

1IEE06 - Laboratorio 11

- Materiales permitidos: Wiki del curso, apuntes de clase, consultar foros, tutoriales o documentación de python online.
- Está prohibido el uso de cualquier modelo de lenguaje como ChatGPT o alguno similar. A cualquier alumno que se le detecte que ha consultado un modelo de lenguaje se le pondrá nota 0(cero) en el laboratorio.
- Usted debe subir a Paideia 1 solo archivo comprimido(.zip o .rar) con el nombre L11_CODIGOPUCP.zip o L11_CODIGOPUCP.rar. Este archivo comprimido debe tener archivos de python(extensión .py) para cada pregunta. No se aceptarán soluciones en Jupyter notebook.
- Está prohibido usar cualquier librería como ayuda para leer o escribir en los archivos .csv
- El horario máximo permitido para subir el archivo es a las 10:00:00 pm. Pasada esa hora, habrá una penalidad de 2 puntos por cada minuto extra que se demore en entregar su archivo.

Pregunta 1 (5 puntos)

Se desea instalar unos dispositivos que miden variables meteorológicas en distintas partes del país. Para que estos dispositivos envíen sus datos sensados a una central en Lima, se están considerando 3 posibles tecnologías: SMS, 3G, o satelital. Cada una de estas tecnologías tiene un tiempo de latencia, el cual se puede modelar usando las siguientes fórmulas:

Latencia de SMS = Tiempo_de_procesamiento_de_envío +
Tiempo_de_procesamiento_de_recepción

Donde:

Tiempo de procesamiento de envío: Valor aleatorio entre [10ms, 100ms]

Tiempo de procesamiento de recepción: Valor aleatorio entre [10ms, 100ms]

Latencia de 3G = 2 x (Tiempo_de_ida_y_vuelta + Tiempo_de_procesamiento)

Donde:

Tiempo de ida y vuelta: Valor aleatorio entre [100ms, 300ms]

Tiempo de procesamiento de recepción: Valor aleatorio entre [10ms, 100ms]

Latencia de satelital = 2 x Retardo_de_propagación + Tiempo_de_procesamiento

Donde:

Retardo de propagación: Valor aleatorio entre [500ms, 700ms]

Tiempo de procesamiento de recepción: Valor aleatorio entre [10ms, 100ms]

a) (3 puntos) Su trabajo consiste en simular los tiempos de simulación de las 3 tecnologías usando corrutinas con `asyncio`. Para ello cree un archivo llamado `pregunta1.py` en el que escribirá un programa que ejecute 3 corrutinas de manera asíncrona. Cada una de las 3 corrutinas va a simular la latencia de las 3 tecnologías.

- Para generar un número aleatorio que simule la latencia, puede hacer uso de la función `random.randint()`.
- Para simular la latencia en sí, deberá hacer uso de `asyncio.sleep()`. Esta función recibirá como parámetro de entrada el valor aleatorio que ha generado haciendo uso de `random.randint()`. De esta manera, cada rutina simulará la latencia de la tecnología.
Una vez que el `asyncio.sleep()` termina de ejecutarse, la corrutina debe imprimir el siguiente mensaje:

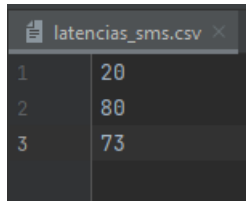
“La corrutina de la tecnología X tuvo una latencia de Y ms.”

Donde X puede ser “SMS”, “3G”, “Satelital” y Y es el resultado de la fórmula de latencia de cada tecnología.

b) (2 puntos) Modifique su programa de la parte a) para que cada una de las corrutinas guarde en un archivo `.csv` los tiempos de latencia usando la librería `aiofiles`. Como son 3 tecnologías, se espera generar 3 archivos, los cuales deben tener los siguientes nombres:

- `latencias_sms.csv`
- `latencias_3g.csv`
- `latencias_satelital.csv`

Cada vez que usted ejecute su programa, se debe guardar en los archivos las latencias sin sobrescribir los valores antiguos. Por ejemplo, después de ejecutarlo 3 veces, se espera encontrar 3 valores en el archivo `latencias_sms.csv`:



1	20
2	80
3	73

Pregunta 2 (4 puntos)

En clase se escribieron 2 programas: *generador_notas_sync.py*, y *generador_notas_async.py*

Al ejecutar ambos programas, se observó en clase que el tiempo de ejecución del programa síncrono es mucho menor que su versión asíncrona.

Modifique ambos archivos (póngales de nombre *p2_generador_notas_sync.py* y *p2_generador_notas_async.py*) para probar si es possible hacer que la versión asíncrona sea más rápida que la síncrona. Imprima los tiempos de ejecución de sus programas modificados, y comente si su programa cumplió con el objetivo deseado.

Indicaciones a tomar cuenta para la solución de esta pregunta:

- I. En la parte superior de su archivo, escriba en formato de comentario qué modificaciones le va a hacer al archivo que el professor hizo en clase, y explique cómo espera que esas modificaciones logren el objetivo deseado. Si no escribe este texto en la parte superior del archivo, se le pondrá puntaje cero (0).
- II. Solo se le asignará puntaje en la medida que usted escriba código que haga lo que ha puesto como comentario en la parte superior del archivo. No se asignará puntaje solo por haber copiado y pegado el código visto en clase, o por hacer cambios que no siguen el plan que usted ha justificado que va a hacer.

Pregunta 3 (8 puntos)

Se le pide realizar la simulación del campeonato mundial de ajedrez que se realiza en 2 etapas: fase de rondas y fase final

Los 6 participantes en la fase de rondas son los siguientes:

- Magnus Carlsen
- Vladimir Kramnik
- Peter Svidler
- Levon Aronian
- Boris Gelfand
- Alexander Grischuk

A continuación, se describen las 2 etapas:

Fase de rondas

Son 5 rondas y en cada ronda se juegan 3 partidas. Como en total son $5 \times 3 = 15$ partidas, al final de las 15 rondas todos habrán jugado contra todos. Cada ronda se juega en días distintos.

Ronda 1 – Día 1		
Levon	vs	Magnus
Boris	vs	Alexander
Peter	vs	Vladimir
Ronda 2 – Día 2		
Magnus	vs	Vladimir
Alexander	vs	Peter
Levon	vs	Boris
Ronda 3 – Día 3		
Boris	vs	Magnus
Peter	vs	Levon
Vladimir	vs	Alexander

Ronda 4 – Día 4

Magnus	vs	Alexander
Levon	vs	Vladimir
Boris	vs	Peter

Ronda 5 – Día 5

Peter	vs	Magnus
Vladimir	vs	Boris
Alexander	vs	Levon

Indicaciones para esta ronda:

-Para simular las partidas, cada una durará 0.15 segundos, para ello usará la función `sleep()` de Python para cuando le pidan implementación síncrona, o la función `asyncio.sleep()` para cuando le pidan la implementación asíncrona.

- Por cada partida ganada el ganador recibe 1 punto, y el perdedor 0 puntos. Asuma que no hay empates.

- Para este laboratorio, la manera en la cual se decidirá quién ganará en cada partida es comparando los ratings de cada jugador (ver columna “rating” del archivo `players.csv` disponible en Paideia). Quien tiene mayor rating, gana.

- Al finalizar las 5 rondas, clasifica a la siguiente fase solo 1 jugador: aquel que tenga mayor puntaje.

Fase final

El ganador de la fase de rondas se enfrenta con el campeón vigente, para este laboratorio lo llamaremos “Anand” (el campeón del torneo del año anterior).

Fase final

Ganador de fase de rondas	vs	Anand
---------------------------	----	-------

En la fase final los 2 finalistas jugarán 12 partidas, y el que obtiene el mejor puntaje al final de las 12 partidas gana el campeonato.

En esta fase los resultados posibles por cada partida para cada jugador son: ganar (1 punto), empatar (0.5 punto), perder (0 puntos).

Los resultados de cada partida en esta fase van a ser determinados de manera aleatoria.

Duración de cada partida: 0.15 segundos

- a) Leer el contenido del archivo 'players.csv' **(1 pto)**
- b) Desarrollar la función *fase_rondas_async()* que implementa la fase de rondas de manera asíncrona. Esta función debe retornar el nombre del único jugador que pasará a la siguiente fase. Nota: Deberá asumir que los 3 partidos de cada ronda se juegan en simultáneo; así que tiene que implementar ejecución de corrutinas para cada ronda. **(3 ptos)**
- c) Desarrollar la función *fase_rondas_sync()* que hace lo mismo que la parte b) pero de manera síncrona. **(2 ptos)**
- d) Desarrollar la función *fase_final_sync()* que calculará los resultados de la fase final de manera síncrona. **(2 ptos)**