República Bolivariana de Venezuela

Ministerio de Poder Popular para la Educación

Universidad Rafael Urdaneta

Sistemas Operativos

**Informe de**

**Fifo, Lifo y Round Robin**

Estudiante:

David Mundo 30249427

Maracaibo 29 de marzo del 2025

1. Introducción

Este informe analiza un programa de simulación de planificación de actividades que implementa tres algoritmos clásicos: FIFO (First-In, First-Out), LIFO (Last-In, First-Out) y Round Robin (RR). El objetivo es evaluar su eficiencia en términos de:

Tiempo de respuesta (Tiempo Time, T).

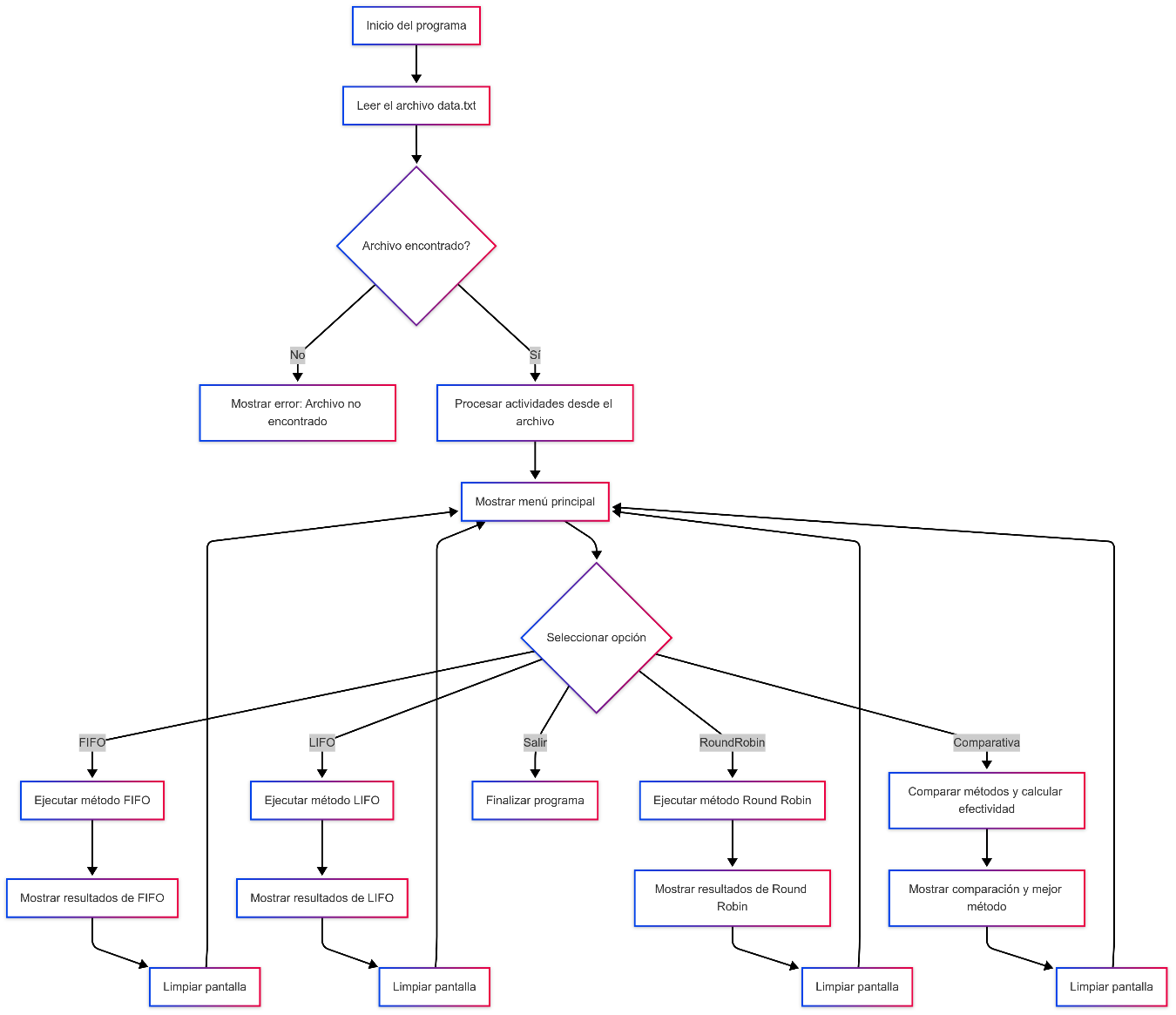
Tiempo de espera (E).

Índice de respuesta (I).

Se incluyen diagramas de flujo y un análisis detallado del código Python para entender su funcionamiento.

2. Diagramas de Flujo

2.1. Diagrama General del Programa



Proceso:

Lectura de datos: Carga actividades desde data.txt (formato: Nombre (ti, t)).

Menú interactivo: Permite seleccionar entre FIFO, LIFO, RR o comparativa.

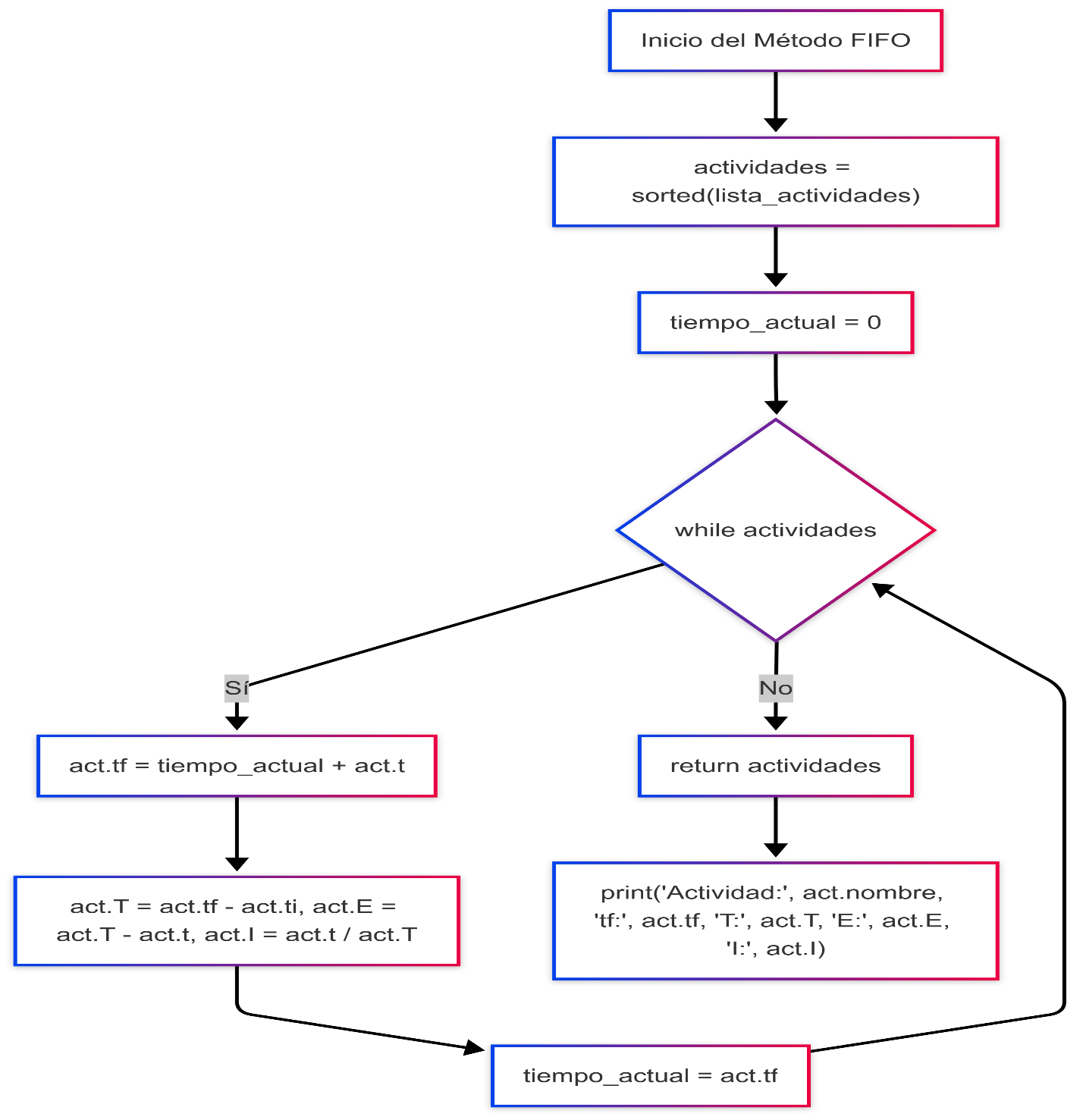
Resultados: Muestra métricas y compara eficiencia.

Puntos clave:

Validación de archivo.

Modularidad (cada método es independiente).

2.2. Diagrama de FIFO



Lógica:

Ordena actividades por tiempo de llegada (ti).

Procesa en orden secuencial, actualizando tf = tiempo\_actual + t.

Calcula:

T = tf - ti (Turnaround Time).

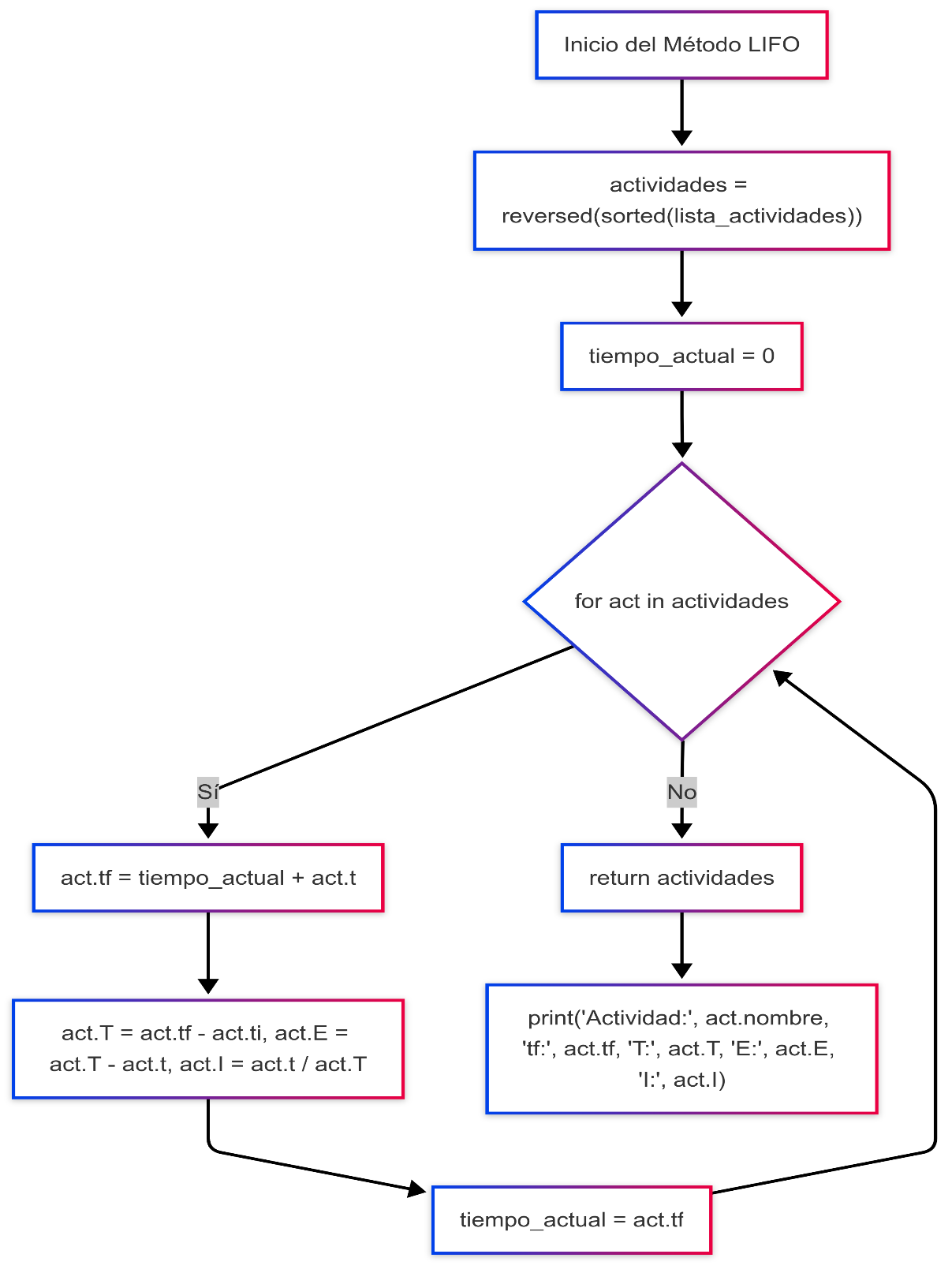
E = T - t (Tiempo de espera).

I = t / T (Índice de respuesta).

Ventaja: Simple y justo para tareas cortas.

Desventaja: Puede causar esperas largas (efecto "convoy").

2.3. **Diagrama de LIFO**



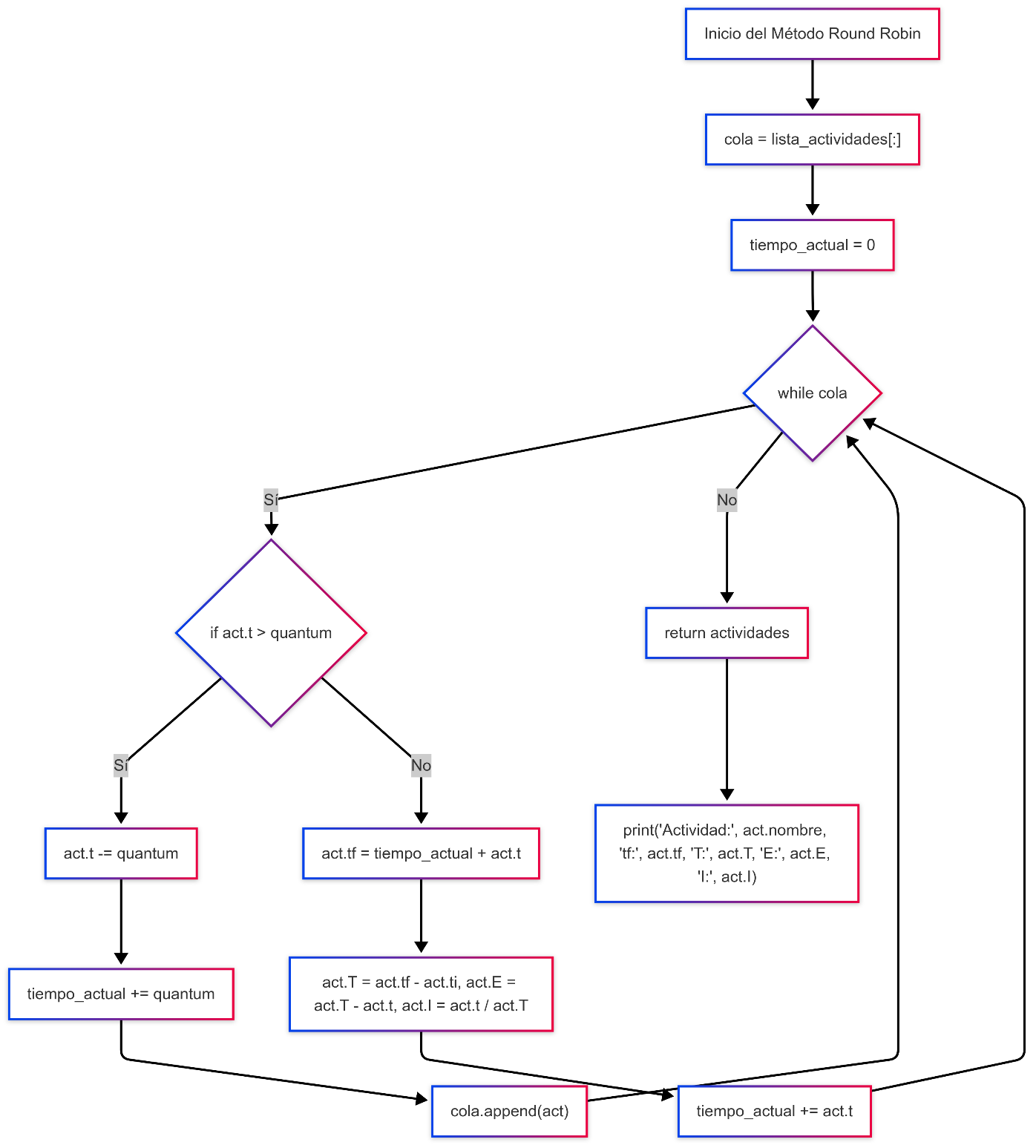
Lógica:

Similar a FIFO, pero procesa en orden inverso (últimas actividades primero).

Caso de uso: Útil si las tareas más recientes son prioritarias.

Riesgo: Inanición (tareas antiguas pueden no ejecutarse pronto).

2.4. **Diagrama de Round Robin**



Lógica:

Asigna un quantum (ej. 4 unidades) por actividad.

Si una actividad no termina, se reencola y se resta el tiempo usado.

Repite hasta que todas finalicen.

Ventaja: Equitativo para tareas largas y cortas.

Desventaja: Overhead por cambios de contexto.

3. **Análisis del Código Python**

3.1. Estructura Principal

Clase Actividad:

python

Copy

class Actividad:

def \_\_init\_\_(self, nombre, ti, t):

self.nombre = nombre # Identificador

self.ti = ti # Tiempo inicial

self.t = t # Duración

self.tf = 0 # Tiempo final

self.T = 0 # Turnaround Time

self.E = 0 # Tiempo de espera

self.I = 0 # Índice de respuesta

3.2. Métodos de Planificación

FIFO:

python

Copy

def FIFO(actividades):

tiempo\_actual = 0

for act in actividades:

tiempo\_actual = max(tiempo\_actual, act.ti) # Evita solapamientos

act.tf = tiempo\_actual + act.t

act.T = act.tf - act.ti

act.E = act.T - act.t

act.I = act.t / act.T

tiempo\_actual = act.tf

return actividades

Round Robin:

python

Copy

def RR(actividades, quantum):

cola = actividades.copy()

tiempo\_actual = 0

while cola:

act = cola.pop(0)

if tiempo\_actual < act.ti:

tiempo\_actual = act.ti # Espera hasta que la actividad esté lista

if act.t > quantum:

act.t -= quantum

tiempo\_actual += quantum

cola.append(act) # Reencola si no ha terminado

else:

tiempo\_actual += act.t

act.tf = tiempo\_actual

act.T = act.tf - act.ti

act.E = act.T - act.t

act.I = act.t / act.T

return actividades

4. **Comparativa de Métodos**

4.1. Métricas Clave

Métrica FIFO LIFO Round Robin

Tiempo (T) Bajo para tareas cortas Variable (depende del orden) Alto (por el quantum)

Espera (E) Bajo si no hay tareas largas Riesgo de inanición Equilibrado

Índice (I) Alto (≈1 si E ≈ 0) Variable Menor (por overhead)

4.2. Recomendaciones

Usar FIFO si las tareas son cortas y homogéneas.

Evitar LIFO en sistemas críticos (riesgo de inanición).

Round Robin es ideal para entornos interactivos (ej. sistemas operativos).

5. **Conclusiones**

El programa simula eficientemente los algoritmos, con una interfaz clara y métricas bien definidas.

La comparativa integrada permite elegir el mejor método según el caso de uso.