

PRACTICAS CURSO DE PYTHON II

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

```
#!/usr/bin/env python
```

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

Para Windows:

```
# -*- coding: cp1252 -*-
```

BITS:

1. Generar el bit de paridad de un numero binario.

LISTAS:

1. Dada una lista de números (con repeticiones) obtener la **moda** (el valor que más se repite)
2. A partir de una lista de números (no ordenada) calcular la **mediana**.

Cálculo de la mediana. 1 Ordenamos los datos de menor a mayor. 2 Si la serie tiene un número impar de medidas la **mediana** es la puntuación central de la misma. 3 Si la serie tiene un número par de puntuaciones la **mediana** es la **media** entre las dos puntuaciones centrales.

3. Dada una lista de números comprobar si está ordenados ascendentemente.

DICCIONARIOS:

1. Simular el comportamiento de un cajero automático. Donde el sistema monetario se almacena una lista. Se teclea un importe y se muestra en un diccionario con el desglose de billetes.

CADENAS:

1. A partir de un texto generar un histograma. Que indique el número de repeticiones de cada carácter.
2. Comprobar si una palabra es un palíndromo: Ejemplos:

Anana, Arañara, Arenera, Arenera, Arepera

FUNCIONES

1. Implementar un función recursiva para la calcular la sucesión de **Fibonacci**:

0, 1, 1, 2, 3, 5, 8, 13, 21 ...

2. Calcular el cociente de dos números (división entera) mediante restas con una función recursiva.

PROGRAMACIÓN ORIENTADA A OBJETOS

1. Convertir el programa del cajero en una clase. Que pueda recibir un sistema monetario y a partir de este calcular el desglose de billetes con un importe dado.

2. Crear una clase que represente un punto en el plano, vendrá definido por sus dos coordenadas x e y. Tendrá que cumplir todas estas particularidades:

a. Se podrá crear un punto a partir de otro punto, a partir de dos coordenadas o crearlo a 0.

b. Métodos para sumar y restar puntos, pero no modificar el punto. La suma o diferencia se devolverá en un punto nuevo.

c. Desplazar un punto a partir de un escalar.

d. Calcular la distancia entre dos puntos: p1 (x1, y1) y p2(x2, y2) $d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$

e. Hacer un método que devuelva el punto como un String con este formato: (8.5, -9)

f. Hacer otra clase a parte que defina dos o tres puntos y pruebe todos los métodos.

g. Sobrecarga de operadores.

3. Diseñar la factura de un operador de telefonía. Donde se registran los SMS (fecha, numero destino, hora, tarifa), las llamadas (fecha, hora, numero destino, tarifa y tiempo de conexión en HORAS). y los datos del usuario. Considerar la posibilidad de que se efectúen llamadas internacionales donde se puede añadir una cuota de roaming. Los importes se calcularán en base a unas tarifas, donde el SMS se tarifica por unidad, la llamada por SG. y a la llamada internacional se le añade una cuota de roaming.

En la factura se muestra la fecha, el número de factura, datos del clientes y el desglose del consumo.

Se mostrará la siguiente información:

```

Numero: 16/0001
Fecha: 30/09/2016
Cliente: 56.777.888G Juan Perez 600998833

1/09/2016 14:15 808585585 0.05 --> 0.05 €
10/09/2016 08:09 608585445 0.05 --> 0.05 €
12/09/2016 12:58 608564585 0.05 --> 0.05 €
19/09/2016 14:15 208585585 0.05 --> 0.05 €
9/09/2016 14:15 208585585 0.025 0.023 h. --> 2.07 €
19/09/2016 4:15 208585585 0.025 0.08 h. --> 7.2 €
29/09/2016 16:15 208555585 0.025 0.12 h. --> 10.8 €
2/09/2016 09:15 208555585 0.025 0.12 h. R: 1 --> 11.8 €

Base imponible: 32.07 €
IVA: 6.73 €
TOTAL: 38.8 €

```

4. Modificar el código anterior, cada cliente dispone de su tarifa donde se indica los precios.

Tarifa:

- precio SMS
- precio SG en una llamada nacional
- Cuota Roaming en una llamada internacional.

PROGRAMACIÓN FUNCIONAL:

1. Utilizando **List Compresion** generar:

- Una lista con los 30 primeros números de la sucesión de fibonacci.
- Dada dos listas generar todas las posibles combinaciones.

```

L1: [3, 2, 1]
L2: [5, 6, 4]
LL: [(3, 5), (3, 6), (3, 4), (2, 5), (2, 6), (2, 4), (1, 5), (1, 6), (1, 4)]

```
- Cargar en una lista el abecedario utilizando la tabla Ascii y la función **chr(num)**.
- A partir de un texto cargar una lista con las palabras de 5 letras.

2. Implementar la función recursiva $\text{Siguiente}([1,2,5]) \rightarrow [2,3,6]$. Que devuelva la lista con los siguientes números.

3. Una función que sume los elementos de dos listas.

4. Funciones Lambda.

- $\text{menor}(x,y) \rightarrow$ Devuelve el menor de dos números.

5. Con map y filter:

- Con **map**:
Aplicar dos funciones, simultáneamente (utilizar una lista con las funciones).
Por ejemplo calcular el cuadrado y el cubo de una lista de números. Utilizar funciones lambda para los cálculos.
- Con **filter**: De la lista de cubo obtener con filter los que son múltiplos de 8. Utilizar funciones lambda.

GENERADORES:

1. Escribir y probar un generador que convierta un numero expresado en binario a decimal.

Por ejemplo: 10011 → 1 + 2 + 0 + 0 + 16 = 19

El generador irá generando las potencias de 2 para cada dígito binario.

DECORADOR:

1. Hacer una prueba colocando varios decoradores en una función.

EXCEPCIONES:

1. Tenemos un programa que genera números aleatorios dentro de un bucle for de 1 a 32000, cada vuelta del bucle se generan dos números aleatorios (de 1 a 100 y se resta. Y se realiza la siguiente operación: 12345 / resta, tenemos que capturar la posible división por cero. Los distintos cocientes de 12345 / resta se imprimen por pantalla y cuando salta la exception se contabiliza, una vez que termine el programa se imprime el número de divisiones por 0.

2. Diseñar excepciones personalizadas para el cajero automático. Por ejemplo, importe que no sea múltiplo de 10.

MÓDULOS Y PAQUETES:

1. A partir de las clases de la generación de facturas 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. A partir de un fichero de texto grande, se pide al usuario el número de particiones y se divide este en N particiones. Implementarlo con POO. El programa genera un informe similar a este: (La última partición llevará el resto si no son exactas las particiones).

```
Informe:
Fichero:  Contenido.txt
NumChars Total: 5668
Particiones:  10
NumChars Particion: 566

Generando ficheros ...
Contenido_0.txt
Contenido_1.txt
Contenido_2.txt
Contenido_3.txt
Contenido_4.txt
Contenido_5.txt
Contenido_6.txt
Contenido_7.txt
Contenido_8.txt
Contenido_9.txt
```

EXPRESIONES REGULARES:

1. Comprobar la validez del siguiente código:

Palabra COD o S/N, un guión bajo, 3 vocales mayúsculas, guión bajo y 6 números que no pueden empezar por 0.

COD_AEE_800959

S/N_UOO_958474

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. Pruebas: propiedades, ejecutar SQL, ...
2. Implementar una clase TerremotoBBDD que importe un fichero CSV a la tabla de Terremotos. Utilizar la BBDD terremotos.dat.
3. 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 Punto para probar los métodos. Por ejemplo, sumar dos Puntos.
3. Aplicar pruebas tipo unittest a la clase Punto u otra con preparación y destrucción del contexto.