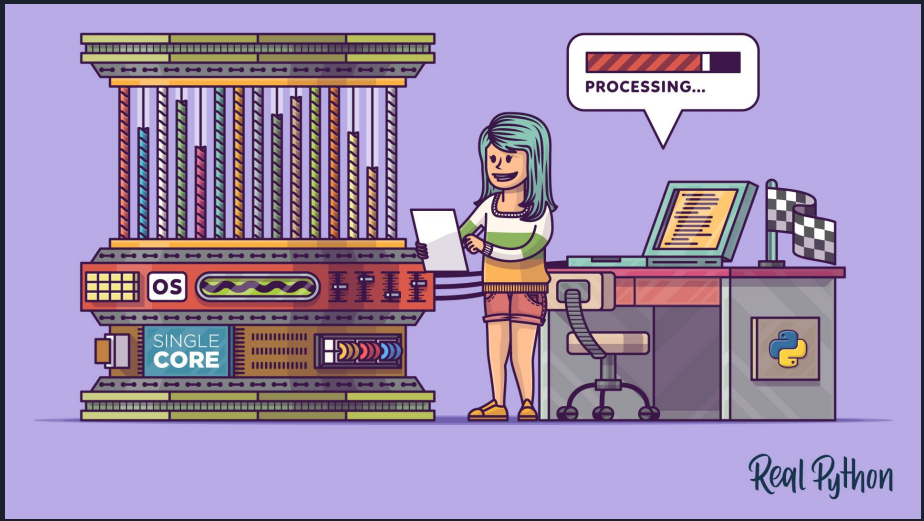
# Programación concurrente con hilos (threading) y Logging en Python

Proyectos de Computación Aplicados a Ingeniería Electrónica (0980) - FIUSAC

MSc. Ing. Iván René Morales - Junio 2020

# ¿Qué son hilos?

- Flujos individuales de ejecución de código en "paralelo"
- Se utiliza un *scheduler* para asignar tiempo de cómputo a cada hilo
- En realidad, solo un hilo se ejecuta a la vez, pero en tiempos cortos
  - Paralelismo simulado
- Se ejecuta en un solo núcleo del procesador
  - Todo corre sobre un mismo proceso, que se subdivide en hilos

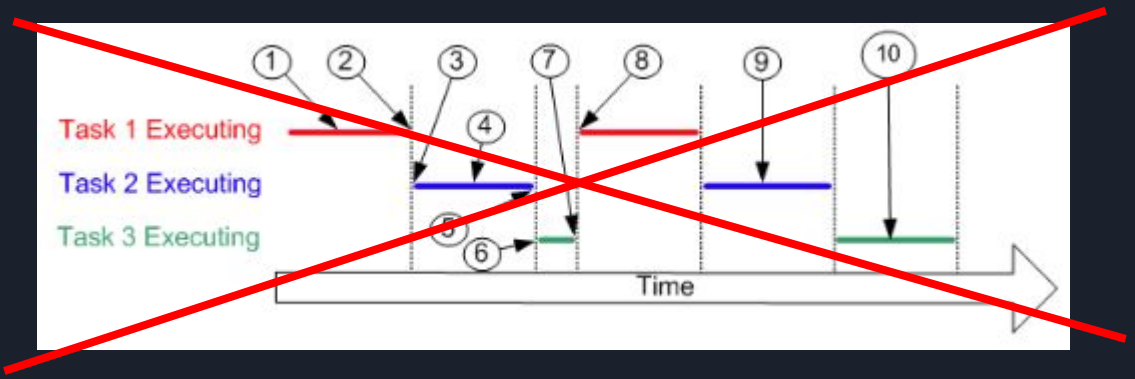


# Ventajas de hilos en Python

- Procesos bloqueantes quedan en el fondo
  - Ejemplos de aplicaciones
    - Esperando respuesta de un servidor/cliente en un socket
    - Transferencia de información o archivos en el fondo
    - Entradas de usuario con *input*
- Python se encarga del control de tiempo de uso de CPU
- Múltiples tareas "simples" (sin uso intensivo de CPU)
- Control individual de cada hilo
- Mecanismos de sincronización
- Mecanismos de intercambio de información entre hilos
- Demonios (daemons) ¿*Systick*?

# Desventajas de hilos en Python

- Debe cuidarse el acceso a recursos compartidos
  - Mutexes, semáforos
- No es posible controlar prioridades entre hilos
- La ventana de tiempo de ejecución de cada hilo no es determinística
- El *scheduler* de Python depende también del *scheduler* del Kernel del Sistema Operativo
  - Esto agrega mayor incerteza en las ventanas temporales de ejecución de hilos
- Deadlocks :S
- Hilos en ejecución pueden prevenir que se ejecute la salida del script principal



# Logging

- Herramienta potente para depuración y control de flujo
- Librería *logging* incluida por defecto en Python
  - Librerías y módulos normalmente utilizan *logging* por defecto
  - Fácil interacción con librerías de terceros
- Distintos niveles de mensajes
  - DEBUG
  - INFO
  - WARNING
  - ERROR
  - CRITICAL
- Configuración de visibilidad (en función de nivel) de mensajes
- Configuración de formato
- Reemplaza **por completo** el uso de *print* durante el desarrollo para depuración

# ¿Procesamiento paralelo multi-núcleo?

- **Multiprocessing**
- Crea procesos concurrentes independientes
- Cada proceso es manejado por el Sistema Operativo para maximizar su capacidad de ejecución
  - Se convierte en ejecución multinúcleo
- Interfaz similar a Threading
- Usado para procesamiento intensivo de datos
- Python NO es un lenguaje para procesamiento intensivo de datos
  - Es mucho más eficiente OpenMPI con C++

# Referencias


- Python 3 - Thread-based parallelism
  - https://docs.python.org/3/library/threading.html
- Python 3 - Logging
  - https://docs.python.org/3.6/library/logging.html
- Threading & Logging with Python 3.6
  - https://realpython.com/intro-to-python-threading/
- Logging with Python
  - https://realpython.com/python-logging/
- Gestión de operaciones concurrentes en un proceso con Python 3
  - https://rico-schmidt.name/pymotw-3/threading/index.html
- Manejo adecuado de operaciones bloqueantes con hilos
  - https://www.geeksforgeeks.org/start-and-stop-a-thread-in-python/
- Procesamiento multinúcleo
  - https://www.geeksforgeeks.org/multiprocessing-python-set-1/