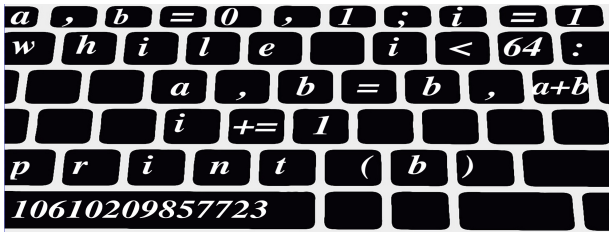

FUNDAMENTOS DE COMPUTACIÓN
Grado en ingeniería en tecnologías Industriales
Universidad de Cantabria

Complementos de Programación
Diciembre del 2020

---

1. Implementar una función de acuerdo con la documentación descrita abajo. Necesitas usar la variable global:

   FECHA= 07-01-2020

```
def edad(cadena_cumple):
""" Entrada (str) ---> SALIDA (int)
  Requisito previo: <<cadena_cumple>> tiene el formato DD-MM-AAAA,
representa un día, un mes y un año anterior a FECHA.
   Devuelve la edad(int)  de la persona con fecha de nacimiento
<<cadena_cumple>>  al día de la FECHA. """
```

```
>>> edad('07-01-2001')
19
>>> edad('08-01-2001')
18
>>> edad('31-10-1969')
50
```

```
| def edad(cadena_cumple):
|
|
|
|
|
|
|
|
```

2. Se dispone de un archivo (potencialmente grande) llamado `skype_datos.txt` que contiene información sobre los usuarios de Skype. Cada usuario tiene tres piezas de información en el archivo, cada uno etiquetado con una de las palabras clave USUARIO, LUGAR y FECHA DE NACIMIENTO. El orden de la información puede variar, pero las tres piezas para cada usuario aparecerán antes de que comience el próximo usuario. Cada palabra clave está en una sola línea, y la siguiente contiene la información particular para el usuario.
   Un ejemplo de un posible archivo `skype_datos.txt` se muestra a la derecha.
   Queremos un nuevo archivo que contenga las edades de los usuarios del archivo `skype_datos.txt` con una línea por usuario, en el mismo orden en que aparecen en el archivo de datos. Por ejemplo, la solución basada en el archivo de `skype_data.txt` sería:

```
19
18
50
```

```
USUARIO
skyper108
LUGAR
Fontecha
FECHA DE NACIMIENTO
07-01-2001
USUARIO
lamilana
FECHA DE NACIMIENTO
08-01-2001
LUGAR
Argüeso
LUGAR
Santander
USUARIO
liguerucu
FECHA DE NACIMIENTO
31-10-1969
```

   Implementar una función de acuerdo con la documentación descrita abajo, generando un nuevo archivo. Deberás usar la función del ejercicio anterior para calcular la edad.

```
def escribir_edad_archivo(fichero_datos,  fichero_edad):
"""
Entrada (fichero_datos para leer, y fichero_edad  para escribir ) ---> SALIDA  NoneType
  Requisito previo:  fichero_datos   debe ser un fichero valido para Skype,
Escribir la edad de cada  usuario  en fichero_edad  obtenida desde fichero_datos.
 """
```

3. Considerar la siguiente función escrita en Python:

```
def saludar(L):
    """
    La entrada L es una lista (list) ----> la salida NoneType
    """
     for x in L:
     print("Hola")
```

En cada uno de los siguientes 4 códigos se ejecuta para una lista `L` de objetos Python de longitud `n`: `len(L)=n`. Para cada uno de ellos, escribir una fórmula para expresar el número de veces que muestra por pantalla la palabra `Hola`. La fórmula puede depender de `n`. En la última columna señalar cuándo la dependencia en `n` es constante, lineal, cuadrática u otra.

Código Python	Número de veces que Hola se muestra por pantalla	Dependencia en n
`saludar(L)`		constante lineal cuadrático otro

`i=0`
`while i < len(L):`
`  saludar(L[i:i+1])`
`  i = i+1`
		constante
		lineal
		cuadrático
		otra

`i=0`
`while i < len(L):`
`  saludar(L[i:])`
`  i = i+1`
		constante
		lineal
		cuadrático
		otra

`i=0`
`while not( i < len(L)):`
`  saludar(L)`
`  i = i+1`
		constante
		lineal
		cuadrático
		otra

4. Se considera la siguiente clase:

```
class Bicicleta(object):
    def __init__(self, pi, pl):
        self.piñones = pi
        self.platos = pl
        self.color = ""
```

Marcar, en cada caso, las afirmaciones correctas. Puede haber más de una.

(a) ¿Cuáles de las siguientes instrucciones crea un objeto Bicicleta con 4 piñones y dos platos ?

```
1)  Bicicleta(mibici, 4, 2)
2)  mibici = Bicicleta(4, 2)
3)  mibici = Bicicleta(4,2,"blanca")
4)  mibici = Bicicleta(2,4)
```

(b) ¿Cuáles de los siguientes métodos cambia el color de un objeto Bicicleta ?

```
1)  def pintar(c):
        color= c
2)  def pintar(self,c):
        color =c
3)  def pintar(c):
        self.c =c
4)  def pintar(self,c):
        self.color=c
```

(c) Adaptar el método especial `__eq__` para comparar cuando dos objetos Bicicleta son iguales.

5. **Variaciones con repetición.**— Contamos con cinco semáforos numerados, cada uno de los cuales podemos colocar en tres posiciones (verde, rojo o ámbar). ¿Cuántos mensajes podemos codificar con estos semáforos? Es decir: ¿de cuántas maneras distintas podemos configurar el conjunto?

   Dados m dispositivos etiquetados (de modo que podemos distinguir uno de otro) que pueden adoptar n posiciones distintas (cada uno de ellos), ¿de cuántas maneras se puede configurar el conjunto?

   Si los dispositivos fueran indistinguibles (si no estuvieran etiquetados), la respuesta sería:

$$\binom{n+m-1}{m} = \binom{n+m-1}{n-1} = \frac{(n+m-1)!}{m!(n-1)!}.$$

   En particular, si no conserváramos la posición que ocupa cada dígito, con m bits solo podríamos representar $\binom{m+1}{1} = m+1$ números (poniendo m ceros y ningún uno, $m-1$ ceros y un uno,..., o ningún cero y m unos).

6. El denominado método de la bisección permite encontrar una aproximación a un cero de una función matemática $f(x)$ en un intervalo $[a, b]$ si $f(x)$ es continua en dicho intervalo y $f(a)$ y $f(b)$ son de distinto signo. Consiste en dividir el intervalo en dos partes iguales. Sea c el punto medio del intervalo, si el signo de $f(c)$ tiene el mismo que $f(a)$, aplicamos el método al intervalo $[c, b]$ en caso contrario aplicamos el método al intervalo $[a, c]$. El algoritmo finaliza cuando encontramos un punto c tal que $f(c) = 0$ o cuando la longitud del intervalo de búsqueda es menor que un cierto número real positivo ϵ. Implementa en PYTHON el método de la bisección para un polinomio $f(x)$, un intervalo $[a, b]$ tal que $f(a)f(b) < 0$ y un real positivo ϵ.

7. Diremos que una entrada está bien formateada con respecto a una lista de etiquetas si:

   1. La entrada es una lista.
   2. La entrada tiene una longitud de al menos dos.
   3. El primer elemento de la entrada está en las etiquetas de la lista.
   4. Cada uno de los elementos restantes en la entrada es una cadena o una lista bien formateada.

   Algunos ejemplos de entradas bien formateadas y no bien formateadas:

ENTRADA	ETIQUETAS	BIEN FORMATEADAS
`["VP",["V", "hola"]]`	`["VP", "V" ]`	`True`
`["NP",["N", "a","or","b"],"c"]`	`["NP", "V", "N"]`	`True`
`["NP",[2,"oui", ["1","no"]],"no"]`	`["NP","1",2]`	`True`
`["VP",["V", "hola"]]`	`["VP"]`	`False: "V" no es etiqueta`
`["VP", ["V"]]`	`["VP", "V" ]`	`False: la list ["V"] es corta`
`"VP"`	`["VP", "V" ]`	`False: la entrada no es una lista`

   Implementar la siguiente función recursivamente según la documentación:

```
def  bien_formateada (entrada, etiquetas):
        "" "Devuelve: el booleno True si la <entrada>  está bien formateada  con respecto a <etiquetas> ;
          False en caso  contrario.
       Pre: <etiquetas> es una lista (posiblemente vacía) "" "
```

8. Demostrar que el número de movimientos del algoritmo de las Torres de Hanoi es $2^n - 1$, donde n es el número de discos.

9. Completar la siguiente tabla, donde, para cada función $f(n)$ y tiempo t, la siguiente tabla muestra el tamaño más grande n que un problema puede ser resuelto en tiempo t, asumiendo que el algoritmo necesita $f(n)$ microsegundos.

	1 segundo	1 minuto	1 hora	1 dia	1 mes	1 año	1 siglo
$\log n$	2^{1000}	2^{60000}					
$\sqrt{n}$	10^6	$36 \cdot 10^8$					
n	10^3	$6 \cdot 10^4$	$6^2 \cdot 10^5$				
$n \log n$	140	4895					
n^2	31	244		9295			
n^3	10	39	153	442		3519	
2^n	9	15	21	26	31	34	41

10. Consideramos el siguiente programa (hemos omitido la documentación para la brevedad del examen, pero hemos incluído los números de cada línea).

```
1    class Prof(object):
2       def __init__(self, n):
3          self.lnombre = n
4
5     ljl2 = Prof("Turing")
6     srm2 = Prof("Galois")
7
8     clasesprof = srm2
9     clasesprof.sabio = True
10    clasesprof = ljl2
11    print("¿Es el Prof"+srm2.lnombre+"sabio ?" + str(srm2.sabio))
12    print("¿Es el Prof"+ljl2.lnombre+" sabio ?" + str(ljl2.sabio))
```

Escribe los resultados y/o errores que se generarían ejecutando este código, en el orden en que son producidos. Para cada una que escribas, indica qué número de línea lo produce. En el caso de errores, es necesario explicar cuál es el problema, no tienes que saber precisamente lo que Python imprimiría por pantalla.

11. **Nim.**— Para jugar al *Nim* hacen falta dos contrincantes, que tienen que retirar, por turnos, piedras de un tablero. En este hay varios montones y cada jugador, cuando le toca mover, puede retirar cuantas piedras quiera (una por lo menos), pero solo de un montón. El objetivo es retirar la última piedra, de modo que pierde el que ya no puede coger ninguna.

    Existe un conjunto de situaciones del tablero (distribuciones de piedras en montones) desde las que el jugador al que le toca mover puede asegurarse la victoria. Por ejemplo, si en el tablero hay un montón con dos piedras y otro con una (posición [2,1]), el jugador ganará si quita una piedra del montón con dos, dejándole a su adversario [1,1], y asegurándose que hará el último movimiento.

    Se trata de describir el conjunto de posiciones ganadoras y la estrategia que debe seguir el jugador para ganar.

12. A continuación aparecen tres funciones f, g y h de Python que toman como argumento una lista L de números de longitud n.

```
def f(L):
    sum = 0
    i=1
    while i < len(L):
         sum = sum + L[i]
         i = i * 2
    return sum
```

```
def g(L):
    for  i in range(len(L)):
       L[i] = L[i] + 1
    sum = 0
    for x in L:
        sum = sum + x
    return sum
```

```
def h(L):
    sum = 0
    for x in L:
        for i in range(1, 101):
            if x >= i:
                sum = sum + 1
    return sum
```

¿Cuál de los siguientes afirmaciones describe con mayor precisión cómo crece el tiempo de ejecución de cada una de ellas a medida que n crece?

(a) Crece linealmente, como n
(b) Crece cuadráticamente com n^2
(c) Crece menos que linealmente
(d) Crece más que cuadráticamente

SOLUCIÓN para f:	SOLUCIÓN para g:	SOLUCIÓN para h:

13. Diseña un programa recursivo y otro iterativo, tales que dados un número x y un entero positivo n, devuelvan el valor x^n.

14. El *doble factorial* $n!!$ de un número n se define como el producto de todos los números enteros positivos, no mayores que n y con la misma paridad de este número. Por ejemplo:

$$7!! = 1 \times 3 \times 5 \times 7, \qquad 6!! = 2 \times 4 \times 6.$$

Escribe un programa (función) iterativo y otro recursivo para calcular el *doble factorial* de un entero positivo $n > 0$.

15. En términos generales una *transformada* es una función que toma un valor en un sistema y lo convierte a un valor en otro sistema. Por ejemplo la transformación de coordenadas cartesianas a polares o la famosa transformada de Fourier. La transformada de óndula (wavelet) es una herramienta importante en el procesamiento de señales. En este problema, trabajaremos con el tipo más simple de transformación wavelet, la transformada de Haar.

    (a) En cada paso del cálculo, la transformada de Haar reducirá a la mitad el tamaño de la lista original usando dos funciones: `pares_media` y `pares_diferencia`. La función `pares_media` agrupa los elementos de la lista en pares consecutivos y los promedia. Por ejemplo, `pares_media`$([9,7,3,5])$ es $[8,4]$ puesto que $(9+7)/2$ es 8 y $(3+5)/2$ es 4. La función `pares_diferencia` también agrupa los elementos de la lista en pares consecutivos y realiza el promedio de la diferencia. Por ejemplo, `pares_diferencia`$([9,7,3,5])$ es $[1,-1]$ puesto que $(9-7)/2$ es 1 y $(3-5)/2$ es -1.

    Se pide implementar la función `pares_media(L)` usando el bucle FOR, y la función `pares_diferencia` usando el bucle WHILE, donde el argumento L debe ser una lista de números de longitud una potencia de 2.

    (b) La transformada Haar, `transformada_haar(L)`, es un procedimiento que tiene como argumento una lista L de 2^n elementos y produce una lista de otros 2^n elementos. Si la lista L tiene longitud 2, calcula `pares_media(L)` y `pares_diferencia(L)` y devuelve la lista [`pares_media(L)`, `pares_diferencia(L)`]. Si la longitud de la lista L es 2^n donde $n > 1$, entonces sigue recursivamente las siguientes etapas:

    - Computar la función `pares_media(L)`.
    - Computar la funcion `transformada_haar(pares_media(L))`.
    - Computar la función `pares_diferencia(L)`.
    - Concatenar `transformada_haar(pares_media(L))` y `pares_diferencia(L)`.

    Ejemplo: La transformada Haar de la lista $L = [9,7,3,5]$

    - Computar `pares_media(L)` es decir, $[8,4]$
    - Computar la `transformada_haar`([8,4]), es decir, [6,2]
    - Computar `pares_diferencia(L)` es decir, [1,-1].
    - Concatenar las dos listas anteriores, es decir, la salida es [6,2,1,-1]

    Se pide implementar la función `transformada_haar(L)`, donde el argumento L debe ser una lista de números de longitud una potencia de 2.