**EJERCICIOS DE PYTHON II** DE JAVIER ESCALADA (EXAMENES) CURSO 2019-2020.

**Ejercicio** 1. **Escribe una función** que, dada una matriz tridimensional cuadrada, tome todos los valores de dicha matriz en el triángulo superior. Para borrar los valores anteriores, se les asignará 0. Esta función tiene que asegurarse de que la matriz recibida, es bidimensional y cuadrada. En caso de que esta condición no se cumpla, tiene que devolver la lista vacía. Posteriormente, escribe un programa que pruebe dicha función. A continuación se muestran varios ejemplos.

|  |
| --- |
| >>> dejar \_solo\_triangulo\_superior ( [ [1 ,1 ,1] , [2 ,2 ,2] , [3 ,3 ,3] ])  ( [1, 1, 1], [0, 2, 2] , [0, 0, 3]  >>> dejar\_solo\_triangulo\_superior ( [ [ 1] , [2, 2, 2] , [3, 3, 3] ] )  [ ]  >>> dejar\_solo\_triangulo\_superior ( [ [1, 2, 3] ] )  [ ]  >>> dejar\_solo\_triangulo\_superior ( [ [4 ,5] ] ) , [2 ,7] ] )]  [ [4 ,5] ] , [0 , 7] ]  >>> dejar\_solo\_triangulo\_superior ( [ [5] ] )  [ [ 5] ] |

**Ejercicio 2. Escribe una función** que, dada **una lista de listas**, devuelva como resultado otra lista de listas, donde la primera lista contenga los primeros elementos de las listas de entrada, la segunda lista contenga los segundos elementos de las listas de entrada, la tercera lista contenga los terceros elementos de las listas de entrada y así sucesivamente hasta terminar de procesar la lista más pequeña de todas las que se introdujeron. Posteriormente, **escribe un programa que pruebe dicha función**. A continuación, se muestran varios ejemplos.

|  |
| --- |
| >>> entrelaza\_lista ([1, 2, 3], [4, 5, 6])  ( [1, 4 ], [2, 5 ], [3, 6 ]  >>> entrelaza\_listas ( ( [7, 8], (100, 200, 300, 400, 500 ] ])  [ [ 7, 100], [, 200]  >>> entrelaza listas ( [ [1, 1, 1], [1] )  [ ]  >>> entrelaza\_listas ( [ [ 1, 2, 3 ] ])  [ 1], [ 2], [ 3 ]  >>> entrelaza\_listas ( [ [1, 2,], [7 ,8, 9, ],[ 0, 0, 0 ] ])  ( [ [ 1, 7, 0], [2, 8, 0 ] ) |

**Ejercicio 3. Escribe una función** que, dada una cadena de caracteres que contiene números enteros separados por puntos y comas (;), **devuelva una lista** con los números enteros positivos que se encuentran dentro de ella. Posteriormente, **escribe un** **programa que pruebe dicha función**. A continuación, se muestran varios ejemplos.

|  |
| --- |
| >>> extrae\_numeros\_positivos ( “1 ; -2; 3; -4; -5; 6”)  [1, 3 , 6]  >>> extrae\_numeros\_positivos ( “ -1”)  [ ]  >>> extrae\_numeros\_positivos ( “-1 , 0”)  [ ]  >>> extrae\_numeros\_positivos ( “ “ )  [ ] |

**Ejercicio 4. Escribe una función** que, dados una cadena de caracteres y un tamaño **n** entero positivo, devuelva una lista con los n-gramas que no contengan espacios que se pueden extraer de la lista de caracteres. Si el tamaño es 3, la función tiene que extraer 3-gramas, es decir, subcadenas de tamaño 3. Si el tamaño es 5, la función debe devolver 5-grams, es decir, subcadenas de tamaño 5. La función debe comproba que la cadena de caracteres no esté vacía, en caso contrario devolverá la lista vacía. Posteriormente, escribe un programa que pruebe dicha función. A continuación, se muestran varios ejemplos.

|  |
| --- |
| >>> extraer n\_gramas (“ “, 3)  [ ]  >>> extraer n\_gramas (“hola”, 0)  [ ]  >>> extraer n\_gramas (“ hola mundo”, 3 )  ( ‘hol’, ‘ola’, ‘mun’, ‘und’, ‘ndo’)  >>> extraer n\_gramas ( “hola mundo”, 2)  [‘ho’, ‘ol’, ‘la’, ‘mu’, ‘un’, ‘nd’, ‘do’]  >>> extrae\_ngramas (“hola mundo”, 4)  [‘hola’, ‘mund’, ‘undo’]  >>> extrae\_ngramas (“hola mundo”, 5 )  [‘ mundo']  >>> extrae\_ngramas (“hola mundo”), 6)  [ ] |

**Ejercicio 1.**

**Escribe una función** que dada una matriz tridimensional cuadrada, borre todos los valores impares, dejando solo los pares. Para borrar los valores anteriores, se les asignará 0. Esta función tiene que asegurarse de que la matriz recibida, es bidimensional y cuadrada. En caso de que esta condición no se cumpla, tiene que devolver la lista vacía. Posteriormente, escribe un programa que pruebe dicha función. A continuación se muestran varios ejemplos.

|  |
| --- |
| >>> dejar\_solo\_pares ([1, 1, 1], [2, 2, 2], [3, 3, 3]  [0, 0, 0] , [2, 2, 2], [0, 0, 0]  >>> dejar\_solo\_pares ([ [1], [2, 2, 2], [3, 3, 3]  [ ]  >>> dejar\_solo\_pares ( [ 1,2,3 ] )  [ ]  >>> dejar\_solo\_pares ( [ [1,2,3 ] , [4, 5 ,6] ] )  [ ]  >>> dejar\_solo\_pares ( [4 ,5] , [ 2, 7] )  ( [4, 0] , [2, 0]  >>> dejar\_solo\_pares ( [ [5 ] )  [ [ 0 ] ] |

**Ejercicio 2. Escribe una función** que, dado un tamaño entero positivo, devuelva la matriz identidad de dicho tamaño, es decir la matriz que tiene unos en la diagonal principal y ceros en todas las demás posiciones. La función tiene que comprobar que el tamaño se efectivamente un entero positivo, en caso contrario tiene que devolver la lista vacía. Posteriormente, escribe un programa que pruebe dicha función. A continuación, se muestran ejemplos.

|  |
| --- |
| >>> Crear\_ matriz\_ identidad [-1]  [ ]  >>> Crear\_matriz\_identidad [0]  [ ]  >>> Crear\_matriz\_identidad [2]  [ [1 , 0] ,[0 ,1] ]  >>> Crear\_matrirz\_identidad [ 3]  [ [1, 0, 0] ,[0, 1, 0 ] , [0, 0, 1 ] ]  >>> Crear matriz identidad [ 4 ]  [ [ 1 , 0, 0,0] , [0 ,1 , 0 , 0] , [ 0 ,0 , 1, 0], [0 ,0, 0, 1]]  [ [ 1, 0 , 0, 0] ] |

**Ejercicio 3.**

**Escribe** **una función** que dados:

* Una cadena de caracteres que contiene números reales separados por puntos y comas.

( **;** )

* Un valor mínimo
* Y un valor máximo.

Devuelve otra cadena de caracteres que contiene los reales que están en el rango [mínimo, máximo] separados por puntos y comas (;) posteriormente, escribe un programa que pruebe dicha función. A continuación se muestran varios ejemplos.

|  |
| --- |
| >>> filtrar \_reales ( …….. )  “ “  >>> filtrar\_reales (“1.3 **;** 5.9 **;** 20. 5 **;** ..” **; …..** )  >>> filtrar\_reales(“1.0”,0 ,2  “1.0” |

**Ejercicio 4. Escribe una función** que dada una cadena de caracteres, compruebe si esa cadena es un palíndromo y devuelva True o False. Posteriormente, escribe un programa que pruebe dicha función. A continuación, se muestran varios ejemplos.

|  |
| --- |
| >>> es\_palindromo [“ “]  True  >>> es\_palindromo [“a” ]  True  >>> es\_palindromo [“se es”]  True  >>> es\_palindromo [“somos”]  True  >>> es\_palindromo [“anna”]  True  >>> es\_palindromo [“año”]  False  >>> es\_palindromo [“sometemos”]  True  >>> es\_palindromo [“reconocer”]  True  >>>> es\_palindromo [“Isaac no ronca asi”  True  >>> es\_palindromo [“lavan esa base naval”]  True  >>>>es\_palindromo [“no subas abuson”]  True |

**EVALUACIÓN EXTRAORDINARIA DE MAYO. CURSO 2019-2020**

**Profesor: Javier Escalada Gómez.**

**Python I:**

**Programa 1**. **Problema 6 práctico de mayo.**

Escribe un programa que pida por teclado dos números inicio y fin que sean enteros positivos. A continuación, suma los números que sean divisibles por 2 o por 3 y resta los que sean divisibles por 6 desde inicio hasta fin (no incluido).

**Ejemplo de ejecución del programa**

|  |
| --- |
| Escribe el valor de inicio > **1**  Escribe el valor de fin > **10**  El resultado es : **20** |

**Programa 2. Problema 4 práctico de mayo.**

Escribe un programa que pida dos números base y altura que sean enteros mayores o iguales que 2 y que imprima un rectángulo con esas dimensiones.

**Ejemplo de ejecución del programa**

|  |
| --- |
| Escribe el valor de base > **10**  Escribe el valor de altura >  **4**  **~~i~~ - - - - - - - -**  ~~i~~  i i  i i  ~~i~~ **- - - - - - - -** ~~i~~ |

**Python I:**

**Programa 1. Problema 1 práctico de mayo.**

Escribe un programa que pida por teclado dos números **fin** y **divisor** que sean enteros positivos. A continuación, suma los números que sean divisibles por **divisor** desde **0 hasta** fin (no icluido). Para finalizar muestra el resultado de la suma.

**Ejemplo de ejecución del programa**

|  |
| --- |
| Escribe el valor de fin > **9**  Escribe el valor del divisor > **3**  El resultado es: **9** |

**Programa 2. Problema 2 práctico de mayo.**

Escribe un programa que pida números enteros y que calcule el divisor más grande posible para ambos números, es decir, el Máximo Común Divisor.

**Ejemplo de ejecución del programa**

|  |
| --- |
| Escribe el primer numero> **100**  Escribe el primer numero > **55**  5 es el máximo común divisor de 100 y 55 |

**Python II: (Funciones, lista y cadenas)**

**Programa 1. Problema 3 práctico de mayo (Matrices)**

Escribe una función que, dada una matriz bidimensional cuadrada, borre todos los valores impares, dejando solo los pares. Para borrar los valores anteriores, se les asigna 0. Esta función tiene que asegurarse de que la matriz recibida, es bidimensional y cuadrada. En caso de que esta condición no se cumpla, tiene que devolver la lista vacía. Posteriormente, escribe un programa que pruebe dicha función. A continuación, se muestran varios ejemplos.

|  |
| --- |
| >>> dejar\_solo\_pares ( [ [ 1, 1,1 ] ] ) , [2, 2, 2] , [3, 3, 3] ] )  [ [ 0, 0, 0 ] ] , [ 2, 2, 2 ] , [0, 0, 0]  >>> dejar\_soilo\_pares ( [ [1] ] , [2, 2, 2], [3, 3, 3] ] )  [ ]  >>> dejar\_solo\_pares ( [ [1, 2, 3 ,] ] )  [ ]  >>> dejar\_solo\_pares ( [ [ 1, 2, 3] ] ) , [4, 5, 6] ] )  [ ]  >>> dejar\_solo\_pares ( [ [4, 5] ] , [2, 7] ] )  [ [4, 0] , [2, 0] ]  >>> dejar\_solo\_pares ( [ [ 5 ] ] )  [ [ 0] ] |

**Programa 2. Problema 5 práctico de mayo (listas/cadenas)**

Escribe una función que dada una cadena de caracteres que contiene números enteros separados por puntos y comas [ **;** ] devuelva una lista con los números enteros positivos que se encuentran dentro de ella.Posteriormente, escribe un programa que pruebe dicha función. A continuación, se muestran varios ejemplos.

|  |
| --- |
| >>> Extrae\_ números \_positivos [“1 ; -2; 3 ;-4 ;-5;6”]  [1, 3, 6]  >>> Extrae\_números\_positivos [“-1”]  [ ]  >>> Extrae\_números\_positivos[“-1;0”]  [ ]  >>> Extrae\_números\_positivos [“ “]  [ ] |

**Examen de EXCEL.**

En el fichero Excel se pueden ver datos de población censada este año para cada municipio del Principado de Asturias.

**Parte 1:**

1). Calcula en la celda **k2** el total de hombres.

2). Calcula en la celda **k3** el total de mujeres.

3). Calcula en la celda **D** la población de cada municipio.

4). Calcula en la celda **k4** el total de la población

5). Calcula en la columna **E** el porcentaje de población con respecto a la población total.

6). Formatea en la columna **F** para cada municipio una función que muestre **“Si”** cuando la población masculina y femenina de ese municipio sea **mayor que** la media geométrica menos 1000 y **menor que** la media geométrica más 1000 (es decir que el valor esté en el intervalo abierto (media.geom**-**1000, media.geom**+**1000)) y **“No”** en caso contrario.

**Recuerda:** No es lo mismo la media geométrica que la media aritmética o promedio.

**Parte 2:**

1. Calcula en la celda **k7** la suma de los hombres de los municipios cuya población (masculina y femenina) sea mayor o igual que la media geométrica.
2. . Calcula en la celda **k8** la cantidad de municipios en los que la diferencia entre hombres y mujeres (y la diferencia entre mujeres y hombres) es mayor que 100.

**Recuerda:** No hace falta hacer esto cálculos en una fórmula enorme se pueden dividir los cálculos en varias celdas.

**Parte 3:**

En una segunda hoja llamada “Gráfica” crea un gráfico de columnas agrupadas que represente:

1). El municipio.

2). El porcentaje de hombres con respecto a la población total del Principado de Asturias.

3). El porcentaje de mujeres con respecto a la población total del Principado de Asturias de Oviedo, Gijón, Avilés y del resto de municipios agrupados bajo el nombre de Otros.

**Recuerda:** Para calcular los datos necesarios para **hacer** los **gráficos** no se pueden copiar ni escribir a mano los valores, hay que usar referencias entre páginas.

**Examen de ACCESS.**

**Parte 1:**

….. Transformaciones necesarias para que sólo existan relaciones uno-a-uno o uno-a-muchos entre entidades. Identifica las claves primarias (CP) y las claves ajenas (CA) de cada entidad.

Una comunidad autónoma quiere rediseñar su base de datos de hospitales. Para ello quiere tener un registro de cada hospital, cada médico y cada paciente.

De cada una de estas entidades, se desea almacenar la siguiente información:

1. **Hospital:** id, dirección.
2. **Área:** id, nombre y descripción.
3. **Médico:** id, NIF, nombre y apellidos.
4. **Paciente:** id, historial, fecha\_de\_ingreso, NIF, nombre, apellidos, ingresado.

Las relaciones se definen de la siguiente forma:

1. Un hospital está dividido en áreas.
2. Cada médico trabaja en un hospital.
3. Un médico puede estar o no asociado a un área
4. Un hospital puede ingresar a varios pacientes.
5. Un paciente puede ser tratado por varios médicos y un médico puede tratar varios pacientes. No obstante, todo paciente tiene asignado un médico responsable, que se encarga de coordinar todo lo relativo a ese paciente.

**Parte 2:**

Crea la base de datos Hospitales. Accdb a partir de la información anterior, teniendo en cuenta:

1. Todas las relaciones tienen que ser creadas de forma que impidan el borrado de entidades relacionadas.
2. El nombre de un área es obligatorio y único.
3. El historial de un paciente puede tener más de 255 caracteres.
4. El NIF de un paciente y un médico es obligatorio, único y debe tener 9 caracteres.
5. El nombre y los apellidos de un paciente y un médico son obligatorios.
6. La fecha de ingreso es obligatoria y no puede ser anterior a 1 de enero del 2020
7. Un paciente está ingresado hasta que el médico responsable diga lo contrario.
8. Los médicos tienen que estar obligatoriamente asociados a **un** hospital pero **no** necesariamente a **un** área.
9. Los pacientes tienen que estar obligatoriamente asociados a un hospital.
10. Las áreas tienen que estar obligatoriamente asociadas a un hospital.

**Parte 3:**

Mete datos en cada una de las tablas creadas y crea la consulta agregada que quieras con los datos que has metido.