import random # For simulating the synthetic market

def analyze\_market(past\_ticks, threshold, digits\_list):

"""Analyzes the recent market ticks for the specified conditions."""

if len(past\_ticks) < threshold:

return False # Not enough data to analyze

# Check for consecutive occurrences and allowed digits

for digit in digits\_list:

if past\_ticks.count(str(digit)) < threshold:

return False

return True

def get\_past\_ticks(num\_ticks):

"""Simulates getting past ticks from the synthetic market."""

# In a real implementation, you'd fetch this from the market API

return [str(random.randint(0, 9)) for \_ in range(num\_ticks)]

def place\_trade(over\_under, target):

"""Simulates placing a trade in the synthetic market."""

# In a real implementation, this would interact with your broker API

market\_outcome = random.randint(0, 9) # Simulate market outcome

return market\_outcome > target if over\_under == "over" else market\_outcome < target

def main():

current\_strategy = "over1" # Start with "over 1" strategy

loss\_recovery\_mode = False

balance = 1000 # Starting balance (adjust as needed)

stake = 10 # Stake amount per trade (adjust as needed)

while True:

past\_ticks = get\_past\_ticks(5) # Get the last 5 ticks

if loss\_recovery\_mode:

if analyze\_market(past\_ticks, 4, [0, 1, 2, 3, 4]):

trade\_result = place\_trade("over", 4)

if trade\_result:

balance += stake

loss\_recovery\_mode = False # Switch back to over 1

print("Loss recovered! Switching back to Over 1 strategy.")

else:

balance -= stake

print("Loss recovery failed. Retrying...")

else: # Regular "over 1" strategy

if analyze\_market(past\_ticks, 3, [0, 1]):

trade\_result = place\_trade("over", 1)

if trade\_result:

balance += stake

else:

balance -= stake

loss\_recovery\_mode = True # Enter loss recovery mode

print("Loss incurred. Switching to Over 4 recovery strategy.")

print(f"Current balance: {balance}")

if \_\_name\_\_ == "\_\_main\_\_":

main()