1) Design and implement a **console-based Library Book Management System** in Python. The system should store available books in a dictionary, where each key represents a book title (string) and the corresponding value represents the number of copies available (integer). The program must allow the user to borrow books by entering the book name and the number of copies they wish to borrow. If the entered book name does not exist in the dictionary, handle the situation using a **KeyError** exception and display an appropriate message. If the user enters a non-integer value for the number of copies, handle it using a **ValueError** exception. Additionally, if the user tries to borrow more copies than are available, create and raise a **custom exception** to display a proper error message. The program should run in a loop, allowing multiple borrowing attempts, and should only terminate when the user types "exit". Even if an error occurs during the process, the program should continue running, and separate try-except blocks should be used to ensure smooth execution for each borrowing attempt.
2) Write a Python program to simulate an **online shopping cart**.
    a. Maintain an **inventory dictionary** where keys are product names and values are their prices.
    b. The user can enter an item name and quantity to purchase.
    c. Handle the following exceptions:
        i. **KeyError** if the product does not exist in the inventory.
        ii. **ValueError** if the entered quantity is not a number.
        iii. **ZeroDivisionError** if you attempt to calculate a discount percentage where the total price is zero.
    d. Use a **list of tuples** to store purchased items in the format (item_name, quantity, total_price).
    e. At the end, display a bill showing all purchased items, their quantities, and prices.
    f. Ensure that **invalid inputs** do not crash the program but instead prompt the user again.
3) You are writing a signup system for a web application.
    a. Create a custom exception InvalidDomainError that inherits from Exception.
    b. Write a function validate_email(email) that:
        i. Checks if the email contains "@". If not, raise ValueError.
        ii. Checks if the domain (part after "@") is "example.com". If not, raise InvalidDomainError.
        iii. Returns "Valid email" if all checks pass.
    c. Use try–except to:
        i. Catch ValueError and print "Email must contain '@' symbol.".
        ii. Catch InvalidDomainError and print "Email domain must be 'example.com'.".
        iii. Catch any other exception and print "Unexpected error: <error message>"