**ASSIGNMENT-4**

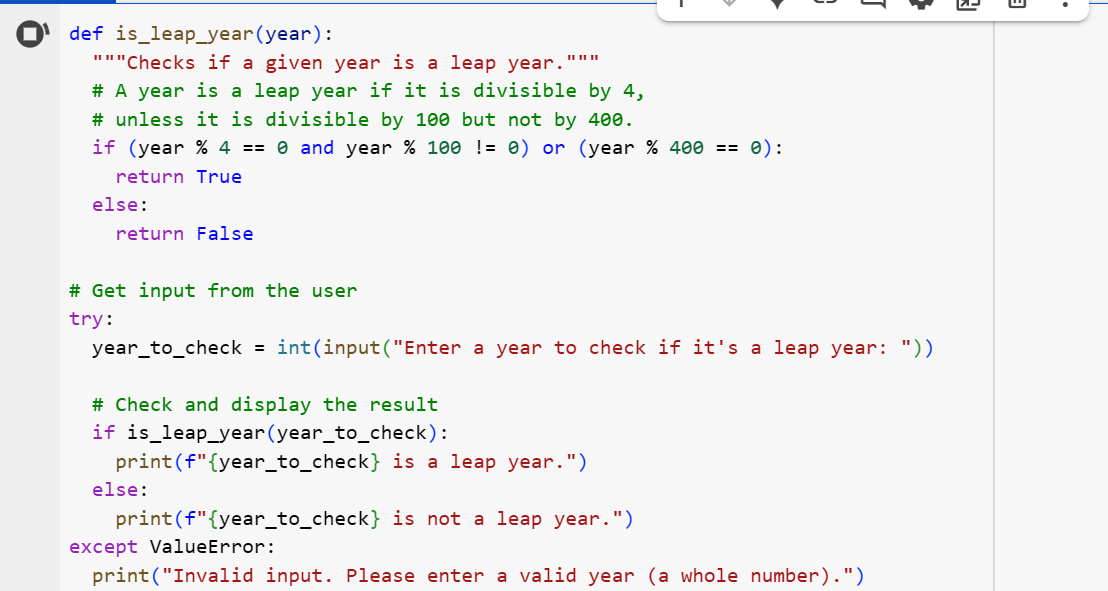
**NAME : G.SAI RAJ**

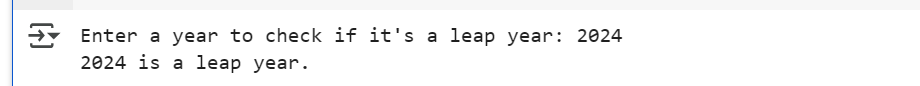
**HT NO: 2403A52149**

**BATCH:** 24BTCAIAIB06

**TASK-1**: Write a python program that checks the given year is leap year using functions.

**CODE AND OUPUT:**

****

****

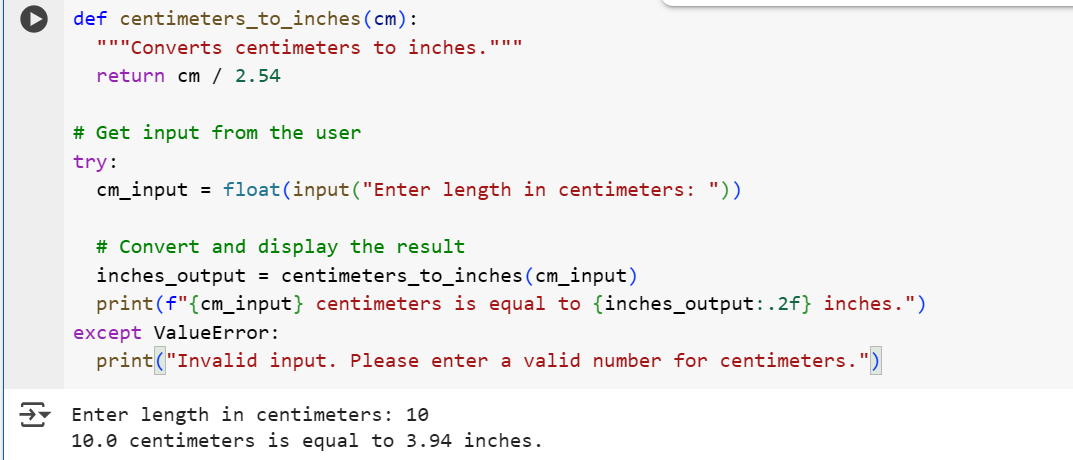
**EXPLANATION:**

1. **def is\_leap\_year(year):**: This line defines a function named is\_leap\_year that takes one argument, year, which is the year you want to check.
2. **"""Checks if a given year is a leap year."""**: This is a docstring explaining what the function does.
3. **if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):**: This is the core logic for checking a leap year based on the Gregorian calendar rules:
   * year % 4 == 0: Checks if the year is divisible by 4.
   * year % 100 != 0: Checks if the year is *not* divisible by 100.
   * year % 400 == 0: Checks if the year is divisible by 400.
   * The condition (year % 4 == 0 and year % 100 != 0) covers years divisible by 4 but not by 100 (e.g., 2004, 2008).
   * The condition (year % 400 == 0) covers years divisible by 400 (e.g., 2000, 2400).
   * The or combines these two conditions: a year is a leap year if it satisfies either of these.
4. **return True**: If the condition in the if statement is true, the function returns True, indicating it's a leap year.
5. **else: return False**: If the condition is false, the function returns False, indicating it's not a leap year.
6. **try...except ValueError:**: This block handles user input:
   * **year\_to\_check = int(input("Enter a year to check if it's a leap year: "))**: This prompts the user to enter a year and attempts to convert the input string into an integer using int().
   * **try:**: The code inside the try block is executed. If a ValueError occurs during the int() conversion (e.g., if the user enters text that cannot be converted to an integer), the code in the except block is executed.
   * **except ValueError:**: If a ValueError occurs, this block catches the error.
   * **print("Invalid input. Please enter a valid year (a whole number).")**: This prints an error message if the user's input was not a valid integer.
7. **if is\_leap\_year(year\_to\_check): ... else: ...**: If the input was successfully converted to an integer (i.e., no ValueError), the is\_leap\_year function is called with the year\_to\_check. Based on the boolean value returned by the function (True or False), the code prints whether the year is a leap year or not.

**TASK-2:**

Write a program in python that converts centimeters into inches using one shot**.**

**CODE AND OUPUT:**

****

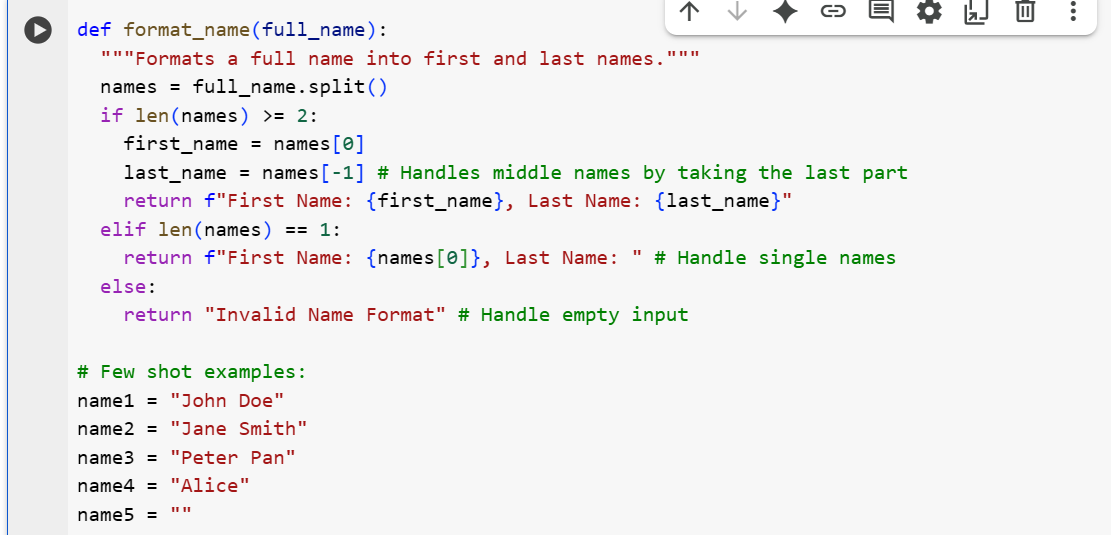
**EXPLANATION:**

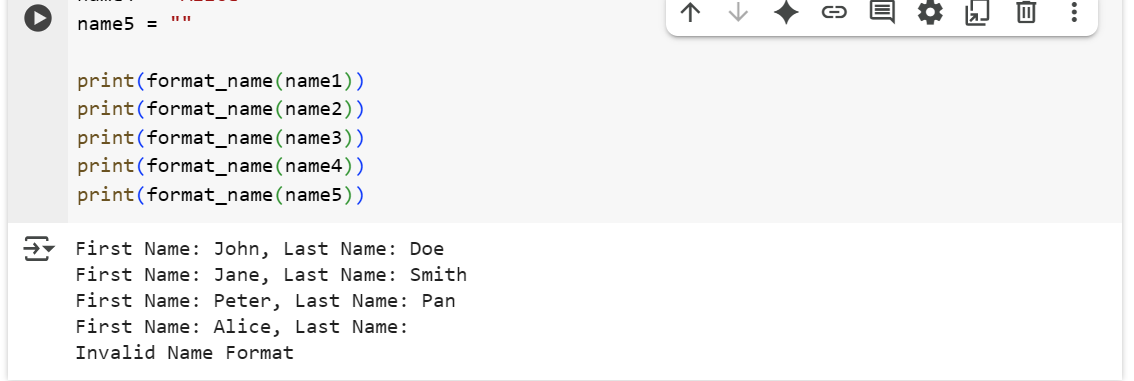
1. **def centimeters\_to\_inches(cm):**: This line defines a function named centimeters\_to\_inches that takes one argument, cm, which is the length in centimeters you want to convert.
2. **"""Converts centimeters to inches."""**: This is a docstring explaining what the function does.
3. **return cm / 2.54**: This line performs the conversion. Since there are 2.54 centimeters in 1 inch, dividing the number of centimeters by 2.54 gives the equivalent length in inches. The function returns this calculated value.
4. **try...except ValueError:**: This block handles user input:
   * **cm\_input = float(input("Enter length in centimeters: "))**: This prompts the user to enter a length in centimeters and attempts to convert the input string into a floating-point number using float(). Using float() allows for decimal values.
   * **try:**: The code inside the try block is executed. If a ValueError occurs during the float() conversion (e.g., if the user enters text that cannot be converted to a number), the code in the except block is executed.
   * **except ValueError:**: If a ValueError occurs, this block catches the error.
   * **print("Invalid input. Please enter a valid number for centimeters.")**: This prints an error message if the user's input was not a valid number.
5. **inches\_output = centimeters\_to\_inches(cm\_input)**: If the input was successfully converted to a float (i.e., no ValueError), the centimeters\_to\_inches function is called with the user's input (cm\_input) to perform the conversion. The result is stored in the inches\_output variable.
6. **print(f"{cm\_input} centimeters is equal to {inches\_output:.2f} inches.")**: This line prints the original centimeter value and the calculated inches value using an f-string for formatted output. The :.2f inside the f-string formats the inches\_output to two decimal places.

**TASK-3:**

Write a python program that formats full names into “first” and “last” using few shot.

**CODE AND OUPUT:**

****

****

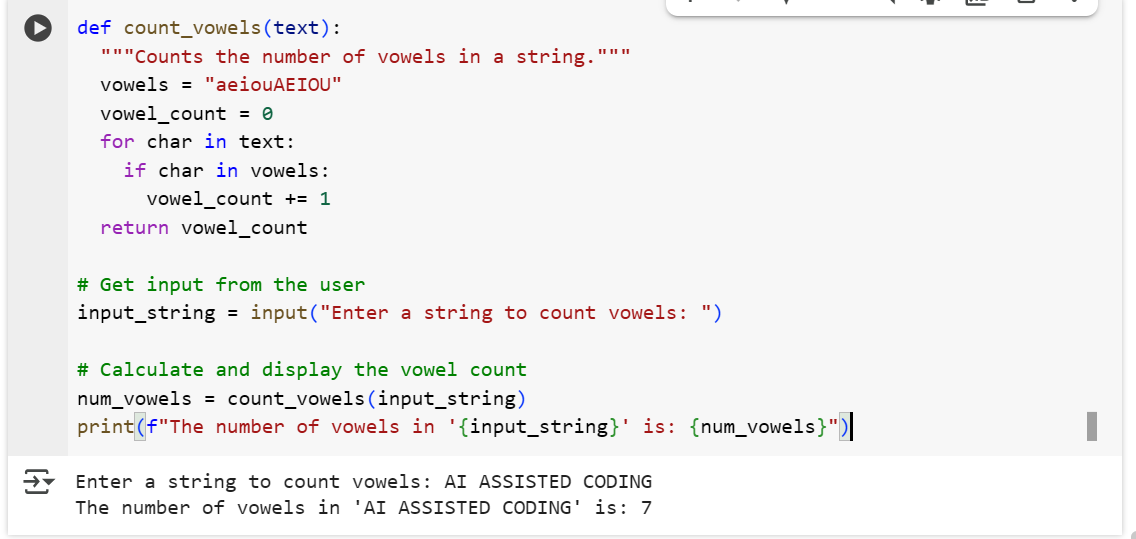
**EXPLANATION:**

1. **def format\_name(full\_name):**: This defines a function named format\_name that takes one argument, full\_name, which is the string containing the full name.
2. **"""Formats a full name into first and last names."""**: This is a docstring explaining the function's purpose.
3. **names = full\_name.split()**: This line splits the full\_name string into a list of words using spaces as the delimiter. For example, "John Doe" becomes ['John', 'Doe'].
4. **if len(names) >= 2:**: This checks if the list of names has two or more elements. This condition is true for names with at least a first and a last name.
   * **first\_name = names[0]**: The first element of the names list (the first name) is assigned to the first\_name variable.
   * **last\_name = names[-1]**: The last element of the names list (the last name) is assigned to the last\_name variable. Using [-1] is a convenient way to get the last element of a list, which helps handle names with middle names (it will take the last part as the last name).
   * **return f"First Name: {first\_name}, Last Name: {last\_name}"**: The function returns a formatted string showing "First Name: ..." and "Last Name: ...".
5. **elif len(names) == 1:**: This checks if the list of names has only one element. This condition is true for single names.
   * **return f"First Name: {names[0]}, Last Name: "**: The function returns a formatted string showing the single name as the "First Name" and an empty "Last Name".
6. **else: return "Invalid Name Format"**: If the list of names is empty (meaning the input full\_name was an empty string or contained only spaces), this condition is met, and the function returns the string "Invalid Name Format".
7. **# Few shot examples:**: This is a comment indicating the start of the example usage.
8. **name1 = "John Doe"**, **name2 = "Jane Smith"**, etc.: These lines define several example strings with different name formats.
9. **print(format\_name(name1))**, **print(format\_name(name2))**, etc.: These lines call the format\_name function with each of the example names and print the returned formatted string. This demonstrates how the function behaves with different inputs.

**TASK-4:**

Write a python program that counts the number of vowels in a string.

**CODE AND OUPUT:**

****

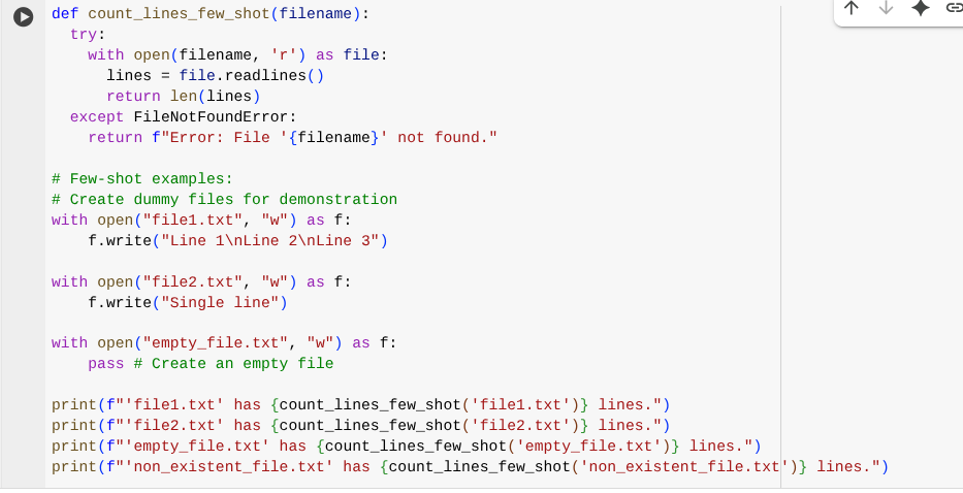
**EXPLANATION:**

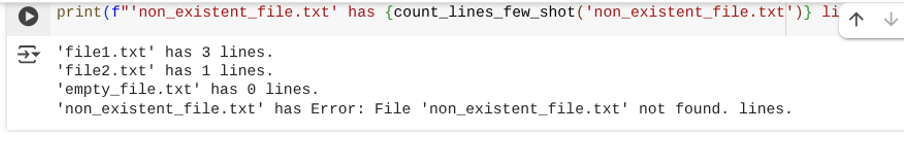
1. **def count\_vowels(text):**: This defines a function named count\_vowels that takes one argument, text, which is the string you want to analyze.
2. **"""Counts the number of vowels in a string."""**: This is a docstring explaining what the function does.
3. **vowels = "aeiouAEIOU"**: This line creates a string named vowels containing all lowercase and uppercase vowel characters. This makes it easy to check if a character is a vowel.
4. **vowel\_count = 0**: Initializes a variable vowel\_count to 0. This variable will store the total number of vowels found in the string.
5. **for char in text:**: This loop iterates through each character (char) in the input text string.
6. **if char in vowels:**: Inside the loop, this checks if the current char is present in the vowels string. If it is, it means the character is a vowel.
7. **vowel\_count += 1**: If the character is a vowel, the vowel\_count is incremented by 1.
8. **return vowel\_count**: After the loop has processed all characters in the string, the function returns the final vowel\_count.
9. **input\_string = input("Enter a string to count vowels: ")**: This line prompts the user to enter a string using the input() function and stores the entered string in the input\_string variable.
10. **num\_vowels = count\_vowels(input\_string)**: This line calls the count\_vowels function with the user-provided input\_string and stores the returned vowel count in the num\_vowels variable.
11. **print(f"The number of vowels in '{input\_string}' is: {num\_vowels}")**: This line prints the original input string and the calculated number of vowels using an f-string for formatted output.

**TASK-5:**

Write a python program that reads a .txt file and returns the number of lines using few short.

**CODE AND OUPUT:**

****

****

**EXPLANATION:**

1. **def count\_lines\_few\_shot(filename):**: Defines a function to count lines in a file.
2. **"""..."""**: A docstring explaining the function's purpose, arguments, and return value.
3. **try:**: Starts a block to handle potential errors, like the file not being found.
4. **with open(filename, 'r') as file:**: Opens the specified file for reading.
5. **lines = file.readlines()**: Reads all lines into a list.
6. **return len(lines)**: Returns the number of lines in the list.
7. **except FileNotFoundError:**: Catches the error if the file doesn't exist.
8. **return f"Error: File '{filename}' not found."**: Returns an error message if the file is not found.
9. **# Few-shot examples:**: Comments indicating the following lines are examples.
10. **# Create dummy files for demonstration**: Comment explaining dummy file creation.
11. **with open("file1.txt", "w") as f: f.write("Line 1\nLine 2\nLine 3")**: Creates "file1.txt" with three lines.
12. **with open("file2.txt", "w") as f: f.write("Single line")**: Creates "file2.txt" with one line.
13. **with open("empty\_file.txt", "w") as f: pass**: Creates an empty "empty\_file.txt".
14. **print(f"'file1.txt' has {count\_lines\_few\_shot('file1.txt')} lines.")**: Prints the line count for "file1.txt".
15. **print(f"'non\_existent\_file.txt'has {count\_lines\_few\_shot('non\_existent\_file.txt')} lines.")**: Prints the result for a non-existent file (showing the error handling).