BOLETÍN DE PROBLEMAS DE PROGRAMACIÓN

PROBLEMAS DE CONDICIONALES

PROBLEMA 1

Escriba un programa que permita al usuario jugar a adivinar un número del 1 al 10. El programa generará un número aleatorio, y pedirá al usuario que lo adivine hasta en 3 ocasiones. Si el usuario adivina dicho número se le felicitará y terminará el programa. En caso contrario, se le indicará al usuario si el número que ha de adivinar es mayor o menor que el número que el usuario ha indicado, y se le volverá a dar de nuevo la oportunidad de que lo adivine hasta en 2 ocasiones más.
Necesitarás generar números aleatorios, lee este artículo:
https://www.mclibre.org/consultar/python/lecciones/python-biblioteca-random.html

PROBLEMA 2

Escriba un programa que pida al usuario dos números enteros. Una vez obtenidos los dos números, el programa mostrará al usuario una serie de opciones numeradas (un menú), y esperará que el usuario teclee la opción deseada. Entre las opciones del menú se le deberán ofrecer al usuario las siguientes:

1. Sumar los números.
2. Restar los números.
3. Multiplicar los números.
4. Dividir los números.

Una vez que el usuario haya escogido la opción, se realizará la operación con los números asociada a dicha opción y se ofrecerá el resultado de dicha operación por pantalla. En el caso de que el usuario seleccione una opción que no aparece en el menú se le indicará que la opción es incorrecta. En cualquier caso, el programa terminará.

PROBLEMA 3

Escriba un programa que simule el funcionamiento de un termostato. Para ello, se le pedirá al usuario la temperatura actual, la temperatura deseada por debajo de la cual se activará la calefacción y la temperatura por encima de la cual se activará el aire acondicionado frío. Tras la entrada de datos se mostrará alguno de los siguientes mensajes: 'HACE FRÍO. SE VA A ENCENDER LA CALEFACCIÓN', 'HACE CALOR. SE VA A ENCENDER EL AIRE ACONDICIONADO' o 'LA TEMPERATURA ES LA IDEAL'.

PROBLEMA 4

Escriba un programa que pida al usuario el día, mes (en formato numérico) y año, y le muestre por pantalla dicha fecha indicando el mes por su nombre (por ejemplo, si se indica día 1, mes 8 y año 2020, el programa tiene que devolver el resultado '1 de Agosto de 2020').

PROBLEMA 5

Escriba un programa que indique si el año introducido por un usuario es o no bisiesto. Los años bisiestos son múltiplos de cuatro, sin embargo, los múltiplos de 100 sólo son bisiestos cuando a la vez son múltiplos de 400 (por ejemplo 1800 no es bisiesto, mientras que 2000 si lo es).

PROBLEMA 6

Escriba un programa que pida tres números al usuario e indique cual es el mayor y cual es el menor de los tres números.

PROBLEMA 7

Escriba un programa que escriba la calificación correspondiente a una nota que se le pedirá al usuario que introduzca por teclado, de acuerdo con el siguiente criterio:

- 0 a <5.0 Suspenso
- 5 a <7 Aprobado
- 7 a <9 Notable
- 9 a <10 Sobresaliente
- 10 Matrícula de honor

PROBLEMA 8

Cuatro enteros entre 0 y 100 representan las puntuaciones de un estudiante de una universidad americana. Escriba un programa que encuentre la media de estas puntuaciones y que visualice la calificación según se establece en el siguiente cuadro:

Media de puntuación	Calificación
90-100	A
80-89	B
70-79	C
60-69	D
0-59	E

PROBLEMA 9

Escriba un programa que calcule el número de días de un mes, dado el valor numérico del mismo. Por ejemplo, el número 5 corresponderá a mayo. No hay que tener en cuenta los años bisiestos.

PROBLEMA 10

Escriba un programa que pida al usuario un número de hasta tres cifras y muestre un mensaje indicando si tiene 1, 2, o 3 cifras.

PROBLEMA 11

De un operario se conoce su sueldo y los años de antigüedad. Escriba un programa que lea los datos de un trabajador, en concreto su sueldo actual y su antigüedad. A partir de estos datos el programa deberá calcular el nuevo sueldo que le corresponde según los siguientes criterios:

a) Si el sueldo es inferior a 1500 y su antigüedad es igual o superior a 10 años, otorgarle un aumento del 20 %, mostrar el sueldo a pagar.
b) Si el sueldo es inferior a 1500 pero su antigüedad es menor a 10 años, otorgarle un aumento de 5 %.
c) Si el sueldo es mayor o igual a 1500 mostrar el sueldo en la página sin cambios.

PROBLEMA 12

Escriba un programa que resuelva la ecuación cuadrática (ax2 + bx + c =0) mostrando sus soluciones o indicando que no tiene solución cuando así sea.

PROBLEMAS DE BUCLES

PROBLEMA 13

Escribir una aplicación que visualice el siguiente triángulo isósceles:

```
*
***
*****
*******
*********
```

PROBLEMA 14

La constante pi (3.141592) se utiliza en matemáticas; un método sencillo de calcular su valor es:

$$\pi = 2 * \frac{2}{1} * \frac{2}{3} * \frac{4}{3} * \frac{4}{5} * \frac{6}{5} * \frac{6}{7} * \frac{8}{7} * \frac{8}{9} * \ldots * \frac{2n}{2n-1} * \frac{2n}{2n+1}$$

Escriba un programa que efectúe este cálculo con un número de términos *n* especificados por el usuario.

PROBLEMA 15

Escriba un programa que calcule y muestre por pantalla la descomposición en factores de un número introducido por el usuario.

PROBLEMA 16

Escriba un programa que calcule y muestre por pantalla la suma de la serie 1/1 + ½ + 1/3 + … + 1/(n-1) + 1/n. El valor *n* será introducido por el usuario del programa.

PROBLEMA 17

Escriba un programa que imprima la lista de números del uno al cien a excepción de aquellos números que sean múltiplos de 2 o de 3.

PROBLEMA 18

Escriba un programa que, dado un número entero por parte del usuario, nos indique si dicho número es o no es primo.

PROBLEMA 19

Escriba un programa que pida la edad de 10 personas y muestre cuantas de ellas son menores de edad, cuantas mayores y la edad media del grupo.

PROBLEMA 20

Escriba un programa que calcule la suma de todos los números que sean divisible por 3 y por 7 entre el número 1 y el 1000.

PROBLEMA 21

Escriba un programa donde el usuario ingrese 10 números enteros. Tras esto se mostrará la suma de los números positivos y la media de los negativos. En caso de que no se introduzcan números positivos o negativos, en lugar del valor calculado se mostrará un mensaje indicando de ello: 'No hay números positivos' o 'No hay números negativos'.

PROBLEMA 22

Escriba un programa que pida caracteres y que imprima 'VOCAL' si se trata de una vocal y 'NO VOCAL' en caso contrario. La entrada de datos terminará al introducir un espacio y tras esto se mostrará el número total de vocales y el número total de no vocales de la entrada de datos.

PROBLEMA 23

Escriba un programa que muestre por pantalla todos los números divisibles por 7 entre dos números introducidos por el usuario.

PROBLEMA 24

Escriba un programa que dibuje una matriz cuadrada donde cada valor sea la suma de su fila y columna. Se pedirá al usuario que indique el número de filas de la matriz.
Ejemplo de la matriz 3x3:

2	3	4
3	4	5
4	5	6

PROBLEMAS DE LISTAS Y TUPLAS

PROBLEMA 25

Ejercicios de recorrido y búsqueda

1) Algoritmo que sume todos los elementos de un vector t de N reales.
2) Algoritmo que calcule el producto escalar de dos vectores t1 y t2 de N reales.
3) Algoritmo que calcule el máximo y el mínimo de un vector de N reales.
4) Algoritmo que invierta un vector t de N reales.
5) Algoritmo que sume los elementos de la fila k de una matriz m de NxN reales.
6) Algoritmo que sume dos matrices m1 y m2, devolviendo el resultado en m3. m1 y m2 son matrices de NxN reales.
7) Algoritmo que localice un elemento x en un vector t de N reales, devolviendo la posición p y una variable lógica exito. (Búsqueda lineal).
8) Algoritmo que indique si un vector t de N enteros está ordenado.
9) Algoritmo que localice un elemento x en una matriz t de N reales, devolviendo la posición (p,q) y una variable lógica exito. (Búsqueda lineal).
10) Algoritmo que calcula la traspuesta de una matriz m de NxN reales respecto de la diagonal principal.
11) Se define la diagonal secundaria como los elementos que ocupan las posiciones (1,N), (2,N-1), (3,N-2) ... (N,1). Algoritmo que sume los elementos de la diagonal secundaria de una matriz m de NxN reales.
12) Algoritmo que multiplique dos matrices m1(MxL) y m2(LxN), devolviendo el resultado en m3(MxN).
13) Algoritmo que ordene un vector t de N reales por el método de selección. (Se localiza el menor y se coloca en la primera posición; así, sucesivamente).

PROBLEMA 26

Escriba un programa que utilizando listas pida al usuario los 9 números necesarios para conformar una matriz de 3x3. Tras esto el programa mostrará la matriz introducida y también la matriz al cuadrado.

PROBLEMA 27

Escriba un programa que pida la introducción de seis números al usuario, los tres primeros se almacenarán en una lista y se ordenarán, los siguientes se almacenarán en otra lista y también se ordenarán. Se mostrará por pantalla el contenido de la primera lista, el contenido de la segunda lista y una tupla conformada por los seis elementos en el orden que se obtenga de concatenar la primera y la segunda lista.

PROBLEMA 28

Escriba un programa que pida la introducción de una lista de palabras. Al terminar la introducción de datos se mostrará por pantalla la lista de palabras introducidas. A continuación, se creará una nueva lista que solo contenga las palabras que contenga las palabras que comiencen por vocal y una tupla con las palabras que comiencen por consonante. Ambas estructuras de datos no contendrán palabras repetidas y se mostrarán por pantalla antes de finalizar el programa.

PROBLEMA 29

Escriba un programa que pida la introducción de cinco números al usuario. Al terminar la introducción de datos se mostrará por pantalla la tupla creada con los cinco números. A continuación, se creará una lista que contendrá tuplas formadas por cada número y la palabra 'PAR' o 'IMPAR'. Dicha lista se mostrará por pantalla antes de finalizar el programa.

Ejemplo de salidas del programa:

```
( 22 , 12, 17, 8, 51 )
```

```
[ (22, 'PAR'), (12, 'PAR'), (17, 'IMPAR), (8, 'PAR'), (51, IMPAR)]
```

PROBLEMA 30

Escriba un programa que pida la introducción de los datos de 3 equipos de fútbol y sus resultados en un torneo: partidos ganados (3 puntos), partidos perdidos (0 puntos) y partidos empatados (1 punto). Al terminar la introducción de datos se mostrará por pantalla una lista de tuplas con los datos de los tres equipos. A continuación, se generará una nueva lista formada por tuplas que contendrán dos elementos: el nombre del equipo y los puntos obtenidos en el torneo. Antes de finalizar el programa se mostrará por pantalla esta última lista ordenada por puntuación descendente.

Ejemplo de salida del programa:

```
[ ('Boquerones', 2, 6, 2)  , ('Verdolagas',  4, 2, 4) , ('Palanganas', 3, 4, 3) ]
```

```
[ ('Verdolagas', 16) , ('Palanganas', 12), ('Boquerones', 8) ]
```

PROBLEMA 31

Escriba un programa que permita crear una lista de palabras y que, a continuación, muestre este menú de opciones:

1. Contar: Me pide una cadena, y me dice cuántas veces aparece en la lista.
2. Modificar: Me pide una cadena, y otra cadena a modificar, y modifica todas las apariciones de la primera por la segunda en la lista.
3. Eliminar: Me pide una cadena, y la elimina de la lista.
4. Terminar

Tras la ejecución de una opción volverá aparecer el menú de opciones y justo debajo, la lista de palabras en su estado actual. El programa no finalizará hasta que el usuario seleccione la opción de terminar.

PROBLEMA 32

Escriba un programa que solicite los datos de movimientos de una cuenta bancaria. Cada movimiento estará formado por un concepto y un número que podrá ser positivo (ingresos) o negativo (gatos). Crear una lista de tuplas formada por los movimientos que sean ingresos y otra lista de tuplas formada por los movimientos que sean gastos. Mostrar por pantalla las listas creadas, junto con el total de ingresos y el total de gastos.

PROBLEMA 33

Escriba un programa que solicite los nombres de cinco trabajadores y el día de la semana que estarán de guardia (lunes a viernes). Cuando se introduzca el nombre de un trabajador que ya esté registrado no se tomará en cuenta esa entrada de datos. Tampoco se tomará en cuenta cuando se introduzca un día que ya esté registrado o que no sea válido (lunes a viernes). Utilizar como estructuras de datos solo listas y tuplas para realizar el programa. El resultado final se mostrará por pantalla como una lista de tuplas, donde cada trabajador solo trabajará un día de la semana.

```
Ejemplo de salida del programa:

[ (‘Pablo’, ‘lunes’), (‘David’, ‘martes’), (‘María’, ‘miércoles’), (‘Marta’, ‘jueves’), (‘Juan’, ‘viernes’) ]
```

PROBLEMAS DE DICCIONARIOS

PROBLEMA 34

Escriba un programa que solicite los nombres de cinco trabajadores y el día de la semana que estarán de guardia (lunes a viernes). Cuando se introduzca el nombre de un trabajador que ya esté registrado no se tomará en cuenta esa entrada de datos. Tampoco se tomará en cuenta cuando se introduzca un día que ya esté registrado o que no sea válido (lunes a viernes). Utilizar como estructura de datos un diccionario. El resultado final se mostrará por pantalla como un diccionario donde cada trabajador solo trabajará un día de la semana.

Ejemplo de salida del programa:

```
{'Pablo': 'lunes', 'David': 'martes', 'María': 'miércoles', 'Marta': 'jueves', 'Juan': 'viernes'}
```

PROBLEMA 35

Escriba un programa que permita el control de asistencia de los empleados a una empresa. La entrada de datos es una lista de nombres de empleado que el sistema va generando según los empleados pasan por el arco de entrada, por ejemplo: [‘Pedro’, ‘María’, ‘Pedro’, ‘Juan’, ‘María’, ‘Marta’] . Esta lista se debe procesar y registrar en un diccionario de la siguiente manera:

- Si es la primera vez que lo detecta, entonces es que está entrando en el edificio y por tanto almacenaríamos el valor True asociado al nombre del trabajador.
- Si detecta el paso en una posterior ocasión, negará el valor booleano asociado al nombre del trabajador. De esta forma registraríamos una salida del edificio (True → False) o una nueva entrada (False → True)

El programa al finalizar mostrará el contenido del diccionario, junto con una lista de las personas que están en el edificio.

Ejemplo de salida del programa:

```
{'Pedro': False, 'Maria': False, 'Juan': True, 'Marta': True}

[ ‘Juan’, ‘Marta’ ]
```

PROBLEMA 36

Escriba un programa que pida al usuario que introduzca un texto. Como resultado de procesar el texto se construirá y mostrará por pantalla un diccionario que contenga el número de veces que aparece cada palabra en el texto. Se deben obviar los signos de puntuación del texto como los puntos, las comas, los dos puntos, los signos de exclamación e interrogación. Se considerará que los espacios separan las palabras del texto.

PROBLEMA 37

Escriba un programa que permita llevar la contabilidad de una empresa. La empresa identifica las cuentas contables con un nombre: ‘Ventas’, ‘Compras’, ‘Efectivo’, ‘Mercadería’, … Cada vez que se produce un movimiento en una cuenta contable se anota el importe, que puede ser negativo o positivo, agregándolo a una tupla que contendrá los movimientos de dicha cuenta.

La entrada de datos será una sucesión de nombres de cuenta e importe del movimiento a registrar. Dicha entrada de datos finalizará al escribir ‘x’ como nombre de cuenta.

Al finalizar se mostrará el diccionario creado, junto con la cuenta con un saldo mayor y la cuenta con un saldo menor.

Ejemplo de salida del programa:

```
{'Ventas': (100, -50, 25, 75), 'Compras': (60, -30, -10), 'Efectivo': (20, 10, 5),
'Mercadería': (-10, -20, 80)}

Cuenta con el saldo mayor: (‘Ventas’, 150)

Cuenta con el saldo menor: (‘Compras’, 20)
```

PROBLEMAS DE FUNCIONES

PROBLEMA 38

Escriba un programa que contenga una función de nombre TextoCentrado, que reciba como parámetro un texto y lo escriba centrado en pantalla (suponiendo una anchura de 70 columnas), también subrayará el mensaje utilizando el carácter guion bajo '_'.

Se incluirá el código necesario que muestre el resultado de invocar la función con diferentes textos, todos con menos de 70 caracteres.

PROBLEMA 39

Escriba un programa que pida dos números enteros al usuario y diga si alguno de ellos es múltiplo del otro. Para ello crea una función MultiploDe que reciba los dos números, y devuelve un valor True si el primero es múltiplo del segundo y el valor False en caso contrario.

PROBLEMA 40

Escriba un programa con un menú donde se pueda elegir la opción de convertir a segundos, convertir a horas, minutos y segundos o salir del programa.

Para ello escribir dos funciones que se especifican a continuación:

- Función CalcularSegundos, que toma tres parámetros de entrada: horas, minutos y segundos. Devuelve la cantidad de segundos correspondientes a los datos recibidos como parámetros.
- Función CalcularTiempo, que toma como parámetro una cantidad de segundos y devuelve el resultado como una tupla que contiene la cantidad de horas, minutos y segundos correspondientes al parámetro de entrada.

PROBLEMA 41

Escriba un programa que calcule el día juliano a partir de una fecha introducida por teclado (día, mes y año). Para ello cree las funciones:

- Función LeerFecha, pide al usuario la introducción de una fecha solicitando el día, el mes y el año. Devuelve una tupla con los tres valores.
- Función DiaJuliano, que tomando como parámetros el día, el mes y el año devuelva el día juliano correspondiente.

PROBLEMAS DE FICHEROS

PROBLEMA 42

Escriba un programa que pida al usuario los nombres de dos ficheros de texto y agregue el contenido del segundo fichero al primero de ellos.

PROBLEMA 43

Escriba un programa que pida al usuario el nombre de un fichero de texto a ordenar, el número de la columna del fichero que se toma para la ordenación y el nombre del fichero que contendrá la salida ordenada.

Por ejemplo, dado el siguiente fichero si se pide ordenar por la columna 14 la ordenación se debe realizar tomando en cuenta una cadena de texto desde dicha columna hasta el final de la línea. En el ejemplo el fichero resultado sería un fichero ordenado por el campo apellidos.

```
Aitor       ;Menta          ;44509383L
Jacinto     ;Mate           ;33456356Z
María       ;Uncafe         ;47892843K
Elena       ;Nito           ;56783937K
```

PROBLEMA 44

Escriba un programa que pida al usuario los nombres de un fichero de texto de entrada y el nombre de un fichero para almacenar la salida del programa. El programa procesará el fichero de texto de entrada, contando la frecuencia de cada palabra que aparezca en el texto, se recomienda para ello utilizar un diccionario. El fichero de salida contendrá una línea de texto por cada palabra del texto original y dicha línea tendrá el formato: *palabra,ocurrencias*.

- No se debe hacer distinción de mayúsculas y minúsculas. Es decir, para el programa “De” y “de” son la misma palabra.
- Se deben obviar los signos de puntuación del texto como los puntos, las comas, los dos puntos, los signos de exclamación e interrogación.
- Se considerará que los espacios separan las palabras del texto.

Ejemplo de fichero de salida:

```
DE,7
PERSONA,4
EL,5
MESA,3
```

PROBLEMA 45

Escriba un programa que pida al usuario los nombres de un fichero de texto de entrada y el nombre de un fichero para almacenar la salida del programa. También se pedirá al usuario un número menor que 26 que indicará el desplazamiento para la "encriptación" que se va a realizar.

El proceso consistirá en "encriptar" el texto del fichero de entrada cambiando cada carácter del texto de entrada por el que le corresponda en el alfabeto usando el valor del desplazamiento introducido. Todo carácter no alfabético pasará al fichero de salida directamente, sin "encriptación". De esta forma, tomando como desplazamiento el número 3 se tendrían las siguientes sustituciones: A → D, B → E, C→F, ..., W→Z, X→A, Y→B y Z→C

Ejemplo de fichero de entrada y salida, tomando un desplazamiento igual a 1:

Fichero de entrada:

```
Hola,
mundo.
```

Fichero de salida:

```
Ipmb,
nuñep.
```

PROBLEMA 46

Escriba un programa que pida al usuario el nombre de un fichero de texto. El fichero de texto contendrá números reales separados por el carácter punto y coma, pudiendo tener varias líneas con este formato. Como resultado de procesar el fichero se mostrarán por pantalla los siguientes valores:

- Valor máximo
- Valor mínimo
- Suma de todos los valores
- Media de todos los valores

Ejemplo de fichero de entrada:

```
1.25;2.75;0.001;21
3.45;4.52;12;-3.834
7.1;-5.2;4.32;8.27
```

PROBLEMA 47

Escriba un programa que pida al usuario el nombre de tres ficheros de texto, los dos primeros con datos de entrada y el tercero para almacenar la salida.

- El primer fichero contiene simplemente texto.
- El segundo fichero contendrá las sustituciones de caracteres a realizar, una por cada línea.
- El tercer fichero contendrá el texto del primer fichero, pero sustituyendo los caracteres que se especifiquen en el segundo fichero.

Ejemplo de ficheros de entrada y salida:

Ejemplo del primer fichero de entrada:

```
En un lugar
de la Mancha
de cuyo nombre
```

Ejemplo del segundo fichero de entrada:

```
a;o
e;o
m;p
```

Ejemplo del fichero de salida:

```
On un lugor
do lo poncho
do cuyo nombro
```

PROBLEMAS DE BASES DE DATOS

PROBLEMA 48

En aula virtual se proporciona un fichero SQLite (“equipo.db”) conteniendo la base de datos del equipo de fútbol con la siguiente estructura:

```
CREATE TABLE jugadores(
dni TEXT PRIMARY KEY,
fecha_baja TEXT NULL,
peso INTEGER NOT NULL,
altura REAL NOT NULL )
```

1. Se pide realizar la función llamada ***altura_media,*** que acceda a la base de datos y devuelva la media de altura de los futbolistas en activo (fecha_baja IS null).
2. Se pide realizar la función **jugadores_altos,** que acceda a la base de datos y genere un fichero de texto “jugadores.txt” que contenga una línea con el DNI de cada jugador en activo que supere la altura media.
3. Se pide realizar un **programa principal** que invoque las dos funciones anteriores a modo de demostración.

PROBLEMA 49

Se proporciona un fichero SQLite (“equipo.db”) conteniendo la base de datos del equipo de fútbol con la siguiente estructura:

```
CREATE TABLE jugadores(
dni TEXT PRIMARY KEY,
fecha_baja TEXT NULL,
peso INTEGER NOT NULL,
altura REAL NOT NULL )
```

1. Se pide realizar la función llamada ***peso_medio*** que acceda a la base de datos y devuelva la media de peso de los futbolistas que **no** estén en activo (fecha_baja IS NOT null).
2. Se pide realizar la función **jugadores_retirados_delgados** que acceda a la base de datos y genere un fichero de texto “jugadores.txt” que contenga una línea con el DNI de cada jugador no en activo por debajo del peso medio obtenido con la función del apartado anterior.
3. Se pide realizar un **programa principal** que invoque las dos funciones anteriores a modo de demostración.

PROBLEMA 50

Se proporciona un fichero SQLite (“equipo.db”) conteniendo la base de datos del equipo de fútbol con la siguiente estructura:

```
CREATE TABLE jugadores(
dni TEXT PRIMARY KEY,
fecha_baja TEXT NULL,
peso INTEGER NOT NULL,
altura REAL NOT NULL )
```

1. Se pide realizar una función llamada ***bd_to_diccionario*** que acceda a la base de datos y devuelva un diccionario con la siguiente estructura: { dni: (fecha_baja, peso, altura) } . La clave de los elementos del diccionario será el dni y el valor asociado será una tupla formada por la fecha de baja, el peso y la altura.
2. Se pide realizar una función llamada ***diccionario_to_fichero*** que recibiendo un diccionario con la estructura { dni: (fecha_baja, peso, altura) } cree un fichero de texto de nombre “jugadores.txt” en formato CSV. Cada línea del fichero tendrá la siguiente estructura: dni,fecha_baja,peso,altura
3. Se pide realizar un **programa principal** que invoque las dos funciones anteriores a modo de demostración.

PROBLEMA 51

Se proporciona un fichero SQLite (“equipo.db”) conteniendo la base de datos del equipo de fútbol con la siguiente estructura:

```
CREATE TABLE jugadores(
dni TEXT PRIMARY KEY,
fecha_baja TEXT NULL,
peso INTEGER NOT NULL,
altura REAL NOT NULL )
```

1. Se pide realizar una función llamada ***filtrar_peso*** que reciba como parámetros un valor de peso mínimo y un valor de peso máximo. Debe acceder a la base de datos y devolver una lista de los DNI de los jugadores cuyo peso esté en el intervalo especificado.
2. Se pide realizar una función llamada ***lista_to_fichero*** que recibiendo una lista de los DNI de varios jugadores cree un fichero de texto, de nombre “jugadores.txt”, en formato CSV donde cada línea serán los datos de un jugador cuyo DNI esté en la lista de DNI recibida. Cada línea del fichero tendrá la siguiente estructura: dni,fecha_baja,peso,altura
3. Se pide realizar un **programa principal** que invoque las dos funciones anteriores a modo de demostración.

PROBLEMA 52

Se proporciona un fichero SQLite (“equipo.db”) conteniendo la base de datos del equipo de fútbol con la siguiente estructura:

```
CREATE TABLE jugadores(
dni TEXT PRIMARY KEY,
fecha_baja TEXT NULL,
peso INTEGER NOT NULL,
altura REAL NOT NULL )
```

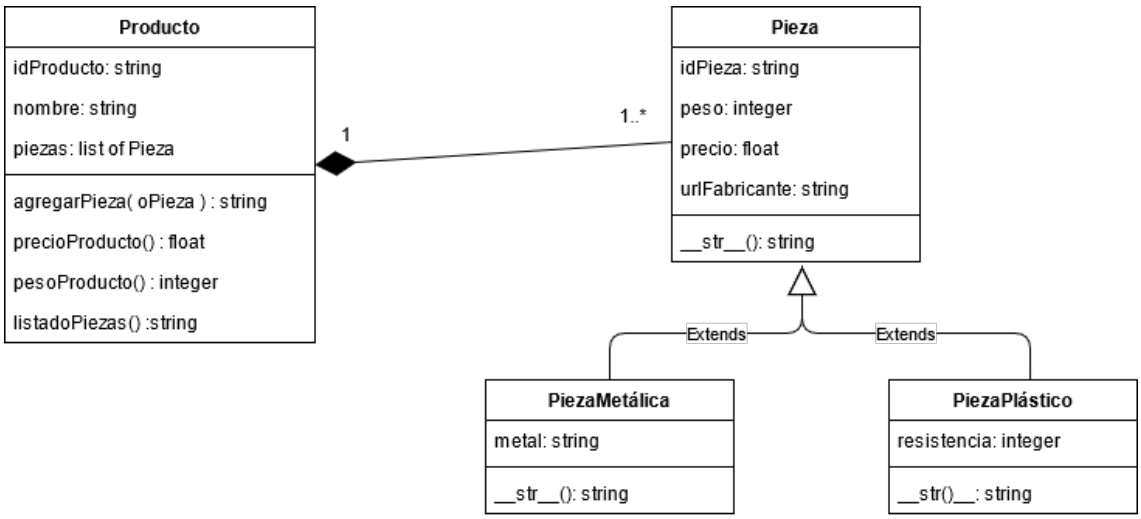
1. Se pide realizar una función llamada ***filtrar_altura_peso*** que reciba como parámetros un valor de altura mínima y un valor de peso máximo. Debe acceder a la base de datos y devolver una tupla con los DNI de los jugadores cuya altura esté por encima de la altura mínima recibida como parámetro y cuyo peso esté por debajo del peso máximo recibido como parámetro.
2. Se pide realizar una función llamada ***lista_to_fichero*** que recibiendo una tupla de los DNI de varios jugadores cree un fichero de texto, de nombre “jugadores.txt”, donde cada línea serán los datos de un jugador cuyo DNI esté en la tupla de DNI recibida. Cada línea del fichero tendrá la siguiente estructura y formato, utilizando guiones como separadores de campos: dni- fecha_baja-peso-altura

3. Se pide realizar un **programa principal** que invoque las dos funciones anteriores a modo de demostración.

PROBLEMAS DE PROGRAMACIÓN ORIENTADA A OBJETOS

PROBLEMA 53

Escriba un programa donde implemente las clases que aparecen en el siguiente diagrama UML. Es necesario crear un pequeño programa principal a modo de demostración que instancie diversos objetos de las clases existentes y se utilicen los métodos implementados.

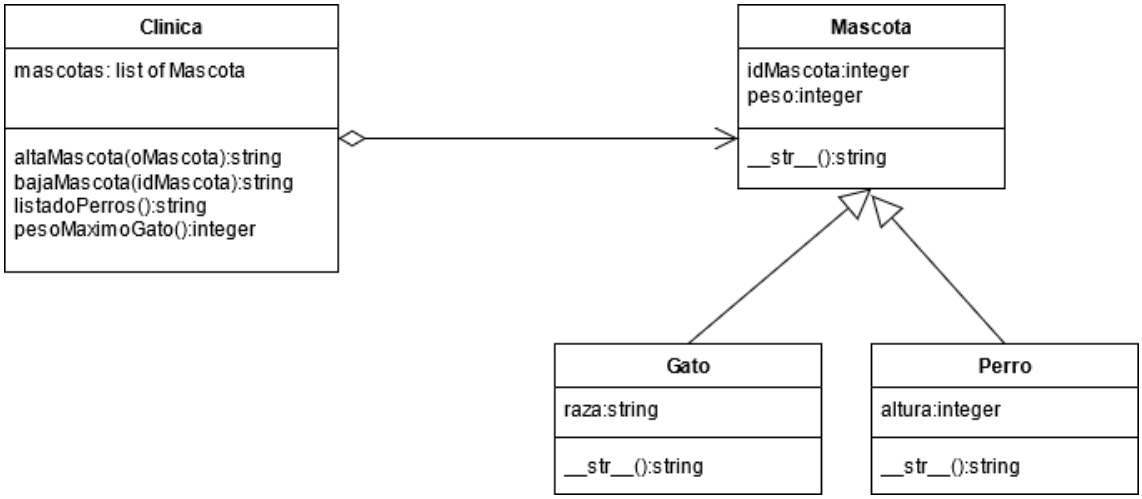


- Cada clase se creará en un módulo distinto del paquete Fabrica.
- Se deja de cuenta del programador definir los métodos constructores más apropiados para cada clase.
- Se realizará la encapsulación mediante propiedades, utilizando para ello el decorador @property, para los atributos de las clases: *Pieza, PiezaMetalica y PiezaPlastico*.
- Los atributos de todas las clases se declararán privados.

- El método agregarPieza(oPieza) recibe un objeto que puede ser una PiezaMetalica o una PiezaPlástico. Inserta la pieza en la lista de piezas de la clase Producto, siempre que la pieza no esté repetida. Devuelve una cadena de texto con los mensajes “Alta OK” o “Pieza repetida”, según sea el caso.
- El método precioProducto() devuelve el precio del producto calculado de forma que se sume el precio en euros de todas las piezas que componen el producto.
- El método pesoProducto() devuelve el peso del producto calculado de forma que se sume el peso en gramos de todas las piezas que componen el producto.
- El método listadoPiezas() devuelve una cadena de texto con un listado que mostrará en cada fila la información de las piezas que componen el Producto.

PROBLEMA 54

Escriba un programa donde implemente las clases que aparecen en el siguiente diagrama UML. Es necesario crear un pequeño programa principal a modo de demostración que instancie diversos objetos de las clases existentes y se utilicen los métodos implementados.

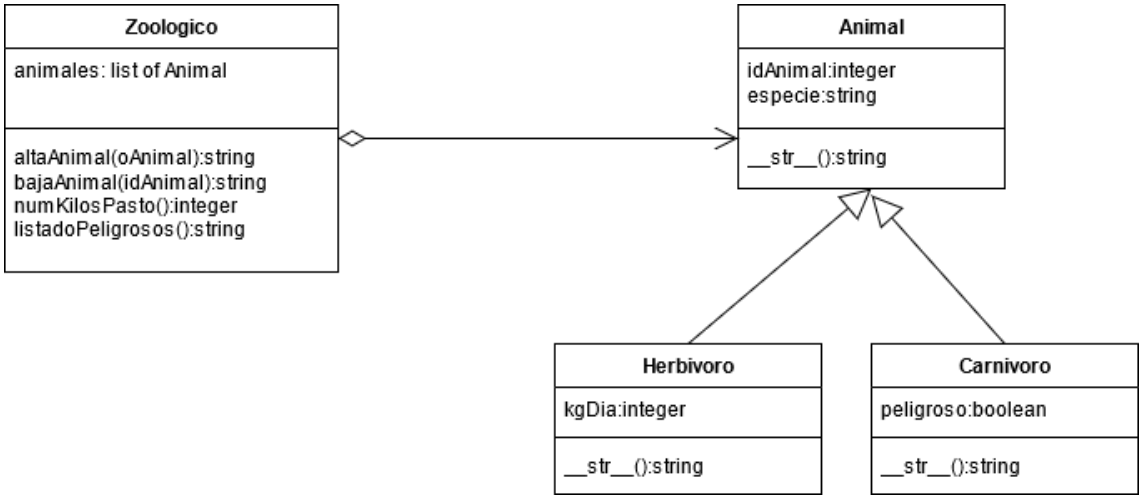


- Cada clase se creará en un módulo distinto del paquete Clinica.
- Se deja de cuenta del programador definir los métodos constructores más apropiados para cada clase.
- Se realizará la encapsulación mediante propiedades, utilizando para ello el decorador @property, para los atributos de las clases: *Mascota, Gato y Perro*.
- Los atributos de todas las clases se declararán privados.

- El método altaMascota(oMascota) recibe un objeto que puede ser un Gato o un Perro. Inserta la mascota en la lista de mascotas de la clase Clinica, siempre que la mascota no esté repetida. Devuelve una cadena de texto con los mensajes “Alta OK” o “Mascota repetida”, según sea el caso.
- El método bajaMascota(idMascota) recibe el identificador numérico de una mascota y elimina dicha mascota de la lista de mascotas de la clase Clínica. Devuelve una cadena de texto con los mensajes “Mascota dada de baja” o “Mascota no localizada”, según sea el caso.
- El método pesoMaximoGato() devuelve el peso en gramos del gato registrado en la clínica que tenga mayor peso.
- El método listadoPerros() devuelve una cadena de texto con un listado que mostrará en cada fila la información de los perros registrados en la clínica.

PROBLEMA 55

Escriba un programa donde implemente las clases que aparecen en el siguiente diagrama UML. Es necesario crear un pequeño programa principal a modo de demostración que instancie diversos objetos de las clases existentes y se utilicen los métodos implementados.

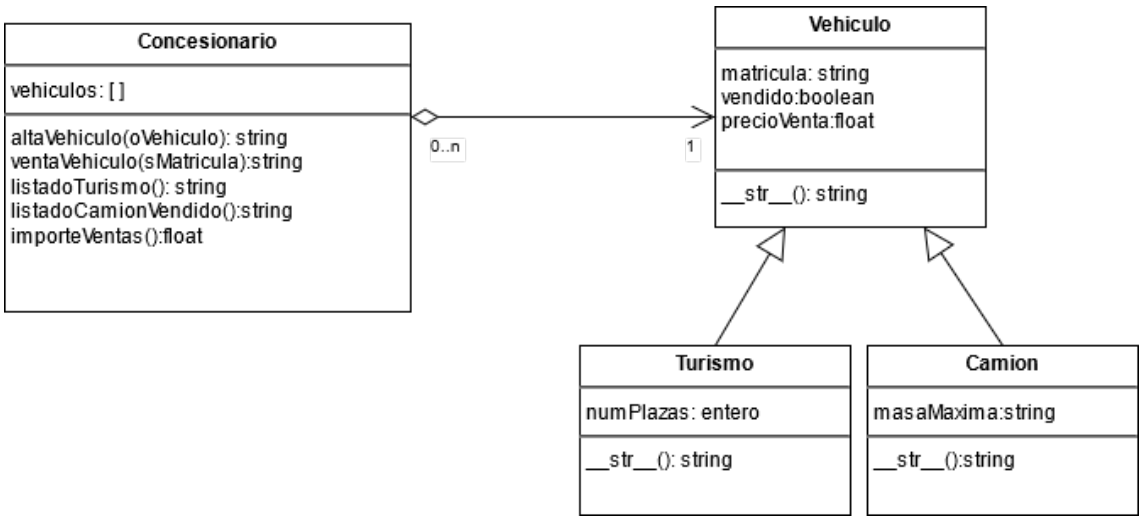


- Cada clase se creará en un módulo distinto del paquete Zoologico.
- Se deja de cuenta del programador definir los métodos constructores más apropiados para cada clase.
- Se realizará la encapsulación mediante propiedades, utilizando para ello el decorador @property, para los atributos de las clases: *Animal, Herbivoro y Carnivoro*.
- Los atributos de todas las clases se declararán privados.

- El método altaAnimal(oAnimal) recibe un objeto que puede ser un Herbivoro o un Carnivoro. Inserta el animal en la lista de animales de la clase Zoologico, siempre que el animal no esté repetido. Devuelve una cadena de texto con los mensajes “Alta OK” o “Animal repetido”, según sea el caso.
- El método bajaAnimal(idAnimal) recibe el identificador numérico de un animal y elimina dicho animal de la lista de animales de la clase Zoologico. Devuelve una cadena de texto con los mensajes “Animal dado de baja” o “Animal no localizado”, según sea el caso.
- El método numKilosPasto() devuelve el peso en kilogramos del pasto necesario para alimentar en un día a los herbívoros registrados en el zoológico.
- El método listadoPeligrosos() devuelve una cadena de texto con un listado que mostrará en cada fila la información de los carnívoros considerados peligrosos y que estén registrados en el zoológico.

PROBLEMA 56

Escriba un programa donde implemente las clases que aparecen en el siguiente diagrama UML. Es necesario crear un pequeño programa principal a modo de demostración que instancie diversos objetos de las clases existentes y se utilicen los métodos implementados.



- Cada clase se creará en un módulo distinto del paquete Concesionario.
- Se deja de cuenta del programador definir los métodos constructores más apropiados para cada clase, aunque hay que tener en cuenta que cuando se crea un vehículo NO puede estar vendido.
- Se realizará la encapsulación mediante propiedades, utilizando para ello el decorador @property, para los atributos de las clases: *Vehículo, Turismo y Camión*.
- Los atributos de todas las clases se declararán privados.

- El método altaVehiculo(oVehiculo) recibe un objeto que puede ser un Turismo o un Camión. Inserta el vehículo en la lista de vehículos de la clase Concesionario, siempre que el vehículo no esté repetido tomando como clave la matrícula. Devuelve una cadena de texto con los mensajes “Alta OK” o “Vehículo repetido”, según sea el caso.
- El método ventaVehiculo recibe la matrícula de un vehículo que se va a vender y actualiza el estado del vehículo en la lista de vehículos a vendido (vendido=True). Devuelve una cadena de texto con los mensajes “Vehículo vendido”, “Matricula no localizada” o “Vehículo vendido anteriormente”, según sea el caso.
- El método importeVentas() devuelve la suma de los precios de venta de todos los vehículos que se encuentren en la lista de vehículos y que se encuentren vendidos.
- El método listadoTurismo() devuelve una cadena de texto con un listado que mostrará en cada fila la información de los turismos.
- El método listadoCamionVendido() deveulve una cadena de texto con un listado que mostrará en cada fila la información de los camiones en estado vendido.

PROBLEMAS DE PROGRAMACIÓN FUNCIONAL

PROBLEMA 57

Dado el diccionario de datos de ejemplo, realice las operaciones que se piden utilizando funciones de orden superior (filter, map, reduce), funciones lambda y comprensión de listas. Cualquier solución que utilice bucles no se considerará válida. Se pueden dar soluciones utilizando resultados (listas, generadores, tuplas, …) auxiliares o intermedios.

- La clave del diccionario está compuesta por una tupla que indica el tipo de barco (ferry o mercante) y un identificador del barco (id_barco).
- Los valores asociados dependen del tipo de barco:
  - Si se trata de un ferry el valor asociado es una tupla formada por la carga en Kg y el número de pasajeros.
  - Si se trata de un mercante el valor asociado es una tupla formada por la carga en Kg y la autonomía de la embarcación en Km.

```
diccionario = { ("ferry",1) : [ 2500 ,350 ],                    # [ carga, pasajeros ]
                ("mercante",2) : [ 120000, 6500 ],              # [ carga, autonomía ]
                ("mercante",3) : [ 200000, 3200 ],              # [ carga, pasajeros ]
                ("ferry",4) : [ 3520 , 420] ,                   # [ carga, pasajeros ]
              }
```

a) Escriba una función obtenerPasajeros(diccionario), que tomando como parámetro de entrada el diccionario anterior, devuelva una tupla con los pasajeros de cada ferry.

   Para el diccionario dado el resultado sería: (350, 420)

b) Escribir una función mayorCarga(diccionario), que tomando como parámetro de entrada el diccionario anterior, obtenga en una lista de un único elemento el id_barco del mercante con mayor carga.

   Para el diccionario dado el resultado sería: [3]

PROBLEMA 58

Dado el diccionario de datos de ejemplo, realice las operaciones que se piden utilizando funciones de orden superior (filter, map, reduce), funciones lambda y comprensión de listas. Cualquier solución que utilice bucles no se considerará válida. Se pueden dar soluciones utilizando resultados (listas, generadores, tuplas, …) auxiliares o intermedios.

- La clave del diccionario está compuesta por una tupla que indica el tipo de barco (ferry o mercante) y un identificador del barco (id_barco).
- Los valores asociados dependen del tipo de barco:
  - Si se trata de un ferry el valor asociado es una tupla formada por la carga en Kg y el número de pasajeros.
  - Si se trata de un mercante el valor asociado es una tupla formada por la carga en Kg y la autonomía de la embarcación en Km.

```
diccionario = { ("ferry",1) : [ 2500 ,350 ],                    # [ carga, pasajeros ]
                ("mercante",2) : [ 120000, 6500 ],              # [ carga, autonomía ]
                ("mercante",3) : [ 200000, 3200 ],              # [ carga, pasajeros ]
                ("ferry",4) : [ 3520 , 420] ,                   # [ carga, pasajeros ]
              }
```

c) Escriba una función totalCarga(diccionario), que tomando como parámetro de entrada el diccionario anterior, devuelva una lista con un único elemento que contenga la suma de la carga de todos los barcos.

   Para el diccionario dado el resultado sería: [326020]

d) Escribir una función obtenerCarga(diccionario), que tomando como parámetro de entrada el diccionario anterior, devuelva una tupla con las cargas de todos los barcos mercantes.

   Para el diccionario dado el resultado sería: (120000, 200000)

PROBLEMA 59

Dado el diccionario de datos de ejemplo, realice las operaciones que se piden utilizando funciones de orden superior (filter, map, reduce), funciones lambda y comprensión de listas. Cualquier solución que utilice bucles no se considerará válida. Se pueden dar soluciones utilizando resultados (listas, generadores, tuplas, …) auxiliares o intermedios.

```
diccionario = { (1, "jarabe") : ("SI",120),      # ( financiado, volumen)
                (2,"pastilla") : ("SI",20),      # ( financiado, num_pastillas)
                (3,"jarabe") : ("NO",45),
                (4,"jarabe") : ("NO",75),
                (5,"jarabe") : ("SI",35),
                (6,"jarabe") : ("NO",90),
                (7,"pastilla") : ("NO",30),
                (8,"pastilla") : ("SI",15),
              }
```

- La clave del diccionario está compuesta por una tupla que indica el tipo de medicamento (jarabe o pastilla) y un identificador del medicamento (id_medicamento).
- Los valores asociados dependen del tipo de medicamento:
  - Si se trata de un jarabe el valor asociado es una tupla formada por:
    - El texto “SI” o “NO, que indica si el medicamento está financiado por la Seguridad Social.
    - El volumen en ml del medicamento.
  - Si se trata de una pastilla el valor asociado es una tupla formada por:
    - El texto “SI” o “NO, que indica si el medicamento está financiado por la Seguridad Social.
    - El número de pastillas de un envase del medicamento.

a) Escriba una función medicamentosFinanciados(diccionario), que tomando como parámetro de entrada el diccionario anterior, obtenga una tupla con los id_medicamento de los medicamentos financiados por la Seguridad Social.

   Para el diccionario dado el resultado sería: (1, 2, 5, 8)

b) Escriba una función mayorVolumenNoFinanciado(diccionario), que tomando como parámetro de entrada el diccionario anterior, obtenga en una lista de un único elemento el id_medicamento del “jarabe” no financiado con envase de mayor volumen.

   Para el diccionario dado el resultado sería: [6]

PROBLEMA 60

Dado el diccionario de datos de ejemplo, realice las operaciones que se piden utilizando funciones de orden superior (filter, map, reduce), funciones lambda y comprensión de listas. Cualquier solución que utilice bucles no se considerará válida. Se pueden dar soluciones utilizando resultados (listas, generadores, tuplas, …) auxiliares o intermedios.

```
diccionario = { (1, "jarabe") : ("SI",120),      # ( financiado, volumen)
                (2,"pastilla") : ("SI",20),      # ( financiado, num_pastillas)
                (3,"jarabe") : ("NO",45),
                (4,"jarabe") : ("NO",75),
                (5,"jarabe") : ("SI",35),
                (6,"jarabe") : ("NO",90),
                (7,"pastilla") : ("NO",30),
                (8,"pastilla") : ("SI",15),
              }
```

- La clave del diccionario está compuesta por una tupla que indica el tipo de medicamento (jarabe o pastilla) y un identificador del medicamento (id_medicamento).
- Los valores asociados dependen del tipo de medicamento:
    - Si se trata de un jarabe el valor asociado es una tupla formada por:
        - El texto “SI” o “NO, que indica si el medicamento está financiado por la Seguridad Social.
        - El volumen en ml del medicamento.
    - Si se trata de una pastilla el valor asociado es una tupla formada por:
        - El texto “SI” o “NO, que indica si el medicamento está financiado por la Seguridad Social.
        - El número de pastillas de un envase del medicamento.

c) Escriba una función pastillasNoFinanciadas(diccionario), que tomando como parámetro de entrada el diccionario anterior, obtenga una tupla con los id_medicamento de los medicamentos de tipo “pastilla” no financiados por la Seguridad Social.

Para el diccionario dado el resultado sería: (7)

d) Escriba una función menorEnvase(diccionario), que tomando como parámetro de entrada el diccionario anterior, obtenga en una lista de un único elemento el valor mínimo para un envase de medicamento de tipo “pastilla”

Para el diccionario dado el resultado sería: [15]

PROBLEMAS DE PANDAS

PROBLEMA 61

Utilice para este ejercicio el conjunto de datos en formato CSV "felicidad-mundial-2020.csv" que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Muestre los nombres (Country name) y la región (Regional indicator) de países donde la renta per cápita (Logged GDP per capita) esté entre 10.000 y 11.000 dólares y el parámetro de libertad (Freedom to make life choices) sea inferior a 0.900.
3. Construya y muestre un nuevo dataframe que contenga los siguientes datos agrupados por región (Regional indicator):
    - Puntuación media de felicidad (Ladder score)
    - Expectativa media de vida saludable (Healthy life expectancy)
    - Media de renta per cápita (Logged GDP per cápita)

PROBLEMA 62

Utilice para este ejercicio el conjunto de datos en formato CSV "felicidad-mundial-2020.csv" que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Muestre la puntuación media de felicidad (Ladder score) y expectativa de vida saludable (Healthy life expectancy) agrupados por región (Regional indicator).
3. Construya y muestre un nuevo dataframe que contenga los datos de:
    - El país con mayor puntuación y el país con menor puntuación (Ladder score)
    - El país con mayor renta y el país con menor renta per capita (Logged GDP per capita).
    - El país más generoso y el país menos generoso (Generosity)

PROBLEMA 63

Utilice para este ejercicio el conjunto de datos en formato CSV "felicidad-mundial-2020.csv" que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Muestre los nombres (Country name) y la esperanza de vida con salud (Healthy life expectancy) de los cinco países donde las personas tienen una menor esperanza de vida con salud (Healthy life expectancy).
3. Construya y muestre un nuevo dataframe que contenga la media, el mínimo y el máximo de las siguientes columnas:
    - Ladder score
    - Freedom to make life choices
    - Perceptions of corruption

PROBLEMAS DE MATPLOTLIB

PROBLEMA 64

Utilice para este ejercicio el conjunto de datos en formato CSV “felicidad-mundial-2020.csv” que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Mostrar en una gráfica de barras, para los tres países con mayor puntuación de felicidad (Ladder Score) y los tres países con menor puntuación de felicidad (Ladder Score), los siguientes datos:
    - Puntuación media de felicidad (Ladder score)
    - Media de renta per cápita (Logged GDP per cápita)
3. Guardar en un fichero “grafica.png” la gráfica generada.

PROBLEMA 65

Utilice para este ejercicio el conjunto de datos en formato CSV “felicidad-mundial-2020.csv” que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Mostrar en una gráfica de sectores (de tarta) el número de países agrupados por región que se incluye en el estudio. Es decir, aparecerá cada región (Regional indicator) con el número de países de dicha región que aparece en el estudio.
3. Guardar en un fichero “grafica.png” la gráfica generada.

PROBLEMA 66

Utilice para este ejercicio el conjunto de datos en formato CSV “felicidad-mundial-2020.csv” que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Mostrar en una gráfica un histograma que muestre la distribución de la variable expectativa de vida con salud (Healthy life expectancy) en 10 intervalos de los 100 países con mayor puntuación en felicidad (Ladder score).
3. Guardar en un fichero “grafica.png” la gráfica generada.

PROBLEMA 67

Utilice para este ejercicio el conjunto de datos en formato CSV “felicidad-mundial-2020.csv” que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Mostrar en una gráfica de puntos (scatter) los diez países con menor puntuación de felicidad (Ladder score) frente a la variable libertad para elegir en la vida (Freedom to make life choices).
3. Guardar en un fichero “grafica.png” la gráfica generada.

PROBLEMA 68

Utilice para este ejercicio el conjunto de datos en formato CSV “felicidad-mundial-2020.csv” que se puede descargar desde blackboard.

Escriba un programa que haga las siguientes operaciones con el anterior conjunto de datos:

1. Cargue en un dataframe de pandas todos los datos.
2. Mostrar en una gráfica de puntos (scatter) los diez países con menor puntuación de felicidad (Ladder score) frente a la variable libertad para elegir en la vida (Freedom to make life choices).
3. Guardar en un fichero “grafica.png” la gráfica generada.