Fundamentos de la Programación. Grado en Física - Curso 2021/2022

Examen de teoría de la convocatoria de febrero (11/01/2022)

Instrucciones

1. Tienes **3 horas** para realizar el examen. Lee tranquilamente el examen y decide por dónde empezar.
2. No puedes utilizar apuntes ni programas ya escritos por ti o por otros para hacer el examen. Tampoco puedes comunicarte con nadie (excepto con el profesor) durante el examen. Puedes consultar al profesor por instrucciones concretas de Python que no recuerdes y puedes consultar los materiales disponibles en MoodleUA.
3. Finalmente debes entregar tu examen con todos los ejercicios que hayas realizado en un archivo comprimido en `zip` cuyo nombre debe ser tu DNI (`NUMERO_DNI.zip`).
4. Debes entregar dicho archivo comprimido a través de la tarea creada en MoodleUA. Dicha entrega se cerrará automáticamente a la hora fijada para la finalización del examen, de modo que debes tener cuidado en no retrasarte.

Ejercicios

1. (ej1.py) **(3 puntos)** Una matriz dispersa es una matriz con un alto porcentaje de elementos nulos. Una matriz dispersa con k elementos nulos se puede representar almacenando los elementos no nulos en una matriz de $k+1$ filas y 3 columnas. En las columnas 2 y 3 de la primera fila se almacena la dimensión de la matriz, el número de filas en la 2 y el de columnas en la 3 (el valor de la primera columna es irrevelante). El resto de filas contienen la fila, la columna y el valor de los elementos no nulos, respectivamente. Por ejemplo, la siguiente matriz dispersa se puede codificar como se muestra (observa que la primera fila/columna tiene índice 1):

$$\begin{pmatrix} 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 5 & 5 \\ 1 & 3 & 3 \\ 3 & 2 & 6 \\ 3 & 3 & 1 \\ 5 & 5 & 1 \end{pmatrix}$$

   a) Implementa una función `aDispersa(M)` a la que se le pasa como párametro una matriz en la codificación ordinaria de Python y devuelve la codificación dispera de dicha matriz.

   b) Implementa otra función `deDispersa(M)` a la que se le pasa una matriz codificada como dispersa y devuelve la matriz con la codificación ordinaria de Python.

   c) En el programa principal se debe leer una matriz de elementos enteros de un fichero de entrada y escribir su codificación como matriz dispersa en otro fichero de salida. Los nombres de ambos ficheros se pasarán por línea de comandos y debes tratar los posibles errores en caso de que los archivos no estén disponibles o no se llame correctamente al programa.

   Ejemplo: Si el archivo `matriz.dat` contiene

   ```
   0 0 3 0 0
   0 0 0 0 0
   0 6 1 0 0
   0 0 0 0 0
   0 0 3 0 1
   ```

   y se invoca al programa desde la consola con la instrucción

   ```
   $ python3 ej1.py matriz.dat salida.txt
   ```

   el archivo `salida.txt` debe contener

   ```
   0 5 5
   1 3 3
   3 2 6
   3 3 1
   5 3 3
   5 5 1
   ```

2. (ej2.py) **(2.5 puntos)** Implementa una función llamada `compara` a la que se le pasan por parámetro dos cadenas. El método debe devolver una matriz de enteros que tenga el mismo número de filas que caracteres tiene el primer parámetro y tantas columnas como caracteres tenga el segundo parámetro. El contenido de la posición `[i][j]` de la matriz lo determinan el carácter del primer parámetro correspondiente a la fila `i` y el carácter del segundo parámetro correspondiente a la columna `j` de la siguiente manera:

   - si el carácter `i` de la primera cadena es menor que el carácter `j` de la segunda, se asigna a la posición `[i][j]` de la matriz un -1;
   - si el carácter `i` de la primera cade4na es mayor que el carácter `j` de la segunda, se asigna a la posición `[i][j]` de la matriz un 1;
   - si ambos caracteres son iguales, se asigna a la posición `[i][j]` de la matriz un 0.

   En el programa principal se deben pedir dos cadenas al usario e imprimir la matriz devuelta por la función en el formato del ejemplo.

   Ejemplo: Si las cadenas introducidas por el usuario son `moda` y `codo` la función debe devolver la matriz (en notación Python):

   `[[1, -1, 1, -1], [1, 0, 1, 0], [1, -1, 0, -1], [-1, -1, -1, -1]]`.

   y en el programa principal se debe imprimir

```
   c  o  d  o
m  1 -1  1 -1
o  1  0  1  0
d  1 -1  0 -1
a -1 -1 -1 -1
```

3. (ej3.py) **(2.5 puntos)** Deseamos dibujar la gráfica de la función Integral Exponencial entre 0.1 y 2. Esta función se define como

$$Ie(x) = \int_x^\infty \frac{e^{-t}}{t} dt$$

   Para ello vamos a proceder en dos pasos:

   a) Calculamos la función para 100 puntos equiespaciados en el intervalo anterior y los guardamos en un archivo de texto llamado `datos.txt` con el formato:

   x_1 $Ie(x_1)$
   x_2 $Ie(x_2)$
   ...
   x_{100} $Ie(x_{100})$

   b) Leemos los datos del archivo anterior, creamos la gráfica y la guardamos con el nombre `ej3.png`.

   Para realizar este ejercicio debes usar las librerías `numpy`, `matplotlib.pyplot` y el método `quad` de la librería `scipy.integrate`.

4. (ej4.py) **(2 puntos)** Nos pasan el siguiente código que describe la clase `Cuenta`:

```
class Cuenta(object):

    def __init__(self,titular,cantidad=0):
        self.titular=titular
        self.cantidad = cantidad


    def __str__(self):
        return "Cuenta\n"+"Titular: " + self.titular+ " - Cantidad: "+str(self.cantidad)
```

   a) Añade a esta clase dos métodos `ingresar(self,cantidad)` y `retirar(self,cantidad)` que añadan/resten `cantidad` a la cantidad previa en la cuenta.

   b) En el programa principal crea una Cuenta con 1000 € cuyo titular es `Pepe Botero`. Ingresa en dicha cuenta 500 € y retira 800 €. Muestra por pantalla la cuenta después de realizar dichas operaciones.