Examen de Programación
Julio 2023
(Grados en Física y Matemáticas)

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

1) Se desea escribir la función coste_billete que calcula y retorna el coste, en libras, de la entrada a determinados servicios del Museo de Ciencias de Londres. Los servicios disponibles son EXPLORER, IMAX o DISCOVERY. El coste depende del servicio y del tipo de cliente (ADULTO o MENOR). El cálculo se hace según los valores de esta tabla, que vienen en libras:

	EXPLORER	IMAX	DISCOVERY
ADULTO	25.00	11.00	6.00
MENOR	22.00	10.00	5.00

La función recibe como parámetros el tipo de cliente (string) y el servicio (string), y retorna un número real. Se puede suponer que tanto el tipo de cliente como el servicio son strings correctos en mayúsculas. Se valorará que la tabla de tarifas se guarde en forma de tupla bidimensional.

2) Se dispone de unos datos extraídos de una encuesta electoral y en particular de una lista con las edades de muchos encuestados (números enteros). Se desea escribir una función a la que se le pasa esta lista como parámetro y retorna una tupla con 5 valores conteniendo los números de encuestados que están en los siguientes rangos de edad:

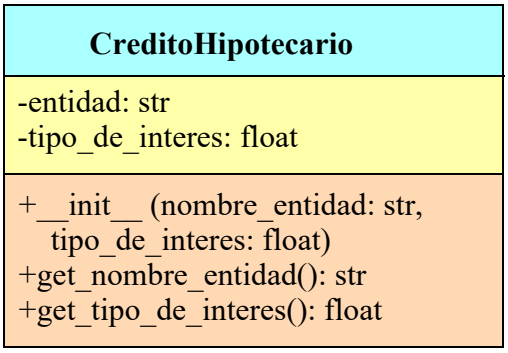
menor o igual que 25
entre 26 y 35
entre 36 y 45
entre 46 y 65
mayor que 65

3) Escribir una función iterativa que calcula y retorna el desarrollo en serie de la función:

$$f(x) = e^{-x^2} = \sum_{i=0}^{n} (-1)^i \cdot \frac{x^{2i}}{i!}$$

La función recibe como parámetros los valores x y n. Se valora que para hacer más eficiente el cálculo se tenga en cuenta que los valores del factorial $i!$ y de la potencia x^{2i} se pueden obtener en la iteración número i para la iteración siguiente multiplicando el factorial anterior por i+1 y la potencia anterior por x^2. Por su parte el signo se puede calcular con una variable que inicialmente vale 1 y a la que se cambia de signo a cada paso.

4) Se dispone de la clase CreditoHipotecario que contiene datos sobre tipos de interés aplicados a las hipotecas por diversas entidades bancarias y responde al diagrama de clases que se muestra.



Se pide escribir una función a la que se le pasa como parámetro una lista de objetos de la clase CreditoHipotecario y que retorna el nombre de la entidad bancaria cuyo tipo de interés es el más bajo de la lista, o la palabra "Ninguna" si la lista está vacía.

Nota 1: Se valora la eficiencia del código.

Nota 2: Se recuerda que es necesario documentar todas las funciones.

Examen de Programación
Julio 2023
(Grados en Física y Matemáticas)

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

Se desea escribir una clase para gestionar información de lugares turísticos en una agencia de viajes. Se dispone para ello de un fichero que contiene datos sobre lugares turísticos.

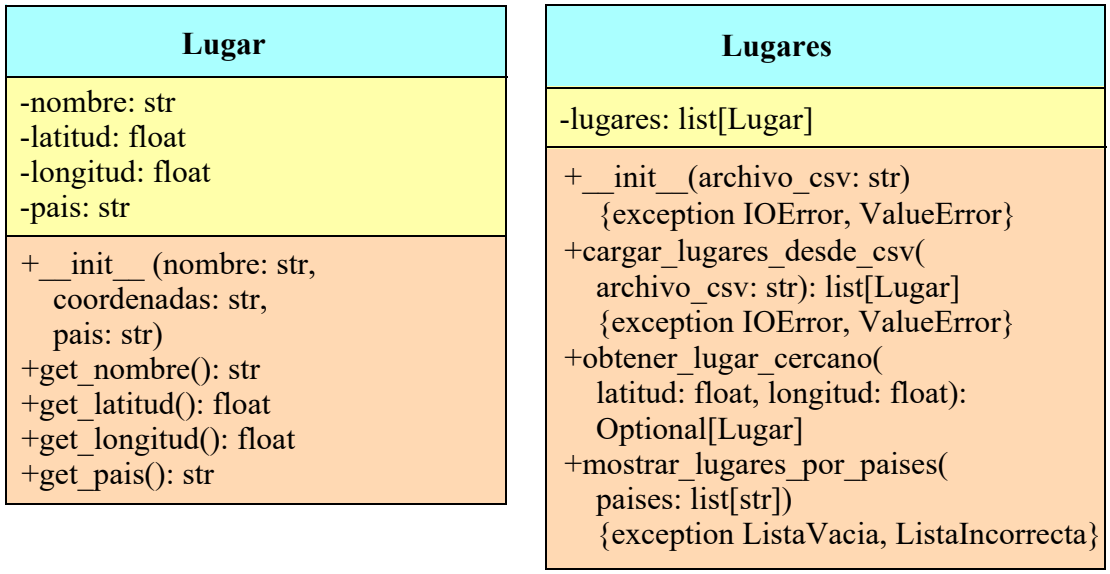
Se dispone de un módulo llamado `lugar_turistico.py` con la clase `Lugar` ya hecha. Los objetos de esta clase actúan como contenedores de los datos de un lugar turístico. Más abajo se muestra su diagrama de clases. Sus atributos son:

- __nombre: El nombre del lugar
- __latitud: La latitud del lugar, en grados
- __longitud: La longitud del lugar en grados
- __pais: El país en que se encuentra el lugar

El constructor recibe como parámetros el nombre del lugar, las coordenadas y el país. Las coordenadas se reciben como un texto conteniendo la latitud y longitud en dos números reales separados por una coma, como en este ejemplo: `56.43416667,-2.387222222`

La clase `Lugar` dispone de métodos observadores de los cuatro atributos.

Se desea crear en un módulo aparte llamado `turismo.py` una constante llamada `DIFERENCIA` de valor igual a 2 grados y la clase `Lugares` con el siguiente diagrama de clases. Esta clase contiene como único atributo una lista de objetos de la clase `Lugar` y operaciones para trabajar con ellos. Sus métodos son:



- *constructor*: crea la lista `lugares` obteniendo los datos mediante una llamada al método `cargar_lugares_desde_csv()`, usando el parámetro como nombre del archivo. No trataremos `IOError` ni `ValueError`, que podrían lanzarse al invocar a `cargar_lugares_desde_csv()` y por tanto se propagarán al siguiente bloque, en su caso.

- cargar_lugares_desde_csv(): lee del fichero cuyo nombre se indica en el parámetro archivo_csv los datos de los lugares turísticos y los añade a una lista que se retornará al final del método.

  El fichero contiene una línea de encabezamiento que se ignorará, y una línea por lugar turístico con datos separados por comas, como se muestra en el ejemplo. En primer lugar aparece el nombre del lugar, luego las coordenadas entre comillas y separadas por una coma, y finalmente el nombre del país. Ejemplo con las primeras líneas:

```
Location,"Latitude,Longitude",Country
Bell Rock Lighthouse,"56.43416667,-2.387222222",UK
Brooklyn Bridge,"40.70555556,-73.99638889",USA
Catacombs of Kom el Shoqafa,"31.178558,29.892954",Egypt
Channel Tunnel,"51.0125,1.5041",France
Chichen Itza,"20.68277778,-88.56861111",Mexico
```

  Si el fichero no existe se lanzará automáticamente IOError. Si el fichero tiene errores se genera automáticamente ValueError. No trataremos ninguna de estas excepciones, dejándolas pasar al siguiente bloque.

  Pistas: si se usa el método split(",") para separar los datos usando la coma como separador, se separarán también las dos coordenadas (latitud y longitud). Será preciso volver a juntarlas para crear el objeto de la clase Lugar, recordando quitar las comillas. Además, hay que acordarse de quitar el salto de línea presente al final de la línea.

- obtener_lugar_cercano(): Retorna el primer lugar de la lista que sea cercano a unas coordenadas específicas (latitud y longitud). La cercanía se establece si las diferencias entre la latitud y longitud que se pasan como parámetros y las del lugar buscado son ambas menores que la constante DIFERENCIA en valor absoluto. Si no hay ningún lugar que cumpla el criterio de cercanía se retornará None.

- mostrar_lugares_por_paises(): Muestra en pantalla un listado de los datos de los lugares que están en una lista de países escritos en minúsculas que se pasa como parámetro. Si esta lista está vacía se lanzará ListaVacia. Si los países de la lista no tienen todas sus letras en minúsculas se lanzará ListaIncorrecta. Estas excepciones están ya definidas en el módulo turismo.py. En caso de que no haya errores se mostrará el listado, con los datos en columnas, de acuerdo al siguiente ejemplo:

```
Name                                          Latitude(°)  Longitude(°) Country
--------------------------------------------------------------------------------
Bell Rock Lighthouse                          56.43416667   -2.38722222 UK
Brooklyn Bridge                               40.70555556  -73.99638889 USA
Catacombs of Kom el Shoqafa                   31.17855800   29.89295400 Egypt
```

  Pistas: En la comparación de los países con los de la lista paises se puede usar el método de los strings lower(), que retorna el mismo string en minúsculas. Por ejemplo, "Pepe".lower() retorna "pepe". Por otro lado, para comprobar si todas las letras de un string son minúsculas puede usarse el método islower(), que retorna True si son todas minúsculas y False en caso contrario. Por ejemplo, "Pepe".islower() retorna False.

Se pide, además, escribir un programa principal que pruebe todos los métodos de la clase Lugares a partir de los datos del fichero llamado "locations.csv" realizando los siguiente pasos:

a. Crear un objeto de la clase Lugares

b. probar el método mostrar_lugares_por_paises()

c. probar el método obtener_lugar_cercano()

Si se lanza IOError al crear el objeto se deben abandonar todos los pasos restantes del programa y poner un mensaje de error. No es preciso tratar la excepción ValueError. Si se lanzan ListaVacia o ListaIncorrecta al llamar a mostrar_lugares_por_paises() se debe poner un mensaje de error que explique lo que ha ocurrido y continuar con los siguientes pasos.

Valoración:

1) imports, constante, encabezamiento de la clase y constructor: (0,5 puntos)

2) cargar_lugares_desde_csv: (1.5 puntos)

3) mostrar_lugares_por_paises: (1 punto)

4) obtener_lugar_cercano: (1 punto)

5) main: (1 punto)