Examen

Junio

17 de Junio 2014

Informática
Año 2013/2014
Facultad de CC.
Matemáticas

▷ 1. Codificación de datos (4 puntos)

- **(2 puntos)** Una compañía desea transmitir datos por teléfono. Pero están preocupados por si sus teléfonos estuvieran intervenidos. Todos sus datos se transmiten como enteros de cualquier número de dígitos. Se te pide que escribas un programa que cifre los datos para poderlos transmitir con mayor seguridad. Tu programa deberá leer un entero introducido por el usuario y cifrarlo como sigue:

  - sustituye cada dígito d por el resto de dividir $d + 7$ entre 10,
  - a continuación intercambia los dígitos que ocupan las posiciones 1 y 3, 5 y 7 ... etc.
  - para terminar intercambia los dígitos que ocupan las posiciones 2 y 4, 6 y 8 ... etc.

  Asignamos la posición 1 al dígito menos significativo. El programa debe mostrar por pantalla el resultado de cifrar el número original.

- **(2 puntos)** Para completar el trabajo debemos escribir el programa a utilizar al otro lado del teléfono. Debes escribir un programa que reciba un número cifrado, así como el número de cifras del original y escriba por pantalla el número del que procede.

Ejemplo Supongamos que el número original es 35871. El primer paso lo transforma en el 02548. A continuación intercambiamos el 8 y el 5 (notar que el cero, que ocupa la posición 5 no se intercambia con nadie pues no hay dígito en la posición 7) y obtenemos 02845. Para terminar intercambiamos el 2 y el 4 obteniendo 04825. Pero, como entero, el cero a la izquierda es indiferente con lo cual la respuesta es 4825.

▷ 2. Monstruos S.A. (6 puntos)

Monstruos S.A. es la compañía eléctrica más importante de Monstruópolis. Su método de producción se basa en emparejar cada niño humano con el monstruo asustador más adecuado, siguiendo unos criterios estrictos de compatibilidad. De este modo, se obtienen gritos de la mejor calidad, que una vez refinados y tratados se convierten en energía limpia y fiable. Sin embargo, últimamente los asustadores han podido comprobar que realizar su trabajo les resulta muy complicado, ya que cada vez es más difícil asustar a los humanos. Como consecuencia de la escasez de gritos, la energía eléctrica se ha convertido en un producto escaso en Monstruópolis, lo que ha obligado a las autoridades a controlar su consumo, sobre todo en el parque automovilístico de la ciudad. Por este motivo, y con el propósito de limitar el gasto por habitante, se ha decidido elaborar una base de datos que ha de almacenar la información siguiente.

1. Solo algunos modelos de ciertas marcas de automóviles pueden transitar por las carreteras de Monstruópolis. Por esta razón, se guarda en una estructura de datos toda la información de dichas **especificaciones**, que incluye la **marca**, el **modelo** y su **consumo medio** por cada 100 km (medido en kWh).

2. Los **vehículos** del parque automovilístico se identifican unívocamente mediante su número de **matrícula**. Durante el proceso de registro, también se pide al propietario que aporte los datos correspondientes al **color** y la **especificación** (*m*arca, modelo y consumo medio). Además, todo vehículo se matricula a nombre de un único ciudadano mayor de edad, aunque es posible que éste tenga en propiedad varios automóviles.

3. La información sobre los **propietarios** se compone de su **NIF**, el **nombre**, **primer apellido** y **edad**, la fecha de **caducidad** de su carnet de conducir y una dirección válida de **e-mail**. Por ejemplo, Mike Wazowski es un propietario de 25 años. Su NIF es 69696969W, su carnet de conducir caduca el 23-03-2015 y su e-mail es ojitoSalton@monstruos.sa. Este monstruo ha matriculado un Nissan Moco verde cuyo consumo medio es de 18 kWh/100 km y un Lamborghini Reventón rojo algo más eficiente (consumo medio de 14,4 kWh/100 km).

AmpliacionMultasAcabas de sustituir al programador que estaba implementando el sistema de gestión del parque automovilístico de Monstruopolis.

Su codigo inicial es el siguiente:

```
class especificacion(object):
    def __init__(self,marca,modelo,consumo):
        '''
        mairca:string
        modelo:string
        consumo:float
        '''
        self.marca=marca
        self.modelo=modelo
        self.consumo=consumo
    def __eq__(self,otra):
        '''
        comprueba la igualdad de dos especificaciones
        '''
        return self.marca==otra.marca and self.modelo==otra.modelo and self.consumo==

class vehiculo(object):
    def __init__(self,matricula,color,especif,prop,lista_espec):
        '''
        matricula:str
        color:string
        especif:especificacion
        prop:propietario
        lista_espec: lista de especificaciones
        '''
        if especif not in lista_espec:
            raise Exception('El coche con especificacion: '+str(especif)+' no puede c
        else:
            self.matricula=matricula
            self.color=color
            self.especif=especif
            self.prop=prop

class permiso_conducir(object):
    def __init__(self,fecha_cad,puntos):
        self.fecha=fecha_cad
        self.puntos=puntos
```

```
class propietario(object):
    def __init__(self,nif,nombre,apellido,permiso,edad,email):
        self.nif=nif
        self.nombre=nombre
        self.apellido=apellido
        self.permiso=permiso
        self.edad=edad
        self.email=email

class parque_automoviles(object):
        def __init__(self):
            self.listaPopietarios=[]
            self.listaVehiculos=[]
        def registrar_vehiculo(self,v):
             if v.propietario not in self.listaPopietarios:
                  self.listaPopietarios.append(v.prop)
             if v not in listaVehiculos:
                  self.listaVehiculos.append(v)
```

Aparte de la gestión de los parámetros erróneos en las definiciones de los métodos generando las correspondientes excepciones, te tienes que poner con urgencia a implementar otra funcionalidad al sistema. La policía municipal de Monstruópolis quiere gestionar las **multas**. La información de cada multa consta de la matrícula del vehículo con el que se ha cometido la infracción, el importe (en euros) de la multa y el número de puntos a retirar del permiso de circulación del infractor.

- **(4 puntos)** Desarrolla e implementa en Python las clases adecuadas para almacenar la información del parque automovilístico de Monstruópolis. Las clases deben cumplir los requisitos siguientes.

  1. El constructor de la clase **Vehículo** comprueba la corrección de sus parámetros de entrada: Dada una lista de especificaciones, la correspondiente al nuevo automóvil aparece en dicha lista.
  2. El constructor de la clase **Propietario** comprueba la corrección de sus parámetros de entrada: La edad supera los 17 años, la fecha de caducidad es posterior a la actual y los vehículos que se van a registrar satisfacen los requisitos del apartado anterior.

     Indicación: El constructor **datetime.date(aa,mm,dd)** genera un objeto de la clase **Date** que representa la fecha **dd-mm-aa**. El método **datetime.date.today()** genera un objeto de la clase **Date** que representa la fecha actual.

- **(2 puntos)** Desarrolla e implementa en Python una función que, dado el NIF de un propietario, devuelva la lista de sus automóviles ordenada ascendentemente por el consumo medio.