Estructuras iterativas: bucles acotados `for_in`

A través de los bucles, podemos hacer que una instrucción o secuencia de instrucciones se repitan (un número determinado) de veces. La instrucción básica es `for`.

Sintaxis:

```
for <<variable>> in <<secuencia>>:
    <<cuerpo del bucle>>
```

Bucles `for_in` con listas

In [1]:
```
def nota_media(lista_de_notas):
    """
    Calcula la media de una lista de notas numéricas

    Parameters
    ----------
    notas : [float]
        Lista de notas, no vacía

    Returns
    -------
    float
        Media de las notas

    Example
    -------
    >>> nota_media([5, 6, 9, 10])
    7.5
    """
    suma = 0.0
    for nota in lista_de_notas:
        suma = suma + nota
    return suma / len(lista_de_notas)

nota_media([4.5, 6, 5]), nota_media([4, 6, 5, 7, 5, 6, 8]), nota_media([5, 6, 9, 10])
```

Out[1]: (5.166666666666667, 5.857142857142857, 7.5)

In [2]:
```
def nombres_cortos(lista_de_nombres, n):
    """
    La función filtra una lista_de_nombres,
    devolviendo únicamente los de longitud menor o igual que n.

    Parameters
    ----------
    lista_de_nombres : [string]
        Lista of strings
    n : int
        Longitud maxima

    Returns
    -------
    [string]
        Lista de los nombres de lista_de_nombres con long <= n

    Example
    -------
    >>> nombres_cortos(['Ana', 'Marta', 'Patricia', 'Alba', 'Silvia', 'Gloria', 'Lara'], 3)
    ['Ana']
    """
    n_cortos = []
    for nombre in lista_de_nombres:
        if len(nombre) <= n:
            n_cortos.append(nombre)
    return n_cortos

lista = ['Ana', 'Marta', 'Patricia', 'Alba', 'Silvia', 'Gloria', 'Lara']
nombres_cortos(lista, 5), nombres_cortos(lista, 3)
```

Out[2]: (['Ana', 'Marta', 'Alba', 'Lara'], ['Ana'])

La función `range()`

La función `range()` genera un rango, algo muy parecido a una lista, y que se puede convertir en una lista sencillamente:

In [3]:
```
range(10)
```

Out[3]: range(0, 10)

In [4]:
```
list(range(10))
```

Out[4]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Esta función es muy flexible: con dos parámetros, toma el primero como el valor inicial; y con tres, el tercero da el incremento entre elementos:

In [5]:
```
list(range(5, 10)), list(range(5, 20, 3)), list(range(10, 5, -1))
```

Out[5]: ([5, 6, 7, 8, 9], [5, 8, 11, 14, 17], [10, 9, 8, 7, 6])

Bucles `for_in` para listas generadas con `range()`

La función `range` se usa con frecuencia para diseñar bucles `for_in`:

In [6]:
```
def lista_aleatoria(n):
    """
    genera una lista aleatoria de n enteros aleatorios entre 0 y 100.

    Parameters
    ----------
    n : int
        Longitud de la lista resultante
        n >= 0

    Returns
    -------
    [int]
        Lista de n enteros aleatorios entre 0 y 100

    Example
    -------
    >>> lista_aleatoria(3)
    [1, 88, 31]
    """
    import random
    lista = []
    for x in range(n):
        lista.append(random.randint(0, 100))
    return lista

lista_aleatoria(3), lista_aleatoria(3), lista_aleatoria(5)
```

Out[6]: ([19, 26, 77], [59, 56, 88], [69, 3, 65, 46, 82])

In [7]:
```
def multiplos_de_7_y_5(n):
    """
    Genera la lista de enteros de [0...n] que son
    multiplos de 7 y de 5 simultáneamente.

    Parameters
    ----------
    n : int
        Límite superior

    Returns
    -------
    [int]
        Lista de múltiplos de 7 y 5

    Example
    -------
    >>> multiplos_de_7_y_5(100)
    [0, 35, 70]
    """
    multiplos = []
    for x in range(n):
        if (x % 5 == 0) and (x % 7 == 0):
            multiplos.append(x)
    return multiplos

multiplos_de_7_y_5(100)
```

Out[7]: [0, 35, 70]

Otra versión, saltando ya en el bucle de 7 en 7:

In [8]:
```
def multiplos_de_7_y_5(n):

    multiplos = []
    for x in range(7, n, 7):
        if x % 5 == 0:
            multiplos.append(x)
    return multiplos

multiplos_de_7_y_5(100)
```

Out[8]: [35, 70]

In [9]:
```
def lista_inversa(lista_inicial):
    """
    Genera una lista como la dada, pero en orden invertido.

    Parameters
    ----------
    lista_inicial : list
        Lista original

    Returns
    -------
    list:
        Lista invertida

    Example
    -------
    >>> lista_inversa([1, 2, 3, 4])
    [4, 3, 2, 1]
    """
    lista_inv = []
    for i in lista_inicial:
        lista_inv.insert(0, i)
    return lista_inv

lista_inversa([1,2,3,4]), lista_inversa(["hola","buenas","tardes"])
```

Out[9]: ([4, 3, 2, 1], ['tardes', 'buenas', 'hola'])

Bucles `for_in` en strings

In [10]:
```
for c in 'hola':
    print(c, end='')
```

hola

In [11]:
```
for c in 'Buenas Tardes Ángel':
    print(c.lower(), ord(c.lower()), c.upper(), ord(c.upper()))
```

```
b 98 B 66
u 117 U 85
e 101 E 69
n 110 N 78
a 97 A 65
s 115 S 83
  32   32
t 116 T 84
a 97 A 65
r 114 R 82
d 100 D 68
e 101 E 69
s 115 S 83
  32   32
á 225 Á 193
n 110 N 78
g 103 G 71
e 101 E 69
l 108 L 76
```

In [12]:
```
for c in 'buenas tardes':
    print('{0}\t{1}\t{2}\t{3}'.format(c.lower(), ord(c.lower()), c.upper() ,ord(c.upper())))
```

```
b       98      B       66
u       117     U       85
e       101     E       69
n       110     N       78
a       97      A       65
s       115     S       83
        32              32
t       116     T       84
a       97      A       65
r       114     R       82
d       100     D       68
e       101     E       69
s       115     S       83
```

In [13]:
```
def cuenta_letras(letra, palabra):
    """
    Cuenta el número de apariciones de una letra en la palabra dada

    Parameters
    ----------
    letra : string
        Letter to count the occurrences
    word : string
        Word

    Result
    ------
    int
        Número de veces que la letra aparece en la palabra

    Example
    -------
    >>> cuenta_letras('o', 'pelirrojo')
    2
    """
    contador = 0
    for char in palabra:
        if char == letra:
            contador = contador + 1
    return contador
```

In [14]:
```
cuenta_letras('o', 'pelirrojo')
```

Out[14]: 2

In [15]:
```
cuenta_letras('j', 'pelirrojo')
```

Out[15]: 1

In [16]:
```
cuenta_letras('a', 'pelirrojo')
```

Out[16]: 0

In [17]:
```
cuenta_letras('J', 'pelirrojo')
```

Out[17]: 0