**Question 1**

(a)

Python code

while True:

bp\_input = input("Enter blood pressure readings (mmHg): ")

if bp\_input == "":

print("Thank you and keep monitoring your blood pressure.")

break

else:

bp\_list = bp\_input.split("/")

systolic = int(bp\_list[0])

diastolic = int(bp\_list[1])

if systolic < 90 or diastolic < 60:

print("Your blood pressure is low.")

elif systolic >= 90 and systolic <= 119 and diastolic >= 60 and diastolic <= 79:

print("Your blood pressure is normal.")

elif systolic >= 120 and systolic <= 129 and diastolic >= 80 and diastolic <= 84:

print("You have Elevated blood pressure.")

elif systolic >= 130 and systolic <= 139 or diastolic >= 85 and diastolic <= 89:

print("You have Stage 1 Hypertension.")

elif systolic >= 140 or diastolic >= 90:

print("You have Stage 2 Hypertension.")

b)

while True:

bp\_input = input("Enter blood pressure readings (mmHg): ")

if bp\_input == "":

print("Thank you and keep monitoring your blood pressure.")

break

else:

bp\_list = bp\_input.split("/")

systolic = int(bp\_list[0])

diastolic = int(bp\_list[1])

if systolic <= 0 or diastolic <= 0:

print("You have entered invalid BP numbers.")

elif systolic < 90 or diastolic < 60:

print("Your blood pressure is low.")

elif systolic >= 90 and systolic <= 119 and diastolic >= 60 and diastolic <= 79:

print("Your blood pressure is normal.")

elif systolic >= 120 and systolic <= 129 and diastolic >= 80 and diastolic <= 84:

print("You have Elevated blood pressure.")

elif systolic >= 130 and systolic <= 139 or diastolic >= 85 and diastolic <= 89:

print("You have Stage 1 Hypertension.")

elif systolic >= 140 or diastolic >= 90:

print("You have Stage 2 Hypertension.")

elif systolic >= 180 or diastolic >= 120:

print("Hypertensive Crisis!! Consult your doctor immediately.")

**Question 2**

a)

Here's the implementation of the ticketValidator function:

def ticketValidator(numbers):

# Check if all numbers are within 1 to 49

for num in numbers:

if num < 1 or num > 49:

return False

# Check if all numbers are unique

if len(numbers) != len(set(numbers)):

return False

return True

# Test cases

print(ticketValidator([5, 47, 6, 32, 49])) # True

print(ticketValidator([5, 6, 7, 6, 45, 31])) # False

print(ticketValidator([51, 6, 7])) # False

Explanation

This function takes a list of numbers and returns True if all the numbers are unique and within the range of 1 to 49. Otherwise, it returns False.

b)

def ticketValidator(numbers):

# Check if all numbers are unique and within 1 to 49

if len(numbers) != len(set(numbers)):

return False

for number in numbers:

if number < 1 or number > 49:

return False

return True

def quickPick():

# Generate 6 unique random numbers from 1 to 49

numbers = []

while len(numbers) < 6:

number = random.randint(1, 49)

if number not in numbers:

numbers.append(number)

return numbers

c)

import random

def ticketValidator(numbers):

# Check if all numbers are unique and within 1 to 49

if len(numbers) != len(set(numbers)):

return False

for number in numbers:

if number < 1 or number > 49:

return False

return True

def quickPick():

# Generate 6 unique random numbers from 1 to 49

numbers = []

while len(numbers) < 6:

number = random.randint(1, 49)

if number not in numbers:

numbers.append(number)

return numbers

while True:

print("TOTO Menu")

print("1. Ticket Entry")

print("2. Quick Pick")

print("0. Exit")

option = int(input("Enter option: "))

if option == 0:

print("Good luck to you!!")

break

if option == 1:

numbers = []

for i in range(1, 7):

number = int(input(f"Enter pick {i}: "))

numbers.append(number)

if ticketValidator(numbers):

print(f"Your TOTO ticket {numbers} is valid")

else:

print(f"{numbers} is an Invalid TOTO ticket")

elif option == 2:

numbers = quickPick()

print(f"This is your lucky ticket: {numbers}")

Explanation

In this program, we have defined the ticketValidator function that takes a list of 6 numbers and checks if they are all unique and within 1 to 49. We have also defined the quickPick function that generates 6 unique random numbers from 1 to 49.

The program then presents a menu to the user, where they can choose to enter their own TOTO ticket or get a quick pick from the computer. The program loops until the user chooses to exit by entering 0.

If the user chooses option 1 to enter their own ticket, the program prompts them to enter their 6 picks and then checks if the ticket is valid using the ticketValidator function. If the ticket is valid, the program displays a message confirming that the ticket is valid. If the ticket is invalid, the program displays a message indicating that the ticket is invalid.

If the user chooses option 2 to get a quick pick, the program generates 6 unique random numbers using the quickPick function and displays the ticket to the user.

Finally, if the user chooses option 0 to exit, the program displays a farewell message and exits the loop.

**Question 3**

1. Here's the implementation for the required functions:

import math

# Function to read pricing data from a file and store it in a dictionary

def readPricing(filename, pricing):

with open(filename, "r") as file:

for line in file:

char, price = line.strip().split()

pricing[char] = int(price)

# Function to display pricing table according to font size

def showPricing(charPricing, fontPricing, fontSize):

# Calculate the factor by which the pricing needs to be multiplied based on font size

factor = 1

if fontSize == 36:

factor = 1.5

elif fontSize == 48:

factor = 2

# Round the prices to the nearest integer and display the pricing table

print("Pricing for font size {}pts".format(fontSize))

print("="\*27)

row = ""

for char, price in charPricing.items():

# Calculate the new price based on the factor and round it to the nearest integer

newPrice = round(price \* factor)

# Add the character and its price to the current row

row += "{} {} ".format(char, newPrice)

# If the row has 10 characters, display it and start a new row

if len(row) >= 20:

print(row)

row = ""

# If there are any characters left in the last row, display it

if row != "":

print(row)

# Main function to tie everything together

def main():

# Create empty dictionaries for character pricing and font pricing

charPricing = {}

fontPricing = {}

# Read character pricing and font pricing from files and store them in the dictionaries

readPricing("char-pricing.txt", charPricing)

readPricing("font-pricing.txt", fontPricing)

# Display the pricing table for font size 24pts

showPricing(charPricing, fontPricing, 24)

# Display the pricing table for font size 36pts

showPricing(charPricing, fontPricing, 36)

# Display the pricing table for font size 48pts

showPricing(charPricing, fontPricing, 48)

# Call the main function to run the program

main()

def main():

characters = load\_characters\_pricings()

font\_sizes = load\_font\_sizes\_pricings()

while True:

print("MonkeyPrint Embroidery Services")

print("===============================")

print("1. Display Pricing Table")

print("2. Request for Quote")

print("3. Add/Update Characters’ Pricings")

print("4. Add/Update Font Sizes’ Pricings")

print("0. Exit")

selection = input("Enter selection: ")

if selection == "1":

font\_size = int(input("Enter font size: "))

if font\_size in font\_sizes:

display\_characters\_pricings(characters, font\_sizes[font\_size])

else:

print("Invalid font size!")

elif selection == "2":

quote = input("Enter characters: ")

font\_size = int(input("Enter font size: "))

if all(c in characters for c in quote) and font\_size in font\_sizes:

pricing = compute\_pricing(quote, characters, font\_sizes[font\_size])

if pricing > 8:

discounted\_pricing = min(pricing, pricing - min(characters.values()))

print(f"Pricing: ${pricing:.2f}")

print(f"After 5% discount: ${discounted\_pricing:.2f}")

else:

lowest\_priced\_char = min(characters, key=characters.get)

if lowest\_priced\_char in quote:

print(f"Pricing: ${pricing:.2f}")

print(f"After 1 free character: ${pricing - characters[lowest\_priced\_char]:.2f}")

else:

print(f"Pricing: ${pricing:.2f}")

else:

print("Invalid characters or font size!")

elif selection == "3":

add\_update\_characters\_pricings(characters)

elif selection == "4":

add\_update\_font\_sizes\_pricings(font\_sizes)

elif selection == "0":

break

else:

print("Invalid selection!")

**Question 4**

import random

def print\_board(board):

for row in board:

print(" ".join(row))

def get\_move(player):

move = input(f"Player {player}, enter your move in the format 'row,column': ")

while not valid\_move(move):

move = input(f"Invalid move. Player {player}, enter your move in the format 'row,column': ")

return move

def valid\_move(move):

try:

row, col = move.split(",")

return row.isdigit() and col.isdigit() and int(row) in range(3) and int(col) in range(3)

except:

return False

def game\_over(board):

for row in board:

if len(set(row)) == 1 and row[0] != " ":

return True

for col in range(3):

if len(set([board[row][col] for row in range(3)])) == 1 and board[0][col] != " ":

return True

if len(set([board[row][row] for row in range(3)])) == 1 and board[0][0] != " ":

return True

if len(set([board[row][2-row] for row in range(3)])) == 1 and board[0][2] != " ":

return True

return False

def play\_game():

board = [[" " for \_ in range(3)] for \_ in range(3)]

players = ["X", "O"]

turn = random.choice(players)

print(f"Player {turn} will go first.")

while not game\_over(board):

print\_board(board)

move = get\_move(turn)

row, col = move.split(",")

board[int(row)][int(col)] = turn

turn = players[(players.index(turn)+1)%2]

print\_board(board)

print(f"Player {turn} wins!")

play\_game()

Explanation

This implementation uses a 3x3 board and two players, X and O. The game starts with a random player and each player takes turns placing their symbol on the board. The game ends when one player has three in a row (horizontally, vertically, or diagonally) or when the board is full.