

1IEE14 - Laboratorio 3

Instrucciones para el laboratorio:

- 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 usar Github CoPilot. 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 L3_CODIGOPUCP.zip o L3_CODIGOPUCP.rar el cual contendrá sus archivos de Python para cada pregunta. En caso se le pida responder de manera teórica o incluir capturas de pantalla, deberá adjuntar un documento PDF con sus respuestas.
- 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 (1 punto):

Descargue los archivos *pregunta1.py* y *foobar.txt* de Paideia. Mueva los archivos al WSL Ubuntu dentro de una misma carpeta.

Lo que hace el archivo de Python es leer el contenido del archivo *foobar.txt* usando dos objetos: *file1* y *file2*.

- a) (0 puntos) Lea y ejecute el programa. Vea cómo *file1* y *file2* son objetos independientes que cada uno hace la lectura por separado del archivo. Este apartado vale 0 puntos porque se le da instrucciones solo para que entienda el script de Python.
- b) (1 punto) Modifique el programa para que *c1* contenga los 2 primeros caracteres leídos de *file1*, mientras que *c2* contenga el tercer carácter leído de *file2*. **Pista:** Para poder leer el tercer carácter o cualquier carácter en la posición *n*, puede hacer uso de la función *seek()* como se explica en la [referencia de Python](#).

Pregunta 2 (4 puntos):

Descargue el archivo *pregunta2.py* de Paideia. Mueva el archivo al WSL Ubuntu.

El script de Python lo que hace es leer el contenido del standard input(que viene a ser el teclado), y lo reenvía al standard output (que viene a ser el terminal para que el usuario lo visualice).

- a) (0 puntos) Lea y ejecute el programa. Mientras el programa está en ejecución, tipee varios caracteres y luego presione *Enter*. Verá que lo que escribió antes del *Enter* se imprime nuevamente. Eso es lo que hace el programa: El texto ingresado por teclado lo redirecciona al estándar output (el terminal). Una vez que termine de hacer sus pruebas,

para terminar la ejecución del programa tipee *Ctrl+C*. Este apartado vale 0 puntos porque se le da instrucciones solo para que entienda el script de Python.

- b) (1 punto) Haga una copia del script, renómbrelo `pregunta2b.py` y modifique el código para que en vez de escribir cada línea en el estándar output(el terminal), se escriba cada línea en un archivo llamado `output_parte2b.txt`
Por ejemplo, si usted tipea en teclado:

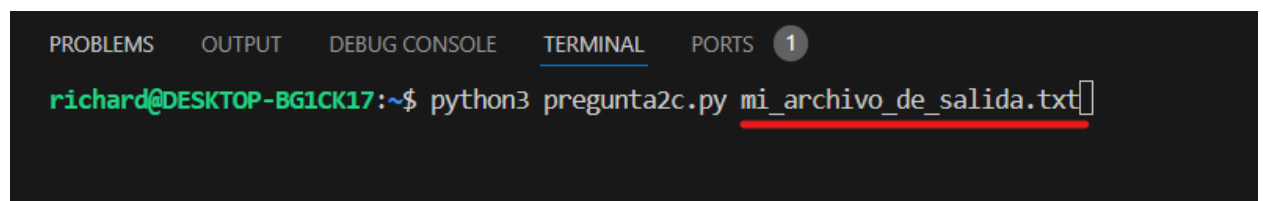
Mi primera línea
Mi 2da línea
Esta es mi última línea

Y luego tipea *Ctrl+C*, entonces debería haberse creado un archivo llamado `output_parte2b.txt` que cuando lo abres contenga lo mismo:

Mi primera línea
Mi 2da línea
Esta es mi última línea

- c) (2 puntos) Haga una copia de `pregunta2b.py`, renómbrelo `pregunta2c.py`, y modifique el código para que el programa reciba un argumento adicional opcional por línea de comandos: El nombre del archivo de salida. Es decir, a diferencia de la 1b donde el archivo se llamaba `output_parte2b.txt`, ahora el usuario ingresará el nombre del archivo. Este argumento es opcional: Si el usuario no ingresa el nombre del archivo, el programa usará el estándar output como salida, caso contrario lo guarda en el archivo.

Nota 1: Cuando se le pide “recibir un argumento por línea de comandos” se le pide hacer lo que se hizo en el lab 1 dirigido: Que el usuario al momento de ejecutar el programa le pasa un argumento al programa, por ejemplo:

A screenshot of a terminal window with a dark background. At the top, there is a navigation bar with tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. To the right of the tabs is a circular icon with the number '1'. Below the tabs, the terminal shows a prompt 'richard@DESKTOP-BG1CK17:~\$' followed by the command 'python3 pregunta2c.py mi_archivo_de_salida.txt'. The command is highlighted with a red underline.

Lo que no debe hacer es usar la función `input()`, porque eso no es recibir argumentos por línea de comandos.

Nota 2: En cualquier pregunta que se le diga que la entrada debe ser ingresada por línea de comandos, se refiere a como se muestra en el screenshot de arriba.

- d) (1 punto) Haga una copia de pregunta2c.py, renómbrelo pregunta2d.py, y modifique el código para que ahora el usuario tenga que obligatoriamente ingresar 2 argumentos por línea de comandos: 1) un nombre del archivo de entrada, y 2) un nombre del archivo de salida. Lo que hará el programa en Python es leer todo el contenido del archivo de entrada y escribirlo en el archivo de salida.
El resultado final debe ser que el contenido del archivo de salida es igual al del archivo de entrada.

Pregunta 3 (4 puntos)

Descargue de Paideia el archivo Log_de_medidores.csv. Este archivo contiene el registro(en inglés "Log") de eventos ocurridos en medidores electrónicos digitales. Cada columna significa lo siguiente:

Fecha-hora: La fecha y hora a la que se registró el evento

Evento: El nombre del evento

Severidad: Indica si es una advertencia (WARN), si solo es información (INFO), o si es un error en el medidor (ERROR).

Descripción: Da una descripción más detallada del evento.

La columna más importante es **Severidad**, porque lo que se quiere es que Ud. escriba un programa de Python (nombre del archivo: **pregunta3.py**) que reciba como argumento por línea de comandos el tipo de severidad a analizar la cual puede ser:

-WARN

-INFO

-ERROR

Y en base a eso su programa va a imprimir solo la primera línea en el archivo csv que encuentre con ese nivel de severidad. A continuación, el programa se debe quedar en un bucle a la espera de que el usuario interactúe: Si el usuario ingresa una 's', significa que quiere ver la siguiente línea en la cual ocurrió ese tipo de severidad por lo que programa imprimirá la siguiente línea que encuentre. Si el usuario vuelve a tipear 's', el programa busca la siguiente ocurrencia y así sucesivamente. Si el usuario tipea 'p', significa que quiere parar el programa y ahí acaba la ejecución.

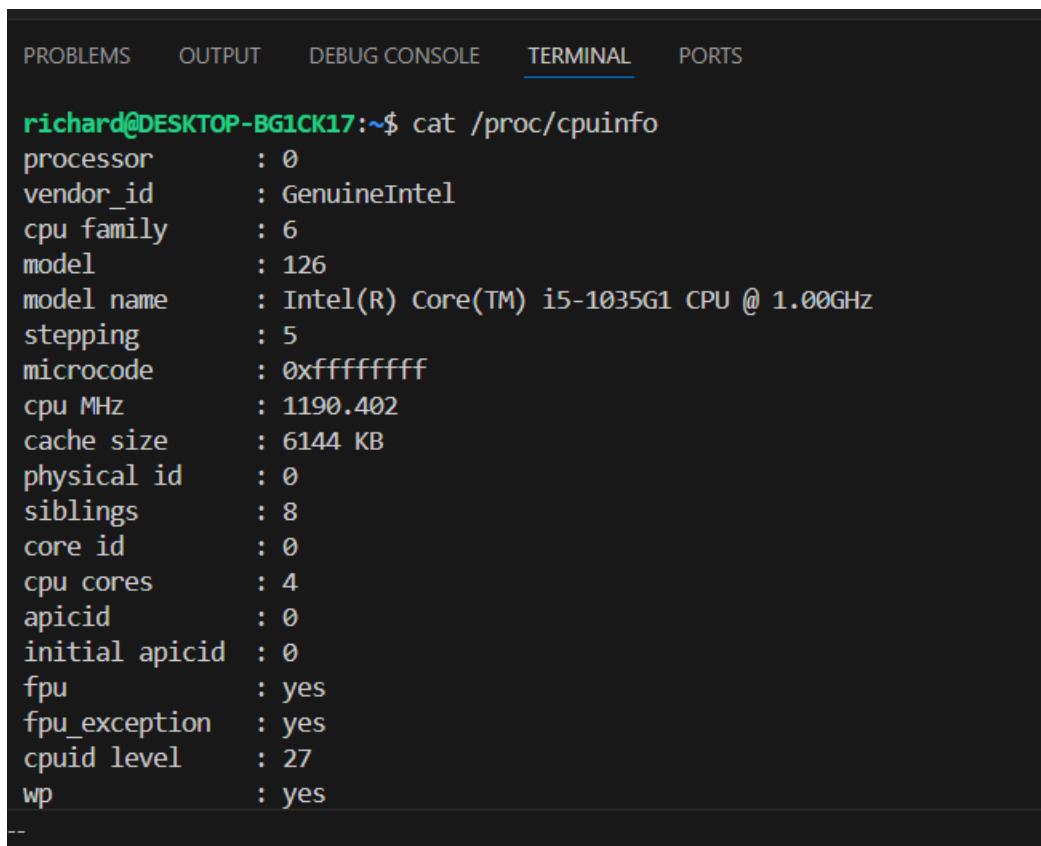
Si es que el usuario tipea 's', y el script ya no encuentra una siguiente línea con ese nivel de severidad, el script ya no imprime nada y simplemente acaba.

Nota: Puede usar la librería csv o también puede manipular el archivo sin librerías. Cualquier opción es válida.

Pregunta 4 (5 puntos)

Linux posee archivos virtuales que permiten obtener información de la computadora y del sistema en general. Estos pueden ser leídos en Python como si fueran cualquier archivo como los que ha visto en clase. En esta pregunta leeremos el contenido del archivo virtual `/proc/cpuinfo`, que provee información de los CPUs.

- a) (0 puntos) Tipee en el terminal `cat /proc/cpuinfo`. El comando `cat` se usa para mostrar el contenido de archivos en el terminal. Lo que está haciendo usted al ejecutar `cat /proc/cpuinfo` es ver el contenido del archivo `/proc/cpuinfo`. Verá algo similar a esto:



```
richard@DESKTOP-BG1CK17:~$ cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 126
model name     : Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz
stepping       : 5
microcode      : 0xffffffff
cpu MHz        : 1190.402
cache size     : 6144 KB
physical id    : 0
siblings       : 8
core id        : 0
cpu cores      : 4
apicid         : 0
initial apicid : 0
fpu            : yes
fpu_exception  : yes
cpuid level    : 27
wp             : yes
--
```

Es decir, `/proc/cpuinfo` contiene información de los CPUs de su computadora. Este apartado vale 0 puntos porque se le da instrucciones solo para que entienda el concepto de archivos virtuales.

- b) (3 puntos) Cree un archivo en llamado **`pregunta4b.py`** y escriba el código necesario para leer el contenido de `/proc/cpuinfo` y luego analizar su contenido para imprimir la cantidad de procesadores que tiene su computadora.

Así es como se debe ver la ejecución de su programa (obviamente la cantidad que a usted le salga puede ser distinta, dependerá de su computadora):

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

richard@DESKTOP-BG1CK17:~$ python3 pregunta5b.py
Numero de procesadores: 8
```

Aclaraciones:

- Recuerde que `/proc/cpuinfo` puede ser leído como cualquier otro archivo, es decir lo puede leer así:

```
with open("/proc/cpuinfo", 'r') as f:
    data = f.read()
```

- El cálculo de los procesadores lo debe hacer el algoritmo de su programa Python en base a lo que lea de `/proc/cpuinfo`.
 - Sugerencia para resolver el ejercicio: Ud. tendrá que buscar cuántas veces se repite la palabra "processor" en el texto que lee de `/proc/cpuinfo`. Para ello puede iterar por cada fila del archivo e ir contando las filas que contenga la palabra "processor". Un operador que le puede ayudar a saber si es que una palabra se encuentra dentro de una cadena de texto es el operador `in`: [Operadores in & not in en python](#)
- c) (2 puntos) Al igual que `/proc/cpuinfo`, existe también `/proc/meminfo`, cuyo contenido se ve así:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

richard@DESKTOP-BG1CK17:~$ cat /proc/meminfo
MemTotal:      8026616 kB
MemFree:       6647180 kB
MemAvailable:  6728656 kB
Buffers:       46952 kB
Cached:        222800 kB
SwapCached:    0 kB
Active:        146468 kB
Inactive:      1046936 kB
Active(anon):   204 kB
Inactive(anon): 923508 kB
Active(file):   146264 kB
Inactive(file): 123428 kB
Unevictable:    0 kB
Mlocked:        0 kB
SwapTotal:     2097152 kB
SwapFree:      2097152 kB
Dirty:         0 kB
Writeback:     0 kB
AnonPages:     919140 kB
```

Cree el archivo **pregunta4c.py** y escriba el código necesario para que su programa imprima cada 2 segundos la cantidad de memoria disponible de manera porcentual, la cual se calcula así:

Cantidad de memoria disponible (en porcentaje): $(\text{MemAvailable} / \text{MemTotal}) * 100$

Donde MemAvailable y MemTotal son valores que se obtienen leyendo el archivo */proc/meminfo* tal como se aprecia en la figura de arriba.

Su programa deberá estar en un bucle siempre para imprimir cada 2 segundos el valor porcentual calculado; y para detener su ejecución puede tipear Ctrl+c.