Examen de Programación
Junio 2021
(Grados en Física y Matemáticas)

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

1) Escribir una función que determina si una subida de precio es baja, media, alta o en realidad es una rebaja, en función de la tabla que se muestra a la derecha. El incremento se calcula de esta manera:

$$\frac{(precioNuevo - precioAntiguo)}{precioAntiguo} \cdot 100$$

incremento	caso	carácter retornado
< 0%	rebaja	R
del 0% al 3%	bajo	B
>3% hasta 5%	medio	M
>5%	alto	A

La función recibe como parámetros (números reales) el precio antiguo y el nuevo y retorna el carácter que se indica en la tabla según el caso encontrado. Se valora la eficiencia y tamaño compacto del código.

2) Escribir una función a la que se le pasa como parámetro un texto conteniendo palabras separadas por espacios en blanco y retorna una tupla de dos números enteros que indican respectivamente cuántos artículos masculinos contiene el texto y cuántos femeninos. Los artículos masculinos son "el", "los", "un" y "unos". Los femeninos son "la", "las", "una" y "unas". Se valora la eficiencia.

3) El índice de Atkinson es una medida útil de la desigualdad de una distribución de los ingresos de un grupo de personas. Se calcula como:

$$A = 1 - \frac{1}{\mu}\left(\frac{1}{N}\sum_{i=1}^{N} y_i^{1-\varepsilon}\right)^{\frac{1}{(1-\varepsilon)}}$$

siendo N el número de personas, y_i los ingresos de la persona i (i entre 1 y N), μ el promedio de todos los ingresos y ε un coeficiente que sirve para ponderar los ingresos más altos o más bajos.

Se pide escribir una función que tome como parámetros una lista de números reales con los ingresos de un grupo de personas y el valor de ε. La función debe retornar el índice de Atkinson, A, aunque si la lista está vacía se debe lanzar la excepción `DatosInsuficientesError`, ya importada.

4) Dentro del directorio del usuario se dispone de un directorio llamado `copias` y de otro directorio llamado `prac3` que contiene ficheros python (con extensión ".py"), módulos python compilados (con extensión ".pyc"), un directorio llamado `doc` con los documentos del proyecto, y ficheros de texto con la extensión ".txt". Suponiendo que el directorio de trabajo inicial es el del usuario, y utilizando exclusivamente rutas relativas, escribir las órdenes linux/unix necesarias para:

a) crear dentro de `copias` un nuevo directorio vacío llamado `copia_prac3`

b) copiar todos los módulos python de `prac3` a `copia_prac3`

c) hacer lo mismo con los ficheros de texto

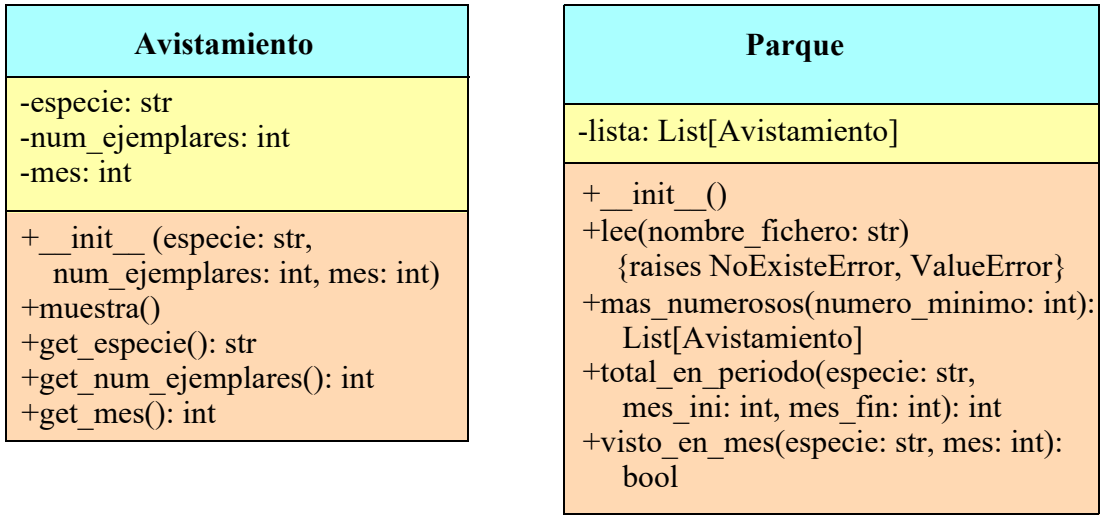
d) mover el directorio `doc` de `prac3` a `copia_prac3`

e) borrar todos los módulos python compilados de `prac3`

Examen de Programación
Junio 2021
(Grados en Física y Matemáticas)

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

Se dispone en el módulo avistamientos.py de una clase llamada Avistamiento que permite guardar los datos de un avistamiento de una especie de aves, anotando el nombre de la especie, el número de ejemplares, y el mes en que se realizó el avistamiento (entre 1 y 12). La clase responde al siguiente diagrama de clases:



Los métodos de la clase Avistamiento hacen lo siguiente:

- *constructor*: crea el objeto poniendo los datos del avistamiento a los valores indicados
- muestra(): escribe en la pantalla los datos del avistamiento en una línea
- get_especie(): retorna la especie
- get_num_ejemplares(): retorna el número de ejemplares
- get_mes(): retorna el mes en que se realizó el avistamiento, entre 1 y 12

Lo que se pide es escribir en el mismo módulo avistamientos.py una clase llamada Parque que responda al diagrama de clases que se muestra arriba y que contenga como atributo una lista de objetos de la clase Avistamiento con los avistamientos de aves en un parque natural, y que tenga también operaciones para trabajar con esta lista. La lista se leerá de un fichero de texto y no tiene por qué estar ordenada. Los métodos de la clase deben hacer lo siguiente:

- *constructor*: crea la lista vacía
- lee(): lee los datos de los avistamientos 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 avistamiento con sus datos separados por ";", como en el siguiente ejemplo:

```
Num;Mes;Especie
2;1;cernícalo vulgar
4;1;chorlitejo patinegro
12;2;abejaruco
...
```

El primer dato de cada línea es el número de ejemplares, el segundo es el mes, y luego viene la especie, compuesta por una o varias palabras. Es preciso tener en cuenta que al final de la especie habrá un salto de línea que es preciso ignorar tomando en cuenta todos los caracteres de la especie excepto el último.

Si el fichero no existe, se debe lanzar la excepción `NoExisteError` definida en el mismo módulo. Si el fichero tiene errores se genera automáticamente `ValueError`. Debemos tratar esta excepción poniendo un mensaje de error 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("/")
```

- `mas_numerosos()`: retorna una lista que contiene todos aquellos avistamientos cuyo número de ejemplares (obtenido con `get_num_ejemplares()`) es mayor que el parámetro `numero_minimo`.
- `total_en_periodo()`: retorna la suma total de los números de ejemplares de la especie indicada vistos en todos los avistamientos comprendidos entre los meses `mes_ini` y `mes_fin`, ambos inclusive.
- `visto_en_mes()`: retorna un booleano que indica si la especie indicada ha sido vista alguna vez o no en el mes indicado. Utilizar un algoritmo de búsqueda.

Se pide, además, escribir un programa principal que pruebe todos los métodos de la clase `Parque` a partir de los datos del fichero de avistamientos llamado `"avistamientos.txt"`. Si se lanza `NoExisteError` se deben abandonar todos los pasos restantes del programa y poner en pantalla un mensaje de error. No es preciso tratar la excepción `ValueError`. Con una sola llamada a cada método es suficiente. Para mostrar en pantalla los avistamientos retornados por `mas_numerosos()` puede usarse el método `muestra()` de los avistamientos.

Valoración:

1) Constructor: (0,2 puntos)

2) `lee`: (1 punto)

3) `mas_numerosos`: (0,8 puntos)

4) `total_en_periodo`: (1 punto)

5) `visto_en_mes`: (1 punto)

6) `main`: (1 punto)