Double-click (or enter) to edit

ATM Withdrawal: Write a function that simulates an ATM withdrawal. The function should check if the account balance is sufficient for the withdrawal amount and raise an exception if not. Handle scenarios where the input withdrawal amount is not a number or is negative.

```
def atm_withdrawal(balance, amount):
    try:
        amount = float(amount)  # Convert to float to handle both integers and floats
        if amount <= 0:
            raise ValueError("Withdrawal amount must be positive.")
        if amount >= balance:
            raise ValueError("Insufficient balance for the withdrawal.")

        balance -= amount
        print(f"Successfully withdrew ${amount:.2f}. Remaining balance: ${balance:.2f}.")
        return balance

    except ValueError as e:
        print(f"Error: {e}")

# Example usage


atm_withdrawal(5000, 150)  # Successful withdrawal
atm_withdrawal(500, "abc")  # Invalid input
atm_withdrawal(450, -50)  # Invalid amount
atm_withdrawal(300, 400)  # Insufficient balance
```

```
Successfully withdrew $150.00. Remaining balance: $4850.00.
Error: could not convert string to float: 'abc'
Error: Withdrawal amount must be positive.
Error: Insufficient balance for the withdrawal.
```

User Login System: Create a function that simulates a user login system. It should raise an exception if the username or password is incorrect and handle cases where the input values are empty strings.

```
def check(username, password):
  try:
    user=input("Enter the username: ")
    password1=input("Enter the password: ")
    if username == "" or password == "":
        raise ValueError("Username or password cannot be empty")
    if(username==user and password1==password):
      print("Correct password")
    else:
      raise ValueError("Incorrect username or password")
  except ValueError as e:
    print(e)
check("mehak", 123)
```

```
Enter the username: 
Enter the password: 
Username or password cannot be empty
```

Online Shopping Cart: Write a function that adds items to an online shopping cart. Handle scenarios where the item is out of stock, the item ID is invalid, or the quantity requested is more than the available stock.

```
def add_items(items):
  cart={}
  if cart
```

```
import random

def get_user_choice():
    user_choice = input("Enter your choice (rock, paper, scissors): ").lower()
    while user_choice not in ["rock", "paper", "scissors"]:
        user_choice = input("Invalid choice. Please enter rock, paper, or scissors: ").lower()
    return user_choice

def get_computer_choice():
    return random.choice(["rock", "paper", "scissors"])

def determine_winner(user_choice, computer_choice):
    if user_choice == computer_choice:
        return "It's a tie!"
    elif (user_choice == "rock" and computer_choice == "scissors") or \
         (user_choice == "paper" and computer_choice == "rock") or \
         (user_choice == "scissors" and computer_choice == "paper"):
```

```
import random

def play_game():
    choices = ["rock", "paper", "scissors"]

    user_choice = input("Enter your choice (rock, paper, scissors): ").lower()
    while user_choice not in choices:
        user_choice = input("Invalid choice. Please enter rock, paper, or scissors: ").lower()

    computer_choice = random.choice(choices)

    print(f"\nYou chose: {user_choice}")
    print(f"Computer chose: {computer_choice}")

    if user_choice == computer_choice:
        result = "It's a tie!"
    elif (user_choice == "rock" and computer_choice == "scissors") or \
         (user_choice == "paper" and computer_choice == "rock") or \
         (user_choice == "scissors" and computer_choice == "paper"):
        result = "You win!"
    else:
        result = "You lose!"

    print(result)

if __name__ == "__main__":
    play_game()
```

```
Enter your choice (rock, paper, scissors): rock

You chose: rock
Computer chose: rock
It's a tie!
```