Examen de Programación
Junio 2022
(Grados en Física y Matemáticas)

Primera parte (1.25 puntos por cuestión, 50% nota del examen)

1) Escribir una función que calcule y retorne el precio de la electricidad, en euros/kWh, que se aplica a un consumo para la hora indicada y teniendo en cuenta si es día laborable o fin de semana.

	Horario		Precio electricidad consumida
Tarifa	Laborable	Fin de semana	€/kWh
Punta	10-14h 18-22h	-	0.133118
Llano	8-10h 14-18h 22-24h	-	0.041772
Valle	0-8h	0-24h	0.006001

   La función recibe como parámetros la hora a la que se hace el consumo (número entero) y un booleano que indica si el consumo es en día laborable (`True`) o en fin de semana (`False`). Observar que las tarifas se acaban al inicio de su hora final. Por ejemplo, a las 10h la tarifa aplicable es la tarifa "Punta", pues la tarifa "Llano" acabó a esa misma hora.

   La función retorna el precio aplicable según la tabla, en euros/kWh. Si la hora indicada no está entre 0 y 23 se retorna `math.nan`, para indicar el error.

   Se valora la eficiencia y tamaño compacto del código.

2) Escribir una función a la que se le pasa como parámetro una lista conteniendo números reales y que retorne una tupla de dos números enteros que indiquen cuántos de los elementos de la lista son menores que cero, y cuántos son mayores que 100.

3) La OCDE (Organización para la Cooperación y Desarrollo Económico) utiliza el índice de *Theil* para medir las desigualdades entre regiones. Se calcula como:

$$T = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{y_i}{\bar{y}} \cdot \ln\left(\frac{y_i}{\bar{y}}\right)$$

   siendo N el número de regiones a analizar, y_i la variable de interés de la región i (i entre 1 y N) e $\bar{y}$ la media de la variable de interés para todas las regiones. La variable de interés puede ser la esperanza de vida, los ingresos familiares, la tasa de alfabetización, etc.

   Se pide escribir una función que tome como parámetro una lista de números reales con los valores de las variables de interés de un grupo de regiones. La función debe retornar el índice de Theil, T, aunque si la lista está vacía se debe lanzar la excepción `DatosInsuficientes`, ya importada.

4) Dentro del directorio del usuario se dispone de dos directorios llamados `instrucciones` y `experimento`. El directorio `instrucciones` contiene, entre otros, ficheros de texto (con extensión ".txt"). El directorio `experimento` contiene a su vez, entre otros, varios ficheros con datos binarios, acabados en ".data", varios ficheros de hojas de datos acabados en

".csv" y un directorio llamado pictures, que contiene imágenes con extensión ".png". Suponiendo que el directorio de trabajo inicial es el del usuario, y utilizando exclusivamente *rutas relativas*, escribir las órdenes linux/unix necesarias para crear un resumen de los datos almacenados, realizando los siguientes pasos:

a) crear en el directorio del usuario un nuevo directorio llamado resumen

b) copiar todos los ficheros acabados en ".txt" de instrucciones a resumen

c) mover todos los ficheros acabados en ".csv" de experimento a resumen

d) crear dentro de resumen un directorio llamado datos_binarios

e) copiar todos los ficheros acabados en ".data" de experimento a datos_binarios

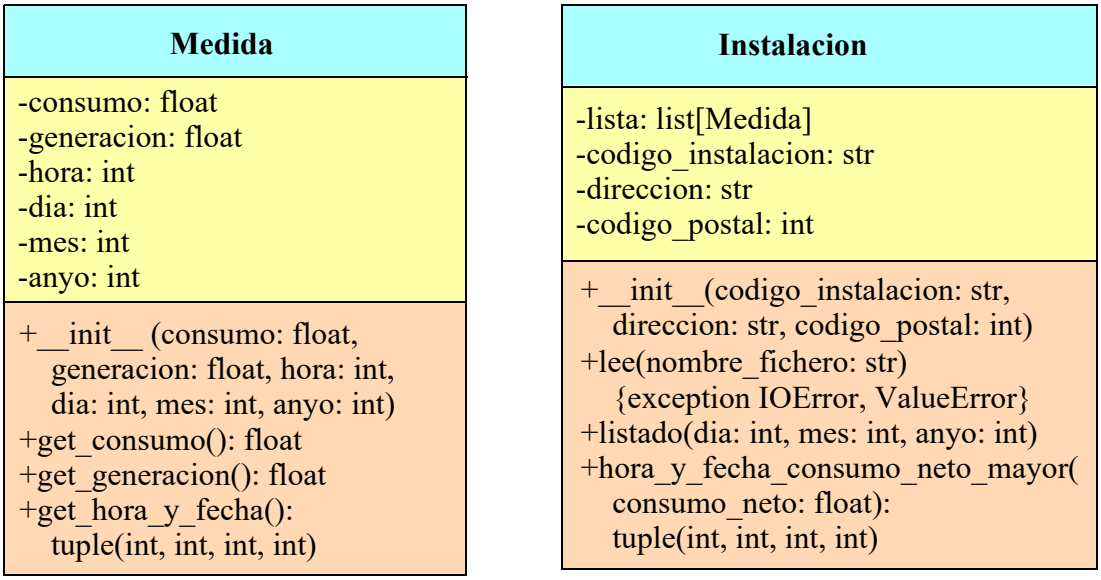
f) mover el directorio pictures de experimento a resumen

g) borrar todos los ficheros que estén todavía en experimento y sean del año pasado. Estos ficheros tienen en su nombre la secuencia "2021"

Examen de Programación
Junio 2022
(Grados en Física y Matemáticas)

Segunda parte (5 puntos, 50% nota del examen)

Se dispone en el módulo `instalacion_solar.py` de una clase llamada `Medida` que permite guardar los datos de la medida del consumo y generación eléctrica de una instalación doméstica que tiene aparatos de consumo y paneles solares capaces de generar electricidad. Los datos se refieren a una hora de una fecha concreta. La clase responde al siguiente diagrama de clases:



Los métodos de la clase `Medida` hacen lo siguiente:

- *constructor*: crea el objeto poniendo los datos de la medida a los valores indicados
- `get_consumo()`: retorna el consumo eléctrico en kWh
- `get_generacion()`: retorna la generación eléctrica en kWh
- `get_hora_y_fecha()`: retorna la hora, día, mes y año de la medida, en forma de tupla

Lo que se pide es escribir en el mismo módulo `instalacion_solar.py` una clase llamada `Instalacion` que responda al diagrama de clases que se muestra arriba y que contenga como atributo una lista de objetos de la clase `Medida` con medidas de consumo y generación de una instalación eléctrica, a lo largo de varias horas y fechas. La lista está ordenada por fecha y hora de más antigua a más nueva. Pueden faltar las medidas de algunas horas si el contador ha estado apagado algún tiempo. La clase tiene otros tres atributos: el código de la instalación, la dirección y el código postal. Los métodos de la clase deben hacer lo siguiente:

- *constructor*: crea la lista vacía y da valor a los otros tres atributos usando los valores que se pasan como parámetros.
- `lee()`: lee los datos de las medidas eléctricas del fichero cuyo nombre se pasa como argumento y los añade al atributo `lista`. El fichero contiene una línea de encabezamiento que se ignora, y una línea por cada medida con sus datos separados por ";", como en el siguiente ejemplo. Observar que los números reales están en formato español, con una coma como separador decimal:

```
Consumo (kW*h);Generación (kW*h);Hora;Día;Mes;Año
0,3;0,0;0;1;1;2022
0,2;0,0;1;1;1;2022
0,2;0,0;2;1;1;2022
...
```

El primer dato de cada línea es el consumo en kWh, el segundo es la generación, también en kWh, ambos números reales. Luego vienen la hora (de 0 a 23), día (de 1 a 31), mes (de 1 a 12) y año, que son números enteros.

Si el fichero no existe se lanzará automáticamente `IOError`. Si el fichero tiene errores se genera automáticamente `ValueError`. Debemos tratar estas excepciones poniendo un mensaje descriptivo del error concreto que ha ocurrido y volviendo a lanzar la misma excepción.

Pista: para separar una línea leída del fichero en sus datos usando un separador, puede usarse el método `split()` de los strings. Por ejemplo, si el separador fuese `"/"`:

```
linea.split("/")
```

- `listado()`: Hace un listado de los consumos y generaciones de la instalación en el día, mes y año indicados. Para este listado primero se ponen cuatro líneas de encabezamiento con:
    - código de la instalación
    - dirección y código postal
    - fecha
    - explicación de las columnas con los datos

  A continuación se pone una medida por línea, con su consumo, generación y hora, escritos en columnas, usando dos decimales para los dos primeros. Este es un ejemplo del formato:

```
Instalación de código: X1234
Situada en: Avda Ilustración, 12, 39005, Santander Código postal:39005
Listado de las medidas en la fecha: 1/1/2022
Consumo (kW*h)   Generación (kW*h)   Hora
          0.30                0.00      0
          0.20                0.00      1
          0.20                0.00      2
...
```

  Si no hubiese ningún dato en la fecha indicada, se pondrá el texto `"No hay medidas en esta fecha"` tras las cuatro líneas de encabezamiento.

- `hora_y_fecha_consumo_neto_mayor()`: Busca si existe un consumo neto (consumo - generación) superior al valor indicado en el parámetro, en kWh. Si se encuentra, retorna la hora y fecha en forma de tupla con la hora, día, mes y año. Si no se encuentra, retorna `None`.

Se pide, además, escribir un programa principal que pruebe todos los métodos de la clase `Instalacion` a partir de los datos del fichero de medidas llamado `"medidas.txt"`. Si se lanza `IOError` se deben abandonar todos los pasos restantes del programa y no hacer nada más, es decir poner un tratamiento de excepción vacío. No es preciso tratar la excepción `ValueError`. Con una sola llamada a cada método es suficiente.

Valoración:

1) Constructor: (0,5 puntos)
2) lee: (1.5 puntos)
3) listado: (1 punto)
4) hora_y_fecha_consumo_neto_mayor: (1 punto)
5) main: (1 punto)