Task 1:

def PlayGame(Targets, NumbersAllowed, TrainingGame, MaxTarget, MaxNumber):

Score = 0

GameOver = False

while not GameOver:

DisplayState(Targets, NumbersAllowed, Score)

UserInput = input("Enter an expression or enter QUIT to exit the game: ")

print()

#added IF statement

if UserInput != "QUIT":

if CheckIfUserInputValid(UserInput):

UserInputInRPN = ConvertToRPN(UserInput)

if CheckNumbersUsedAreAllInNumbersAllowed(NumbersAllowed, UserInputInRPN, MaxNumber):

IsTarget, Score = CheckIfUserInputEvaluationIsATarget(Targets, UserInputInRPN, Score)

if IsTarget:

NumbersAllowed = RemoveNumbersUsed(UserInput, MaxNumber, NumbersAllowed)

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber)

Score -= 1

if Targets[0] != -1:

GameOver = True

else:

Targets = UpdateTargets(Targets, TrainingGame, MaxTarget)

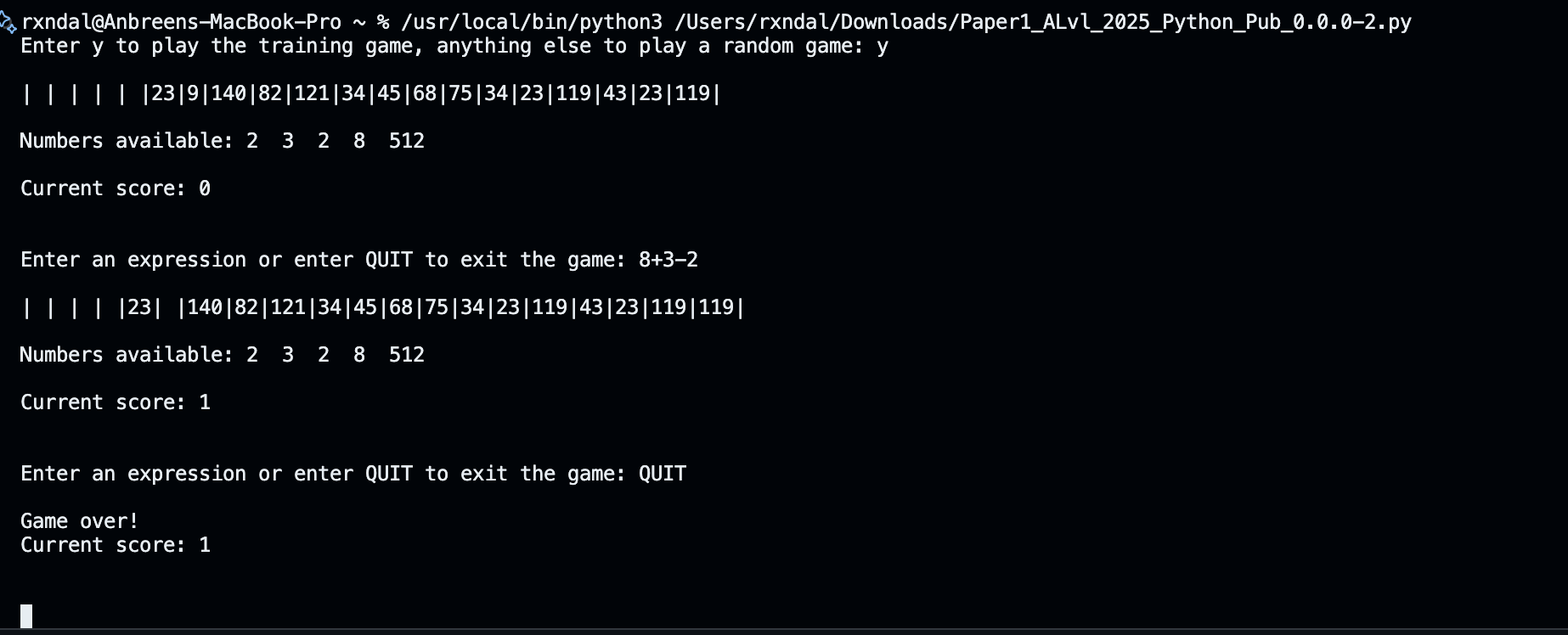
#If QUIT is entered, then GameOver = True

else:

GameOver = True

print("Game over!")

DisplayScore(Score)



Task 2:

if IsTarget:

NumbersAllowed = RemoveNumbersUsed(UserInput, MaxNumber, NumbersAllowed)

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber)

#added error message in case expression does not evaluate to a target

else:

print("Error, expression does not evaluate to a target. ")

def CheckNumbersUsedAreAllInNumbersAllowed(NumbersAllowed, UserInputInRPN, MaxNumber):

Temp = []

for Item in NumbersAllowed:

Temp.append(Item)

for Item in UserInputInRPN:

if CheckValidNumber(Item, MaxNumber):

if int(Item) > MaxNumber:

print("Error, expression contains numbers greater than the max number. ")

return False

else:

if int(Item) in Temp:

Temp.remove(int(Item))

else:

print("Error, expression does not contain valid numbers from the valid number list. ")

return False

return True

def CheckIfUserInputValid(UserInput):

#modified to search for a number / 0 before it searches normally

if re.search("^([0-9]+[\\/])+[0]+$", UserInput) is not None:

print("Error, expression divides by 0 which is not acceptable!")

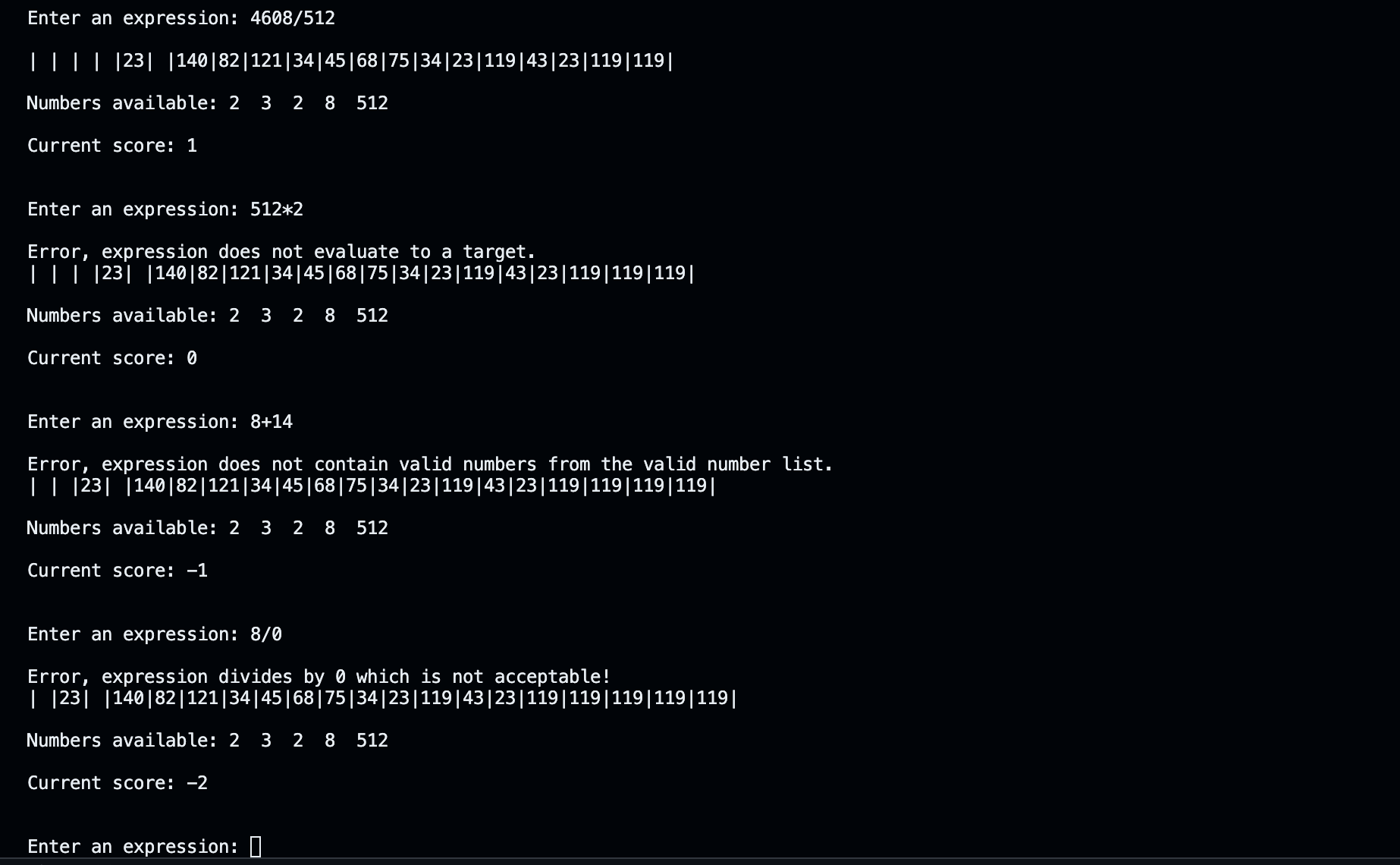
return False

elif re.search("^([0-9]+[\\+\\-\\\*\\/])+[0-9]+$", UserInput) is not None:

return True

else:

return False



Task 3:

def UpdateTargets(Targets, TrainingGame, MaxTarget):

Targets = Targets[1:]

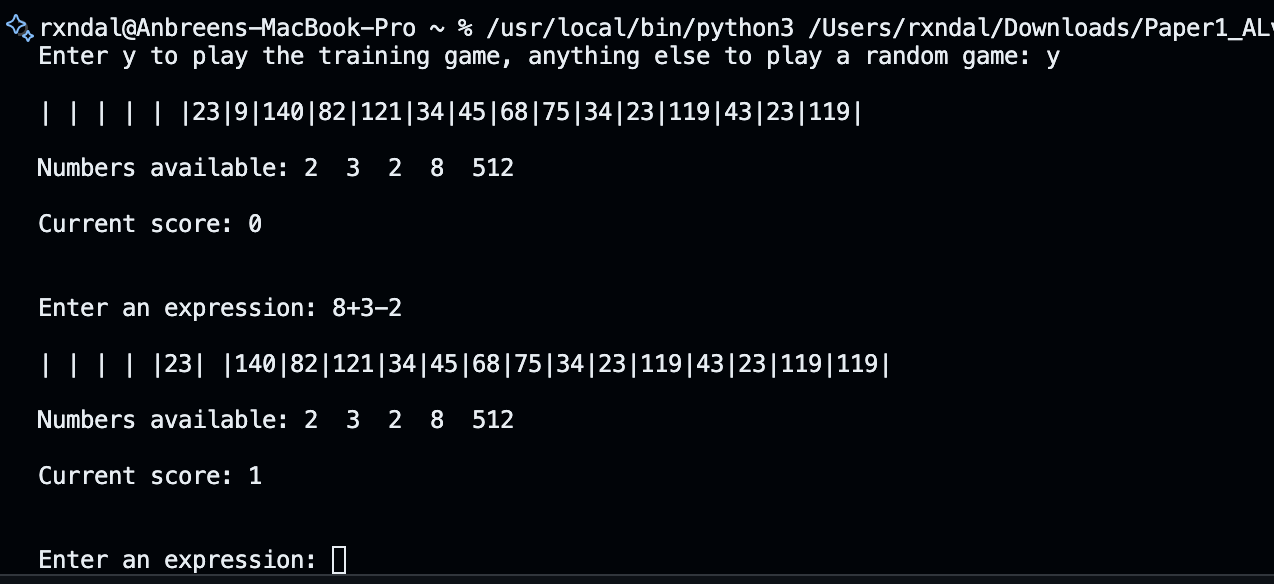
if TrainingGame:

Targets.append(Targets[-1])

else:

Targets.append(GetTarget(MaxTarget))

return Targets



Task 4:

def FillNumbers(NumbersAllowed, TrainingGame, MaxNumber, LargeNumbers=None, num\_large=0, num\_standard=5):

if TrainingGame:

return [2, 3, 2, 8, 512]

else:

if LargeNumbers is not None:

NumbersAllowed.extend(random.sample(LargeNumbers, num\_large))

while len(NumbersAllowed) < num\_large + num\_standard:

NumbersAllowed.append(GetNumber(MaxNumber))

return NumbersAllowed

def Main():

NumbersAllowed = []

Targets = []

MaxNumberOfTargets = 20

MaxTarget = 0

MaxNumber = 0

TrainingGame = False

LargeNumbers = [25, 50, 75, 100]

Choice = input("Enter y to play the training game, anything else to play a random game: ").lower()

print()

if Choice == "y":

MaxNumber = 1000

MaxTarget = 1000

TrainingGame = True

Targets = [-1, -1, -1, -1, -1, 23, 9, 140, 82, 121, 34, 45, 68, 75, 34, 23, 119, 43, 23, 119]

else:

MaxNumber = 10

MaxTarget = 50

Targets = CreateTargets(MaxNumberOfTargets, MaxTarget)

print("Select difficulty level:")

print("1. Easy")

print("2. Standard")

print("3. Medium")

print("4. Hard")

Difficulty = input("Enter the number corresponding to your choice: ")

if Difficulty == "1":

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber, LargeNumbers, 1, 4)

elif Difficulty == "2":

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber, LargeNumbers, 0, 5)

elif Difficulty == "3":

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber, LargeNumbers, 2, 3)

elif Difficulty == "4":

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber, LargeNumbers, 4, 1)

else:

print("Invalid choice. Defaulting to Standard difficulty.")

NumbersAllowed = FillNumbers(NumbersAllowed, TrainingGame, MaxNumber, LargeNumbers, 0, 5)

PlayGame(Targets, NumbersAllowed, TrainingGame, MaxTarget, MaxNumber)

input()