```
#!/usr/bin/env python3
"""
Toddavery Lacrosse Shoe Algorithm
A comprehensive shoe customization program with flowchart-based logic

Author: Toddavery
Date: 2025
Requirements: Python 3.6+

To run in VS Code:
1. Save this file as 'toddavery_lacrosse_shoe_customizer.py'
2. Press F5 or go to Run > Start Debugging
3. Or use terminal: python toddavery_lacrosse_shoe_customizer.py
"""

import random
import os
import sys
from datetime import datetime
from typing import Optional, Tuple

class LacrosseShoeCustomizer:
    """
    Main class for the Toddavery Lacrosse Shoe Customization Algorithm

    Implements all flowchart requirements:
    - 8 numbered steps with proper flow control
    - Boolean logic for validation
    - Conditional statements for pricing
    - Decision points with error handling
    - Loop functionality for program restart
    """

    def __init__(self):
        """Initialize the customizer with default values"""
        self.name: str = ""
        self.color: str = ""
        self.size: float = 0.0
        self.traction: str = ""
        self.support: str = ""
        self.design: str = ""
        self.base_cost: float = 100.0
        self.discount: float = 0.0
        self.final_price: float = 0.0
        self.discount_reason: str = ""

        # Configuration constants
        self.COLORS = ["Red", "Blue", "White", "Black"]
        self.TRACTION_TYPES = ["Turf", "Grass", "All-Terrain"]
        self.SUPPORT_LEVELS = ["Low", "Mid", "High"]
        self.DESIGNS = ["Classic TA", "Modern TA", "Bold TA", "Minimal TA"]
        self.MIN_SIZE = 5.0
        self.MAX_SIZE = 15.0

    def clear_screen(self):
        """Clear the console screen for better user experience"""
        os.system('cls' if os.name == 'nt' else 'clear')

    def display_welcome_message(self) -> None:
        """
        Step 1: Display Welcome Message
        Shows the main program banner and introduction
        """
        self.clear_screen()
        print("🏃" + "=" * 60 + "🏃")
        print(" " * 15 + "TODDAVERY LACROSSE SHOE CUSTOMIZER")
        print("🏃" + "=" * 60 + "🏃")
        print("\n✨ Welcome to the ultimate lacrosse shoe customization experience!")
        print("🎯 Create your perfect custom lacrosse shoes step by step")
        print("📋 Follow our advanced algorithm for the best results\n")
        print("🚀 Let's get started with your custom shoe journey!")
        print("-" * 60)

    def get_user_name(self) -> bool:
        """
        Step 2: Get User's Name with Boolean Validation

        Boolean Logic: name != "" AND name.isalpha() == True

        Returns:
            bool: True if valid name obtained, False otherwise
        """
        print("\n👤 STEP 2: User Information")
        print("=" * 30)

        while True:
            try:
                self.name = input("🔤 Please enter your full name: ").strip()
            # Boolean Check: name != "" AND name.isalpha() == True
            if self.name != "" and self.name.replace(" ", "").isalpha():
                print(f"✅ Perfect! Hello, {self.name}!")
                print(f"🎊 Welcome to your personalized shoe customization experience!")
                input("\nPress Enter to continue...")
                return True
            else:
                print("❌ Invalid name detected!")
                print("⚠️  Name must contain only letters and spaces (no numbers or symbols)")
                print("🔄 Please try again...\n")

        except KeyboardInterrupt:
            print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
            sys.exit(0)
        except Exception as e:
            print(f"❌ Unexpected error: {e}")
            print("🔄 Please try again...\n")

    def choose_color(self) -> None:
        """
        Step 3.1: Choose Color
        Allows user to select from predefined color options
        """
        print("\n🎨 STEP 3.1: Color Selection")
        print("=" * 30)
        print("Choose your preferred shoe color:")

        for i, color in enumerate(self.COLORS, 1):
            print(f"  {i}. {color} {'🔴' if color == 'Red' else '🔵' if color == 'Blue' else '⚪' if color ==
'White' else '⚫'}")

        while True:
            try:
                choice = input(f"\n🎯 Enter your choice (1-{len(self.COLORS)}): ").strip()
                choice_num = int(choice)

                if 1 <= choice_num <= len(self.COLORS):
                    self.color = self.COLORS[choice_num - 1]
                    print(f"✅ Excellent choice! Color selected: {self.color}")
                    break
                else:
                    print(f"❌ Invalid choice! Please enter a number between 1 and
{len(self.COLORS)}")
```

```
        except ValueError:
            print("❌ Invalid input! Please enter a valid number")
        except KeyboardInterrupt:
            print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
            sys.exit(0)

def choose_size(self) -> None:
    """
    Step 3.2: Choose Size with Try/Except and Boolean Validation

    Boolean Logic: size >= 5.0 AND size <= 15.0
    """
    print("\n👟 STEP 3.2: Size Selection")
    print("=" * 30)
    print(f"Enter your shoe size (Range: {self.MIN_SIZE} - {self.MAX_SIZE})")

    while True:
        try:
            size_input = input(f"\n📏 Your shoe size: ").strip()
            self.size = float(size_input)

            # Boolean Check: size >= 5.0 AND size <= 15.0
            if self.size >= self.MIN_SIZE and self.size <= self.MAX_SIZE:
                print(f"✅ Perfect fit! Size selected: {self.size}")
                break
            else:
                print(f"❌ Invalid size range!")
                print(f"⚠️  Size must be between {self.MIN_SIZE} and {self.MAX_SIZE}")
                print("🔄 Please try again...")

        except ValueError:
            print("❌ Invalid input! Please enter a valid number (e.g., 9.5, 10, 11.5)")
        except KeyboardInterrupt:
            print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
            sys.exit(0)

def choose_traction(self) -> None:
    """
    Step 3.3: Choose Traction Type
    Allows user to select traction type for different playing surfaces
    """
    print("\n🏃 STEP 3.3: Traction Selection")
    print("=" * 30)
```

```
        print("Choose your traction type based on playing surface:")

        traction_info = {
            "Turf": "🌱 Artificial turf surfaces",
            "Grass": "🌿 Natural grass fields",
            "All-Terrain": "🌍 Multiple surface types"
        }

        for i, traction in enumerate(self.TRACTION_TYPES, 1):
            print(f"  {i}. {traction} - {traction_info[traction]}")

        while True:
            try:
                choice = input(f"\n🎯 Enter your choice (1-{len(self.TRACTION_TYPES)}): ").strip()
                choice_num = int(choice)

                if 1 <= choice_num <= len(self.TRACTION_TYPES):
                    self.traction = self.TRACTION_TYPES[choice_num - 1]
                    print(f"✅ Great selection! Traction type: {self.traction}")
                    break
                else:
                    print(f"❌ Invalid choice! Please enter a number between 1 and
{len(self.TRACTION_TYPES)}")

            except ValueError:
                print("❌ Invalid input! Please enter a valid number")
            except KeyboardInterrupt:
                print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
                sys.exit(0)

    def choose_support(self) -> None:
        """
        Step 3.4: Choose Support Level
        Allows user to select ankle support level
        """
        print("\n🦵 STEP 3.4: Support Level Selection")
        print("=" * 30)
        print("Choose your preferred ankle support level:")

        support_info = {
            "Low": "💨 Lightweight, maximum mobility",
            "Mid": "⚖️ Balanced support and mobility",
            "High": "🛡️ Maximum support and stability"
        }
        for i, support in enumerate(self.SUPPORT_LEVELS, 1):
            print(f"  {i}. {support} - {support_info[support]}")

        while True:
            try:
                choice = input(f"\n🎯 Enter your choice (1-{len(self.SUPPORT_LEVELS)}): ").strip()
                choice_num = int(choice)

                if 1 <= choice_num <= len(self.SUPPORT_LEVELS):
                    self.support = self.SUPPORT_LEVELS[choice_num - 1]
                    print(f"✅ Perfect choice! Support level: {self.support}")
                    break
                else:
                    print(f"❌ Invalid choice! Please enter a number between 1 and
{len(self.SUPPORT_LEVELS)}")

            except ValueError:
                print("❌ Invalid input! Please enter a valid number")
            except KeyboardInterrupt:
                print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
                sys.exit(0)

    def calculate_cost(self) -> None:
        """
        Step 4.1: Calculate Cost Based on Support Level

        Conditional Logic:
        IF support == "High" THEN +$20
        ELIF support == "Mid" THEN +$10
        ELSE +$0
        """
        print("\n🧮 STEP 4.1: Cost Calculation")
        print("=" * 30)

        # Base cost calculation
        base_price = 100.0

        # Conditional pricing based on support level
        if self.support == "High":
            support_cost = 20.0
        elif self.support == "Mid":
            support_cost = 10.0
        else:  # Low support
            support_cost = 0.0

        self.base_cost = base_price + support_cost

        print(f"💰 Cost Breakdown:")
        print(f"   Base shoe price: ${base_price:.2f}")
        print(f"   {self.support} support add-on: +${support_cost:.2f}")
        print(f"   Subtotal: ${self.base_cost:.2f}")

    def calculate_discount(self) -> None:
        """
        Step 4.2: Calculate Random Discount with Reason
        Applies a random discount with explanation
        """
        print("\n🎁 STEP 4.2: Discount Calculation")
        print("=" * 30)

        # Random discount options with reasons
        discount_options = [
            (5, "New customer welcome discount! 🎉"),
            (10, "Lucky day special offer! 🍀"),
            (15, "Student athlete discount! 🎓"),
            (20, "Flash sale - you're in luck! ⚡"),
            (8, "Loyalty program bonus! 💎"),
            (12, "Seasonal promotion active! 🌟"),
            (0, "No discount today, but you're getting premium quality! 💪")
        ]

        discount_data = random.choice(discount_options)
        self.discount = discount_data[0]
        self.discount_reason = discount_data[1]

        print(f"🎊 Discount Applied: {self.discount}%")
        print(f"📝 Reason: {self.discount_reason}")

    def calculate_final_price(self) -> None:
        """
        Step 4.3: Calculate Final Price
        Applies discount to base cost to get final price
        """
        print("\n💸 STEP 4.3: Final Price Calculation")
        print("=" * 30)

        discount_amount = self.base_cost * (self.discount / 100)
        self.final_price = self.base_cost - discount_amount

        print(f"📊 Final Price Breakdown:")
        print(f"   Subtotal: ${self.base_cost:.2f}")
        print(f"   Discount ({self.discount}%): -${discount_amount:.2f}")
        print(f"   🎯 FINAL PRICE: ${self.final_price:.2f}")

    def choose_design(self) -> None:
        """
        Step 5: Optional TA Initial Design Selection

        Boolean Logic: user_wants_design == True
        """
        print("\n✨ STEP 5: Optional TA Design")
        print("=" * 30)
        print("Would you like to add a TA (Team/Athletic) initial design?")
        print("This adds a personalized touch to your shoes!")

        while True:
            try:
                wants_design = input("\n🎨 Add TA design? (yes/no): ").strip().lower()

                # Boolean Check: user_wants_design == True
                if wants_design in ['yes', 'y', 'true', '1']:
                    print("\n🎨 Available TA Design Options:")

                    design_info = {
                        "Classic TA": "🏛️ Traditional style lettering",
                        "Modern TA": "🚀 Contemporary design",
                        "Bold TA": "💪 Strong, prominent style",
                        "Minimal TA": "✨ Clean, subtle approach"
                    }

                    for i, design in enumerate(self.DESIGNS, 1):
                        print(f"  {i}. {design} - {design_info[design]}")

                    while True:
                        try:
                            choice = input(f"\n🎯 Choose design (1-{len(self.DESIGNS)}): ").strip()
                            choice_num = int(choice)

                            if 1 <= choice_num <= len(self.DESIGNS):
                                self.design = self.DESIGNS[choice_num - 1]
                                print(f"✅ Design selected: {self.design}")
                                break
                            else:
                                print(f"❌ Invalid choice! Please enter 1-{len(self.DESIGNS)}")

                        except ValueError:
                            print("❌ Invalid input! Please enter a valid number")
                    break

                elif wants_design in ['no', 'n', 'false', '0']:
                    self.design = "No design selected"
                    print("✅ No design selected - clean, classic look!")
                    break
                else:
                    print("❌ Please enter 'yes' or 'no'")

            except KeyboardInterrupt:
                print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
                sys.exit(0)

    def show_summary(self) -> None:
        """
        Step 6: Show Complete Customization Summary
        Displays all selected options and final details
        """
        print("\n" + "🏆" + "=" * 58 + "🏆")
        print(" " * 15 + "YOUR CUSTOM LACROSSE SHOE SUMMARY")
        print("🏆" + "=" * 58 + "🏆")

        print(f"\n👤 Customer Information:")
        print(f"   Name: {self.name}")

        print(f"\n🎨 Shoe Specifications:")
        print(f"   Color: {self.color}")
        print(f"   Size: {self.size}")
        print(f"   Traction Type: {self.traction}")
        print(f"   Support Level: {self.support}")
        print(f"   Design: {self.design}")

        print(f"\n💰 Pricing Details:")
        print(f"   Base Cost: ${self.base_cost:.2f}")
        print(f"   Discount: {self.discount}% - {self.discount_reason}")
        print(f"   🎯 Final Price: ${self.final_price:.2f}")

        print(f"\n📅 Order Date: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
        print("🏆" + "=" * 58 + "🏆")

        input("\n📋 Press Enter to continue...")

    def save_to_file(self) -> None:
        """
        Step 7: Optional Save Receipt to File

        Boolean Logic: save_receipt == True
        """
        print("\n💾 STEP 7: Save Receipt")
        print("=" * 30)
        print("Would you like to save your receipt to a file?")
        print("This creates a permanent record of your custom shoe order.")

        while True:
            try:
                save_choice = input("\n💾 Save receipt? (yes/no): ").strip().lower()

                # Boolean Check: save_receipt == True
                if save_choice in ['yes', 'y', 'true', '1']:
                    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                    safe_name = "".join(c for c in self.name if c.isalnum() or c in (' ', '-', '_')).rstrip()
                    filename = f"lacrosse_shoe_receipt_{safe_name}_{timestamp}.txt"

                    try:
                        with open(filename, 'w', encoding='utf-8') as file:
                            file.write("=" * 50 + "\n")
                            file.write("TODDAVERY LACROSSE SHOE RECEIPT\n")
                            file.write("=" * 50 + "\n\n")

                            file.write(f"Date: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")
                            file.write(f"Customer: {self.name}\n\n")

                            file.write("SHOE SPECIFICATIONS:\n")
                            file.write(f"Color: {self.color}\n")
                            file.write(f"Size: {self.size}\n")
                            file.write(f"Traction Type: {self.traction}\n")
                            file.write(f"Support Level: {self.support}\n")
                            file.write(f"Design: {self.design}\n\n")

                            file.write("PRICING DETAILS:\n")
                            file.write(f"Base Cost: ${self.base_cost:.2f}\n")
                            file.write(f"Discount: {self.discount}% - {self.discount_reason}\n")
                    file.write(f"Final Price: ${self.final_price:.2f}\n\n")

                    file.write("=" * 50 + "\n")
                    file.write("Thank you for choosing Toddavery Lacrosse Shoes!\n")
                    file.write("Your custom shoes will be crafted with care.\n")
                    file.write("=" * 50 + "\n")

                print(f"✅ Receipt saved successfully!")
                print(f"📄 File location: {os.path.abspath(filename)}")
                break

            except Exception as e:
                print(f"❌ Error saving file: {e}")
                print("🔄 Continuing without saving...")
                break

        elif save_choice in ['no', 'n', 'false', '0']:
            print("✅ Receipt not saved - continuing...")
            break
        else:
            print("❌ Please enter 'yes' or 'no'")

    except KeyboardInterrupt:
        print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
        sys.exit(0)

    def restart_program(self) -> bool:
        """
        Step 8: Restart Program Check

        Boolean Logic: restart_program == True

        Returns:
            bool: True if user wants to restart, False otherwise
        """
        print("\n🔁 STEP 8: Program Restart")
        print("=" * 30)
        print("Would you like to customize another pair of shoes?")
        print("You can create multiple customizations in one session!")

        while True:
            try:
                restart_choice = input("\n🔄 Customize another pair? (yes/no): ").strip().lower()
            # Boolean Check: restart_program == True
            if restart_choice in ['yes', 'y', 'true', '1']:
                print("\n🔄 Excellent! Starting new customization...")
                print("🎯 Returning to Step 3 - Shoe Customization")
                input("\nPress Enter to continue...")
                return True

            elif restart_choice in ['no', 'n', 'false', '0']:
                print("\n🏁 Thank you for using Toddavery Lacrosse Shoe Customizer!")
                print("🎉 Your custom shoes will be crafted with precision and care.")
                print("👟 Enjoy your new lacrosse shoes and dominate the field!")
                print("\n💪 Have a great day and play hard!")
                return False
            else:
                print("❌ Please enter 'yes' or 'no'")

        except KeyboardInterrupt:
            print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
            sys.exit(0)

def main():
    """
    Main program execution function
    Orchestrates the complete shoe customization workflow
    """
    try:
        # Initialize the customizer
        customizer = LacrosseShoeCustomizer()

        # Step 1: Display Welcome Message
        customizer.display_welcome_message()

        # Step 2: Get User's Name (with validation)
        customizer.get_user_name()

        # Main program loop with restart capability
        while True:
            # Step 3: Complete Shoe Customization Process
            customizer.choose_color()      # Step 3.1
            customizer.choose_size()       # Step 3.2
            customizer.choose_traction()   # Step 3.3
            customizer.choose_support()    # Step 3.4

            # Step 4: Complete Cost Calculation Process
            customizer.calculate_cost()          # Step 4.1
            customizer.calculate_discount()      # Step 4.2
            customizer.calculate_final_price()   # Step 4.3

            # Step 5: Optional Design Selection
            customizer.choose_design()

            # Step 6: Display Complete Summary
            customizer.show_summary()

            # Step 7: Optional File Saving
            customizer.save_to_file()

            # Step 8: Check for Program Restart
            if not customizer.restart_program():
                break

    except KeyboardInterrupt:
        print("\n\n👋 Thanks for using Toddavery Lacrosse Shoe Customizer!")
        print("💫 Come back anytime to create your perfect shoes!")
    except Exception as e:
        print(f"\n❌ An unexpected error occurred: {e}")
        print("🔄 Please restart the program and try again.")
    finally:
        print("\n🎯 Program terminated successfully.")

# Program entry point
if __name__ == "__main__":
    main()
```