Exercise 1 - Testing Battleships

We're going to play a game of "**Bugtleship**"! For this activity you'll be split into groups.
Phase 1 - Groups will be given some functions to implement from the list below.
Phase 2 - Groups will come up with funny arguments to break other groups' functions. Be meticulous.
Phase 3 - Implement error handling to fix the vulnerabilities found by the other groups.

Function to implement:

- remainder() - $(x, y) \rightarrow x \; mod \; y$
- exponent() - $(x, y) \rightarrow x^y$
- factorial() - $(x) \rightarrow x!$
- nand() - $(bool1, \; bool2) \rightarrow bool1 \; nand \; bool2$
- xor() - $(bool1, \; bool2) \rightarrow bool1 \; xor \; bool2$
- is_integer() - $(x) \rightarrow bool$
- is_odd() - $(x) \rightarrow bool$
- get_full_name() - $(title, \; first\_name, \; last\_name) \rightarrow full\_name$
- conceal_password() - $(plain\_password) \rightarrow$ " **... * "

Exercise 2

Draft a small test plan for a login page. The page has 2 input fields for username and password, and 2 buttons for login and forgotten password.

- Define the scope: What will be tested? What's out of scope?
- Specify entry and exit criteria.
- Identify at least 3 test cases

Exercise 3:

Sign-Up Password Validator (with Error Handling)
You're building a user sign-up system. You must validate the user's password according to specific rules. If the password does not meet the criteria, raise and handle custom exceptions.
Requirements:
The password must:

- Be between 8 and 16 characters long.
- Contain at least one uppercase letter.

- Contain at least one digit.

Exercise 4:

Create a program that simulates a simple bank transfer between two accounts (You are going to define a function to perform the transfer). The user must input:

- Their account balance (must be a number ≥ 0)
- The amount they want to transfer (must be a number ≥ 0)
- The recipient's account number (must be exactly 10 digits, numbers only)

You must handle the following errors with meaningful messages:

- • ValueError if any input is not a number.
- • A custom Exception if:
  - The balance is less than the transfer amount.
  - The account number is not exactly 10 digits or contains non-digit characters.
  - The amount is negative or zero.

Exercise 5.

You are going to create a class and define 4 methods

- Add two numbers
- Multiply two numbers
- Subtract two numbers
- Divide two numbers

Exercise 5

Write a function that tests whether a credit card number string:

- Has 16 digits.
- Contains only digits.
- Is grouped as "XXXX-XXXX-XXXX-XXXX" (4 groups separated by -).

Use assertions to validate these.

Exercise 6

Create a function and test the function:
In a factory, each product has a unique identification number.
However, due to a system error, some numbers have been assigned to more than one product.
Find the first identification number that is repeated, where the second occurrence has the smallest index in the list.
In other words:

- If there is more than one repeated number, return the number whose second occurrence appears first in the list.
- If there are no repeated numbers, return -1.

Extension Activities:

Write functions for the following, assertion testing each.

Given an age, test the age is between 0 and 120.
Given a password, test the password is between 9 and 16 characters.
Given a number test if the number is even.
Given a list of numbers, test that the list is not empty.
Given an email, test the email contains @ symbol.
Given a dictionary, test the dictionary has the keys “name”, “age”, and “email”