# PRACTICAS CURSO DE PYTHON

** nota: En la cabecera de los ficheros si añades acentos, … Codificación en **UTF8**

#!/usr/bin/env python

# -*- coding: utf-8 -*-

También en Windows: cp1252

## LISTAS:

1. Implementar una función que realice la unión dos listas (de números).

2. Implementar una función que realice la intersección de dos listas.

3. Implementar la función unionALL de dos listas. Añade todos los elementos de las dos listas.

4.Ordenar una lista de cadenas por la longitud de estas. Ejemplo:

```
L: ['zzz', 'a', 'x', '23xxx', 'aaaavfg']
L.sort:  ['23xxx', 'a', 'aaaavfg', 'x', 'zzz']


L2: ['zzz', 'a', 'x', '23xxx', 'aaaavfg']
L2.sort:  ['a', 'x', 'zzz', '23xxx', 'aaaavfg']
```

## DICCIONARIOS:

**1.** Partiendo de una lista con repetidos convertirla en un diccionario que cuente las ocurrencias de cada elemento de la lista.

## CADENAS:

1. Implementación de algoritmos de encriptado:

**Cifrados por sustitución**

Un **cifrado por sustitución** es un método de cifrado por el que unidades del texto original son sustituidas con texto cifrado siguiendo un sistema regular; las "unidades" pueden ser una sola letra (el caso más común), pares de letras, tríos de letras, mezclas de lo anterior, entre otros. El receptor descifra el texto realizando la sustitución inversa.

Compárese con los **cifrados por transposición**, en los que las unidades del texto original son cambiadas usando una ordenación diferente y normalmente bastante compleja, pero las unidades en sí mismas no son modificadas. Por el contrario, en un cifrado por sustitución, las unidades del texto plano mantienen el mismo orden, lo que hace es sustituir las propias unidades del texto original.

**Sustitución simple**

En los cifrados de **sustitución simple** un carácter en el texto original es reemplazado por un carácter determinado del alfabeto de sustitución. Es decir, se establecen parejas de caracteres donde el segundo elemento de la pareja establece el carácter que sustituye al primer elemento de la pareja.

**Definir las funciones:**

**1.** Generar un alfabeto común para el emisor y el receptor del mensaje que consistirá en las letra mayúsculas, minúsculas, números y espacio en blanco.

**2.** Implementar las funciones codifica letra (devuelve la posición del carácter en el alfabeto) y decodifica letra (recibe la posición que ocupa el carácter y devuelve el carácter).

**3.** Apoyándonos en las anteriores codificar cadena y decodificar cadena. La primera recibe una cadena de texto y devuelve una lista con las posiciones de cada letra y la otra al revés.

**Cifrado de César**

El ejemplo más sencillo de cifrado de sustitución simple es el cifrado de César. El cifrado César, con clave k, utiliza Tk(j)=(j+k)modL, como biyección en el conjunto de caracteres del alfabeto (estrictamente, en el conjunto de índices), con L el cardinal de dicho conjunto. En términos de índices, se usa la permutación:

$$(0,1,2,...,L-1)\mapsto(k,k+1,...,L-1,0,1,2,...,k-1).$$

**4.** Definir una función python, cifradoCesar(texto,clave,alfabeto), que espere un texto y una clave, y considere el anterior como alfabeto por defecto. La función devolverá el texto cambiando cada carácter por el que en el alfabeto ocupa k posiciones a su derecha (si el alfabeto se queda corto, se vuelve a empezar). Así cifradoCesar('Calzado',5) devolverá 'HfpDfit'.

## PROGRAMACIÓN ORIENTADA A OBJETOS:

**1.** Implementar una agenda de teléfonos. Mantener el nombre y el teléfono de cada contacto. Añadir contacto, listar contactos, buscar un contacto, borrar contacto. Definir las clases necesarias.

**2.** Implementar una clase Hora y otra Fecha con las operaciones más habituales. Con herencia múltiple crear una clase FechaHora y crear métodos: constructor, __str__, __add__, __eq__, __cmp__ y propiedades. Probarlos métodos con código ==, cmp(ob1,ob2) , str(obj) …

**3.** En un proceso electoral los ciudadanos pueden distribuir su voto entre varios candidatos, además pueden emitir votos negativos de castigo. Para votar rellenan un vector de tal forma que entre todos los votos + o - sumen 100 puntos. Cada tarjeta electoral tiene un número identificativo de los ciudadanos con capacidad de voto. Los votos cuentan de una forma proporcional a la edad del votante. El mecanismo electoral gestiona la lista de ciudadanos que

votan (inicialmente vacía) e inicializa a cero las puntuaciones de una tabla de marcadores al objeto de contar los puntos de los candidatos. El recuento de votos consiste en lo siguiente:

```
WHILE queden_votos {
   leer_una_ficha
   IF el_número_está_en_el_censo {
      tachar_el_número_del_censo
      IF el_voto_es_válido THEN agregar_puntuación_(puntos*edad)_a_cada_candidato
   }
}
```

Un voto es válido si suma sólo 100 puntos.

Definir las clases e implementar el proceso para calcular el candidato ganador.

4. Implementar una clase Matriz y las operaciones para imprimir (__str__), sumar, multiplicar, get/set sobre un elemento.

## PROGRAMACIÓN FUNCIONAL:

1. Utilizando **List Compresion** generar:

- Una lista con las potencias de 2.
- Números pares del 1 al 100.
- Utilizando la lista anterior obtener los números que no están.
- Generar las 10 tablas de multiplicar en tuplas: (1,1,1),(1,2,2) … (10,1,10),…(10,10,100)

2. Implementar una función recursiva que sea una lista de listas e imprima todos los elementos uno a uno:

```
[1, 2, ['a', 'b', [23, 25, [100, 345], 34], ['c', 'd']]]

Resultado:
1
2
a
b
23
25
100
345
34
c
d
```

3.Una función similar a la anterior que quite las anidaciones de la lista.

```
[1, 2, ['a', 'b', [23, 25, [100, 345], 34], ['c', 'd']]]

Resultado:
[1, 2, 'a', 'b', 23, 25, 100, 345, 34, 'c', 'd']
```

4. Funciones Lambda.

- Construir 3 funciones lambda:
- car(L) → Devuelve el primer elemento de la lista
- cdr(L) → Devuelve el resto de la lista después de quitar el primero elemento.

- empty(L) → Devuelve True si la lista está vacía.
- Utilizando las 3 funciones anteriores construir una función que imprima una lista invertida y otra que devuelva una lista sin anidaciones.

- Escribir una función que devuelva una función lambda y ejecutarla desde fuera. Por ejemplo, una función que se encarga de incrementar una variable pero el valor de incremento se construye con una función lambda.

5. Con map y filter:

- Con **map**:
  A partir de una de texto, obtener la lista de números ascii correspondientes a cada letra:
  Cadena: hola que tal
  L: [104, 111, 108, 97, 32, 113, 117, 101, 32, 116, 97, 108]
- Con **filter**: disponemos de una lista de cadenas filtrar las que sean palabras palíndromos.
  ['hola', 'ana', 'oso', 'abbccdccbba', 'abcdeba']
  ['ana', 'oso', 'abbccdccbba']

## GENERADORES:

1. Escribir y probar un generador de números primos donde podamos indicar los límites inicio y fin.

## DECORADOR:

1.Escribir un decorador que escriba en un fichero de log la siguiente información (cada vez que se llame a una función que graba o borra en la BBDD).

Fecha y hora del sistema, operación y parámetros.

- 17/09/15 17:02:30 operación: grabar Parámetros: admin 1234
- 17/09/15 17:02:30 operación: borrar Parámetros: admin

  Para sacar la fecha y la hora del sistema:

  import time

  fecha = time.strftime("%d/%m/%y") # La fecha actual

  hora = time.strftime("%H:%M:%S") # La hora actual

## EXCEPCIONES:

1. A partir de la clase Time antes implementada, definir una excepción personalizada y modificar la clase Time para tener un control sobre los valores que son válidos al definir una hora.

## MÓDULOS Y PAQUETES:

1. A partir de las clases Fecha, Hora y FechaHora separar cada clase en un fichero distinto (módulo) y crear un fichero fuente para ejecutar el código principal. Añadir sentencias import donde corresponda. Comprobar el valor del atributo __name__ dentro de los módulos imprimiendo este y después ejecutando el módulo o importándolo esto permite elegir si queremos ejecutar o no un código dentro del módulo. Ejecutar los módulos para generar los ficheros pyc (ficheros compilados → aumenta la velocidad de ejecución).

- Para compilar desde **windows**:
- Con el IDLE pulsar F5 en el script.
- En **Linux**:
- Con el programa **geany** o en modo consola: **python -m py_compile mi_script.py**
- **Genera un fichero .pyc**

## E/S Y FICHEROS:

1. Listar el contenido de un fichero por consola.

2. A partir de una serie de ficheros de idioma. Definir una clase con una exception personalizada que permita internacionalizar los mensajes de una aplicación. Partimos de los ficheros es.txt y en.txt para español e inglés. La clase Idioma se inicializa con un código de país: es, en, etc. (controlar posibles errores si no se soporta el idioma) y con un método que le pasemos una clave nos devuelva la palabra traducida al idioma con el que se inicializó la clase. El programa debería reconocer otro idioma por ejemplo: **fr, de** simplemente añadiendo otro fichero: fr.txt o de.txt a la carpeta **lang**.

3. A partir de la anterior tomar el idioma por la línea de comandos con la siguiente sintaxis:

- **python idioma_comandos.py -es | -en | -h**
- Si falta el parámetro mostrar la sintaxis adecuada o con la opción -h
- Si falta el idioma seleccionado emitir un error.

## EXPRESIONES REGULARES:

1. Definir una función que valide una matrícula. Validar una lista de matrículas. Ojo sin vocales.

2.Definir una función que valide un DNI. Validar una lista de DNIs.

## SOCKET:

1. Esquema básico de comunicación servidor / cliente. El cliente solicita por teclado una cadena y la envía al servidor. La comunicación termina cuando el cliente envía la cadena "quit".

## SERIALIZACION:

1.Implementar una función que reciba un fichero (el nombre) y una lista u objeto y lo serialice. Después implementar otra función para hacer la operación a la inversa. Escribir código principal para probarlo.

## BASES DE DATOS:

1. Imprimir las propiedades de SQLite3.

2. Crear una BD, crear una tabla con SQL, insertar un registro y localizarlo con select.

3. Sobre la base de datos anterior implementar una función que importe el contenido de un fichero en la BD, otra que exporte en un fichero el contenido de una tabla de la BD y otra que limpie el contenido de la tabla de empleados. Utilizar el formato CSV para los ficheros. Utilizar sentencias parametrizadas. Las funciones reciben por parámetro el nombre del fichero de la base de datos y el CSV y retornarán el número de líneas importar. Añadir control de excepciones.

4. Implementar una clase que implemente un patrón DAO y permita realizar las operaciones **CRUD** (Create, Read, Update y Delete) sobre un empleado de la BD. Clases Empleado, EmpleadoDAO y un código principal para comprobar el funcionamiento de las clases.

## PRUEBAS

1. Utilizar las funciones implementadas en el apartado de listas para hacer pruebas con doctest.

2. Utilizar la clase Time para probar los métodos. Por ejemplo, sumar dos Time.

3.Aplicar pruebas tipo unittest a la clase Time u otra con preparación y destrucción del contexto.