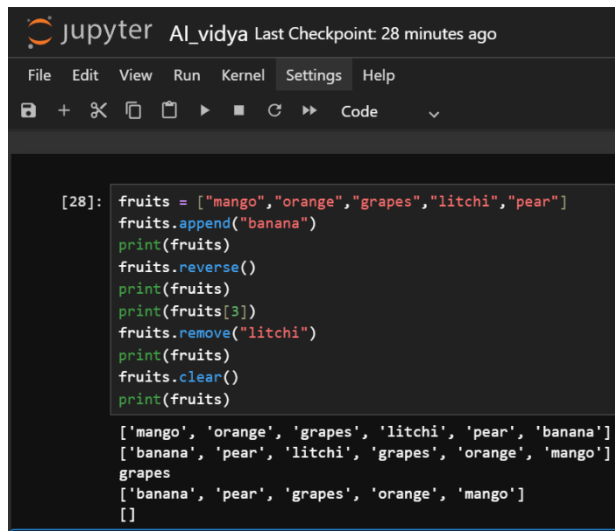
```
num = int(input("Enter any value: "))
for i in range(1,11):
    mul = (num * i)
    print(f"{num} x {i} = {mul}")
```

The output of the code is shown below the code cell:

```
Enter any value: 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

The prompt []: is visible at the bottom of the code cell.

7. Create a Python list that contains the names of 5 different fruits. Perform the given operations on the list.



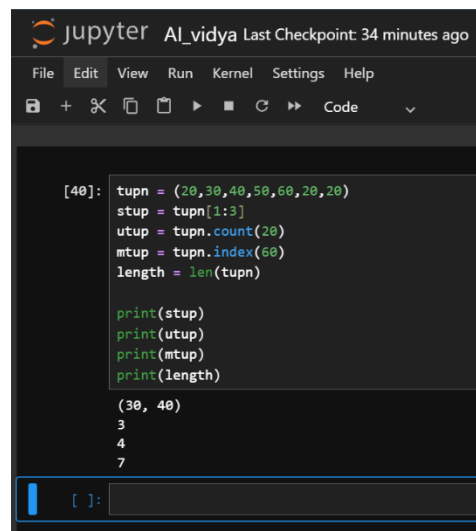
A Jupyter Notebook interface with a dark theme. The title bar says "jupyter AI_vidya Last Checkpoint: 28 minutes ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. The toolbar has icons for saving, opening, running, and other actions. The code cell [28]: contains the following Python code:

```
fruits = ["mango", "orange", "grapes", "litchi", "pear"]
fruits.append("banana")
print(fruits)
fruits.reverse()
print(fruits)
print(fruits[3])
fruits.remove("litchi")
print(fruits)
fruits.clear()
print(fruits)
```

The output of the code is displayed below the code cell:

```
['mango', 'orange', 'grapes', 'litchi', 'pear', 'banana']
['banana', 'pear', 'litchi', 'grapes', 'orange', 'mango']
grapes
['banana', 'pear', 'grapes', 'orange', 'mango']
[]
```

8. Write a Python program that creates a tuple containing 5 numbers. Perform the given operations on the tuple.



A Jupyter Notebook interface with a dark theme. The title bar says "jupyter AI_vidya Last Checkpoint: 34 minutes ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. The toolbar has icons for saving, opening, running, and other actions. The code cell [40]: contains the following Python code:

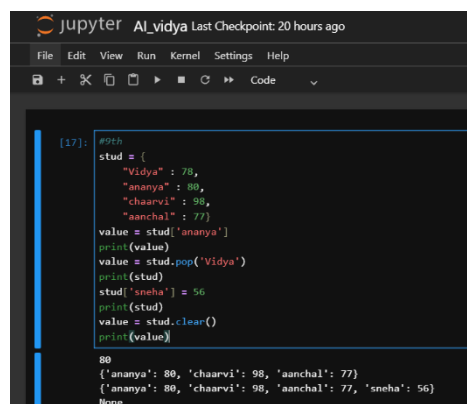
```
tupn = (20, 30, 40, 50, 60, 20, 20)
stup = tupn[1:3]
utup = tupn.count(20)
mtup = tupn.index(60)
length = len(tupn)

print(stup)
print(utup)
print(mtup)
print(length)
```

The output of the code is displayed below the code cell:

```
(30, 40)
3
4
7
```

9. Create a dictionary that stores the names of 3 students as keys and their marks in mathematics as values. Perform the given operations.



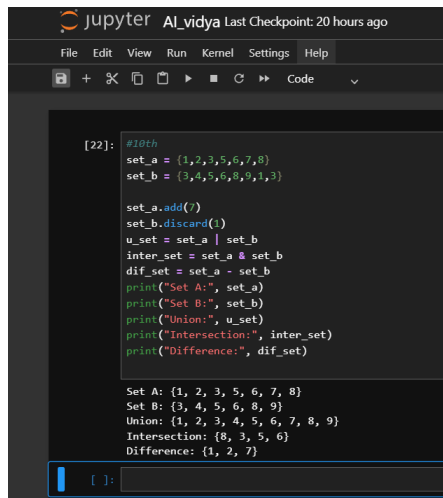
A Jupyter Notebook interface with a dark theme. The title bar says "jupyter AI_vidya Last Checkpoint: 20 hours ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. The toolbar has icons for saving, opening, running, and other actions. The code cell [17]: contains the following Python code:

```
#9th
stud = {
    "Vidya": 78,
    "ananya": 88,
    "chaarvi": 98,
    "aanchal": 77
}
value = stud["ananya"]
print(value)
value = stud.pop("Vidya")
print(stud)
stud["sneha"] = 56
print(stud)
value = stud.clear()
print(value)
```

The output of the code is displayed below the code cell:

```
88
{'ananya': 88, 'chaarvi': 98, 'aanchal': 77}
{'ananya': 88, 'chaarvi': 98, 'aanchal': 77, 'sneha': 56}
None
```

10. Create two sets of integers. Perform the given set operations

A screenshot of a Jupyter Notebook interface. The top bar shows 'Jupyter AI_vidya Last Checkpoint: 20 hours ago'. The menu bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, running, and other actions. The main area shows a code cell with the following Python code:

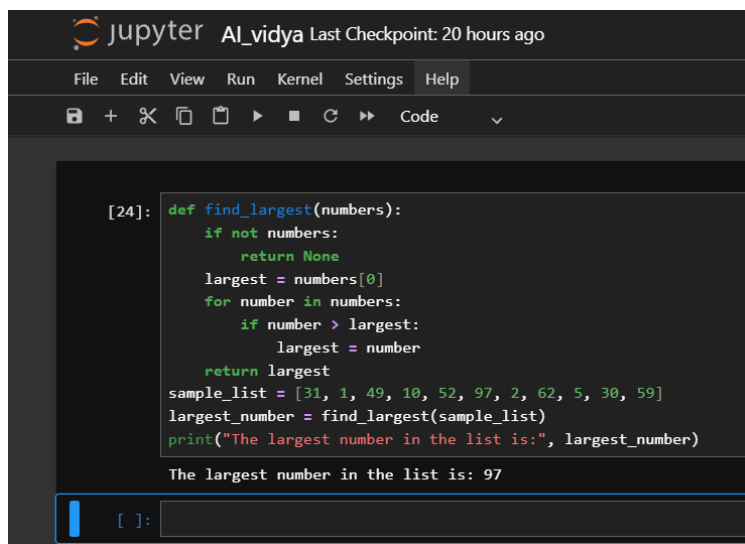
```
[22]: #10th
set_a = {1,2,3,5,6,7,8}
set_b = {3,4,5,6,8,9,1,3}

set_a.add(7)
set_b.discard(1)
u_set = set_a | set_b
inter_set = set_a & set_b
dif_set = set_a - set_b
print("Set A:", set_a)
print("Set B:", set_b)
print("Union:", u_set)
print("Intersection:", inter_set)
print("Difference:", dif_set)
```

The output of the code is displayed below the code cell:

```
Set A: {1, 2, 3, 5, 6, 7, 8}
Set B: {3, 4, 5, 6, 8, 9}
Union: {1, 2, 3, 4, 5, 6, 7, 8, 9}
Intersection: {3, 4, 5, 6}
Difference: {1, 2, 7}
```

11. Write a Python function called `find_largest()` that takes a list of numbers as input and returns the largest number from the list. Test the function with a sample list.

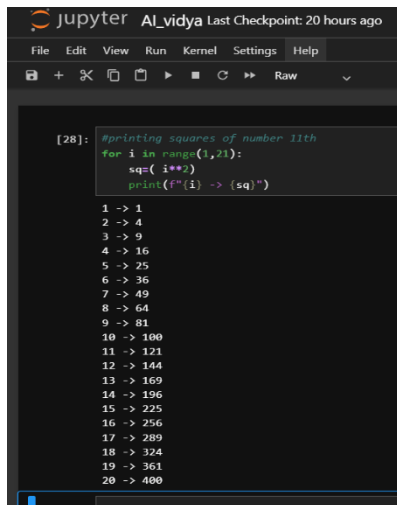
A screenshot of a Jupyter Notebook interface. The top bar shows 'Jupyter AI_vidya Last Checkpoint: 20 hours ago'. The menu bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, running, and other actions. The main area shows a code cell with the following Python code:

```
[24]: def find_largest(numbers):
    if not numbers:
        return None
    largest = numbers[0]
    for number in numbers:
        if number > largest:
            largest = number
    return largest
sample_list = [31, 1, 49, 10, 52, 97, 2, 62, 5, 30, 59]
largest_number = find_largest(sample_list)
print("The largest number in the list is:", largest_number)
```

The output of the code is displayed below the code cell:

```
The largest number in the list is: 97
```

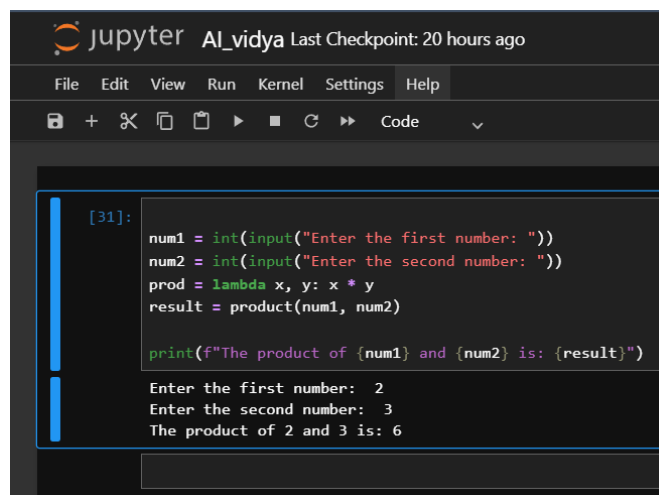
12. Use list comprehension to create a list of squares of all even numbers between 1 and 20.



```
[28]: #printing squares of number 11th
for i in range(1,21):
    sq=( i**2)
    print(f"{i} -> {sq}")

1 -> 1
2 -> 4
3 -> 9
4 -> 16
5 -> 25
6 -> 36
7 -> 49
8 -> 64
9 -> 81
10 -> 100
11 -> 121
12 -> 144
13 -> 169
14 -> 196
15 -> 225
16 -> 256
17 -> 289
18 -> 324
19 -> 361
20 -> 400
```

13. Write a Python script that uses a lambda function to calculate the product of two numbers provided by the user.

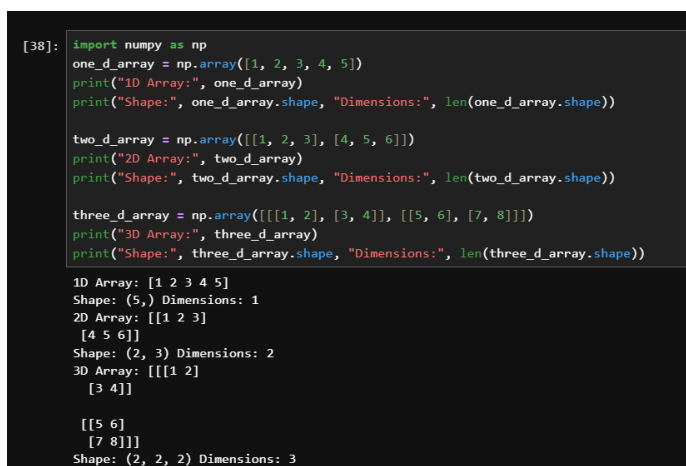


```
[31]: num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
prod = lambda x, y: x * y
result = product(num1, num2)

print(f"The product of {num1} and {num2} is: {result}")

Enter the first number: 2
Enter the second number: 3
The product of 2 and 3 is: 6
```

14. Write a Python program to create a one-dimensional, two-dimensional, and three-dimensional NumPy array. Print the shape and dimensions of each array.



```
[38]: import numpy as np
one_d_array = np.array([1, 2, 3, 4, 5])
print("1D Array:", one_d_array)
print("Shape:", one_d_array.shape, "Dimensions:", len(one_d_array.shape))

two_d_array = np.array([[1, 2, 3], [4, 5, 6]])
print("2D Array:", two_d_array)
print("Shape:", two_d_array.shape, "Dimensions:", len(two_d_array.shape))

three_d_array = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("3D Array:", three_d_array)
print("Shape:", three_d_array.shape, "Dimensions:", len(three_d_array.shape))

1D Array: [1 2 3 4 5]
Shape: (5,) Dimensions: 1
2D Array: [[1 2 3]
 [4 5 6]]
Shape: (2, 3) Dimensions: 2
3D Array: [[[1 2]
 [3 4]]
 [[5 6]
 [7 8]]]
Shape: (2, 2, 2) Dimensions: 3
```

15. Write a Python program to create a 5x5 NumPy array of random integers and Perform array indexing as given

```
[40]: import numpy as np
dom_array = np.random.randint(0, 100, size=(5, 5))

print("5x5 Random Integer Array:\n", dom_array)
print("\nElement at (0, 0):", dom_array[0, 0])
print("First row:", dom_array[0])
print("Last column:", dom_array[:, -1])

5x5 Random Integer Array:
[[55 67 26 76 73]
 [70 98 93 24 74]
 [ 4 76 70 55 31]
 [17  0 34 19 74]
 [73 33 37 20 33]]

Element at (0, 0): 55
First row: [55 67 26 76 73]
Last column: [73 74 31 74 33]
```

16. create a NumPy array of shape (4, 4) containing numbers from 1 to 16. Use slicing to extract for the given conditions

```
[41]: import numpy as np
array_4x4 = np.arange(1, 17).reshape(4, 4)
print("4x4 Array:\n", array_4x4)

sliced_array = array_4x4[:2, -2:]
print("\nSliced Array (first 2 rows, last 2 columns):\n", sliced_array)

4x4 Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Sliced Array (first 2 rows, last 2 columns):
[[3 4]
 [7 8]]
```

17. Write a Python program that creates a 2D array of shape (6, 2) using np.arange() and then reshapes it into a 3D array of shape (2, 3, 2). Flatten the reshaped array and print the result.

```
[ ]: import numpy as np
      array_2d = np.arange(12).reshape(6, 2)
      print("2D Array (6x2):\n", array_2d)

      array_3d = array_2d.reshape(2, 3, 2)
      print("\n3D Array (2x3x2):\n", array_3d)

      flattened_array = array_3d.flatten()
      print("\nFlattened Array:\n", flattened_array)

      2D Array (6x2):
      [[ 0  1]
       [ 2  3]
       [ 4  5]
       [ 6  7]
       [ 8  9]
       [10 11]]

      3D Array (2x3x2):
      [[[ 0  1]
        [ 2  3]
        [ 4  5]]

       [[ 6  7]
        [ 8  9]
        [10 11]]]

      Flattened Array:
      [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

18. Write a Python program to demonstrate broadcasting. Create an array of shape (3, 3) and add a one-dimensional array of shape (1, 3) to it using broadcasting.

```
[49]: import numpy as np
      array_3x3 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
      print("3x3 Array:\n", array_3x3)

      one_d_array = np.array([10, 20, 30])
      print("one Dimensional Array:\n", one_d_array)

      broadcasted_sum = array_3x3 + one_d_array
      print("\nResult aftr Broadcasting:\n", broadcasted_sum)

      3x3 Array:
      [[1 2 3]
       [4 5 6]
       [7 8 9]]

      one Dimensional Array:
      [10 20 30]

      Result aftr Broadcasting:
      [[11 22 33]
       [14 25 36]
       [17 28 39]]
```


19. Create two NumPy arrays of the same shape, A and B. Perform the following arithmetic operations:

Element-wise addition.

Element-wise subtraction.

Element-wise multiplication.

Element-wise division.

```
[52]: A = np.array([[1, 2], [3, 4]])
      B = np.array([[5, 6], [7, 8]])

      add = A + B
      print("Elementwise Addition:\n", add)

      sub = A - B
      print("\nElementwise Subtraction:\n", sub)

      multi = A * B
      print("\nElementwise Multiplication:\n", multi)

      divi = A / B
      print("\nElementwise Division:\n", divi)

      Elementwise Addition:
      [[ 6  8]
       [10 12]]

      Elementwise Subtraction:
      [[-4 -4]
       [-4 -4]]

      Elementwise Multiplication:
      [[ 5 12]
       [21 32]]

      Elementwise Division:
      [[0.2      0.33333333]
       [0.42857143 0.5      ]]
```

20. Create a Pandas DataFrame with the given Name and marks of 3 courses:

Add a new column named 'Total' that represents the sum of all the courses. Add 'Grade' based on the values of the 'Total'. Print the updated DataFrame with the new 'Total' and 'Grade' column.

```
•[58]: import pandas as pd
data = {
    'Name': ['Vidya', 'Aditya', 'Divya'],
    'Course1': [85, 78, 92],
    'Course2': [90, 75, 88],
    'Course3': [80, 85, 95]
}
df = pd.DataFrame(data)
df['Total'] = df[['Course1', 'Course2', 'Course3']].sum(axis=1)

def calculate_grade(total):
    if total >= 250:
        return 'A'
    elif total >= 200:
        return 'B'
    else:
        return 'C'

df['Grade'] = df['Total'].apply(calculate_grade)
print("Updated\n", df)
```

```
Updated
   Name  Course1  Course2  Course3  Total  Grade
0  Vidya      85      90      80    255      A
1  Aditya      78      75      85    238      B
2  Divya      92      88      95    275      A
```



