Comparativo de Implementación del Modelo de Simulación "Sin Fila"

1. Descripción del modelo

El modelo "Sin Fila" representa un sistema donde los pasajeros llegan de forma aleatoria (o predeterminada) y pueden ser atendidos o rechazados según la disponibilidad del aforador (servidor). Cuando el aforador está ocupado, los pasajeros no esperan: simplemente se van sin revisión. Ambos códigos (Mathematica y Python) implementan el mismo proceso lógico con control de tiempo, llegadas/salidas y registro de eventos.

2. Implementación en Wolfram Mathematica

El siguiente fragmento muestra la estructura del modelo original implementado en Wolfram Mathematica:

```
In[16]:= TI = {4, 3, 2, 2, 5, 1, 3}; (*Tiempos entre llegadas en minutos*)
         TA = {1, 3, 2, 5, 1, 2}; (*Tiempos de atención en minutos*)
         TM = 0; (*Tiempo del modelo*)
         TMax = 16; (*Tiempo máximo de simulación*)
         llegada = 4;
         salida = ∞;
         pasajero = 1; (*Contador de pasajeros*)

         desocupado = True; (*Estado del aforador*)
                      verdadero
         ts = 1; (* contador para el tiempo de atención*)
         pasajeroSiendoAtendido = 1;
         evento = {{"Evento", "#pasajero", "TM", "Revisión"}}; (*Lista para guardar eventos*)

         (*Iniciamos la simulación*)|


In[27]:= While[TM ≤ TMax, (*Proceso mientras haya tiempo*)
          mientras
           If[llegada < salida,
           si
              (*Proceso una llegada*)
              TM = llegada;
              Print["Llega el pasajero ", pasajero, " al minuto ", TM];
              escribe
              If[desocupado,
              si
                (*Revisamos al pasajero*)
                desocupado = False;
                             falso
                salida = TM + TA[[ts]];
                ts++;
                pasajeroSiendoAtendido = pasajero;
                Print["Revisamos al pasajero ", pasajeroSiendoAtendido ];
                escribe
```

```
    (*Ingresando un evento*)
    eventoAux = {"Entrada", pasajeroSiendoAtendido, TM, "Rev"};
    evento = Join[evento, {eventoAux}];
             |junta
    ,
    (*El pasajero pasa sin ser revisado*)
    Print["El pasajero ", pasajero, " pasa sin ser revisado"];
    |escribe

    (*Ingresando un evento*)
    eventoAux = {"Entró y salió", pasajero, TM, "No Rev"};
    evento = Join[evento, {eventoAux}];
             |junta
  ]; (*Fin If desocupado*)
  (*Debemos planificar la próxima llegada*)
  pasajero++;
  llegada = TM + TI[[pasajero]];
  ,
  (*Proceso una salida*)
  TM = salida;

  Print["Sale el pasajero ", pasajeroSiendoAtendido, " en el Min ", TM];
  |escribe

  (*Ingresando un evento*)
  eventoAux = {"Salida", pasajeroSiendoAtendido, TM, "Rev"};
  evento = Join[evento, {eventoAux}];
           |junta

  desocupado = True;
               |verdadero
  salida = ∞;
 ](*Fin de If*)×
 Input["Dé enter para continuar"](*Esta instrucción la usamos para poder ir revisando el código sin que se encicle. Si se encicla salimos del Kernel*)
 |entra
]
(*Fin de While*)
```

3. Implementación en Python

La siguiente traducción en Python conserva la lógica estructural, usando variables aleatorias y listas dinámicas:

```python
import random

# ----------------------------
# CONFIGURACIÓN INICIAL
# ----------------------------
TMax = 1000
TM = 0
llegada = random.randint(1, 10)  # Tiempo de la primera llegada
salida = float('inf')
pasajero = 1
desocupado = True
pasajeroSiendoAtendido = 0

eventos = [["Evento", "#Pasajero", "TM", "Revisión"]]

# ----------------------------
# SIMULACIÓN SIN FILA
# ----------------------------
print("\n--- SIMULACIÓN SIN FILA ---")

while TM ⩽ TMax:
    if llegada < salida:
        # Procesar llegada
        TM = llegada

        if desocupado:
            # Inicia revisión
            desocupado = False
            TA = random.randint(1, 10)
            salida = TM + TA
            pasajeroSiendoAtendido = pasajero
            eventos.append(["Entrada", pasajeroSiendoAtendido, TM, "Re
v"])    else:
            # Pasa sin revisión
            eventos.append(["Entró y salió", pasajero, TM, "No Rev"])

        # Planificar próxima llegada
        TI = random.randint(1, 10)
        llegada = TM + TI
        pasajero += 1

    else:
        # Procesar salida
        TM = salida
        eventos.append(["Salida", pasajeroSiendoAtendido, TM, "Rev"])
        desocupado = True
        salida = float('inf')

```

4. Correspondencias entre Mathematica y Python

Concepto	Mathematica	Python
Tiempo máximo de simulación	While[TM <= TMax, ...]	while TM <= TMax:
Tiempo entre llegadas y atención	Listas TI, TA predefinidas	random.randint()
Registro de eventos	Join[evento, {eventoAux}]	eventos.append([...])
Condicional principal	If[llegada < salida, ...]	if llegada < salida:
Revisión y salida	If[desocupado, ..., ...]	if desocupado: ... else:
Salida del bucle	desocupado = True; salida = ∞	desocupado = True; salida = float('inf')
Visualización	MatrixForm	pandas.DataFrame o print tabular
Generación de resultados	Print en tiempo real	Cálculo posterior + print()