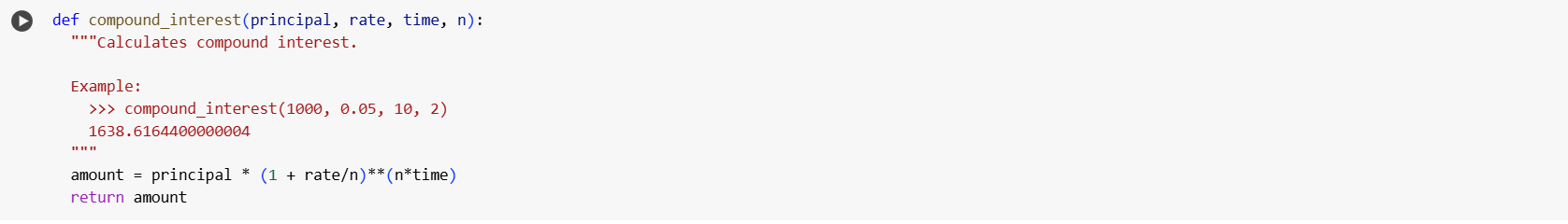
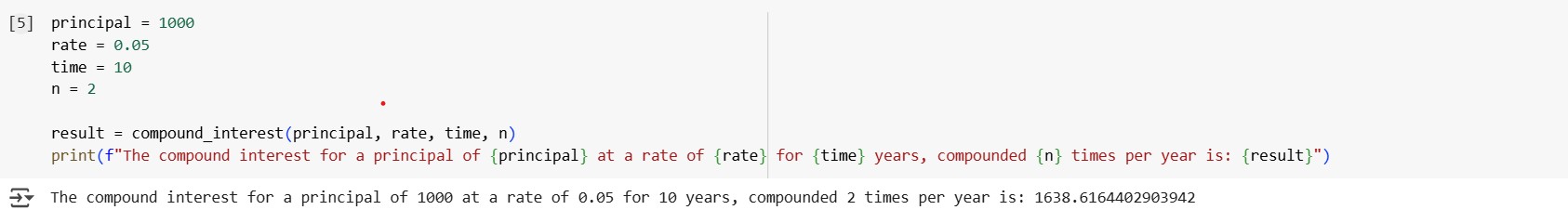
ASSIGNMENT-3.2

TASK-1

CODE:



OUTPUT:



EXPLANATION:

Here is a step-by-step explanation of the code in the last cell:

1. **principal = 1000**: This line assigns the value 1000 to the variable principal. This represents the initial amount of money invested or borrowed.
2. **rate = 0.05**: This line assigns the value 0.05 to the variable rate. This represents the annual interest rate as a decimal (5% is 0.05).
3. **time = 10**: This line assigns the value 10 to the variable time. This represents the number of years the money is invested or borrowed for.
4. **n = 2**: This line assigns the value 2 to the variable n. This represents the number of times that interest is compounded per year. In this case, it's compounded semi-annually (twice a year).
5. **result = compound\_interest(principal, rate, time, n)**: This line calls the compound\_interest function with the values stored in the variables principal, rate, time, and n. The value returned by the function (the calculated compound interest) is then stored in the variable result.
6. **print(f"The compound interest for a principal of {principal} at a rate of {rate} for {time} years, compounded {n} times per year is: {result}")**: This line prints the final output to the console. It uses an f-string to format the output nicely, including the values of the variables and the calculated result.

This code sets up the input values for the compound interest formula and then uses the compound\_interest function you defined earlier to calculate and display the final amount after compound interest.

TASK-2

CODE:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

EXPLANATION:

This code snippet implements the calculation of the average (mean) for the list of numbers provided to the calculate\_statistics function.

Here's a breakdown:

1. **if not data:**: This checks if the input list data is empty. An empty list is considered False in a boolean context.
2. **average = None**: If the list is empty, the average is undefined, so the variable average is set to None.
3. **else:**: This block is executed if the list is not empty.
4. **average = sum(data) / len(data)**: This is the standard formula for calculating the average. sum(data) calculates the sum of all numbers in the list, and len(data) gets the total count of numbers. The sum is then divided by the count to get the average.
5. **median = None** and **mode = None**: These lines are placeholders for the median and mode calculations, which are not yet implemented in this version of the function.
6. **return {"average": average, "median": median, "mode": mode}**: The function returns a dictionary containing the calculated average and the None placeholders for median and mode.

This code successfully calculates the average and handles the edge case of an empty list, which is an important step in building the complete statistics function.

TASK-3

CODE:

A screenshot of a computer

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer code

AI-generated content may be incorrect.

EXPLANATION:

Okay, let's break down the code in the cell where we got the output for the binary conversion (cell ID 89819a24):

1. **decimal\_number = 10**: This line initializes a variable named decimal\_number and assigns it the integer value 10. This is the decimal number that we want to convert to binary.
2. **binary\_output = convert\_to\_binary(decimal\_number)**: This is the core of the execution.
   * It calls the function convert\_to\_binary.
   * It passes the value stored in decimal\_number (which is 10) as the argument num to the function.
   * The convert\_to\_binary function (which we implemented using return bin(num)) takes the 10, converts it to its binary representation (0b1010), and returns this string value.
   * The returned binary string (0b1010) is then stored in the variable binary\_output.
3. **print(f"The binary representation of {decimal\_number} is: {binary\_output}")**: This line prints the result to the console.
   * It uses an f-string, which allows you to embed the values of variables directly within the string using curly braces {}.
   * It constructs the output message: "The binary representation of 10 is: 0b1010".

In simple terms, this code sets a decimal number, uses the function we created to convert it to binary, and then displays the original decimal number and its binary equivalent in a readable sentence.

Does that explanation clarify how the code works? Would you like to know more about the bin() function itself?

TASK-4

CODE: