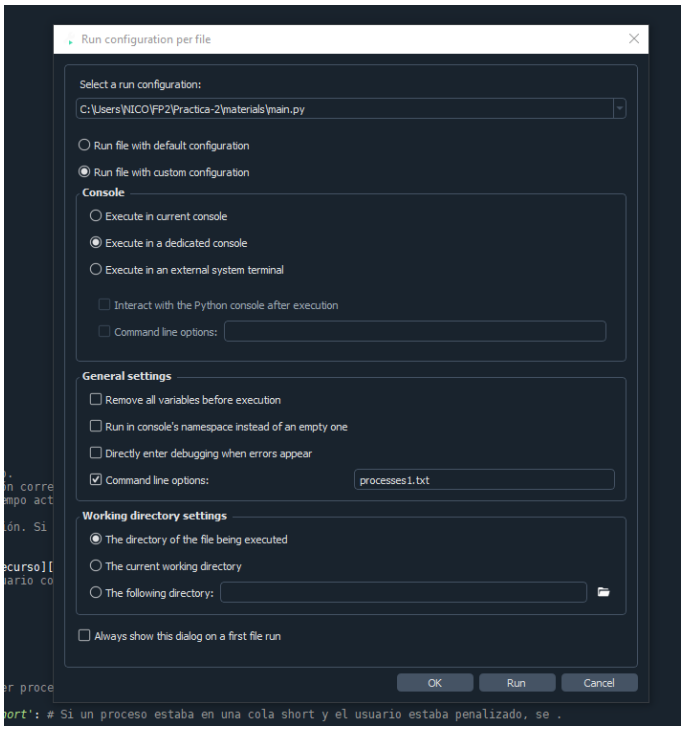
MANUAL DE USUARIO

Nicolás Muñiz Rodríguez : nicolas.muniz@udc.es

Pablo José Pérez Pazos : pablo.perez.pazos@udc.es

· Mecanismo de ejecución:

Para conseguir ejecutar el programa lo primero que debemos hacer es extraer los archivos en código python del archivo en formato zip adjunto (“array_queue.py ” y “main.py”) por ejemplo a una carpeta del escritorio. A continuación, debemos abrir el archivo “main.py” mediante Spyder, habiendo abierto este último desde el navegador de Anaconda. Desde esta ventana, haremos click en la pestaña “Run” y seleccionaremos la opción “Configuration per file…”, que nos abrirá la siguiente pestaña:



Aquí tendremos que tener marcadas las opciones “Run file with custom configuration”, “Execute in current console”, “The directory of the file being executed” y “Command line options” donde, a la derecha de esta última, escribiremos el nombre del archivo de texto que contiene la información de: identificador único del proceso, identificador único del usuario propietario del proceso, tipo de proceso, tiempo estimado de ejecución indicado por el usuario, tiempo de ejecución, en nuestro caso “processes1.txt”. Por último, haremos click en el botón de “Run” de esta última pestaña y observaremos los resultados en la terminal de Spyder.

· Fases de desarrollo:

Primero, comenzamos por construir el lector del archivo de texto, lo cual fue sencillo teniendo el ejemplo de la práctica anterior, y la clase abstracta “Proceso” que hace referencia al concepto general de un proceso en una cola. A esta clase le asociamos las variables fundamentales (ID_proceso, ID_usuario, recurso, tiempo_estimado, tiempo_real...) como variables privadas (denotadas por “_” delante del nombre) y creamos algunos de los métodos necesarios para la correcta funcionalidad de los procesos como *__init__()* y *__str__()*. Además, para en un futuro poder llamar atributos que inicialmente son privados, creamos los *setters* y *getters* de las diferentes variables.

Al completar la clase “Proceso”, creamos una clase denominada “GestorColas” la cuál nos ha facilitado la organización de los procesos en 4 colas según el recurso necesario y la duración aproximada del mismo. En esta clase hemos creado un conjunto (*set* de Python) de usuarios penalizados y dos diccionarios, uno que organiza las cuatro colas (*buffer*) y otro que utilizaremos para marcar si un proceso está o no en ejecución (*ejecucion*). En esta etapa de la práctica nos surgieron ciertas complicaciones como que el programa no detectaba que un proceso actual fuera de tipo “Proceso”, que solucionamos haciendo un par de modificaciones al lector de texto. También decidimos implementar varias funciones, como *ejecutar()* o *penalizar()*, e implementamos los *setters* y *getters* de los atributos de la clase

De entre los problemas más notorios cabe destacar un bucle infinito que nos resultaba al añadir la condición de que el *buffer* no pudiese estar vacío al *while* de la función *main()*, por lo que decidimos eliminarla, solucionando así este bucle infinito. El problema es que el programa, a la hora de decidir si meterse en el *if* que se activaría si el usuario del proceso está penalizado y la cola que se mira es de la variedad “short”, no se cumple la condición por alguna razón, aunque debería de suceder.

Creemos que este fallo resulta de que no realiza los ciclos necesarios, lo que implica que el diccionario de ejecución se vacía después de que se vacíe la cola de registro, pero al añadir la condición de la que hablábamos antes, basada en la función del gestor *ejecucion_is_empty()*, crea de nuevo el bucle infinito. Después de horas de revisiones, cambios al código y de preguntar a nuestros compañeros, no hemos encontrado forma de solucionar este inconveniente, por lo que nos gustaría que nos aconsejasen al respecto. De igual manera el programa funciona y no da errores con ninguno de los dos archivos de texto, y hemos entendido el funcionamiento del *TAD* de las colas, cosa que vemos como un éxito.