Herramientas de Programación

2.2.1 Tratamiento de datos estructurados: listas

Ejercicios con strings y listas

1. ***Comienza_letra***. Diseña una función `comienza_letra(s,letra)` que, dado un texto representado en un string, cuente y devuelva el número de palabras del texto que empiezan por la letra 'letra' (minúscula o mayúscula).
```
>>> comienza_letra('Este día comenzamos a estudiar algoritmos', 'e')
2
```

2. ***PalabraLarga***. Diseña una `palabra_mas_larga(s)` que, dado un texto representado en un string, devuelva la palabra de mayor longitud (la primera encontrada si hay más de una del mismo tamaño).
```
>>> palabra_mas_larga('Ayer comenzamos a estudiar algoritmos')
'comenzamos'
```

3. ***ConvertirFecha***. Diseña una función `convfecha(str)` que reciba una fecha del formato 'dd/mm/aaaa' y devuelva una lista con los tres valores de día, mes y año utilizando la función string.split('/') para extraer los valores a una lista y la función int() para transformarlos a enteros.
```
>>> convfecha('12/07/2017')
[12, 7, 2017]
```

4. **Busca mayúsculas**. Diseñar una función `BuscaTexto(lst)` que reciba una lista lst de strings y devuelva True si hay al menos dos strings en lst que empiecen por mayúscula, o False si no los hay.
   Utilizar la siguiente función `Mayus(str)` que devuelve True si el string str empieza por mayúscula, o False si no. Es obligatorio que la función `BuscaTexto(lst)` use la función `Mayus(str)`, que se define como:
```
def  Mayus(str):
    return str[0].isupper()
```
   Ejemplos:
```
>>> buscaTexto(['hola', 'silla', 'mESA'])
False
>>> buscaTexto(['Cuadrado rojo', 'coche azul', 'Bosque', 'Taza'])
True
>>> buscaTexto(['bolígrafo', 'Aventura', 'bicicleta', 'casco', 'horno'])
False
```

Ejercicios con listas

5. ***ListaMultiplos***. Escribir una función `listaMultiplos(lst,n)` que, dada una lista lst y un número natural n, devuelva una lista con sólo los múltiplos de n de lst.

```
>>> listaMultiplos([1,2,3,4,5,6,7,8,9,10,11,12], 3)
[3, 6, 9, 12]
```

6. ***PositivosNegativos.*** Diseña una función `PositNegat(lst)` que reciba una lista, lst, y devuelva 2 listas, una con los Valores positivos o 0 y otra con los negativos.

```
>>> PositNegat([69, -37, 0,  -27, -59, 83, 1, 45])
([69, 0, 83, 1, 45], [-37, -27, -59])
>>> PositNegat([3, 4, 7, 12])
([3, 4, 7, 12], [])
```

7. ***ParImpar***. Diseña una función `parImpar(lst)` que reciba una lista, lst, y devuelva 2 listas, una con los Valores pares y otra con los impares.

```
>>> parImpar([1, 2, 3, 4, 5, 6, 7, 8])
([2, 4, 6, 8], [1, 3, 5, 7])
>>> parImpar([1, 3, 5])
([], [1, 3, 5])
```

8. ***Eliminar Repetidos***. Diseña una función `eliminarRepet(lst)` en que, dada una lista no vacía, `lst`, devuelva otra sin que tenga elementos repetidos. Ayuda: puedes usar `if item not in lista_nueva` ó `lst[i] not in lst[i+1:]`:

```
>>> eliminarRepet([1, 1, 1, 2, 2, 3, 4, 5, 5, 5, 5, 6, 6, 7])
[1, 2, 3, 4, 5, 6, 7]
```

9. ***Mover elementos lista a la derecha (Right shift)***. Diseña una función `mueve_derecha(lst)` (no productiva) que dada una lista, lst, la devuelva con sus elementos corridos hacia la derecha (y el último de primero). *Right-shift*.

```
>>> lst = [2, 4, 6, 8, 10]
>>> mueve_derecha(lst)
>>> lst
[10, 2, 4, 6, 8]
```

10. ***Mover elementos lista a la izquierda (Left shift)***. Diseña una función `mueve_izquierda(lst)` (no productiva) en que, dada una lista, `lst`, la devuelva con sus elementos corridos hacia la izquierda (y el primero de último). *Left-shift*.

```
>>> lst = [2, 4, 6, 8, 10]
>>> mueve_izquierda(lst)
>>> lst
[4, 6, 8, 10, 2]
```

11. **Lista de temperaturas**. La lista `lst` contiene valores de temperatura corporal de un niño. La primera medida de temperatura se hace en la hora 0, luego la hora 1, etc., lo que corresponde con las posiciones de las tempraturas en la lista.

a) Diseñar una función `incrTempN(lst)` que, dada una lista de temperaturas medidas cada hora a un niño, calcular el número de incrementos de temperatura (en una hora respecto a la hora anterior) de al menos 1 ºC de temperatura cuando se tenia fiebre (temperatura >= 38ºC). La función debe devolver el número incrementos de al menos 1 ºC encontrados cuando la temperatura sea de al menos 38ºC. Si no hay estos incrementos devolverá 0. Ejemplos:

```
>>> incrTempN([36,37,37.2,36.8,38,37.5,37.6,38.1,37,37.3])
0
>>> incrTempN([36,37,37.2,36.8,38,38.2,39.2,38.1,37,37.3])
1
```

b) Diseñar una función `incrTempL(lst)` que, dada una lista de temperaturas medidas cada hora a un niño, calcular los incrementos de temperatura (en una hora respecto a la hora anterior) de al menos 1 ºC de temperatura cuando se tenia fiebre (temperatura >= 38ºC) y devolver estos incrementos en una lista. La función debe devolver una lista con los valores de los incrementos de al menos 1 ºC cuando la temperatura sea de al menos 38ºC. Si no hay incrementos (para valores >= 30ºC) se devolverá la lista vacía. Ejemplos:

```
>>> incrTempL([36,37,37.2,36.8,38,37.5,37.6,38.1,37,37.3])
[]
>>> incrTempL([36,37,37.2,36.8,38,39.5,38.7,39.7,37,37.3])
[1.5, 1.0]
```

c) Diseñar una función `incrTempB(lst)` que, dada una lista de temperaturas medidas cada hora a un niño, buscar si hay en alguna hora un incremento (respecto a la hora anterior) de al menos 1 ºC de temperatura cuando se tenia fiebre (temperatura >= 38ºC). La función debe devolver el valor del primer incremento de al menos 1 ºC encontrado y la hora que ocurrió. Si no hubiera incrementos de 1ºC cuando hay fiebre, la función devuelve -1 tanto para el incremento como para la hora. Ejemplos:

```
>>> incrTempB([36,37,37.2,36.8,38,37.5,37.6,38.1,37,37.3])
(-1, -1)
>>> incrTempB([36,37,37.2,36.8,38,38.2,39.2,38.1,37,37.3])
(1.0, 6)
```