1. **Print Stars (Single Input):**
   Write a Python function `print_stars(n)` that receives a natural number n and prints a line of `n` asterisks '*'.
2. **Draw Rectangle (Two Inputs):**
   Write a Python function `draw_rectangle(rows, cols)` that receives two natural numbers `rows` and `cols` and draws a rectangle of asterisks '*' with `rows` rows and `cols` columns.
3. **Draw Triangle (Single Input):**
   Write a Python function `draw_triangle(n)` that receives a natural number n and draws a triangle of asterisks '*' with a height of n.
4. **Draw Rhombus (Single Input):**
   Write a Python function `draw_rhombus(n)` that receives a natural number n and draws a rhombus of asterisks '*' with a width of n. The rhombus should have n asterisks in the middle.

<u>Input() in python scripts</u>

In Python, you can use the `input()` function to receive user input from the keyboard. This input can be stored in variables and used within your script. Here's a short explanation of how to use `input()` with two examples:

Example 1: Basic Input

```python
# Simple input and output
name = input("Enter your name: ")  # Prompt the user for their name
print("Hello, " + name + "!")     # Print a personalized greeting
```

In this example, the `input()` function displays the message "Enter your name: " and waits for the user to type something. Whatever the user enters is stored in the variable `name`, and then it's used in a greeting message.

Example 2: Calculations with Input

```python
# Input for numeric values and calculations
num1 = float(input("Enter a number: "))   # Prompt for the first number
num2 = float(input("Enter another number: "))  # Prompt for the second number

sum_result = num1 + num2  # Add the two numbers
print("The sum is:", sum_result)

product_result = num1 * num2  # Multiply the two numbers
print("The product is:", product_result)
```

In this example, the `input()` function is used to gather two numeric values. To ensure they are treated as numbers, we use `float()` to convert the user input to floating-point numbers. Then, we perform calculations with these values and display the results.

When using `input()`:

1. Always provide a clear and informative prompt to guide the user.
2. Remember that `input()` returns a string, so convert it to the appropriate data type (e.g., `int` or `float`) if needed.
3. Be mindful of error handling, as user input can be unpredictable, and it's a good practice to validate and handle unexpected input to prevent crashes or errors in your script.

Exercise 1: Determine if a Number is Prime or Composite

Create a Python script that takes a user-input number and determines whether it's a prime or composite number. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. A composite number is a natural number greater than 1 that has divisors other than 1 and itself. Additionally, make sure to handle cases where the input is not a natural number.

Exercise 2: Leap Year Checker

Write a Python script that does the following:

1. Asks the user to input a 4-digit year (e.g., 2024) using the `input()` function.
2. Checks if the input is a valid 4-digit year.
3. Determines whether the entered year is a leap year or not based on the leap year rule.
4. Prints a message indicating whether the year contains February 29th or not.

<u>Leap Year Rule:</u>

In the Gregorian calendar, leap years are determined by the following rule:

- A year is a leap year if it is divisible by 4.
- However, years divisible by 100 are not leap years unless they are also divisible by 400.

For example, if the user enters "2024," the script should print "Leap year!" because 2024 meets the leap year criteria.

Ensure that your script handles valid 4-digit years greater than 0 and provides the appropriate output based on the leap year rule.