Informe de Simulación de Servicio al Cliente

Introducción

Este código simula un sistema de servicio al cliente utilizando la biblioteca Pygame. El sistema consiste en múltiples cajas de atención al cliente, donde los clientes llegan, esperan si las cajas están ocupadas y son atendidos cuando una caja está libre. La simulación calcula el tiempo de espera y atención de los clientes, y presenta estadísticas al final del proceso.

Descripción del Código

1. Importación de Bibliotecas:

```
import pygame
import random
import numpy as np
```

2. Inicialización de Pygame:

```
pygame.init()
```

3. Definición de Colores:

```
BLANCO = (255, 255, 255)
NEGRO = (0, 0, 0)
ROJO = (255, 0, 0)
VERDE = (0, 255, 0)
AZUL = (0, 0, 255)
```

4. Parámetros del Sistema:

Se definen varios parámetros, incluyendo la cantidad de cajas, costos, tiempo de operación, y

parámetros de la distribución normal para la llegada de clientes.

5. Configuración Inicial de Pygame:

```
ANCHO, ALTO = 800, 600
ventana = pygame.display.set_mode((ANCHO, ALTO))
pygame.display.set_caption('Simulación de Servicio al Cliente')
reloj = pygame.time.Clock()
fuente_pequena = pygame.font.Font(None, 36)
```

6. Variables de Simulación:

Se inicializan listas y contadores para gestionar los clientes, las cajas y las estadísticas.

7. Funciones Auxiliares:

- generar_tiempo_atencion: Genera el tiempo de atención basado en una distribución normal.

- generar_tiempos_llegada: Genera los tiempos de llegada de los clientes siguiendo una distribución normal truncada.

8. Generación de Tiempos de Llegada de Clientes:

tiempos_llegada_clientes = generar_tiempos_llegada(CLIENTES_ESPERADOS, MEDIA_LLEGADA, DESV_EST_LLEGADA, LIMITE_INFERIOR, LIMITE_SUPERIOR)

9. Definición de la Clase Cliente:

```
class Cliente:
    def __init__(self, tiempo_llegada):
        self.tiempo_llegada = tiempo_llegada
```

```
        self.tiempo_inicio_atencion = None
        self.tiempo_fin_atencion = None

    def iniciar_atencion(self, tiempo_inicio):
        self.tiempo_inicio_atencion = tiempo_inicio
        self.tiempo_fin_atencion = tiempo_inicio + generar_tiempo_atencion()
        tiempos_atencion.append(self.tiempo_fin_atencion - self.tiempo_inicio_atencion)

    def siendo_atendido(self, tiempo_actual):
            return self.tiempo_inicio_atencion is not None and self.tiempo_inicio_atencion <= tiempo_actual < self.tiempo_fin_atencion

    def atendido(self, tiempo_actual):
        return self.tiempo_fin_atencion is not None and self.tiempo_fin_atencion <= tiempo_actual
```

10. Bucle Principal de la Simulación:

En el bucle principal, se gestionan las llegadas de clientes, la asignación de clientes a las cajas, la gestión de la cola de espera y la actualización de la pantalla.

11. Cálculo de Estadísticas:

```
if tiempos_atencion:
    tiempo_min_atencion = min(tiempos_atencion)
    tiempo_max_atencion = max(tiempos_atencion)
else:
    tiempo_min_atencion = tiempo_max_atencion = 0
```

Informe de Simulación de Servicio al Cliente

```
    if tiempos_espera:
        tiempo_min_espera = min(tiempos_espera)
        tiempo_max_espera = max(tiempos_espera)
    else:
        tiempo_min_espera = tiempo_max_espera = 0
```

Resultados

Total de clientes: {total_clientes}

Clientes atendidos: {clientes_atendidos}

Clientes no atendidos: {clientes_no_atendidos}

Tiempo mínimo de atención en caja: {tiempo_min_atencion / 60:.2f} minutos

Tiempo máximo de atención en caja: {tiempo_max_atencion / 60:.2f} minutos

Tiempo mínimo de espera en cola: {tiempo_min_espera / 60:.2f} minutos

Tiempo máximo de espera en cola: {tiempo_max_espera / 60:.2f} minutos

Costo de operación: {CANT_CAJAS * COSTO_CAJA + clientes_no_atendidos * COSTO_PERDIDA_CLIENTE}