List and Tuples Methods:

Problem 1: List Manipulation
Create a program that takes a list of numbers from the user and appends the squares of these numbers to a new list.

Steps to Follow:
1. Prompt the user for a list of numbers using the `input()` function.
2. Convert the input to a list using the `split()` function.
3. Use a loop to iterate through the list and calculate the squares.
4. Append the squared numbers to a new list.
5. Print the new list.

Problem 2: Tuple Packing and Unpacking
Write a Python program that packs information about a book (title, author, year) into a tuple and then unpacks and displays it.

Steps to Follow:
1. Create a tuple with information about a book (title, author, year).
2. Unpack the tuple and store the values in separate variables.
3. Print each piece of information.

Strings and String Methods:

Problem 3: Reverse String
Develop a program that takes a string from the user and prints its reverse.

Steps to Follow:
1. Prompt the user for a string using the `input()` function.
2. Use string slicing to reverse the string.
3. Print the reversed string.

Problem 4: String Formatting
Create a program that asks the user for their name and age and outputs a message using string formatting.

Steps to Follow:
1. Prompt the user for their name and age using the `input()` function.
2. Use string formatting to create a message like "Hello, [Name]! You are [Age] years old."
3. Print the message.

Indexing:

Problem 5: Extract Substring
Write a Python program that extracts a substring from a given string based on user-defined start and end indexes.

Steps to Follow:
1. Prompt the user for a string and two integer inputs for start and end indexes.
2. Use string indexing to extract the substring.
3. Print the extracted substring.

Problem 6: Last N Characters
Create a program that takes a string and an integer `n`, and prints the last `n` characters of the string.

Steps to Follow:
1. Prompt the user for a string using the `input()` function.
2. Prompt the user for an integer input for `n`.
3. Use string indexing to get the last `n` characters.
4. Print the result.

Conditional Statements and Comparison Operators:

Problem 7: Temperature Converter
Write a Python program that converts temperature from Celsius to Fahrenheit or vice versa based on user input.

Steps to Follow:
1. Prompt the user for a temperature value and the unit (C or F) using the `input()` function.
2. Convert the temperature to the desired unit using conditional statements.
3. Print the converted temperature.

Problem 8: Voting Eligibility
Create a program that checks whether a person is eligible to vote based on their age.

Steps to Follow:
1. Prompt the user for their age using the `input()` function.
2. Convert the input to an integer using the `int()` function.
3. Use conditional statements to check if the person is eligible to vote (age >= 18).
4. Print the result.

Arithmetic Operators and Print Formats:

Problem 9: Currency Converter
Develop a program that converts an amount in one currency to another currency based on a given exchange rate.

Steps to Follow:
1. Prompt the user for an amount in one currency using the `input()` function.
2. Prompt the user for the exchange rate.
3. Convert the amount to the other currency using arithmetic operators.
4. Print the converted amount.

Problem 10: Formatted Division
Write a Python program that takes two numbers as input and prints their division result in a specific format.

Steps to Follow:
1. Prompt the user for two numbers using the `input()` function.
2. Convert the inputs to floats using the `float()` function.
3. Calculate the division result.
4. Use string formatting to display the result with a specific number of decimal places.
5. Print the formatted result.

List and Tuple Methods Continued:

Problem 11: List Sorting
Create a program that takes a list of names from the user and sorts them in alphabetical order.

Steps to Follow:
1. Prompt the user for a list of names using the `input()` function.
2. Convert the input to a list using the `split()` function.
3. Use the `sort()` method to sort the list.
4. Print the sorted list.

Problem 12: Tuple Concatenation
Write a Python program that takes two tuples from the user and concatenates them into a single tuple.

Steps to Follow:
1. Prompt the user for two tuples using the `input()` function.
2. Convert the inputs to tuples using the `eval()` function.
3. Concatenate the two tuples using the `+` operator.
4. Print the concatenated tuple.

String Methods Continued:

Problem 13: Counting Vowels
Develop a program that takes a string from the user and counts the occurrences of vowels (a, e, i, o, u).

Steps to Follow:
1. Prompt the user for a string using the `input()` function.
2. Convert the string to lowercase for consistent comparison.
3. Use the `count()` method to count the occurrences of each vowel.
4. Print the counts.

Problem 14: Title Case Converter
Create a program that takes a sentence from the user and converts it to title case.

Steps to Follow:
1. Prompt the user for a sentence using the `input()` function.
2. Use the `title()` method to convert the sentence to title case.
3. Print the converted sentence.

Indexing and Conditional Statements Combined:

Problem 15: Password Checker
Write a Python program that prompts the user to enter a password and checks its strength based on length and character types.

Steps to Follow:
1. Prompt the user for a password using the `input()` function.
2. Check if the password length is at least 8 characters using a conditional statement.
3. Use string methods to check if the password contains uppercase letters, lowercase letters, numbers, and special characters.
4. Assign a strength level to the password based on the criteria met.
5. Print a message indicating the password's strength level.

Problem 16: String Truncation
Develop a program that takes a long string from the user and truncates it to a specified length.

Steps to Follow:
1. Prompt the user for a long string using the `input()` function.
2. Prompt the user for an integer input representing the desired length.
3. Use string indexing to truncate the string to the desired length.
4. Print the truncated string.

More Arithmetic Operators and Print Formats:

Problem 17: Compound Interest Calculator
Create a program that calculates and prints the compound interest for a given principal amount, interest rate, and time.

Steps to Follow:
1. Prompt the user for the principal amount, interest rate, and time using the `input()` function.
2. Convert the inputs to appropriate data types using the `float()` function.
3. Calculate the

compound interest using the formula: `A = P * (1 + r/n)^(nt)`, where A is the final amount, P is the principal amount, r is the annual interest rate, n is the number of times interest is compounded per year, and t is the time in years.
4. Print the calculated compound interest.

Problem 18: Formatting Decimal Places
Write a Python program that takes a floating-point number from the user and prints it with a specified number of decimal places.

Steps to Follow:
1. Prompt the user for a floating-point number using the `input()` function.
2. Convert the input to a float using the `float()` function.
3. Prompt the user for an integer input representing the desired number of decimal places.
4. Use string formatting to display the number with the specified decimal places.
5. Print the formatted number.

Conditional Statements with Logical Operators:

Problem 19: Leap Year Range
Develop a program that takes a start and end year from the user and prints all the leap years within that range.

Steps to Follow:
1. Prompt the user for a start and end year using the `input()` function.
2. Convert the inputs to integers using the `int()` function.
3. Use a loop to iterate through the range of years.
4. Use conditional statements to determine leap years (divisible by 4, not divisible by 100 unless also divisible by 400).
5. Print the leap years.

Problem 20: Grading System
Create a program that takes a student's score and calculates their grade based on a grading scale.

Steps to Follow:
1. Prompt the user for a numeric score using the `input()` function.
2. Convert the input to a float using the `float()` function.
3. Use conditional statements to determine the grade based on a grading scale (e.g., A for 90-100, B for 80-89, etc.).

4. Print the calculated grade.

Remember to test each problem with different inputs to ensure its correctness and robustness.